POLITECHNIKA WROCŁAWSKA

WYDZIAŁ ELEKTRONIKI

1. KIERUNEK: Informatyka
2. SPECJALNOŚĆ: Inżynieria internetowa
3. PRACA DYPLOMOWA
4. INŻYNIERSKA

Aplikacja mobilna umożliwiająca modulowanie głosu ludzkiego

Mobile Application for Human Voice Modulation

AUTOR:  
Piotr Sotor

AUTOR:

Imię i nazwisko



PROWADZĄCY PRACĘ:

* + - * 1. dr inż. Jacek Mazurkiewicz, W4/K9

1. OCENA PRACY:
2. WROCŁAW 2015

[1 Wstęp 3](#_Toc437997138)

[2 Cel i zakres pracy 4](#_Toc437997139)

[3 Przegląd istniejących rozwiązań 6](#_Toc437997140)

[4 Opis procesu modulacji dźwięku 8](#_Toc437997141)

[5 Przegląd zagadnień modulacji dźwięku 11](#_Toc437997142)

[5.1 Charakterystyki sygnału mowy 11](#_Toc437997143)

[5.2 Twierdzenie Kotielnikowa-Shannona 13](#_Toc437997144)

[5.3 Funkcje okna 14](#_Toc437997145)

[5.4 Cross-fading 14](#_Toc437997146)

[5.5 Dyskretna transformata Fouriera 16](#_Toc437997147)

[5.5.1 Szybka transformata Fouriera (FFT) 17](#_Toc437997148)

[6 Aplikacja realizująca modulowanie głosu 21](#_Toc437997149)

[6.1 Zdefiniowane modulacje 21](#_Toc437997150)

[6.1.1 Filtr o konfigurowalnej przepustowości pasm częstotliwości 21](#_Toc437997151)

[6.1.2 Skalowanie sygnału w dziedzinie częstotliwości 23](#_Toc437997152)

[6.1.3 Rozmycie charakterystyk częstotliwościowych 25](#_Toc437997153)

[6.2 Zrealizowana funkcja okna 27](#_Toc437997154)

[6.3 Zaimplementowane metody cross-fadingu 29](#_Toc437997155)

[6.3.1 Cross-fading liniowy 30](#_Toc437997156)

[6.3.2 Cross-fading sinusowo – cosinusowy 32](#_Toc437997157)

[6.4 Opis programu 33](#_Toc437997158)

[6.4.1 Założenia projektowe 33](#_Toc437997159)

[6.4.1.1 Założenia funkcjonalne 34](#_Toc437997160)

[6.4.1.2 Założenia niefunkcjonalne 34](#_Toc437997161)

[6.4.2 Biblioteki zewnętrzne 35](#_Toc437997162)

[6.4.3 Analiza wydajnościowa 35](#_Toc437997163)

[6.5 Prezentacja systemu 37](#_Toc437997164)

[6.5.1 Opis interfejsu 37](#_Toc437997165)

[6.5.2 Pliki konfiguracyjne 38](#_Toc437997166)

[7 Wnioski i podsumowanie 41](#_Toc437997167)

[Literatura 42](#_Toc437997168)

[Spis rysunków 43](#_Toc437997169)

[Spis listingów 44](#_Toc437997170)

# Wstęp

Przełom tysiącleci niewątpliwie można opatrzyć wieloma etykietami znaczących z punktu widzenia znanej ludzkości historii punktów zwrotnych lub (r)ewolucji. Wymieniać je można długo – od rewolucji obyczajowej, przez rewolucję komputerową, na rewolucji Internetowej kończąc. Niespotykany dotąd w historii szybki postęp w różnych dziedzinach techniki umożliwił konstrukcję czystych ekologicznie samochodów, spoglądanie w zakątki kosmosu dzięki orbitalnym teleskopom, oraz globalną komunikację z niemalże dowolnego miejsca na Ziemi. Dzięki postępowi i miniaturyzacji elektroniki ludzie dosłownie otaczają się komputerami, które codziennie noszą w kieszeniach swoich spodni lub czeluściach torebki czy to w formie kart płatniczych czy telefonów komórkowych. W ciągu zaledwie dwóch dekad urządzenie, które jeszcze w latach dziewięćdziesiątych XX wieku było gadżetem dla biznesmenów, stało się już niemal przedmiotem pierwszej potrzeby. Ludzie wykształcili   
w sobie wewnętrzny przymus pozostawania w ciągłym kontakcie między sobą, a sama komunikacja przybiera coraz to wytworniejsze formy, choć nie można powiedzieć by starsze były wypierane przez nowe. Na ogół nowe formy uzupełniają przestrzenie jeszcze nie zagospodarowane przez poprzedników. Można więc powiedzieć, że z całą pewnością podstawowa funkcja telefonu komórkowego – połączenie telefoniczne – będzie integralną częścią globalnej komunikacji jeszcze przez dłuższy czas.

Wraz z rozpowszechnieniem telefonii komórkowej wiele firm i organizacji zdecydowało się przenieść część niezbędnych interakcji z klientem z kontaktu twarzą w twarz do rozmowy telefonicznej. Nie jest to regułą, ale wiele z nich zastrzega sobie prawo do nagrywania prowadzonych połączeń. Wszystko po to, by w przypadku wątpliwości ze strony klienta firma posiadała niezbity dowód na zawarcie z pracownikiem umowy słownej, która jest równie wiążąca co umowa pisemna.

Dla nikogo nie będzie zaskoczeniem stwierdzenie, że jednym z najbardziej wartościowych surowców w obecnych czasach jest informacja. *Informacja* jest jednak terminem bardzo ogólnym. Jest nią tak imię, nazwisko, jak i lokalizacja najczęściej odwiedzanych sklepów, czy dane cechy personalne takie jak skan siatkówki, odcisk palca lub dokładna charakterystyka mowy, nawyki wymowy. Zwłaszcza te ostatnie mogą stać się przedmiotem nielegalnego wykorzystania i zmanipulowania, biorąc pod uwagę częste przechowywanie nagrań firm bez wykorzystania odpowiednich zabezpieczeń.

Nasuwa się więc pytanie. Czy jedyną reakcją na informację o prawdopodobnym nagrywaniu rozmowy musi być wybór pomiędzy natychmiastowym zakończeniem połączenia,   
a zaryzykowaniem, że nic złego się z tym nagraniem w przyszłości nie stanie? A co, jeśli zostanie dodana do dylematu również trzecia możliwość: zniekształcenie własnego głosu   
w taki sposób, by był zrozumiały, ale utrudniał jednoznaczne zidentyfikowanie rozmówcy?

Poniższa praca jest dopiero pierwszym krokiem ku zaoferowaniu użytkownikowi telefonu komórkowego świetlanej trzeciej opcji wyboru. Przedstawia efekty różnych zabiegów modulujących dźwięk mowy, oraz oszacowuje zdolność współczesnego sprzętu do wykonania złożonej modulacji na dźwięku wysokiej jakości w czasie rzeczywistym, który jest niezbędny z punktu widzenia prowadzenia rozmowy telefonicznej.

# Ce­­­­l i za­­­kres pracy

Celem niniejszej pracy jest opracowanie trzech modulacji stosowanych na sygnale ludzkiej mowy i ich implementacja w aplikacji mobilne na system Android. Wybór platformy docelowej zdeterminowany został dostępnymi urządzeniami testowymi oraz posiadaną przez autora podstawową wiedzą na jego temat. Schemat użycia powstałego programu opiera się na następujących krokach: nagraniu dźwięku, zastosowaniu modulacji i odtworzeniu wyniku. Dodatkowym aspektem ważnym z punktu widzenia tematu pracy było sprawdzenie użyteczności współczesnych urządzeń pod kątem programowego przetwarzania dźwięku   
w czasie rzeczywistym.

W skład założeń projektowych weszła implementacja trzech modulacji. Pierwsza z nich udostępnia podobne możliwości do filtrów dolno- i górnoprzepustowe znanych   
z zaawansowanych zastosowań teorii przetwarzania sygnałów. Skuteczne korzystanie z niej wymaga przynajmniej podstawowej wiedzy teoretycznej dotyczącej budowy   
i charakterystyki dźwięku poddanego przetworzeniu. Druga zastosowana metoda zmiany dźwięku pozwala na jego skalowanie w dziedzinie częstotliwości. Pozwala to na podnoszenie lub obniżanie tonu głosu o założony interwał. Ostatnim zaimplementowanym mechanizmem jest uśrednianie widma sygnału wejściowego. Należy traktować go jako eksperyment, ponieważ nie znaleziono w literaturze podobnej modulacji.

Zastosowanie w przyszłości opracowanych algorytmów do przetwarzania sygnału w czasie rzeczywistym wymusza maksymalną optymalizację działania. Nie zdecydowano się jednak do skorzystania z równoległych implementacji poszczególnych kroków z powodu charakterystyki wykonywanych operacji. Algorytmy nie wymagają wielkiej ilości obliczeń, jednakże problemem jest szybkość spływania nowych danych do przetworzenia.

Powstała aplikacja umożliwia użytkownikowi dostęp do konfiguracji zarówno parametrów nagrywanego dźwięku jak i stosowanej modulacji. W ramach opcji nagrania pozwala na wybór częstotliwości próbkowania rejestrowanego sygnału oraz liczby kanałów. Wśród cech dotyczących przetworzenia sygnału znajdują się oprócz wyboru modulacji także specyfikacja jej współczynników oraz rodzaj cross-fadingu wykorzystywanego w czasie obliczeń.

Poniższa część pracy składa się z pięciu głównych rozdziałów. Rozdział 3 stanowi przegląd istniejących aplikacji związanych z przetwarzaniem dźwięku. Opisano w nim jak można je podzielić ze względu na zastosowanie oraz jakie podobne mechanizmy znalazły zastosowanie w tej pracy. Kolejny rozdział stanowi pobieżny opis operacji wykonywanych w celu zmodulowania dźwięku. Nakreśla kolejność wykonywanych kroków, tłumacząc przy tym ich znaczenie dla pozostałych. Rozdział 5 szczegółowiej przedstawia wcześniej wymienione kroki. Stanowi podstawę teoretyczną dla późniejszej części implementacyjnej, na którą składa się rozdział 6. W tym rozdziale kolejno przedstawiono sposób działania zastosowanych algorytmów modulacji. Dla każdej przedstawiono szczegóły programistyczne oraz zaprezentowano efekt działania na przykładowym sygnale wejściowym. Oprócz samego przetwarzania widma zaimplementowano ponadto mechanizmy okienkowania sygnału   
i cross-fadingu, które również znalazło swoje miejsce w tym rozdziale. Końcową jego część stanowi już opis samej aplikacji od strony użytkownika oraz obszerna analiza wydajnościowa. Ostatnią częścią pracy jest przedstawienie wniosków oraz kilka słów podsumowania.

# Przegląd istniejących rozwiązań

Najpopularniejszym źródłem aplikacji dedykowanych systemowi Android jest sklep internetowy Google Play. Duża liczba dostępnego na nim oprogramowania jest jedną   
z przyczyn popularności tej platformy. Z racji powszechności i oficjalnego wsparcia przegląd istniejących rozwiązań został oparty o zbiór dostępny właśnie na Google Play.

Dostępne aplikacje umożlwiające modulowanie głosu można podzielić na poniżej opisane dwie kategorie. Pierwszą z nich można określić ogólną nazwą odtwarzaczy multimediów. Oferta sklepu zawiera wiele proponowanych aplikacji służących odtwarzaniu muzyki zapisanej z różnych formatach i jakościach. Oprócz odsłuchiwania muzyki i organizowania plików w listy odtwarzania część bardziej zaawansowanych umożliwia modulacje dźwięku. Najczęstszym stosowanym zabiegiem jest zastosowanie korektora. Jest niczym innym niż filtrem tłumiącym lub wzmacniającym określone pasma widma odtwarzanego dźwięku. Uzyskanie podobnego efektu jest jednym z celów poniższej pracy.

Odtwarzanie szeregu plików dźwiękowych wymusza wystąpienie sytuacji rozpoczęcia następnego nagrania w momencie zakończenia poprzedniego. W niektórych aplikacja umożliwia się ustawienie płynnego przechodzenia pomiędzy utworami poprzez stopniowe wyciszanie kończącego się i zgłośnienie kolejnego. Zabieg ten nazywany jest cross-fadingiem i został również wykorzystany w poniższej pracy.

Poniżej przedstawiono przykładową aplikację oferującą powyżej opisane funkcjonalności.

**Neutron Music Player**

Oferowane funkcjonalności:

* obsługa plików w formatach: MP3, WMA, 3GP, FLAC, WAVE i inne,
* organizacji plików w listy utworów, płynne przejście pomiędzy utworami
* graficzny, konfigurowalny korektor
* filtry szumów, hałasu statycznego
* crossfeed poprawiający jakość dźwięku stereo.

W przypadku odtwarzaczy zabiegi modulacyjne dźwięku mają na celu jedynie wzmocnienie wrażeń słuchowych użytkownika, ale nie są główną funkcjonalnością aplikacji. Programy sklasyfikowane tutaj jako rozrywkowe stawiają modulacje dźwięku jako główną funkcjonalność.

Najczęściej przedstawiają taki sam scenariusz użycia. Umożliwiają nagranie dźwięku lub wybranie pliku z pamięci trwałej, nałożenie wybranego efektu modulującego na nagranie   
i odtworzenie nagrania. Powyższy scenariusz został wykorzystany w poniższej pracy. Część aplikacji tej kategorii ponadto integrują w sobie funkcjonalności klienta sieci VoIP (*Voice over Internet Protocol*) co pozwala na prowadzenie rozmów poprzez łącze internetowe wraz   
z wykonaniem modulacji głosu rozmówcy. Nawiązanie połączenia z innym użytkownikiem często wiąże się z dodatkowymi kosztami opłacenia serwera sieci VoIP określonego przez dostawcę aplikacji. Z reguły nie umożliwiają łączenia się z poprzez inne serwery niż zdefiniowane przez producenta. Poniżej przedstawiono przykładową produkt dostępny   
w serwisie Google Play oferująca opisane możliwości.

**Voice changer with effects**

Oferowane funkcjonalności:

* nagrywanie dźwięku,
* odtwarzanie z nałożonym efektem,
* dostępne modulacje:
  + głos cyborga,
  + głos wiewiórki,
  + odtwarzanie wsteczne,
* możliwość ustawienia zmienionego nagrania jako dzwonka telefonu,
* synteza mowy na podstawie tekstu.

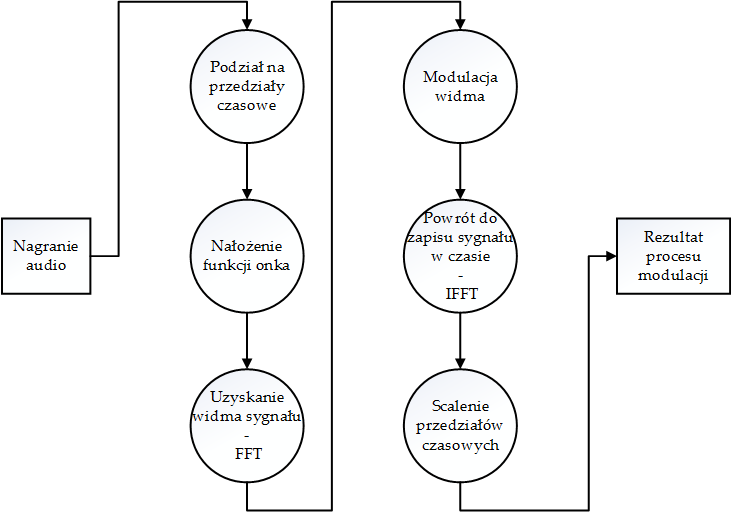
# Opis procesu modulacji dźwięku

Modulacja dźwięku jest niczym innym niż procesem przetwarzania pewnego sygnału. Z tego powodu wykorzystuje się do tego wiele metod powszechnie wykorzystywanych w bardzo różnych dziedzinach technicznych jak na przykład telekomunikacja lub sejsmografii.   
Z punktu widzenia wykorzystywanych algorytmów nie ma różnicy czy przetwarzane są próbki przebiegu fali akustycznej czy sejsmicznej.

W celu przeprowadzenia modulacji dźwięku niezbędne jest uzyskanie zapisu dźwięku   
w postaci fali. Zrealizowane w ten sposób nagranie przedstawia rzeczywisty sygnał ciągły   
w postaci listy wartości wynikającej z procesu próbkowania sygnału i kwantyzacji próbek. Parametrem próbkowania sygnału jest częstotliwość próbkowania, która wyznacza liczbę próbek przypadającą na sekundę nagrania, a parametrem kwantyzacji jest jej rozdzielczość, czyli dokładność odwzorowania realnej wartości na dyskretną rzeczywistość maszyny. Różnica pomiędzy wynikiem kwantyzacji, a rzeczywistą wartością sygnału w danej próbce nosi nazwę *błędu kwantyzacji*. W poniższej pracy umożliwia się nagrywanie i odtwarzanie dźwięku z częstotliwościami próbkowania 8000 Hz, 16000 Hz i 44100 Hz, oraz stosuje się zapis próbek z rozdzielczością 16 bitową, co oznacza, że próbka reprezentowana jest jedną z wartości.

Praca nad sygnałami rzeczywistymi, a nie syntetycznie uzyskanymi w podejściu teoretycznym, wiąże się z koniecznością wzięcia pod uwagę kilku kwestii. Pierwszą rzeczą jest zawsze występujący szum lub zniekształcenia. Można zakwalifikować do nich zarówno odgłosy otoczenia jak i niedoskonałość urządzenia rejestrującego. Rozróżnić je można pod kątem deterministyczności występowania oraz statyczności przebiegu. Tak więc szumy urządzenia, który często ma charakter *szumu białego* można postrzegać jako zakłócenie deterministyczne (ponieważ występuje ciągle) i statyczne (ponieważ nie zmienia się w czasie). Dokładnym przeciwieństwem będzie odgłos przejeżdżającego samochodu lub osoby rozmawiające w tle.

Kolejną cechą sygnałów rzeczywistych, a w szczególności fal akustycznych, jest ich niedeterministyczność, zmienność w czasie i nieokresowość. Zastosowanie transformaty Fouriera na takim sygnale daje w rezultacie wynik nie mający z rzeczywistym sygnałem wiele wspólnego. Dzieje się tak z powodu własności samej transformaty, która zakłada okresowość sygnału. Ponieważ wykorzystanie transformaty Fouriera jest kluczowym elementem   
w zakresie przetwarzania sygnałów niezbędne jest wykonanie pewnych operacji, by rezultaty transformaty były możliwie wiarygodne.

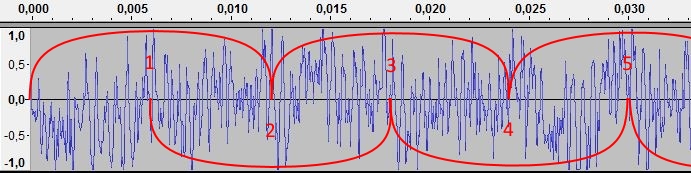


Rysunek 1 Diagram przepływu danych w procesie modulacji sygnału

Rysunek 1 prezentuje poglądowy diagram przypływu danych w procesie modulacji. Pierwszym krokiem jest podział długiego sygnału na krótkie fragmenty, dalej nazywane *oknami czasowymi*. Prowadzi to do uzyskania stosunkowo niezmiennego przebiegu sygnału, który można już uznać za sygnał stacjonarny. W przypadku przetwarzania dźwięku przyjmuje się za optymalną długość okna ciąg próbek reprezentujący nagranie o długości *10 – 20 ms*. Dokładna długość zależy od użytej częstotliwości próbkowania sygnału, ponieważ dąży się do tego, by wybrana liczba próbek miała postać , tj. była potęgą liczby 2. Jest to warunek konieczny by w przetwarzaniu wykorzystać szybką transformatę Fouriera. Dokładna liczba próbek może zostać wyliczona z wzoru:

(4.1)

W przypadku zapisu kilku kanałów przy użyciu jednego pliku należy w tym momencie zadbać o rozdzielenie poszczególnych kanałów. Próbki sygnału dla różnych kanałów zapisywane są naprzemiennie po sobie, a przetwarzanie konkretnych kanałów powinno zostać wykonane niezależnie.



Rysunek 2 Podział sygnału na okna czasowe. fs = 44,1kHz, n = 512, czas okna = 11,6 ms

Podział sygnału na okna czasowe rozwiązuje problem stacjonarności sygnału. Pozorną okresowość sygnału uzyskuje się poprzez zastosowanie *funkcji okna.* W rezultacie jej zastosowania przebieg początku sygnału upodabnia się do przebiegu końca sygnału. Przebieg *funkcji okna* wymaga zadbania o to, by kolejne *okna czasowe* nakładały się na siebie.   
W przeciwnym razie odwzorowanie sygnału w modulacji byłoby przekłamane.

Po wykonaniu *funkcji okna* na fragmencie sygnału należy wykorzystać transformatę Fouriera. W jej wyniku następuje przejście z postrzegania sygnału w dziedzinie czasu na dziedzinę częstotliwości. W rezultacie otrzymuje się informację o amplitudzie danej częstotliwości składowej oraz o fazie danej fali. W kontekście przetwarzania dźwięku nie wykorzystuje się w żadnym stopniu przesunięcia w fazie poszczególnych częstotliwości składowych. Przetwarzanie sygnału w dziedzinie częstotliwości jest kluczowe dla modulacji.

Przeprowadzenie modulacji w znacznym stopniu ogranicza się do odpowiedniej zmiany wartości widma sygnału uzyskanego dzięki transformacie. W niniejszej pracy zaproponowano 3 modulacje: skalowanie częstotliwości, filtr przepustowy oraz rozmycie charakterystyk widma.

Po przetworzeniu widma danego *okna czasowego* należy zastosować na nim odwrotną transformatę Fouriera. Jest to zabieg odwrotny do transformaty Fouriera, a więc na podstawie zapisu sygnału w formie widma uzyskuje przebieg sygnału w dziedzinie czasu.

Ostatnim krokiem całego procesu jest połączenie nakładających się fragmentów *okien czasowych* aby uzyskać z powrotem pełny przebieg sygnału już zmienionego. W tym celu wykorzystany został mechanizm cross-fadingu. Polega on na stopniowym wyciszaniu wcześniejszego *okna czasowego* i zgłośnieniu kolejnego.

# Przegląd zagadnień modulacji dźwięku

Poniższy rozdział stanowi podstawę teoretyczną dla dalszych opisów części praktycznej pracy. W znacznej części jedynie dotyka rozleglejszych zagadnień danego tematu. Przedstawia jednak wystarczającą wiedzę, by w pełni zrozumieć zasadność i konieczność poszczególnych operacji wchodzących w skład modulacji dźwięku. Pokrótce zostaną opisane podstawowe zagadnienia dziedziny przetwarzania sygnałów oraz specyfiki fali dźwiękowej mowy.

## Charakterystyki sygnału mowy

Przed pojęciem próby zniekształcenia nagrania w celu utrudnienia identyfikacji rozmówcy warto zrozumieć mechanizmy towarzyszące procesowi artykułowania słów i ich percepcji. Najważniejszym faktem, który należy przyjąć jest postrzeganie dźwięków nie w kontekście czasu, ale częstotliwości. Jest to bezpośrednią konsekwencją zjawisk zachodzących   
w organach mowy i słuchu człowieka.

Pierwszym elementem układu mowy mającym wpływ na głos są fałdy błony śluzowej krtani często nazywane strunami głosowymi. Odpowiadają one za wydobywanie dźwięcznych części głosek. Należy pamiętać, że zjawisko powstawania dźwięku jest procesem biernym   
z punktu widzenia ułożenia elementów krtani. Jest niczym innym niż efektem turbulentnego przepływu powietrza - brak tam aktywnych rezonatorów generujących dźwięk w podobny sposób jak głośniki. Kształt strun głosowych w znacznym stopniu determinuje jednak widmo wydobywanego dźwięku. Układ nerwowy posiada jednak kontrolę nad powierzchniowymi   
i głębokimi mięśniami krtani co pozwala na zmianę rozwarcia i długości szpary głośni oraz korekty napięci i grubości fałd błony śluzowej. Właśnie te operacje pozwalają na intonowanie wypowiedzi [12]. Ostateczny obraz widma definiowany jest przez układ rezonujący w skład którego wchodzą policzki, wargi i język, a w przypadku głosek nosowych jama nosowa   
i nozdrza.

Ton krtaniowy, nazywany czasem tonem podstawowym, posiada bardzo bogate widmo. Kluczowe dla identyfikacji treści i rozmówcy jest pasmo niższych częstotliwości. Dzięki temu już w początkach telekomunikacji wykorzystanie stosunkowo niskiej częstotliwości próbkowania pozwalało na swobodną i zrozumiałą komunikację. Warto jednak prześledzić wpływ filtrów przepustowych na zrozumiałość semantyczną mowy. Rysunek 3 przedstawia zmianę klarowności przekazu zależnie od granicy przepustowości filtrów. Należy jednak zdawać sobie również sprawę z tego, że wysokość tonu podstawowego jest różna dla każdej osoby. Z reguły kobiety posiadają wyższy ton od mężczyzn. Dla przykładu można podać uśrednione zakresy częstotliwości tonu krtaniowego dla głosów śpiewaczych:

* bas: 80 – 320 Hz,
* baryton: 100 – 400 Hz,
* tenor: 120 - 480 Hz.

Rysunek 3 Wpływ filtracji górno- i dolnoprzepustowej pasma mowy ludzkiej na zrozumiałość [8]

Mowa człowieka są niczym innym niż odpowiednio modulowanym tonem krtaniowym. Gdzieś tkwić musi jednak różnica pomiędzy poszczególnymi dźwiękami. W zrozumieniu tych różnic pomocne będą pojęcia fonemu i formantu. Fonem jest najmniejszą rozpoznawalną w danym języku jednostką fonetyczną. Zarejestrowanie zlepka fonemów w określonej kolejności pozwala na semantyczną interpretację usłyszanego słowa. Jednak dopiero formant dotyka dźwięku na poziomie jego widma. Określa on jakie składowe częstotliwościowe charakteryzują określony fenom. Formant, a właściwie zbiór formantów, pozwala na poprawną interpretację dźwięków, co może zostać wykorzystane w mechanizmach rejestrowania języka naturalnego. Każdy fonem posiada określone częstotliwości swoich formantów. Przykładowe wartości przedstawiono w Tabela 1. Można ponadto wyróżnić różne charakterystyczne formanty na całych grup głosek, np. dla spółgłosek nosowych formant o częstotliwości 2500 Hz [4].

Tabela 1 Wartości pierwszych dwóch formantów wybranych fonemów [4]

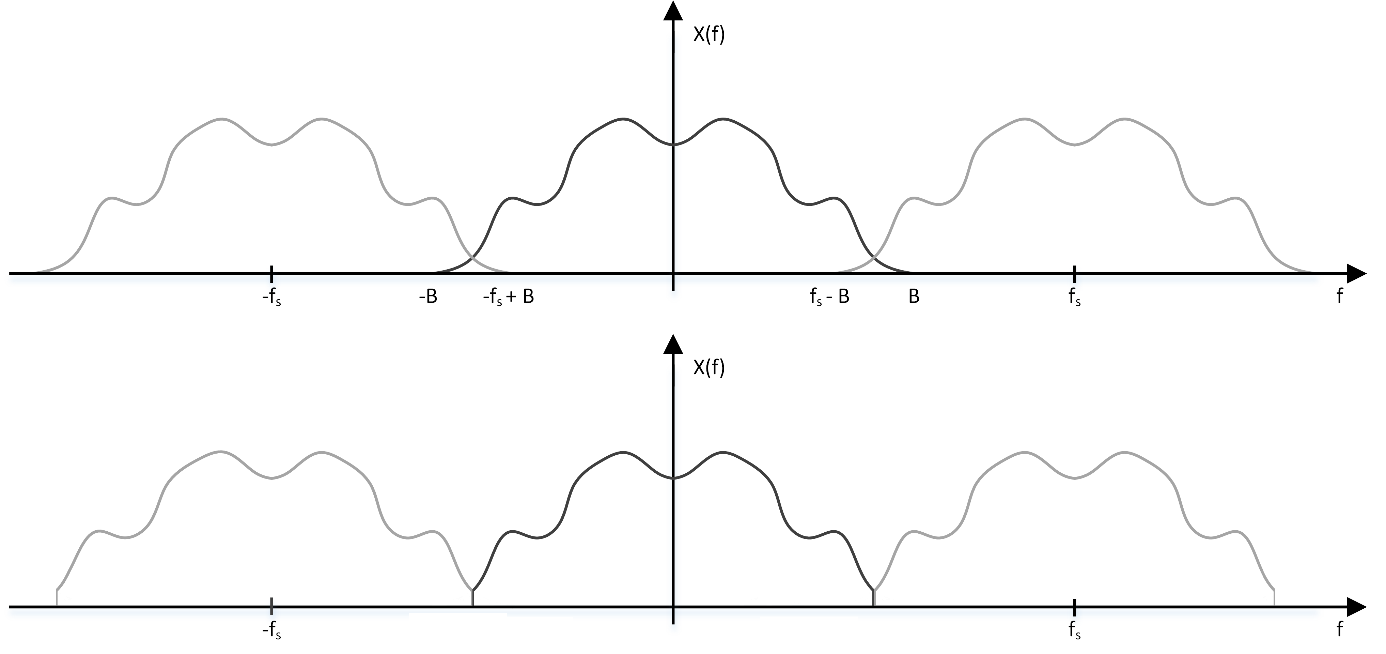
|  |  |  |  |
| --- | --- | --- | --- |
| Fonem | Formant | Formant | Różnica |
| /ʌ/ | 600 | 1170 | 570 |
| /e/ | 390 | 2300 | 1910 |
| /i:/ | 240 | 2400 | 2160 |
| /u:/ | 250 | 595 | 345 |

Znajomość rozkładu formantów w mowie pozwala przewidzieć skutki zastosowania filtrów przepustowych. Oderwanie jednego z pary formantów skutecznie utrudnia rozpoznanie głoski, a w konsekwencji całego słowa.

## Twierdzenie Kotielnikowa-Shannona

Inaczej nazywane twierdzeniem o próbkowaniu. Jest jednym z podstawowych zagadnień dotyczących przetwarzania sygnałów. Wchodzi w skład twierdzeń nazywanych ogólnie *teorią próbkowania*, które opisują metody i procesy transkrypcji sygnałów rzeczywistych (ciągłych, analogowych) na sygnały cyfrowe (dyskretne sygnały czasu dyskretnego). Twierdzenie zostało opublikowane ramach dokumentu *Communication in the Presence of Noise* w roku 1949 [9]. Było to w czasach szerokiego zainteresowania świata naukowego badaniami nad teorią sygnałów. W wyniku tego pojawiały się prace o podobnym wydźwięku przed i po publikacji przez Shannona, stąd twierdzenie bywa nazywane również twiedzeniem *Nyquista–Shannona–Kotelnikowa*, *Whittakera–Shannona–Kotelnikowa* czy *Whittakera–Nyquista–Kotelnikowa–Shannona.*

W swojej pracy Shannon przedstawił wpływ wyboru częstotliwości próbkowania sygnału na dokładność odwzorowania sygnału. Kluczowe w tej kwestii jest postrzeganie sygnału   
w dziedzinie częstotliwości, czyli jego widma. Shannon przedstawia zjawisko *aliasingu* wywołanego przez zbyt niską częstotliwość próbkowania. Zjawisko to jest wynikiem pozornego powielenia widma realnego w przesunięciu o częstotliwość próbkowania   
w wydmowym zapisie sygnału, co zostało przedstawione poglądowo na Rysunek 4. oznacza częstotliwość graniczną widma próbkowanego sygnału. Łatwo zauważyć, że częstotliwości  
 nie posiadają prawidłowej reprezentacji w widmie [12].



Rysunek 4 Zjawisko aliasingu przy złym doborze częstotliwości próbkowania

Tezę twierdzenia można przedstawić następująco: *Sygnał ciągły może być ponownie odtworzony z sygnału dyskretnego, jeśli był próbkowany z częstotliwością co najmniej dwa razy większą od granicznej częstotliwości swego widma.* (Asimo.pl, 2015). Dowód twierdzenia nie jest istotny dla tematu pracy dlatego nie zostanie tutaj przedstawiony. Sprowadza się to do zapisania wzoru (5.1) [1].

(5.1)

Konsekwencją tego twierdzenia dla niniejszej pracy jest konieczność doboru różnych okresów próbkowania w zależności od rejestrowanego sygnału. Ponadto ogranicza od góry spodziewane zakres widma otrzymywany w wyniku stosowania transformaty Fouriera, co ułatwia interpretację jej wyników.

## Funkcje okna

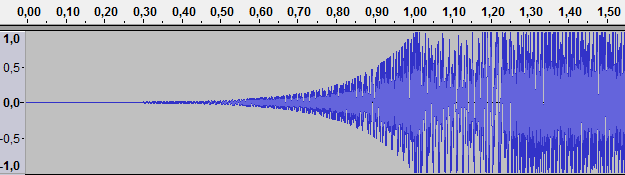
Transformata Fouriera (opisana w rozdziale 5.5) jest narzędziem często wykorzystywanym   
w przetwarzaniu sygnałów. Użycie jej skutkuje dokładnymi rezultatami jednak tylko, gdy spełnione zostaną wymogi dotyczące okresowości sygnału. Żadna ścieżka dźwiękowa, a zapis mowy tym bardziej, nigdy nie cechuje się jednak pełną cyklicznością. Zastosowanie transformaty na nieokresowym sygnale powoduje powstanie w rezultacie *wycieku widma*. Dzieje się tak z powodu nieciągłości końca i początku sygnału. Interpretowane jest to jako gwałtowny skok sygnału co w transformacie obserwowane jest poprzez składowe wysokich częstotliwości.

Teoria przetwarzania sygnałów opracowała jednak metodę pozwalającą wykorzystanie popularnego i szybkiego algorytmu na sygnałach rzeczywistych. Wynikiem nałożenia okna czasowego na zapis pełnego przebiegu nagrania są 10 - 20 milisekundowe porcje danych. Mechanizmem wykorzystywanym w celu ograniczenia wystąpienia wycieku widma jest okienkowanie sygnału. Można przedstawić go jako operację mnożenia przez siebie próbki sygnału oraz odpowiadającą mu wartość funkcji okna. Matematycznie przedstawia to wzór (5.2)

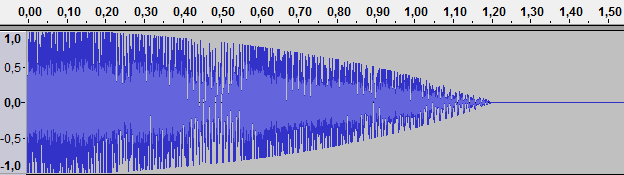
(5.2)

## Cross-fading

Pierwszym krokiem do wyjaśnienia mechaniki działania cross-fadingu jest zrozumienie mechanizmu fadingu. Polega on na stopniowej zmianie głośności próbek dźwięku na początku lub końcu ścieżki nagrania. Stosując na początku najczęściej ma charakter wzrastający (ang. *fade in*), tj. począwszy od znacznego wyciszenia, aż do nominalnej wartości próbek. W przypadku zastosowania na końcu fading przedstawia się analogicznie odwrotnie – opadająco (ang. *fade out*) [3]. W zależności od preferencji słuchowych i przeznaczenia wybiera się odpowiedni przebieg krzywej, według której przebiega zmiana natężenia dźwięku nagrania. Rysunek 5 przedstawia efekt zastosowania wzrastającego przebiegu eksponenciajnym, a Rysunek 6 przedstawia wynik zastosowania cieniowania opadającego   
o charakterze logarytmicznym.

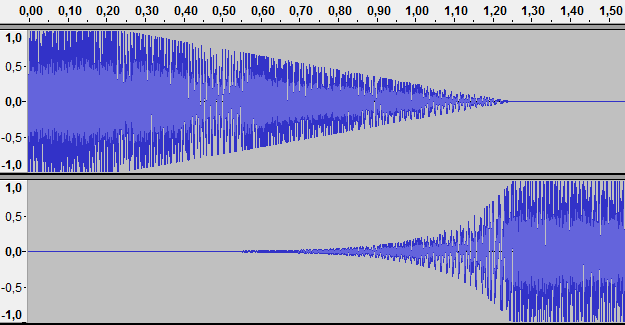


Rysunek 5 Sygnał z eksponencjalnym efektem fade in



Rysunek 6 Sygnał z logarytmicznym efektem fade out

Rozumiejąc działanie fadingu łatwo pojąć czym jest cross-fading. Jest niczym innym niż zastosowaniem dwóch przeciwnie zorientowanych cieniowań na dwóch ścieżkach w tym samym oknie czasowym. Ideałem jest takie dobranie przebiegów wzrastających i opadających, by sumaryczne natężenie dźwięku sygnału nie ulegało znaczącym wahaniom. Nie jest to jednak obowiązkowe i zależy tylko od zamiarów osoby projektującej cross-fading. Rysunek 7 przedstawia przykład zastosowanie różnych fadingów – fade out liniowy, fade in eksponencjalny.



Rysunek 7 Asymetryczny cross-fade dwóch ścieżek dźwiękowych

## Dyskretna transformata Fouriera

Matematyka jako królowa nauk dostarcza niezbędnych narzędzi wykorzystywanych   
w praktycznych problemach. Nie inaczej jest w przypadku transformaty Fouriera. Została opracowana przez francuskiego matematyka o takim samym nazwisku w roku 1807. Początkowo znajdowała zastosowanie w równaniach przewodnictwa cieplnego metali.   
W krótkim czasie doprowadziła jednak do przewrotu w matematyce i powstania nowych teorii. Rezultaty francuskiego naukowca zostały z czasem zaadaptowane w dziedzinach kompresji danych (mp3, jpg), teorii drgań czy właśnie przetwarzaniu sygnałów.

Działanie transformaty polega na rozłożeniu analizowanego sygnału do postaci szeregu Fouriera. Sam szereg opisany może zostać wzorem (5.3), a transformata wzorem (5.4) [5]. Dokładność takiej aproksymacji sygnału zależy od spełnienia warunków stosowania transformaty związanych z okresowością źródła oraz reprezentacji pełnych okresów.

(5.3)

(5.4)

Celem tej pracy nie jest matematyczne opracowanie mechanizmu obliczenia przekształcenia dlatego nie zostanie przeprowadzona dogłębna analiza jej działania oraz podstawy matematyczne jej prawdziwości. Kluczową własnością transformaty jest zmiana dziedziny postrzegania sygnału. Wejściowy zapis jako funkcji czasu daje w rezultacie funkcję   
w dziedzinie częstotliwości. Wartości wynikowe funkcji są liczbami zespolonymi. Część rzeczywista reprezentuje amplitudę składowej częstotliwości, a część urojona przesunięcie   
w fazie. Z punktu widzenia przetwarzania dźwięku faza sygnału jest nieistotna, dlatego   
w dalszy opis opierać się będzie tylko o część rzeczywistą wyniku [12]. Zastosowanie transformaty znacznie upraszcza powszechne operacje takie jak przesunięcie widma lub splot funkcji [5].

Twórca opisywanej transformaty zmarł na długo przed odkryciem szerokiego zastosowania sygnałów dyskretnych. Niezbędne zatem było dostosowanie użytecznego mechanizmu znanego z rzeczywistości ciągłej na dyskretną. Owocem prac nad tym zagadnieniem jest cała rodzina przekształceń Fouriera. W zastosowaniach przetwarzania sygnałów najważniejszym jest dyskretna transformata Fouriera. Główna różnica, jeśli nie jedyna, pomiędzy ciągłą   
i dyskretną transformatą jest zastosowanie operatora sumy w miejscu całki. Zatem wzór (5.4) zamieniony został na wzór (5.5)[10] Uzyskane w ten sposób widmo sygnału może zostać następnie odpowiednio zmienione.

(5.5)

Opisany powyżej mechanizm przekształcenia nazywany jest prostą transformatą Fouriera. Przeciwnym zabiegiem jest odwrotna transformata Fouriera. Pozwala na uzyskanie przebiegu sygnału w dziedzinie czasu na podstawie jego widma [10]. Jest to więc niezbędna operacja dla procesu modulacji. Opisana może zostać wzorem (5.6) w przypadku ciągłej wersji lub wzorem (5.7) dla dyskretnego odpowiednika.

(5.6)

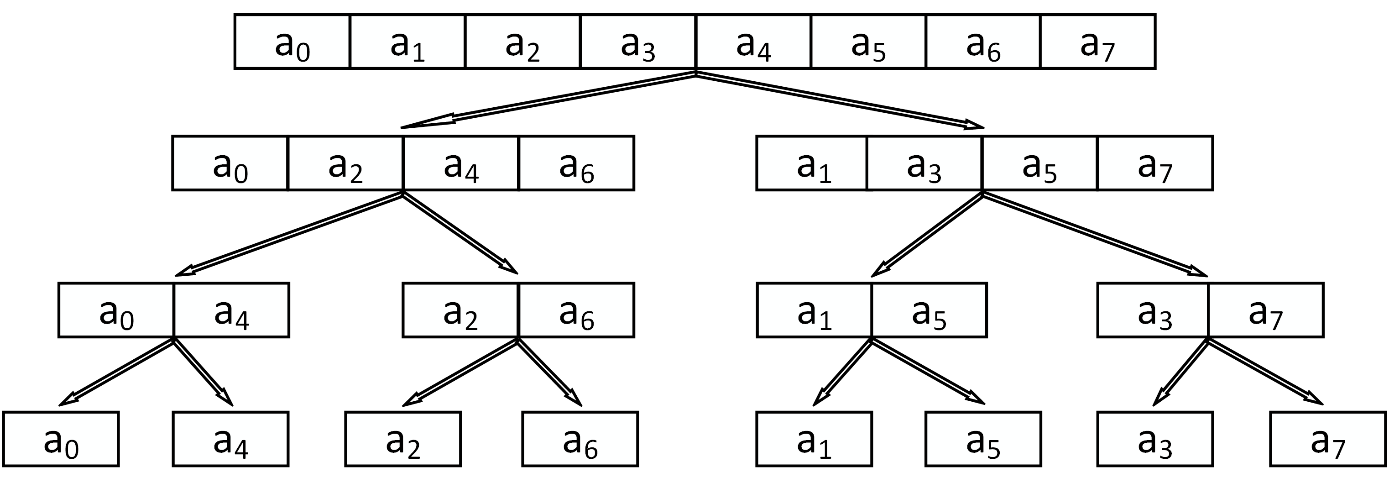
(5.7)

Złożoność obliczeniowa powyższych algorytmów wynosi .

### Szybka transformata Fouriera (FFT)

Szerokie zastosowanie dyskretnej transformaty Fouriera wymusiło optymalizację jej obliczania. Większość szybkich algorytmów obliczania przekształcenia opiera swoje działanie na lemacie Danielsona-Lanczosa [5]. Tożsamość ta mówi o tym, że każdą dyskretną transformatę Fouriera o parzystej liczbie próbek wejściowych N można zastąpić dwoma   
o długości N/2. W przypadku zadania w wejściu sygnału o długości pozwala to na zredukowanie jednego sygnału długości N do N sygnałów o długości 1. W połączeniu   
z wiedzą, że wartość spektralna sygnału jednoelementowego równa jest wartości sygnału pozwala to na znaczne przyspieszenie obliczeń transformat. Pozostaje jednak problem rekompozycji rezultatów do widma całego sygnału.

Podział sygnału na pojedyncze próbki nie jest jednak wykonywany w sposób trywialny. Rozdzielenie zachodzi naprzemiennie, tj. na jeden sygnał wynikowy składają się próbki   
z pozycji parzystych, a na drugi z pozycji nieparzystych. Przykład dekompozycji sygnału   
8-próbkowego przedstawia Rysunek 8.



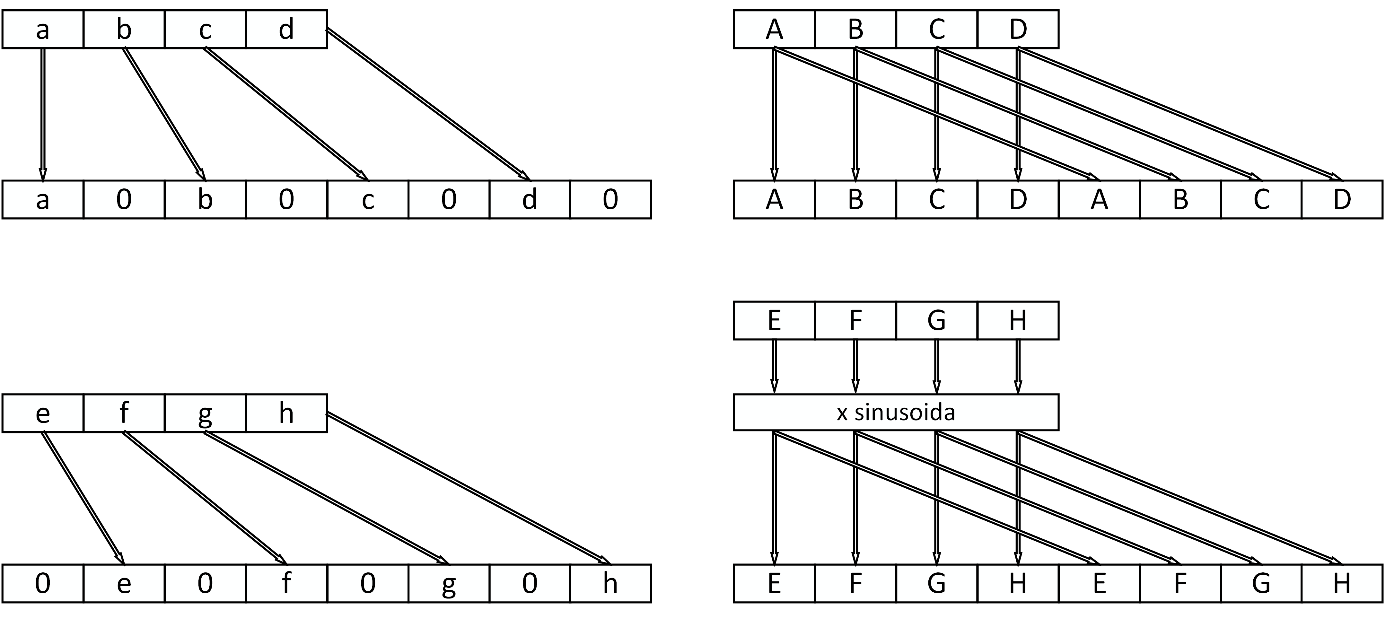
Rysunek 8 Dekompozycja sygnału N-elementowego do N sygnałów 1-elementowych

Taki sam nowy porządek próbek można uzyskać poprzez zastosowanie sortowania zgodnie z odwróconą kolejnością bitów [10]. Zasadę sortowania przedstawiono w Tabela 2.

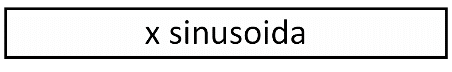
Tabela 2 Sortowanie zgodnie z odwróconą kolejnością bitów

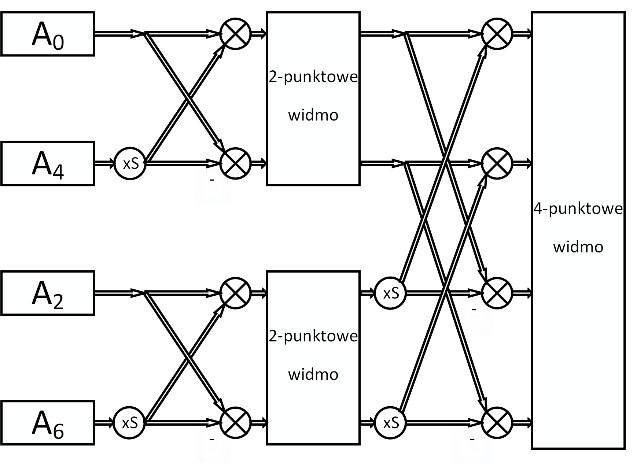
|  |  |  |  |
| --- | --- | --- | --- |
| Wartość dziesiętna początkowa | Zapis binarny | Odwrócenie zapisu binarnego | Nowa wartość dziesiętna |
| 0 | 000 | 000 | 0 |
| 1 | 001 | 100 | 4 |
| 2 | 010 | 010 | 2 |
| 3 | 011 | 110 | 6 |
| 4 | 100 | 001 | 1 |
| 5 | 101 | 101 | 5 |
| 6 | 110 | 011 | 3 |
| 7 | 111 | 111 | 7 |

Następnym krokiem jest scalenie pojedynczych wartości widma w pełne widmo. Wykonuje się to stopniowo dla kolejnych poziomów rozkładu drzewa przedstawionego na Rysunek 8. Potrzebne jest jednak zdefiniowanie operacji łączenia dwóch sygnałów k-elementowych   
w jeden 2k-elementowy. W tym celu stosuje się rozrzedzenie zerami sygnałów składowych. W dziedzinie czasu przedstawia się to następująco. Niech na wejściu do operacji sumowania będą sygnały *abcd* i *efgh.* W wyniku rozrzedzenia powstaną pośrednie sygnały *a0b0c0d*   
i *0e0f0g0h*. Suma pośrednich wyników da w rezultacie zapis *abcdefgh* [10]. Opisaną operacje przestawiono na Rysunek 9 wraz z jej odpowiednikiem w dziedzinie częstotliwości.



Rysunek 9 Rozrzedzanie składowych sygnałów przed sumowaniem

Zrozumiawszy zasadę łączenia sygnałów i składowych widm można przystąpić do końcowej operacji. Rekompozycja składa się z powtórzeń tego samego algorytmu sumowania dla coraz dłuższych widm składowych. Rysunek 10 przedstawia pierwsze dwa kroki algorytmu, a każdy kolejny odbywa się analogicznie. Warto zapamiętać kształt linii przepływu   
w pierwszym kroku, gdyż z jego charakterystycznego ułożenia często nazywany jest motylkiem [10]. Symbole  oraz  oznaczają mnożenie wartości przez sinusoidę o odpowiedniej częstotliwości.



Rysunek 10 Rekompozycja spektrum na pierwszych dwóch poziomach czterech z ośmiu punktów widma

Dzięki podzieleniu obliczeń na kroków algorytm szybkiej transformaty Fouriera posiada złożoność obliczeniową równą .

# Aplikacja realizująca modulowanie głosu

## Zdefiniowane modulacje

Przedmiotem tej pracy jest implementacja i zastosowanie do przetwarzania sygnału dźwiękowego trzech modulacji. Poniżej przedstawiono opis działania każdej z nich, możliwości konfiguracyjne od strony użytkownika aplikacji, listing kodu oraz wynik jego zastosowania.

Danymi wejściowymi dla każdego filtru jest tablica będąca wynikiem transformaty Fouriera oraz parametry zależne od modulacji. Tablica wejściowa jest zbiorem *n* wartości reprezentujących amplitudy kolejnych składowych częstotliwości. Każdy element tablicy odpowiada amplitudzie składowych z pasma o szerokości . Można więc zapisać, że

(6.1)

### Filtr o konfigurowalnej przepustowości pasm częstotliwości

Filtr przepustowy opiera swoje działanie na zmianie amplitud określonych pasm częstotliwości. Odpowiada temu odpowiednia modyfikacja konkretnych wartości tablicy wejściowej. Pełna konfigurowalność pozwala na określenie stopnia wyciszenia określonych składowych. Jest to mechanizm bardzo podobny do korektora stosowanego w odtwarzaczach multimedialnych. Zastosowanie tej samej wartości dla pełnego widma sygnału daje w wyniku zmianę głośności nagrania. Zastosowanie filtru dolnoprzepustowego na sygnale uzyskanym przy wysokiej częstotliwości próbkowania skutkuje obniżeniem jego jakości – wynikiem jest sygnał analogiczny do sygnału próbkowanego z niższą częstotliwością. Górnoprzepustowa konfiguracja umożliwia ukrycie różnych charakterystyk mowy, takich jak ton krtaniowy lub formanty fonemów.

Reprezentację stanowi zbiór par wartości, gdzie pierwsza oznacza częstotliwość, dla której określany jest współczynnik przepustowości będący drugą wartością pary. Umożliwia to graficzną reprezentację w postaci wykresu, którego osiami są częstotliwość i współczynnik przepustowości. Na wykresie punkty reprezentują konkretnie określone pary, a odcinki pomiędzy punktami przebieg wartości dla pozostałych częstotliwości. W przypadku niezdefiniowania wartości filtru dla częstotliwości lub przyjmuje się współczynniki odpowiednio takie jak dla pierwszej lub ostatniej zdefiniowanej.

Listing 1 przedstawia szczegóły implementacyjne tego filtru.

|  |
| --- |
| /\*\*  \* **@param spectrum** - zapis widma wejściowego  \* **@param passPoints** - lista par (częstotliwość, współczynnik)   \* posortowana rosnąco według częstotliwośći  \* **@return** - rezultat modulacji  \*/ **private double**[] filteringPass(**double**[] spectrum, List<Point> passPoints) {  **double**[] result = **null**;  **if** (spectrum != **null**) {  result = **new double**[spectrum.**length**];  List<Point> localPassPoints = **new** ArrayList<>();  **for** (**int** i = 0; i < passPoints.size(); ++i)  localPassPoints.add(  **new** Point(passPoints.get(i).getFrequency(),  passPoints.get(i).getValue()));  **if** (passPoints.size() == 0)  **return** spectrum;  **float** hzPerSpectrumIndex =  WaveRecord.getInstance().getAudioTrackSampleRate() /  2.0f / spectrum.**length**;   Point lower;  Point higher;  **int** lowerIndex = 0;  **int** higherIndex = 0;  **if** (localPassPoints.get(0).getFrequency() != 0)  localPassPoints.add(  0,  **new** Point(0, localPassPoints.get(0).getValue()));   lower = localPassPoints.get(0);  higher = localPassPoints.get(1);   **for** (**int** i = 0; i < spectrum.**length**; ++i) {  **if** (higher.getFrequency() < hzPerSpectrumIndex \* i) {  // change lower, higher points  **if** (higherIndex + 1 < localPassPoints.size())  higherIndex++;  **if** (lowerIndex + 1 < localPassPoints.size())  lowerIndex++;  higher = localPassPoints.get(higherIndex);  lower = localPassPoints.get(lowerIndex);  }  **int** deltaFrequency = higher.getFrequency() -  lower.getFrequency();  **float** deltaValue = higher.getValue() - lower.getValue();  **float** factor = lower.getValue() +  i \* hzPerSpectrumIndex \* deltaValue / deltaFrequency;   result[i] = spectrum[i] \* factor;  }  }  **return** result; } |

Listing 1 Implementacja modulacji: filtr przepustowy

Rysunek 11przedstawia efekt działania filtru przepustowego na sygnale składającym się   
z składowych częstotliwościowych 500 Hz, 1000 Hz, 1500 Hz, 2000 Hz, 2500 Hz, 3000 Hz, 3500 Hz i 4000 Hz. Przepustowość zastosowanego filtru przedstawiono na Rysunek 12.

Rysunek 11 Przykład działania filtru przepustowego

Rysunek 12 Zastosowana przepustowość filtru w przykładzie

### Skalowanie sygnału w dziedzinie częstotliwości

Różnica w wysokości dźwięków postrzegana przez ludzkie ucho w kontekście fizycznym sprowadza się do porównania częstotliwości określonych formantów. W kontekście muzycznym mówi się o *interwałach* dźwiękowych, które pozwalają na jasne określenie różnicy w wysokości dźwięków. Podstawowym interwałem jest oktawa, która w fizycznie odpowiada częstotliwości drugiej harmonicznej niższego dźwięku. Oznacza to, że dźwięk różny o oktawę posiada składowe o dwukrotnie wyższej częstotliwości. W wyniku podzielenia oktawy na mniejsze części otrzymuje się inne interwały.

Parametrem tej modulacji jest współczynnik skalowania częstotliwości. Określa on zarówno kierunek skalowania – w niższe lub wyższe dźwięki – ale i również oczekiwany interwał przekształcenia. Przyjmuje więc wartości rzeczywiste dodatnie. Współczynnik jest wartością przez która multiplikowane są częstotliwości składowe sygnału. Amplituda składowych nie ulega jednak zmianie. Poniżej przedstawiono przykładowe wartości współczynnika wraz   
z odpowiadającym interwałom:

* *f = 0.5 –* oktawa w dół
* *f = 1* – pryma, dźwięk bez zmian
* *f = 1.58 –* kwinta czysta w górę
* *f = 2* – oktawa w górę

Powyższą modulację można wykorzystać wprost do zmiany wysokości dźwięku lub   
w połączeniu z filtrem przepustowym. Drugi wariant może przynosić ciekawe rezultaty   
w przypadku zastosowania najpierw jednostronnie przepuszczalnego filtru, a następnie skalowania z odpowiednim współczynnikiem, by przenieść część pozostałych formantów   
w pasmo wcześniej wyciszone – współczynnik *f > 1* dla pierwszego filtru dolnoprzepustowego, współczynnik *f < 1* dla pierwszego filtru górnoprzepustowego.

Listing 2 przedstawia szczegóły implementacyjne modulacji.

|  |
| --- |
| **private double**[] filteringScale(**double**[] spectrum, **float** scaleFactor) {  **double**[] result = **null**;  **if** (spectrum != **null**) {  result = **new double**[spectrum.**length**];  **for** (**int** i = 0; i < spectrum.**length**; ++i) {  **if** (i \* scaleFactor < spectrum.**length**)  result[(**int**) (i \* scaleFactor)] = spectrum[i];  **else** result[i] = 0;  }  }  **return** result; } |

Listing 2 Implementacja modulacji: skalowanie w dziedzinie częstotliwości

Podobnie jak w pozostałych modulacjach parametr spectrum zawiera widmową reprezentację przetwarzanego okna czasowego. scaleFactor jest odpowiednikiem współczynnika, którego znaczenie opisano powyżej.

Rysunek 13 przedstawia wpływ modulacji na widmo sygnału. Przetwarzanym sygnałem jest dźwięk o składowych 220 Hz, 440 Hz oraz 880 Hz. Współczynnik skali wynosi . Sygnał próbkowano z częstotliwością 8 kHz, czego konsekwencją jest wyskalowanie osi poziomej wykresu.

Rysunek 13 Skalowanie w dziedzinie częstotliwości

### Rozmycie charakterystyk częstotliwościowych

Z punktu widzenia koncepcji jest to w pełni autorski pomysł na modulowanie dźwięku. Nie znaleziono w literaturze podobnych rozwiązań. W wyniku tego należy rozpatrywać tą modulacje w kontekście eksperymentu.

Operacja polega na utracie dokładnej informacji dotyczącej częstotliwości i amplitudy kolejnych formantów widma. Dokonuje się tego poprzez wyznaczenie nowej wartości amplitudy dla danego elementu tablicy przedstawiającej widmo sygnału w wyniku obliczenia średniej arytmetyczne sąsiednich elementów. Liczba sąsiednich pasm branych pod uwagę zależy od wartości parametru wejściowego *r* określającego szerokość rozmycia. Dosłownie parametr określa liczbę elementów uśrednianych zarówno w stronę wyższych jak i niższych częstotliwości. Zatem dla wartości *r = 5* wynikowa wartość i-tego elementu tablicy obliczana jest w oparciu o elementy o indeksach *(i-5, i-4, i-3, i-2, i-1, i, i+1, i+2, i+3, i+4, i+5)*,   
z uwzględnieniem granicznych przypadków na końcach tablicy.

Zastosowanie tej modulacji z niską wartością parametru *r* (dla   
) sprawia, że nagranie zostaje wyraźnie zniekształcone, jednakże nadal treść w nim zawarta jest zrozumiała. Wyższe wartości parametru *r* skutkują nieodwracalną utratą treści.

Listing 3 przedstawia metodę realizującą modulację uśredniania. Po walidacji parametrów wejściowych algorytm przystępuje do obliczenia nowych wartości widma. Wykorzystuje do tego funkcję przedstawioną na Listing 4. Implementuje ona obliczanie średniej arytmetycznej wskazanego przedziału tablicy podanej jako parametr.

|  |
| --- |
| /\*\*  \* **@param spectrum** - zapis widma wejściowego  \* **@param blurRange** - promień przedziału uśredniania  \* **@return** - rezultat  \*/ **private double**[] filteringBlur(**double**[] spectrum, **int** blurRange) {  **double**[] result = **null**;  **if (spectrum != null && blurRange > -1) {**  result = **new double**[spectrum.**length**];  **for** (**int** i = 0; i < spectrum.**length**; ++i)  result[i] = averageValue(spectrum, i, blurRange);  }  **return** result; } |

Listing 3 Implementacja modulacji: rozmycie charakterystyk częstotliwościowych

|  |
| --- |
| /\*\*  \* **@param array** - zapis widma wejściowego  \* **@param middleIndex** - indeks elementu uśrednianego  \* **@param range** - szerokość przedziału usrednianego,   \* uśrednianie elementów (middleIndex-range,   \* middleIndex+range)  \* **@return** \*/ **private double** averageValue(**double**[] array, **int** middleIndex, **int** range) {  **double** result = 0;  **if** (array != **null** && middleIndex >= 0 && range >= 0) {  **if** (middleIndex < array.**length** && range < array.**length**) {  **int** lowerBound = middleIndex - range;  **int** upperBound = middleIndex + range + 1;  **if** (lowerBound < 0)  lowerBound = 0;  **if** (upperBound > array.**length**)  upperBound = array.**length**;  **for** (**int** i = lowerBound; i < upperBound; ++i)  result += array[i];  result /= upperBound - lowerBound;  }  }  **return** result; } |

Listing 4 Funkcja uśredniająca wartości widma w przedziale

Na Rysunek 14 zobrazowano rezultat zastosowania tego filtru na dźwięku o składowych   
220 Hz, 440 Hz, 880 Hz, 1400 Hz oraz 2800 Hz. Sygnał ponownie został próbkowany   
z . Parametr *r* modulacji w tym przypadku wynosił *r = 1*, tj. średnia liczona była   
z 3 kolejnych wartości widma.

Rysunek 14 Uśrednienie charakterystyk częstotliwościowych

## Zrealizowana funkcja okna

Na potrzeby pracy zaimplementowano funkcję okna opracowaną przez austriackiego meteorologa Juliusa von Hanna. Od jego nazwiska funkcję nazywa się *oknem Hanna* lub *oknem Hanninga*. Wybraną ją ponieważ w znacznym stopniu redukuje efekt wycieku widma   
w procesie dyskretnej transformaty Fouriera, co jest ważnym czynnikiem wpływającym na dokładność wyników transformaty. Wysoką redukcję wycieku widma funkcja uzyskuje dzięki temu, że początkowe i końcowe wartości próbek w przetwarzanym oknie czasowym redukowane są do wartości bliskich zeru.

Funkcja okna Hanninga opisana jest wzorem (6.2), który odnosi się do wzoru (5.2):

(6.2)

Wszystkie prezentowane wykresy przedstawiają wartości dla pracy z dźwiękiem nagranym   
z częstotliwością próbkowania 8 kHz, co w przekłada się na zastosowanie 128 elementowych okien czasowych. Rzeczywisty czas trwania takiego okna wynosi .

Przebieg wartości funkcji przedstawia Rysunek 15.

Rysunek 15 Przebieg okna Hanninga

Zastosowanie funkcji okna na oknie czasowym sygnału zostało zaimplementowane w sposób przedstawiony w *Listing 5.* W poniższej implementacji wykorzystano również symetryczność przebiegu funkcji Hanninga.

|  |
| --- |
| **private float**[] windowing(**float**[] timeWindow) {  **float**[] result = **null**;  **if** (timeWindow != **null**) {  result = timeWindow.clone();  **for** (**int** i = 0; i < 0.5 \* result.**length**; ++i) {  **float** factor = getHanning(i,result.**length**);  result[i] \*= factor;  result[result.**length** - 1 - i] \*= factor;  }  }  **return** result; } **private float** getHanning(**int** n, **int** windowLength) {  **return** (**float**) (0.5f\*(1-Math.*cos*((2\*Math.***PI***\*n)/windowLength))); } |

Listing 5 Implementacja funkcji okna Hanninga

Efekt działania funkcji okna na sygnale okna czasowego przedstawia Rysunek 16*.* Przedstawiono na nim przebieg sygnału pierwotnego, będącego wycinkiem pełnego nagrania, przebieg wartości funkcji Hanninga dla 128 próbek sygnału wejściowego oraz przebieg sygnału wynikowego. Sygnał końcowy otrzymano poprzez mnożenie wartości próbki pierwotnej przez wartość funkcji okna dla odpowiedniego numeru próbki.

Rysunek 16 Działanie funkcji okna Hanninga

## Zaimplementowane metody cross-fadingu

Pierwszym krokiem procesu przetwarzania dźwięku był podział nagrania na nakładające się okna czasowe. Następnie każde z osobna, niezależnie, poddano przetwarzaniu. Ostatnim krokiem całego procesu musi być zatem połączenie okien, w celu uzyskania pełnego, zmodulowanego nagrania. Nakładanie się fragmentów wymusza podjęcia decyzji o sposobie określania finalnych wartości w częściach wspólnych.

*Cross-fading* opisany w części teoretycznej tej pracy, tj. w rozdziale 5.4 jest stosowany do płynnego nałożenia na siebie ścieżek dźwiękowych w przeciągu czasu rzędu sekund.   
W przypadku poniższej pracy mechanizm ten został wykorzystany do łączenia ze sobą fragmentów o długości 5 – 10 ms.

Zabieg cross-fadingu jest bezpośrednim krewnym szeroko opisywanego w literaturze fadingu. Różnica miedzy nimi polega na liczbie sygnałów, którymi operują. Fading działa na jednym sygnale, który jest stopniowo wyciszany lub zgłośniony. Cross-fading można opisać jako symetryczną parę tych procesów. Uznano, że na potrzeby pracy niezbędne jest zastosowanie takiego cross-fadingu, by suma obu składowych fadingów była stała. Wszystkie znane metody fadingu i cross-fadingu posiadają złożoność obliczeniową równą O(n), dlatego czas obliczeń nie jest w tym przypadku ważnym kryterium wyboru.

W ramach pracy zrealizowano dwie metody cross-fadingu. Implementacyjnie wynikową wartość danej próbki uzyskuje się w wyniku sumowania próbek sygnałów nakładających się wymnożonych przez różne współczynniki. Obliczenia można przedstawić przy pomocy wzoru (6.3).

(6.3)

Sposób wyliczenia współczynników jest jedyną różnicą pomiędzy metodami cross-fadingu.   
Z tego powodu poniżej przedstawiony *Listing 6* przedstawia kod źródłowy wspólny dla obu metod.

|  |
| --- |
| /\*\*  \* **@param partA** ścieżka ściszana  \* **@param partB** - ścieżka zgłośniona  \*/  **private float**[] crossfade(**float**[] partA, **float**[] partB) {  **float**[] result = **null**;  **if** (partA != **null** && partB != **null**) {  **if** (partA.**length** == partB.**length** && partA.**length** > 0) {  result = **new float**[partA.**length**];  **for** (**int** i = 0; i < result.**length**; ++i) {  **float** factor = getCrossFadeFactor(i, result.**length**);  result[i] = (1 - factor) \* partA[i] + factor \* partB[i];  }  }  } **else if** (partA == **null** && partB != **null**) {  result = partB.clone();  } **else if** (partA != **null** && partB == **null**) {  result = partA.clone();  }  **return** result; } |

Listing 6 Implementacja mechanizmu cross-fadingu

### Cross-fading liniowy

Liniowy przebieg cross-fadingu można wydaje się być najprostszym i najbardziej prymitywnym. Można jednak uznać, że jest najodpowiedniejszym wariantem w przypadku użycia przy okienkowaniu sygnału okna Bartletta lub okna trójkątnego. Jest tak dzięki łudząco podobnym przebiegu wartości.

Wyliczenie wartości współczynnika *f* opiera się na wzorze (6.4).

(6.4)

*Listing 7* przedstawia sposób zaimplementowania funkcji getCrossFadeFactor dla tej metody.

|  |
| --- |
| **private float** getCrossFadeFactor(**int** n, **int** N) {  **if** (N==0)  **return** 0f;  **return** (**float**) n/N; } |

Listing 7 Implementacja getCrossFadeFactor dla liniowego cross-fadingu

Poniżej przedstawiony Rysunek 17 przedstawia przebieg wartości współczynników odpowiednio dla obu ścieżek składowych. Przestawiono przypadek dla czego konsekwencją jest zastosowanie cross-fadingu na 64 próbkach – połowie okna czasowego, które składa się z 128 próbek.

Rysunek 17 Stosunek składowych sygnałów w cross-fadingu liniowym

Rysunek 18 pozwala zaobserwować jak w praktyce zachodzi ujednolicenie wyników modulacji na wspólnym fragmencie okna czasowego.

Rysunek 18 Przykład zastosowania cross-fadingu liniowego

### Cross-fading sinusowo – cosinusowy

­­­­Sinusowo-cosinusowa metoda cross-fadingu jest autorską odpowiedzią na opisywany   
w literaturze sinusowy fading [3].W celu zapewnienia stałości sumy współczynników skorzystano z twierdzenia o jedynce trygonometrycznej. W wyniku tego zamiast funkcji *sinus* i *cosinus* użyto kwadraty tych funkcji.

Wyliczenie wartości współczynnika *f* opera się na wzorze (6.5).

(6.5)

*Listing 8* przedstawia implementacjęfunkcji getCrossFadeFactor dla opisywanej metody cross-fadingu.

|  |
| --- |
| **private float** getCrossFadeFactor(**int** n, **int** N) {  **if** (N==0)  **return** 0f;  **return** (**float**) Math.*pow*(  Math.*sin*((**double**) n / N \* Math.***PI*** / 2.0), 2.0  ); } |

Listing 8 Implementacja getCrossFadeFactor dla sinusowo-cosinusowego cross-fadingu

Podobnie jak w poprzednim przypadku cross-fadingu liniowego został przedstawiony na przykładzie tego samego sygnału. Konsekwencją tego jest po raz kolejny rozpatrywanie 64-próbkowej części wspólnej sygnałów. Pozwoli to jednak na porównanie rezultatów obu wersji.

Rysunek 19 Stosunek składowych sygnałów w cross-fadingu sinusowo-cosinusowym

Rysunek 20 Przykład zastosowania cross-fadigu sinusowo+cosinusowego

## Opis programu

### Założenia projektowe

Praca nad programem wymagała przyjęcia pewnych założeń. Z czasem cześć z nich została zmodyfikowana lub odrzucona zależnie od napotkanych problemów. Pełny zbiór przyjętych założeń zwyczajowo dzieli na dwie lub więcej kategorii. Taki podział wykonano również   
w tym przypadku.

Na założenia funkcjonalne składają się wymogi dotyczące namacalnych możliwości aplikacji. Pojęcie to można sprowadzić do prostego pytania: „Co aplikacja ma umożliwiać?”. W skład tych założeń wchodzą zagadnienia dotyczące interfejsu użytkownika - zawartość poszczególnych aktywności oraz widocznych bądź słyszalnych efektów działania aplikacji.

Założeniami niefunkcjonalnymi nazywa się żądane cechy programu, które nie mają bezpośredniego wpływu na funkcjonalności. Mogą jednak doprowadzić do podniesienia jakości produktu finalnego, lub poprawienia odbioru aplikacji przez użytkowników (ang. u*ser experience*).

#### Założenia funkcjonalne

Lista przyjętych i spełnionych założeń funkcjonalnych przedstawia się następująco:

* aktywność główna umożliwiająca:
  + nagrywanie dźwięku,
  + odtwarzanie dźwięku,
  + modulacje nagrania,
  + reprezentowanie postępu odtwarzania w postaci stopera,
  + reprezentowanie postępu odtwarzania w postaci paska postępu,
  + przejście do aktywności wyboru ustawień,
* aktywność wyboru ustawień umożliwiająca:
  + zapisanie ustawień w formie nowego profilu,
  + zapisanie ustawień w bieżącym profilu,
  + wczytanie ustawień z profilu,
  + wybór parametrów nagrania:
    - częstotliwość próbkowania, dostępne: 8 kHz, 16 kHz, 44,1 kHz,
    - liczba kanałów, dostępne: mono, stereo,
  + wybór parametrów procesu modulacji:
    - metoda cross-fadingu, dostępne: liniowy, sinusowo-cosinusowy,
    - rodzaj modulacji,
    - parametry wybranej modulacji:
      * współczynnik skalowania dla modulacji skalowania dźwięku,
      * szerokość przedziału uśredniania dla modulacji rozmycia częstotliwościowego,
      * zbiór współczynników dla wybranych częstotliwości dla filtru przepuszczającego,
      * możliwość wyboru predefiniowanych konfiguracji filtru przepuszczającego.

#### Założenia niefunkcjonalne

Przyjęte i zrealizowane założenia funkcjonalne zaprezentowano poniżej:

* aplikacja na system Android,
* oparcie o API 15, zgodność z Android 4.0.3,
* brak wykorzystania *Java Native Interface*,
* konstrukcja interfejsów tworzona z myślą o smartfonach,
* pliki konfiguracyjne zgodne z DOM,
* programowo blokowana zmiana orientacji urządzenia,
* minimalna zajętość pamięci trwałej.

### Biblioteki zewnętrzne

Wykorzystanie zewnętrznych bibliotek jest dobrym sposobem na przyspieszenie pracy nad projektem. Pozwala to na skupienie się na głównych zagadnieniach aplikacji bez potrzeby własnoręcznego tworzenia potrzebnych do tego narzędzi. Przed podjęciem decyzji o użyciu konkretnej biblioteki warto zweryfikować jej pochodzenie oraz licencję w oparciu o którą jest rozprowadzana. Niepewne źródło może skutkować nieoptymalną implementacją algorytmów przez bibliotekę lub ich niepełnym przetestowaniem. Ważne jest również, by wraz z biblioteką udostępniana była również jej dokumentacja, sposób instalacji oraz przykłady wykorzystania.

W ramach pracy nad aplikacją wykorzystywano standardowe biblioteki wchodzące w skład API 15. Jedyną zewnętrzną biblioteką użytą w projekcie jest jfftpack [11]**.** Oferuje ona między innymi funkcje odpowiadające za wykonanie transformaty Fouriera oraz odwrotnej transformaty Fouriera. Właśnie te funkcjonalności zostały wykorzystane w aplikacji do szybkiego obliczania transformat.

### Analiza wydajnościowa

Wymagania wydajnościowe stawiane przed aplikacją zależą od zakładanego zastosowania. Jak wspomniano we wstępie do pracy powstała aplikacja ma charakter poznawczy. Pozwala stwierdzić, czy obecne rozwiązania technologiczne są w stanie wydajnie, w czasie rzeczywistym, przetworzyć sygnał. Niezbędne do tego było opracowanie modulacji, których wykonanie może zostać przeanalizowane pod kątem wydajności. Krytyczną kategorią wydajnościową w tym przypadku jest obciążenie obliczeniowe urządzenia. Z racji docelowego przetwarzania sygnału w czasie rzeczywistym złożoność pamięciowa nie jest istotna.

Wszystkie testy przeprowadzono na urządzeniu Sony Ericsson XPeria Neo V pracującym   
z systemem Android 4.0.4. Nie jest to urządzenie klasy premium dlatego można założyć, że na wydajniejszym sprzęcie działanie aplikacji będzie odpowiednio szybsze.

Po zaimplementowaniu wszystkich założonych funkcjonalności przystąpiono do przeprowadzenia szeregu testów. Sprawdzono wpływ parametrów na czas wykonania modulacji. W tym celu wykonano po 10 prób dla każdej z konfiguracji. Wyniki wstępnie przeanalizowano i odseparowano błędy grube, które wynikły z wpływu procesów działających w tle systemu operacyjnego oraz z mechanizmu stronicowania pamięci wirtualnej urządzenia. Błędy stron są konsekwencją wykorzystania do zapisu nagrania pamięci dynamicznej zamiast trwałej. Warto jednak pamiętać fakt, że dostęp do pliku jest z reguły wolniejszy. Ponadto wykorzystanie algorytmów do przetwarzania w czasie rzeczywistym zlikwiduje problem przechowywania pełnego zapisu dźwięku co   
w konsekwencji znacznie ograniczy występowanie błędów stron.

Rysunek 21 przedstawia średni czas pełnego przetwarzania jednego okna czasowego. W skład operacji, które zostają kolejno wykonane wchodzi: załadowanie odpowiednich próbek   
z pamięci, rozdzielenie kanałów (opcjonalnie), nałożenie funkcji okna, transformata Fouriera, modulacja widma, odwrotna transformata Fouriera oraz cross-fading rezultatów. Liczba próbek w oknie czasowym jest bezpośrednio związana z częstotliwością próbkowania zgodnie z wzorem (4.1).

Rysunek 21 Średni czas pełnego przetworzenia jednego okna czasowego

Rezultaty przeprowadzonych pomiarów wskazują na niewystarczającą wydajność maszyny testowej. Niesatysfakcjonujące wyniki wymusiły głębszą analizę czasu wykonania poszczególnych kroków algorytmu. Średnie czasy przejścia przez kolejne kroki przetwarzania przedstawia Rysunek 22. Zastosowano skalę logarytmiczną dla dokładniejszego przedstawienia małych wartości.

Rysunek 22 Średni czas wykonania poszczególnych operacji procesu przetwarzania okna czasowego

Można zauważyć, że suma kroków określonego scenariusza użycia jest mniejsza niż czas średni czas takiego scenariusza na Rysunek 21. Wynika to z wykonania dodatkowych operacji przekazujących wyniki jednego kroku do następnego. Analiza Rysunek 22 nie wskazuje na nieodpowiednią implementację kluczowych operacji nie licząc filtru przepustowego.

Jednoznaczne stwierdzenie czy obecna implementacja modulacji jest w stanie zapewnić ciągłą modulację w czasie rozmowy telefonicznej wymaga dodatkowych testów na nowszych, wydajniejszych urządzeniach.

## Prezentacja systemu

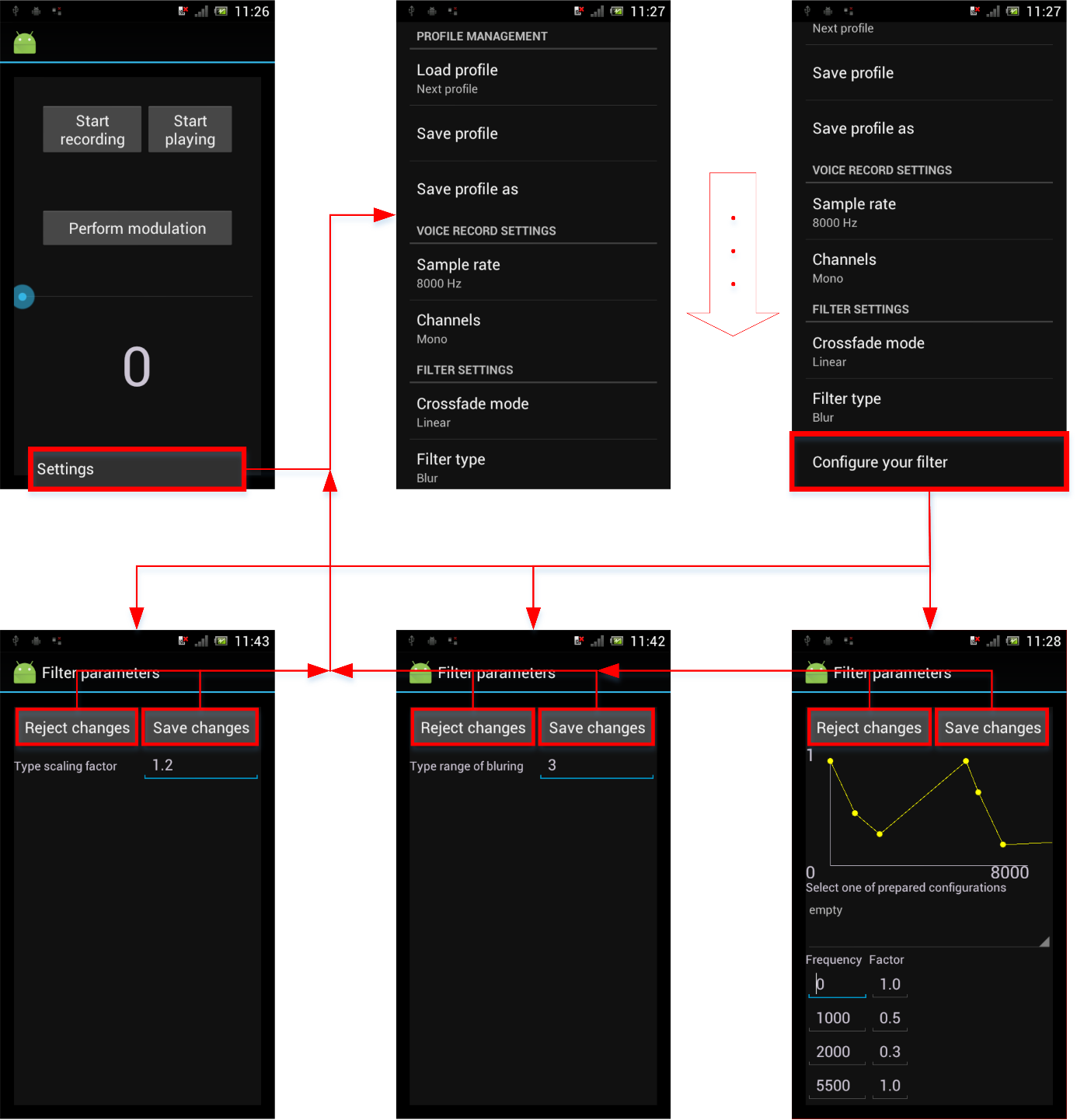
### Opis interfejsu

Mówiąc o interfejsie aplikacji pracującej na systemie Android należy przyjąć pewną konwencję nazewniczą. Programy tego typu opierają swoje działanie o aktywności [6]. Programowany system może składać się z jednej lub większej ich liczby, zawsze jednak któraś z nich będzie tą główną. W przypadku tej pracy takim punktem wejścia aplikacji jest widok umożliwiający nagranie, odtworzenie nagrania i nałożenie modulacji na dźwięk.   
W przypadku odsłuchiwania użyteczny staje się zawarty na ekranie stoper oraz pasek postępu. Niezbędna jest również możliwość przejścia do panelu konfiguracyjnego za co odpowiada stosowny przycisk.

Aktywność wyboru preferencji umożliwia wybór wszystkich dostępnych dla użytkownika opcji rejestracji dźwięku oraz modulacji. Zawiera ponadto interfejs pozwalający na zarządzanie profilami użycia, który oferuje wczytanie profilu, zapisanie zmian bieżącego lub utworzenie nowego na podstawie obecnych ustawień. Konfiguracja filtrów odbywa się   
w osobnym widoku, do którego przejście wyzwalane jest stosowną pozycją interfejsu.

W zależności od aktualnie wybranego rodzaju modulacji aktywność konfiguracji filtrów zawiera różne elementy interfejsu. Dzieje się tak, ponieważ każda z nich wymaga różnego   
i niezależnego parametru lub zbioru parametrów.

Rysunek 23 przedstawia możliwy przepływ sterowania pomiędzy widokami.



Rysunek 23 Przepływ sterowania w aplikacji

### Pliki konfiguracyjne

W oparciu o przyjęte założenia funkcjonalne zaimplementowano możliwość konfiguracji różnych parametrów. Dotyczą one zarówno aspektów jakości nagrywanego dźwięku jak   
i procesu modulacji. Całość zorganizowano w postać profili użycia. Jedynym elementem aplikacji ingerującym w pamięć trwałą urządzenia są pliki xml wykorzystywane do zapisu wybranych przez użytkownika preferencji (*profiles.xml*) oraz trwałego deklarowania bazowych przebiegów filtru przepustowego (*preparedFilters.xml*). Jeśli na wyposażeniu telefonu jest zewnętrzna karta pamięci to pliki przechowywane są na niej, a w przypadku braku to na w pamięci wewnętrznej urządzenia. W przypadku braku wspomnianych plików w pamięci aplikacja utworzy je i wypełni domyślnymi konfiguracjami. Zakładana zgodność z DOM wymusiła zawarcie w plikach sporych rozmiarów zapiski !DOCTYPE. Walidację przeprowadzono z wykorzystaniem narzędzia XML Validator [7].

Przykładową zawartość pliku *profiles.xml* przedstawiono na Listing 9. Poniżej pokrótce opisano znaczenie poszczególnych węzłów drzewa.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE profiles[  <!ELEMENT profiles (profile+)>  <!ATTLIST profiles activeProfile NMTOKEN #REQUIRED>  <!ELEMENT profile (profileName, voiceConfiguration, filterConfiguration, activeSettings)>  <!ELEMENT profileName (#PCDATA)>  <!ELEMENT voiceConfiguration (sampleRate, channels)>  <!ELEMENT sampleRate (#PCDATA)>  <!ELEMENT channels (#PCDATA)>  <!ELEMENT filterConfiguration (blurRange, scaleFactor, passFilterPoints)>  <!ELEMENT blurRange (#PCDATA)>  <!ELEMENT scaleFactor (#PCDATA)>  <!ELEMENT passFilterPoints (point+)>  <!ELEMENT point (#PCDATA)>  <!ATTLIST point frequency NMTOKEN #REQUIRED>  <!ELEMENT activeSettings (crossfade, filterType)>  <!ELEMENT crossfade (#PCDATA)>  <!ELEMENT filterType (#PCDATA)>  ]>  <profiles activeProfile="0">  <profile>  <profileName>My profile</profileName>  <voiceConfiguration>  <sampleRate>8000</sampleRate>  <channels>Mono</channels>  </voiceConfiguration>  <filterConfiguration>  <blurRange>3</blurRange>  <scaleFactor>1.2</scaleFactor>  <passFilterPoints>  <point frequency="0">1.0</point>  <point frequency="1000">0.5</point>  <point frequency="2000">0.3</point>  <point frequency="4000">1.0</point>  </passFilterPoints>  </filterConfiguration>  <activeSettings>  <crossfade>Linear</crossfade>  <filterType>BlurFilter</filterType>  </activeSettings>  </profile>  </profiles> |

Listing 9 Przykładowa zawartość pliku profiles.xml

*Profiles* jest korzeniem drzewa. Atrybut *activeProfile* określa numer aktywnego profilu z pliku. Numeracja zaczyna się od 0, a profile numerowane są zgodnie z kolejnością wystąpienia. *ProfileName* oznacza nazwę pod którą profil będzie widoczny w panelu ustawień aplikacji. *VoiceConfiguration* zawiera informację odpowiednio na temat częstotliwości próbkowania sygnału (*sampleRate*) oraz liczby kanałów (*channels*). *FilterConfiguration* określa wartości wszystkich parametrów modulacji jakie można zastosować. *ActiveSettings* wskazuje na aktualnie wybraną konfigurację modulacji, tj. tryb cross-fadingu (*crossfade*), oraz rodzaj nakładanego filtru (*filterType*).

Listing 10 przedstawia sposób konfiguracji bazowych filtrów przepustowych. Poniżej omówiono znaczenie poszczególnych elementów.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE preparedFilters[  <!ELEMENT preparedFilters (pointset+)>  <!ELEMENT pointset (name, point+)>  <!ELEMENT name (#PCDATA)>  <!ELEMENT point (#PCDATA)>  <!ATTLIST point frequency NMTOKEN #REQUIRED>  ]>  <preparedFilters>  <pointset>  <name>Pass filter 1</name>  <point frequency="0">1.0</point>  <point frequency="1000">0.5</point>  <point frequency="2000">0.3</point>  <point frequency="4000">1.0</point>  </pointset>  <pointset>  <name>Pass filter 2</name>  <point frequency="200">0.0</point>  <point frequency="500">1.0</point>  <point frequency="2000">0.3</point>  <point frequency="4000">1.0</point>  </pointset>  </preparedFilters> |

Listing 10 Przykładowa zawartość pliku preparedFilters.xml

*PreparedFilters* jest korzeniem drzewa. Każdy element *pointset* określa jedną bazową konfigurację filtru przepustowego. Składa się na nią nazwa (*name*), pod którą ustawienie będzie widoczne w interfejsie, oraz zestaw punktów(*point*), tj. par częstotliwość-wartość.

# Wnioski i podsumowanie

W ramach pracy dyplomowej zrealizowano wszystkie postawione przed autorem założenia. Przeprowadzone doświadczenia dotyczące modulacji mowy należy jednak postrzegać jedynie jako wprowadzenie do tego rozległego zagadnienia. Sukcesem można uznać skuteczną implementację filtru przepustowego, ponieważ jego odpowiednie wykorzystanie może nieść pozytywne rezultaty w podejściu utrudnienia identyfikacji rozmówcy. Podstawa techniczno-merytoryczna zastosowania modulacji uśredniającej widmo powinna zostać jeszcze przedyskutowana z osobami bardziej zaznajomionymi z zagadnieniami przetwarzania dźwięku. Najprzyjemniejsze pod względem wrażeń słuchowych rezultaty dawał najprostszy koncepcyjnie filtr skalujący spektrum częstotliwościowe sygnału. Służył on jednak bardziej jako wskaźnik wydajnościowy dla innych modulacji, ponieważ pod względem zapewnienia anonimowości nie przedstawia większej wartości.

Zaimplementowanie dwóch różnych metod cross-fadingu okien modulacji pozwoliło na empiryczne porównanie ich rezultatów. W przypadku stosowania tego mechanizmu dla ścieżek o długości dłuższych niż sekunda różnica jest zauważalna. Ludzkie ucho nie jest jednak w stanie wychwycić różnicy w przypadku rozpatrywania 10 milisekundowych fragmentów. Wnioskiem z tej obserwacji jest rada dla osoby rozwijającej projekt w przyszłości, by ograniczyła się do liniowej wersji cross-fadingu celem zmniejszenia czasu przetwarzania sygnału.

Dalszy rozwój programu mający na celu modulację w czasie rzeczywistym wymusza jednak częściową zmianę założeń lub koncepcji obecnego rozwiązania. Analiza wydajnościowa przedstawiona w rozdziale 6.4.3 wskazuje na to, że czas modulacji okna czasowego jest dłuższy niż samo okno. Wskazuje to na konieczność przyspieszenia obliczeń lub podniesienia wymagań sprzętowych urządzenia. Jednym z możliwych rozwiązań problemu z wydajnością jest zrównoleglenie obliczania transformaty Fouriera oraz operacji modyfikującej widmo sygnału. Równoległa implementacja FFT teoretycznie może posiadać złożoność . Zadanie to jest jednak trudne ze względu na czas wzbudzania nowych wątków lub procesów (zależnie od implementacji) oraz na konieczność komunikacji i synchronizacji działania pomiędzy nimi. Drugą możliwością przyspieszającą działanie algorytmów jest przepisanie ich do języka C oraz wykorzystanie ich poprzez *Java Native Interface.*

Być może z czasem zainteresowanie zaawansowaną obróbką dźwięku na urządzeniach mobilnych wywoła zmiany w obecnych zasadach konstrukcji smartfonów. Idealnym rozwiązaniem znacznie ułatwiającym przetwarzanie sygnałów na sprzęcie przenośnym byłoby rozszerzenie obecnie dostępnych narzędzi zawartych w API systemu operacyjnego. Dodanie mechanizmów umożliwiających skorzystanie z wbudowanego procesora dedykowanego przetwarzaniu sygnałów pozwoliłoby na sprzętową implementację obliczeń transformaty Fouriera. Jest to ciekawe również z tego faktu, iż operacje na dźwięku mają wiele wspólnego z przetwarzaniem obrazu. Audio jest niczym innym niż sygnałem jednowymiarowym, a grafika dwuwymiarowym statycznym (obraz) lub zmiennym w czasie (film).

# Literatura

1. Alan V. Oppenheim, R. W. (1979). *Cyfrowe przetwarzanie sygnałów.* Warszawa: Wydawnictwa Komunikacji i Łączności.
2. Asimo.pl. (2015). *Wortal robotyki*. Pobrano z lokalizacji Dobra strona robotyki: http://www.asimo.pl/teoria/twierdzenieshannona.php [Data dostępu: 30.11.2015]
3. Audacity. (2015). *Audacity 2.1.1 Manual*. Pobrano z lokalizacji Fade and Crossfade: http://manual.audacityteam.org/o/man/fade\_and\_crossfade.html [Data dostępu: 15.11.2015]
4. Carford, J. C. (2001). *A Practical Introduction to Phonetics.* Oxford: Oxford University Press.
5. D.Kincaid, W. (2006). *Analiza Numeryczna.* Warszawa: Wydawnictwa Naukowo Techniczne.
6. Hellman, E. (2015 ). *Platforma Android. Nowe Wyzwania.* Gliwice: Helion.
7. Leporda, A. (2010). *Validome*. Pobrano z lokalizacji Validation Services for your HTML/XHTML/WML: http://www.validome.org/ [Data dostępu: 30.10.2015]
8. Sadłoń, P. (2013). *LiveSound & Installation*. Pobrano z lokalizacji Zrozumiałość mowy: http://livesound.pl/tutoriale/artykuly/4629-zrozumialosc-mowy [Data dostępu: 2.12.2015]
9. Shannon, C. E. (1949). *Communication in the Presence of Noise.* Pobrano z lokalizacji http://nms.csail.mit.edu/spinal/shannonpaper.pdf [Data dostępu: 20.11.2015]
10. Smith, S. W. (2007). *Cyfrowe przetwarzanie sygnałów. Praktyczny poradnik dla inżynierów i naukowców.* Warszawa: Wydawnictwo btc.
11. Swarztrauber, P. N. (2015). *Netlib*. Pobrano z lokalizacji FFTPACK: http://www.netlib.org/fftpack/ [Data dostępu: 15.10.2015]
12. Tadeusiewicz, R. (1988). *Sygnał mowy.* Warszawa: Wydawnictwa Komunikacji   
    i Łączności.

# Spis rysunków

[Rysunek 1 Diagram przepływu danych w procesie modulacji sygnału 9](#_Toc437998241)

[Rysunek 2 Podział sygnału na okna czasowe. fs = 44,1kHz, n = 512, czas okna = 11,6 ms 10](#_Toc437998242)

[Rysunek 3 Wpływ filtracji górno- i dolnoprzepustowej pasma mowy ludzkiej na   
zrozumiałość [8] 12](#_Toc437998243)

[Rysunek 4 Zjawisko aliasingu przy złym doborze częstotliwości próbkowania 13](#_Toc437998244)

[Rysunek 5 Sygnał z eksponencjalnym efektem fade in 15](#_Toc437998245)

[Rysunek 6 Sygnał z logarytmicznym efektem fade out 15](#_Toc437998246)

[Rysunek 7 Asymetryczny cross-fade dwóch ścieżek dźwiękowych 16](#_Toc437998247)

[Rysunek 8 Dekompozycja sygnału N-elementowego do N sygnałów 1-elementowych 18](#_Toc437998248)

[Rysunek 9 Rozrzedzanie składowych sygnałów przed sumowaniem 19](#_Toc437998249)

[Rysunek 10 Rekompozycja spektrum na pierwszych dwóch poziomach czterech z ośmiu punktów widma 20](#_Toc437998250)

[Rysunek 11 Przykład działania filtru przepustowego 23](#_Toc437998251)

[Rysunek 12 Zastosowana przepustowość filtru w przykładzie 23](#_Toc437998252)

[Rysunek 13 Skalowanie w dziedzinie częstotliwości 25](#_Toc437998253)

[Rysunek 14 Uśrednienie charakterystyk częstotliwościowych 27](#_Toc437998254)

[Rysunek 15 Przebieg okna Hanninga 28](#_Toc437998255)

[Rysunek 16 Działanie funkcji okna Hanninga 29](#_Toc437998256)

[Rysunek 17 Stosunek składowych sygnałów w cross-fadingu liniowym 31](#_Toc437998257)

[Rysunek 18 Przykład zastosowania cross-fadingu liniowego 32](#_Toc437998258)

[Rysunek 19 Stosunek składowych sygnałów w cross-fadingu sinusowo-cosinusowym 33](#_Toc437998259)

[Rysunek 20 Przykład zastosowania cross-fadigu sinusowo+cosinusowego 33](#_Toc437998260)

[Rysunek 21 Średni czas pełnego przetworzenia jednego okna czasowego 36](#_Toc437998261)

[Rysunek 22 Średni czas wykonania poszczególnych operacji procesu przetwarzania okna czasowego 37](#_Toc437998262)

[Rysunek 23 Przepływ sterowania w aplikacji 38](#_Toc437998263)

# Spis listingów

[Listing 1 Implementacja modulacji: filtr przepustowy 22](#_Toc437997171)

[Listing 2 Implementacja modulacji: skalowanie w dziedzinie częstotliwości 24](#_Toc437997172)

[Listing 3 Implementacja modulacji: rozmycie charakterystyk częstotliwościowych 26](#_Toc437997173)

[Listing 4 Funkcja uśredniająca wartości widma w przedziale 26](#_Toc437997174)

[Listing 5 Implementacja funkcji okna Hanninga 28](#_Toc437997175)

[Listing 6 Implementacja mechanizmu cross-fadingu 30](#_Toc437997176)

[Listing 7 Implementacja getCrossFadeFactor dla liniowego cross-fadingu 31](#_Toc437997177)

[Listing 8 Implementacja getCrossFadeFactor dla sinusowo-cosinusowego cross-fadingu 32](#_Toc437997178)

[Listing 9 Przykładowa zawartość pliku profiles.xml 39](#_Toc437997179)

[Listing 10 Przykładowa zawartość pliku preparedFilters.xml 40](#_Toc437997180)