

# POLITECHNIKA WROCŁAWSKA

## WYDZIAŁ ELEKTRONIKI

---

KIERUNEK: Elektronika (EKA)  
SPECJALNOŚĆ: Inżynieria akustyczna (EIA)

## PRACA DYPLOMOWA INŻYNIERSKA

Narzędzie do detekcji i ekstrakcji zdarzeń  
akustycznych w materiale audio

A tool for detecting and extracting acoustic  
events in an audio material

AUTOR:  
Konrad Kamil Janowski

PROWADZĄCY PRACĘ:  
dr Maciej Walczyński, Katedra Akustyki i  
Multimediów (W4/K5)

OCENA PRACY:



*Dla rodziców, którzy zawsze mnie  
wspierali i pozwolili mi rozwinąć  
się w kierunku, który pragnąłem.  
Dziękuję.*



# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>3</b>
1.1	Cel i zakres pracy . . . . .	3
1.2	Motywacja . . . . .	3
<b>2</b>	<b>Wprowadzenie</b>	<b>5</b>
2.1	Zdarzenie akustyczne . . . . .	5
2.2	Przyjęta definicja zdarzenia akustycznego . . . . .	5
2.3	Metoda wykrywania zakłóceń . . . . .	7
2.4	Istniejące rozwiązania do wykrywania zdarzeń akustycznych . . . . .	7
<b>3</b>	<b>Metodologia</b>	<b>9</b>
3.1	System kontroli wersji - Git, GitHub, GitKraken . . . . .	9
3.1.1	System kontroli wersji . . . . .	9
3.1.2	Git . . . . .	10
3.1.3	GitKraken . . . . .	10
3.2	Język programowania - Python . . . . .	11
3.3	Środowisko programistyczne - Pycharm . . . . .	12
<b>4</b>	<b>Struktura oraz działanie programu</b>	<b>13</b>
4.1	Struktura programu - aplikacja konsolowa . . . . .	13
<b>5</b>	<b>Notatki</b>	<b>15</b>
<b>A</b>	<b>Kod Programu</b>	<b>17</b>
	<b>Bibliografia</b>	<b>17</b>



# Rozdział 1

## Wstęp

### 1.1 Cel i zakres pracy

Przedmiotem pracy jest oprogramowanie do detekcji i ekstrakcji zdarzeń akustycznych w materiale audio. Oprogramowanie zostało stworzone do obsługi plików w formacie wave o rozdzielczości bitowej szesnaście(16) i dwadzieścia cztery(24) bity oraz częstotliwości próbkowania czterdzieści cztery i sto dziesiątych (44,1) kilo Herca i czterdzieści osiem (48) kilo Herca. Oprogramowanie zostało przetestowane używając plików o tych parametrach. Oprogramowanie zostało stworzone modularnie aby możliwy był jego dalszy rozwój. Z powodu tego założenia, program ma możliwość obsługi dowolnej częstotliwości próbkowania oraz dowolnej rozdzielczości bitowej choć nie jest to zalecane. Praca zawiera cztery moduły:

1. Moduł odczytywania pliku,
2. Moduł przetwarzania sygnału,
3. Moduł detekcji zdarzeń,
4. Moduł zapisu wyników do pliku.

Autor ma nadzieję, że zaprojektowany w ten sposób system pozwoli na późniejszy rozwój i ułatwi powtórne wykorzystywanie tego narzędzia.

### 1.2 Motywacja

Analiza nagrań odgrywa bardzo ważną rolę we współczesnych zastosowaniach akustyki. Jednym z przykładów może być ochrona ludzi przed hałasem. Aby zbadać hałas na stanowisku pracy lub ocenić narażenie na hałas przestrzeni publicznej niejednokrotnie wykonuje się bardzo długie nagrania, trwające nawet kilkanaście godzin. Ze względu na metody analizy nieraz niezbędne jest wyselekcjonowanie z nagrania konkretnych zdarzeń akustycznych i analiza ich w izolacji, nie biorąc pod uwagę poziomu tła w pozostałym czasie.

Z drugiej strony, możemy wyobrazić sobie potrzebę wyizolowania z nagrania bardzo wielu zdarzeń akustycznych, które wydarzają się w krótkim odstępie czasu. Jako przykład może tutaj posłużyć strzelnica, gdzie chcąc policzyć liczbę strzałów w ciągu dnia możemy zrobić nagranie krótkie i na podstawie tego krótkiego wycinka czasu oszacować hałas podczas całego dnia. Innym razem możemy mieć potrzebę wykryć te zdarzenia to celów

z pozoru nie związanych z akustyką. Gdyby ktoś chciał policzyć ile razy została odbita piłka do koszykówki podczas badania jej wytrzymałości, jednym z możliwych sposobów na zliczenie ilości odbić mogłoby być policzenie zdarzeń akustycznych jakimi są odbicia piłki od ziemi. Być może ktoś ze względów medycznych chciałby sprawdzić, ile czasu w ciągu jego snu zajmuje chrapanie i w ten sposób ocenić jak duży problem ma z tą dolegliwością. Również w tym przypadku, przy pewnej kontroli hałasu środowiska mógłby zrobić to za pomocą stworzonego narzędzia. Jak widać, wyszukiwanie zdarzeń akustycznych wraz z czasem ich trwania może być niezwykle użyteczne w wielu z pozoru nie związanych z tym dziedzinach.

Biorąc pod uwagę, że przytoczone wyżej przykłady są jedynie wąskim wycinkiem zastosowań, które mogłyby przyjść do głowy komuś, kto ma widzę i zainteresowanie w innej dziedzinie niż autor pracy nie ulega wątpliwości, że warto posiadać takie oprogramowanie. Oczywiście wszystkie przytoczone sytuacje można analizować ręcznie, odsłuchując nagrane próbki. Analiza takich nagrań jednak pochłania dużo czasu i może być problematyczna. Z pomocą stworzonego narzędzia można proces zupełnie lub częściowo zautomatyzować co w obu przypadkach skutkuje znaczną oszczędnością czasu.

Praca powstała w połowie z chęci stworzenia użytecznego narzędzia a w połowie z chęci Autora do poznania współczesnych metod analizy cyfrowych sygnałów fonicznych. Rozwinięcie wiedzy w zakresie tworzenia oprogramowania w języku Python, pracy z systemem kontroli wersji oraz implementacja algorytmów znanych z książek do realnie działającego programu jest procesem, który autor chciał zgłębić. Przedstawienie wyników w formie zrozumiałej, czytelnej i praktycznej jest nieraz jeszcze większym wyzwaniem i nie można nabyć w tym wprawy inaczej, niż pracując z tymi wynikami i samemu przekonać się na ile są one użyteczne.

Powyższe przesłanki zadecydowały o stworzeniu narzędzia.



# Rozdział 2

## Wprowadzenie

W tym rozdziale zostaną wprowadzone podstawowe pojęcia, którymi autor posługiwał się podczas tworzenia pracy. Omówione zostaną teoretyczne podstawy problemu oraz ogólna struktura logiczna programu.

### 2.1 Zdarzenie akustyczne

Jak wspomniano w powyższym rozdziale, nie ma jednej ogólnej definicji zdarzenia akustycznego, które odnosi się do wszystkich dziedzin akustyki. Jedną z możliwych definicji podana jest w normie [PN-ISO-1996-1:2006, ] dotyczącej akustyki środowiskowej. Zapis normatywny mówi:

"Należy podawać czas trwania zdarzenia w odniesieniu do pewnej cechy dźwięku, jak np, liczba przekroczeń pewnego ustalonego poziomu.

Przykład: czas trwania zdarzenia można zdefiniować jako całkowity czas, w którym poziom ciśnienia akustycznego mieści się w zakresie 10 dB maksymalnego poziomu ciśnienia akustycznego podczas zdarzenia." [PN-ISO-1996-1:2006, ]

Biorąc pod uwagę przytoczone wcześniej możliwe zastosowania stworzonego oprogramowania powyższa definicja dobrze opisuje te sytuacje, które mają być detekowane. Ponadto, wykonanie oprogramowania, które rozpoznaje zdarzenia opisane w normie daje perspektywę na możliwości jego praktycznego zastosowania.

### 2.2 Przyjęta definicja zdarzenia akustycznego

W poprzedniej sekcji przytoczony został zapis normatywny, który opisuje zdarzenie akustyczne na potrzeby akustyki środowiskowej. Sugeruje on, żeby czas trwania zdarzenia akustycznego definiować jako całkowity czas, w którym poziom ciśnienia akustycznego mieści się w zakresie 10 dB maksymalnego poziomu ciśnienia akustycznego podczas zdarzenia. Nie można zapomnieć, że pomiary w akustyce środowiskowej często wykonywane są przez długi czas. Dla przykładu, pomiar hałasu na stanowisku pracy może być wykonywany przez osiem godzin bez przerwy [PN-ISO-9612:2014, ]. Plik z ośmiogodzinnymi pomiarami może być monofoniczny, zapisany w formacie wave o częstotliwości próbkowania czterdziestu ośmiu kilo Herców (48 kHz) oraz rozdzielczości dwudziestu czterech bitów. Każda próbka takiego pliku zajmuje zatem dwadzieścia cztery bity w pamięci a każda sekunda zawiera czterdzieści osiem tysięcy próbek. [Pohlmann, 2000] Przy pomocy prostego

wzoru możemy obliczyć jego rozmiar w pamięci komputera liczonej w bitach:

$$\text{rozdzielczość bitowa} * \text{częstotliwość próbkowania} * \text{długość pliku} = \text{rozmiar pliku} \quad (2.1)$$

$$24 [b] * 48000 \left[\frac{1}{s}\right] * 8 * 60 * 60 [s] = 24 [b] * 48000 \left[\frac{1}{s}\right] * 28800 [s] = 3.31776 * 10^{10} [b] \quad (2.2)$$

Po przeliczeniu tego na jednostki bardziej sprzyjające interpretacji wyniku, przybliżając że

$$1GB \approx 8 * 10^9 [b]$$

otrzymamy:

$$\frac{3.31776 * 10^{10} [b]}{8 * 10^9 \left[\frac{b}{GB}\right]} = 4,1472 [GB] \quad (2.3)$$

Po dokonaniu takich obliczeń, można zauważyć, że przetwarzania takiego pliku nie jest zadaniem trywialnym. Jeżeli program wczytywałby cały plik do pamięci RAM, potrzebowalby on około czterech gigabajtów tej pamięci na samo obsłużenie pliku. Gdy dodamy do tego potrzebę pamięci operacyjnej dla samego programu, potrzeby systemu operacyjnego oraz innych aplikacji działających równolegle z programem zauważamy problem w postaci braku tych zasobów. Nowoczesne stacje robocze poradziłyby sobie z takim obciążeniem, jednak należy pamiętać o kilku rzeczach:

- Nie wszystkie firmy i osoby fizyczne dysponują stacjami roboczymi, posiadającymi duże zasoby pamięci operacyjnej RAM,
- plik może być dłuższy,
- plik może zostać nagrany z wyższą częstotliwością próbkowania,
- może nastąpić potrzeba współdzielenia pamięci z innymi programami bez możliwości ich wyłączenia.

Powyższe czynniki w głównej mierze skłoniły autora do tego aby swój program oprzeć o przetwarzanie sygnału fragmentarycznie. Dokładny sposób działania programu zostanie omówiony w dalszej części pracy. Co istotne w tym punkcie, wracając do definicji przytoczonej w normie [PN-ISO-1996-1:2006, ] zadaniem nietrywialnym jest osiągnąć jednocześnie obie te funkcjonalności:

1. Odczytywanie programu fragmentarycznie.
2. Zapamiętanie wszystkich poprzednich wartości próbek aby w razie potrzeby cofnąć się do poprzedniego fragmentu celem odnalezienia spadku poziomu o 10 dB.

Mając na uwadze powyższe przesłanki oraz możliwość późniejszego rozszerzania funkcjonalności programu, autor zdecydował się aby ograniczyć definicję zdarzenia akustycznego do definicji:

"Należy podawać czas trwania zdarzenia w odniesieniu do pewnej cechy dźwięku, jak np, liczba przekroczeń pewnego ustalonego poziomu." [PN-ISO-1996-1:2006, ]

Powyższa definicja jest mniej praktyczna niż jej rozszerzona wersja. Wymaga ona wiedzy o pewnych danych środowiska i zdarzenia, jak np. poziom tła akustycznego oraz przewidywany poziom ciśnienia akustycznego generowany podczas zdarzenia. Pomimo świadomości tych ograniczeń autor postanowił zastosować tę uproszczoną definicję, aby program realizował fragmentaryczne przetwarzanie danych i tym samym spełniał w całości założenia pracy jednocześnie dając lepsze możliwości na jego rozwój w przyszłości. Autor ma nadzieję, że dodanie innego algorytmu do detekcji bazującego na przetwarzaniu sygnału partiami będzie prostsze niż przyszła zmiana samego centrum oprogramowania, czyli odczytywania i zapisywania wyników.

## 2.3 Metoda wykrywania zakłóceń

Na podstawie informacji, o których mowa powyżej autor podjął decyzję aby wykrywać przekroczenia pewnego ustalonego poziomu ciśnienia akustycznego. Ciśnienie akustyczne to jednak określenie zbyt ubogie aby w pełni precyzyjnie opisać funkcjonalność narzędzia.

## 2.4 Istniejące rozwiązania do wykrywania zdarzeń akustycznych



# Rozdział 3

## Metodologia

W tej części zostanie omówiona metodologia przeprowadzonej pracy. Przedstawione zostaną narzędzia użyte do stworzenia programu, oprogramowanie, środowisko programistyczne oraz język programowania.

### 3.1 System kontroli wersji - Git, GitHub, GitKraken

#### 3.1.1 System kontroli wersji

We współczesnym świecie istnieje bardzo wiele języków programowania. Każdy z nich oferuje nieco inne możliwości i funkcjonalności. Niezależnie jednak od wybranego języka, niezwykle istotnym jest system kontroli wersji.

Jednym z najpopularniejszych współcześnie systemów kontroli wersji jest Git. System kontroli wersji oferuje możliwość ciągłego śledzenia zmian w kodzie programu bez potrzeby ręcznego zapisywania wielu wersji pliku.[Software Freedom Conservancy, 2018]. Problem wersjonowania oprogramowania jest znany od dawna i jest jednym z kluczowych zagadnień pracy nad programem, szczególnie kiedy pracuje się w zespole. Gdy kilka osób pracuje nad jednym fragmentem kodu, wydaje się niemożliwe aby współpracować bez systemu kontroli wersji.

Pomimo iż praca dyplomowa inżynierska jest projektem jednoosobowym, system kontroli wersji spełnia swoją rolę i w takim przypadku. Praca z kodem jest nierozłącznie związana z wielokrotnymi zmianami wcześniej wprowadzonych rozwiązań. Czasami wynika to z faktu odkrycia nowych, lepszych rozwiązań. Bywają też przypadki, kiedy w zaawansowanym stadium pracy okaże się, że już na samym początku został popełniony błąd w logice działania i trzeba coś zmienić w jednej funkcji. Często te zmiany wymagają powrotu do momentu pracy sprzed paru dni a nawet tygodni, ponieważ kolejne fragmenty kodu opierały swoje funkcjonowanie na działaniu tych wadliwych elementów. Nie jest możliwym pamiętanie wszystkich tych zmian i płynny powrót do nich poprzez przepisanie kodu. W takiej sytuacji, system kontroli wersji jest niezawodnym narzędziem, które pomaga zorganizować rozwój oprogramowania.

Dodatkowo nie można nie docenić możliwości porównywania wersji roboczej pliku z dowolną wersją tego pliku przechowywaną na serwerze. Daje to możliwość ocenienia wprowadzonych zmian, zastanowienia się czy wszystkie były konieczne oraz odpowiednie przypomnienia sobie od czego zaczynaliśmy. Wszystkie te elementy pozwalają pracować w sposób efektywny i uporządkowany.

Ostatnim ale na pewno nie najmniej ważnym elementem jest możliwość przechowywania całego programu na zdalnym serwerze. W dobie komputeryzacji bardzo często zdarza

się, że pracujemy na kilku różnych urządzeniach nad tym samym projektem. Korzystając z tego udogodnienia nie występuje problem przenoszenia danych pomiędzy urządzeniami. Dodatkowo w razie utraty sprzętu postępy pracy nie zostają stracone.

Powyżej omówione aspekty jednoznacznie wskazują, że pracując nad oprogramowaniem, chcąc robić to w sposób odpowiedzialny i umożliwiający przyszłą współpracę korzystanie z systemu kontroli wersji jest nieodzownym elementem pracy. Dbanie o czystość i przejrzystość wprowadzanych zmian również jest elementem, którym powinna cechować się dobrze wykonana praca.

### 3.1.2 Git

Jak zostało powiedziane, autor zdecydował się na skorzystanie z Git'a jako systemu do kontroli wersji. Jest to bardzo popularny system. Oferuje prostotę obsługi, przejrzystość, dobrą możliwość współpracy w małych grupach. Dodatkowo twórcy Git'a oferują darmowe prywatne miejsca na serwerze do przechowywania danych. Dzięki temu gestowi oraz wymienionych wyżej zaletach autor zdecydował się wybrać właśnie ten system.

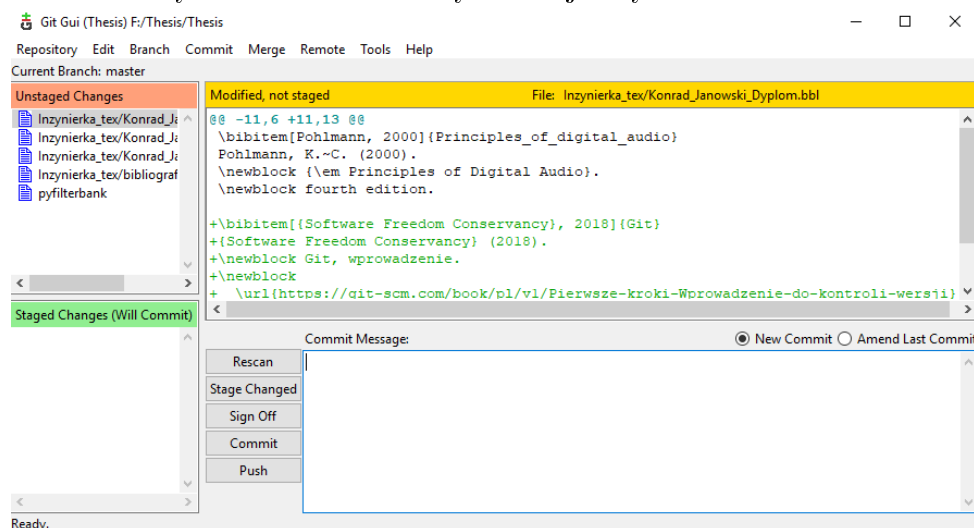
### 3.1.3 GitKraken

Pomimo iż autor zdecydował się tworzyć oprogramowanie w ramach pracy dyplomowej inżynierskiej warto pamiętać o tym, że praca w przeznaczeniu jest skierowana do akustyków. W związku z tym poza efektywnością działania programu ważne jest też jego prostota i przejrzystość aby podczas pracy nad nim oraz przyszłej możliwej rozbudowy można skupiać się na istocie działania a nie odkrywaniu zawitych sposobów obsługi oprogramowania towarzyszącego.

Git w wersji podstawowej jest dostarczany z bardzo ubogim graficznym interfejsem użytkownika. Przykładowe okno zostało zaprezentowane na obrazie (3.1)

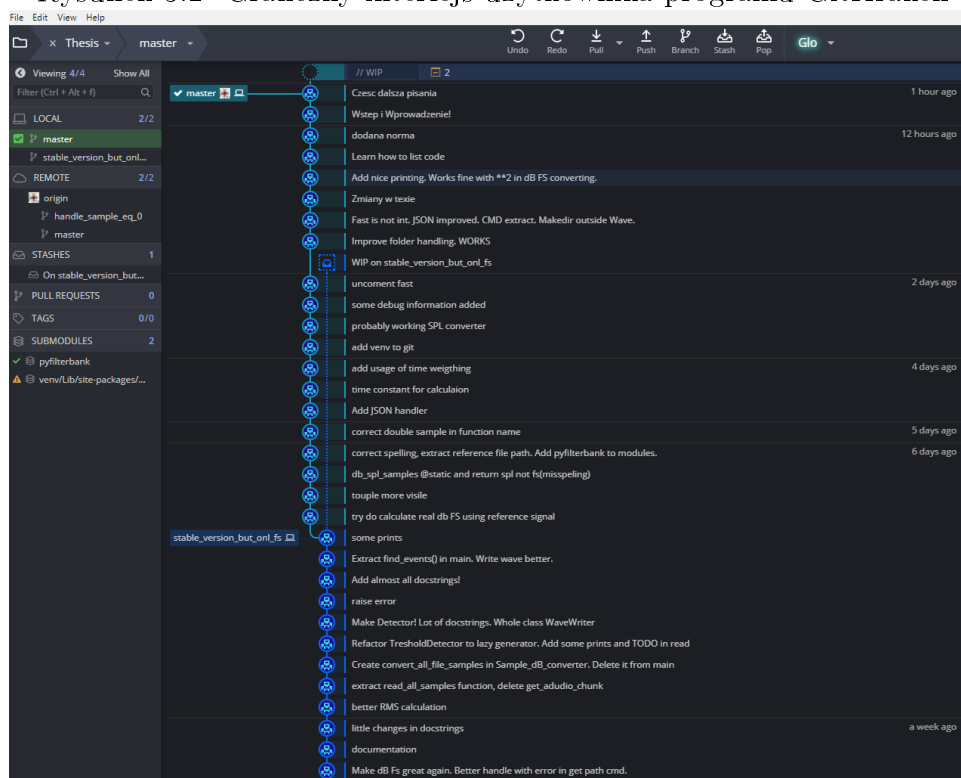
Rysunek 3.1 Graficzny interfejs użytkownika GIT

{git\_gui}



Aby lepiej móc skupić się na logicznej strukturze programu autor zdecydował skorzystać z programu GitKraken, który umożliwia pracę w oparciu o system kontroli wersji Git w bardziej przejrzystej i czytelnej formie. Przykładowe drzewo kolejnych zmian zaprezentowano na obrazie (3.1.3)

Rysunek 3.2 Graficzny interfejs użytkownika programu GitKraken



W opinii autora dobranie odpowiednich narzędzi jest bardzo istotne w procesie tworzenia aplikacji. Wiele twórców oprogramowania, którzy nie są ukierunkowani na programowanie a jedynie używają go jako narzędzia do wykonania innych czynności zaniedbują wyżej wymienione aspekty dbałości o czystość kodu. Dostęp i odpowiednie użycie graficznych narzędzi, które są znacznie wygodniejsze i bardziej intuicyjne niż ich konsolowe lub ubogie odpowiedniki z pewnością może zachęcić twórców do większej dbałości o ten aspekt tworzenia programów

## 3.2 Język programowania - Python

Mając pomysł na działanie programu oraz niezbędne narzędzie do kontrolowania postępów prac należy wybrać język programowania w którym zostanie napisany program. Zdecydowano, że tym językiem będzie Python w wersji 3.7.1. Jest to najnowsza stabilna wersja tego języka[Python Software Foundation, 2018a] w czasie tworzenia oprogramowania.

Wybrano Pythona z kilku powodów. Python jest obecnie jednym z najpopularniejszych języków programowania[Rasmussen College, 2017]. Rozwijanie programu i przyszła współpraca z innymi twórcami może być dzięki temu ułatwiona. Łatwiej znaleźć innych ludzi posługujących się Pythonem niż innymi, mniej popularnymi językami.

Dodatkowo Python jest prosty w składni, intuicyjny i ma nieduży próg wejścia. Nawet osoba, która nie jest doświadczona w programowaniu może stworzyć podstawowe aplikacje. Oczywiście bardziej zaawansowane struktury wymagają wiedzy, umiejętności i doświadczenia ale ta początkowa intuicyjność daje szansę zapoznania się z nim niezawodowym programistom.

Ponadto Python ma rozbudowaną bazę bibliotek i modułów, które można używać ponownie do swoich zastosowań.[Python Software Foundation, 2018b] Daje to ogromne

możliwości tworzenia programów dopasowanych do potrzeb, korzystając z gotowych modułów. Gdyby za każdym razem trzeba było zapisywać podstawowe operacje od nowa, proces kodowania byłby zdecydowanie dłuższy i mógłby zajmować więcej czasu niż osoba zajmująca się inną dziedziną może przeznaczyć na opracowanie narzędzia.

Do tego Python ma doskonale opracowaną dokumentację. Większość modułów jest opisana czytelnie i zrozumiale. Zdecydowanie ułatwia to pracę z tym językiem.

Co ważne, Python jest językiem interpretowanym a nie kompilowanym.[Reitz and schulsser, 2018].

Oznacza to, że do działania potrzebuje interpretera, który działa niezależnie od systemu operacyjnego i jego bibliotek. Co prawda spowalnia to nieco jego działanie w porównaniu to języków kompilowanych takich jak C++ ale daje lepszą możliwość dzielenia się oprogramowaniem. Można skonstruować interpreter tak, by mógł być wysłany razem z kodem programu i osoba odbierająca bez problemu będzie mogła ten kod interpretować. W przypadku języków kompilowanych występują często problemy z udostępnianiem kodu źródłowego właśnie ze względu na ich silną interakcję z systemem operacyjnym. W kontekście przyszłego rozwoju programu dla akustyków jest to ważny aspekt wyboru języka.

Z podanych powodów, Python został wybrany do napisania pracy.

Z wyborem Pythona wiąże się jeszcze jeden wybór, jego wersji. Obecnie najnowszą wersją jest Python 3.7.1 ale równocześnie wspierany jest Python w wersji 2.6.x.[Python Software Foundation] Niestety, zmiana z Pythona wersji drugiej na wersję trzecią wiąże się ze znaczącymi zmianami w funkcjonalności języka i wersje te nie są ze sobą kompatybilne. Wiele modułów aktualnie dostępnych do użytku dalej funkcjonuje w wersji drugiej języka. Można by więc pokusić się o napisanie programu w starszej wersji. Python wersji drugiej przestaje jednak być wspierany wraz z końcem 2020 roku. [Python Software Foundation, 2016]. Oznacza to, że najprawdopodobniej ze względów bezpieczeństwa i braku wsparcia większość nowych modułów będzie pisana w Pythonie w wersji trzeciej a dodatkowo w razie nieprawidłowości w działaniu wersji drugiej, nie będzie można liczyć na żadną pomoc.

### 3.3 Środowisko programistyczne - Pycharm

Pycharm jest środowiskiem dedykowanym do obsługi Pythona. Mając niewielkie doświadczenie z tym środowiskiem autor wybrał je aby je poznać i sprawdzić jego działanie. Każde środowisko oferuje odmienny zestaw funkcjonalności i ułatwień użytkownika.



# Rozdział 4

## Struktura oraz działanie programu

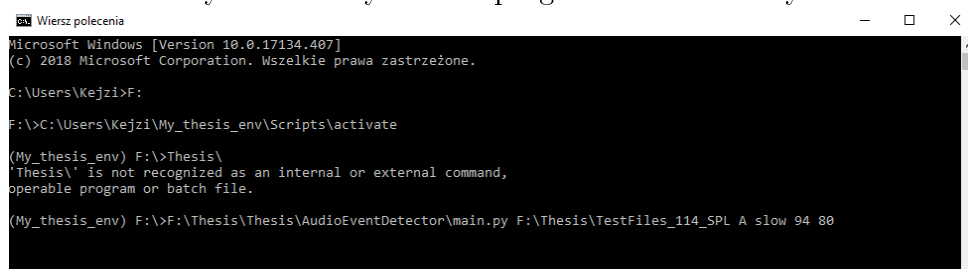
### 4.1 Struktura programu - aplikacja konsolowa

Decydując się na program, należy wybrać w jaki sposób będzie on uruchamiany. Może być jedynie biblioteką, czyli zestawem funkcjonalności, które mogą być wykorzystywane przez inny program. Jest również możliwość skonstruowania rozbudowanego graficznego interfejsu użytkownika. Na ten moment, autor zdecydował aby oprogramowanie działało jako program konsolowy.

Oznacza to, że wywołuje się go z konsoli systemu z odpowiednimi parametrami. Przykładowe wywołanie widzimy na obrazie(4.1)

{wywołanie}

Rysunek 4.1 Przykładowe wywołanie programu w konsoli systemu Windows



```
Wiersz polecenia
Microsoft Windows [Version 10.0.17134.407]
(c) 2018 Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Kejzi>F:
F:\>C:\Users\Kejzi\My_thesis_env\Scripts\activate
(My_thesis_env) F:\>Thesis\
'Thesis\' is not recognized as an internal or external command,
operable program or batch file.
(My_thesis_env) F:\>F:\Thesis\Thesis\AudioEventDetector\main.py F:\Thesis\TestFiles_114_SPL A slow 94 80
```



# Rozdział 5

## Notatki

--



Dodatek A

Kod Programu



# Bibliografia

[PN-ISO-1996-1:2006, ] PN-ISO-1996-1:2006.

[PN-ISO-9612:2014, ] PN-ISO-9612:2014.

[Pohlmann, 2000] Pohlmann, K. C. (2000). *Principles of Digital Audio*. fourth edition.

[Python Software Foundation, 2016] Python Software Foundation (2016). Pep 373 – python 2.7 release schedule. <https://legacy.python.org/dev/peps/pep-0373/>. Accessed: 2010-09-30.

[Python Software Foundation, 2018a] Python Software Foundation (2018a). Python 3.7.1 latest releases. <https://www.python.org/downloads/>. Accessed: 2010-09-30.

[Python Software Foundation, 2018b] Python Software Foundation (2018b). The python package index. <https://pypi.org/>. Accessed: 2010-09-30.

[Rasmussen College, 2017] Rasmussen College (2017). 10 best programming languages based on earnings & opportunities. <https://www.rasmussen.edu/degrees/technology/blog/best-programming-languages-based-on-earnings-and-opportunities/>. Accessed: 2010-09-30.

[Reitz and schulsser, 2018] Reitz, K. and schulsser, T. (2018). *Przewodnik po Pythonie*. Helion.

[Software Freedom Conservancy, 2018] Software Freedom Conservancy (2018). Git, wprowadzenie. <https://git-scm.com/book/pl/v1/Pierwsze-kroki-Wprowadzenie-do-kontroli-wersji>. Accessed: 2010-09-30.