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 4](#_Toc437789604)

[2 Cel i zakres pracy 5](#_Toc437789605)

[3 Przegląd istniejących rozwiązań 6](#_Toc437789606)

[4 Opis procesu modulacji dźwięku 8](#_Toc437789607)

[5 Przegląd zagadnień modulacji dźwięku 11](#_Toc437789608)

[5.1 Charakterystyki sygnału mowy 11](#_Toc437789609)

[5.2 Stacjonarność sygnału 11](#_Toc437789610)

[5.3 Twierdzenie Kotielnikowa-Shannona 11](#_Toc437789611)

[5.4 Funkcje okna 12](#_Toc437789612)

[5.4.1 Porównanie kilku wybranych 13](#_Toc437789613)

[5.5 Cross-fading 13](#_Toc437789614)

[5.6 Transformata Fouriera 14](#_Toc437789615)

[5.6.1 Szybka transformata Fouriera (FFT) 14](#_Toc437789616)

[5.6.2 Odwrotna transformata Fouriera 14](#_Toc437789617)

[6 Aplikacja realizująca modulowanie głosu 15](#_Toc437789618)

[6.1 Zdefiniowane modulacje 15](#_Toc437789619)

[6.1.1 Filtr o konfigurowalnej przepustowości pasm częstotliwości 15](#_Toc437789620)

[6.1.2 Skalowanie sygnału w dziedzinie częstotliwości 17](#_Toc437789621)

[6.1.3 Rozmycie charakterystyk częstotliwościowych 18](#_Toc437789622)

[6.2 Zrealizowana funkcja okna 20](#_Toc437789623)

[6.3 Zaimplementowane metody cross-fadingu 22](#_Toc437789624)

[6.3.1 Cross-fading liniowy 23](#_Toc437789625)

[6.3.2 Cross-fading sinusowo – cosinusowy 25](#_Toc437789626)

[6.4 Opis programu 26](#_Toc437789627)

[6.4.1 Założenia projektowe 26](#_Toc437789628)

[6.4.1.1 Założenia funkcjonalne 27](#_Toc437789629)

[6.4.1.2 Założenia niefunkcjonalne 27](#_Toc437789630)

[6.4.2 Struktura projektu 28](#_Toc437789631)

[6.4.3 Biblioteki zewnętrzne 28](#_Toc437789632)

[6.4.4 Analiza wydajnościowa 28](#_Toc437789633)

[6.5 Prezentacja systemu 30](#_Toc437789634)

[6.5.1 Opis interfejsu 30](#_Toc437789635)

[6.5.2 Pliki konfiguracyjne 30](#_Toc437789636)

[6.6 Przykłady działania 30](#_Toc437789637)

[7 Wnioski i podsumowanie 31](#_Toc437789638)

[8 Literatura 32](#_Toc437789639)

[Spis rysunków 33](#_Toc437789640)

[Spis listingów 34](#_Toc437789641)

# 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 uniemożliwiał 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

[co zostało zrealizowane, jakie przyjęto założenia, przewodnik po pracy, co i gdzie po kolei jest omawiane (akapit o każdym z rozdziałów, króciutkie ich streszczenie, będzie sporo tekstu :D)]

# Przegląd istniejących rozwiązań

Najpopularniejszym źródłem aplikacji dedykowanych systemowi Android jest sklep internetowy Google Play. Duża liczba dostępnych na nim aplikacji jest jedną z przyczyn popularności samego systemu Android. Z racji jego popularności przegląd aplikacji 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. Szeroka oferta sklepu internetowego zawiera wiele proponowanych aplikacji służących odtwarzaniu muzyki zapisanej z różnych formatach i jakościach. Oprócz odtwarzania i organizowania plików w listy odtwarzania (ang.p*laylists*) część bardziej zaawansowanych umożliwia modulacje odtwarzanego dźwięku. Najczęstszym stosowanym zabiegiem jest zastosowanie korektora (ang. *equalizer*). 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 innymi poprzez inne serwery niż zdefiniowane przez producenta.

**Jakieś przykładowe podać**

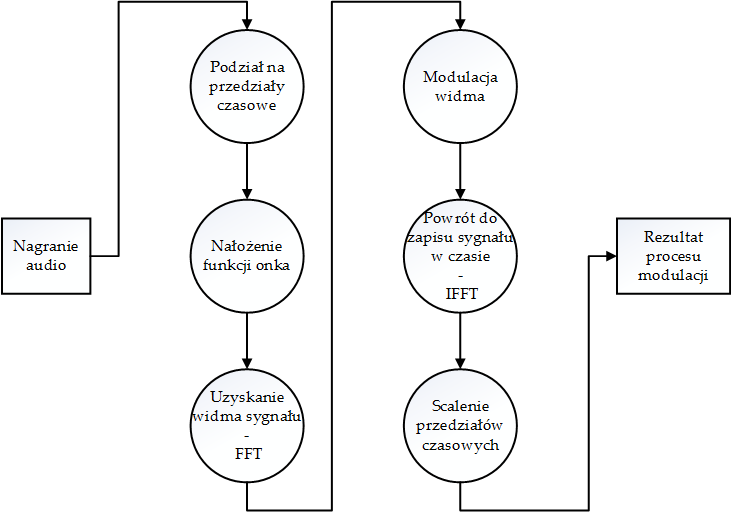
# Opis procesu modulacji dźwięku

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.

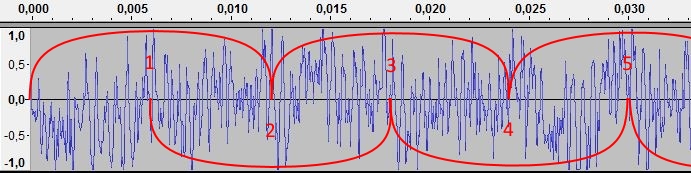


Rysunek Diagram przepływu danych w procesie modulacji sygnału

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 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

[Krótki wstęp o tym, że są to zagadnienia niezbędne dla tego tematu]

## Charakterystyki sygnału mowy

[Formanty dźwięków, ton krtaniowy]

## Stacjonarność sygnału

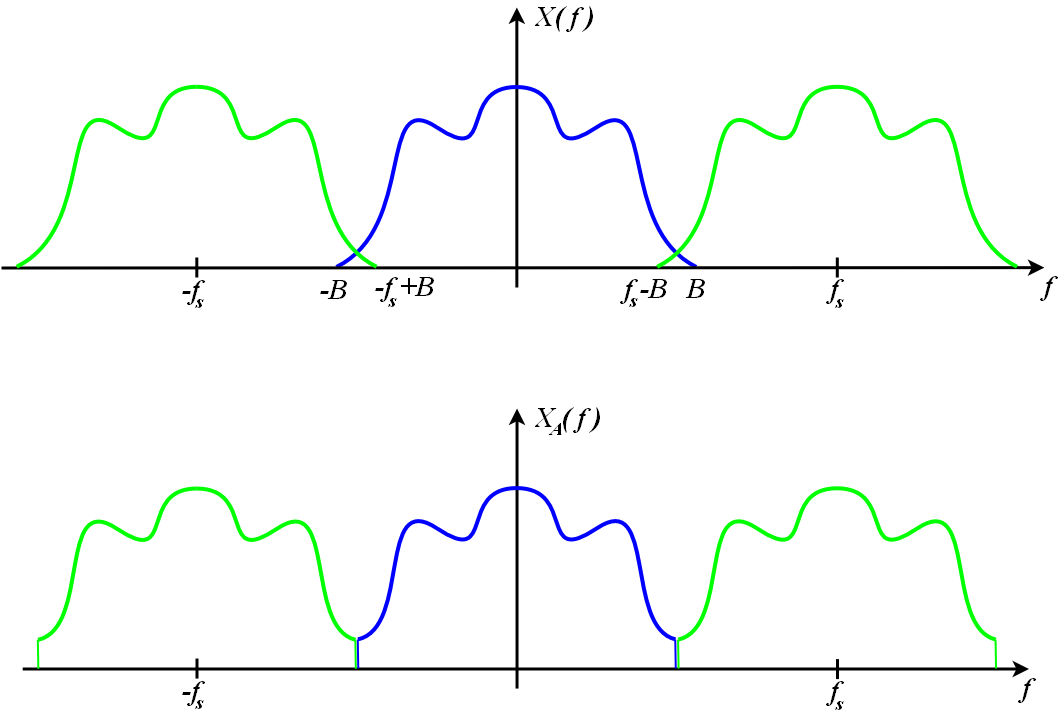
[Czym jest, konsekwencja odnośnie transformaty Fouriera]

## Twierdzenie Kotielnikowa-Shannona

[O czym mówi twierdzenie, jakie są jego konsekwencje dla tematu pracy]

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. 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 rysunku



Rysunek Zjawisko aliasingu, źródło: <https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem#/media/File:AliasedSpectrum.png>

W jaki sposób zapisać źródła internetowe?

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.

Konsekwencją …

## Funkcje okna

[Zapobiega wyciekom widma przy fft]

Transformata Fouriera (opisana w rozdziale 5.6) 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 jest okresowa. Zastosowanie transformaty na nieokresowym sygnale powoduje powstanie w rezultacie procesu *wycieku widma*. Dzieje się tak z powodu nieciągłości końca i początku sygnału, co interpretowane jest jako 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 (5.4.1)

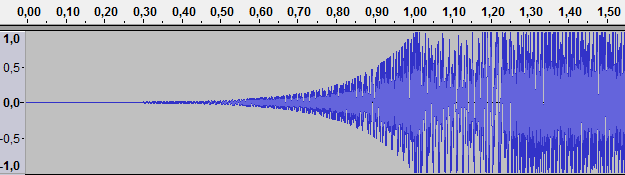
(5.4.1)

### Porównanie kilku wybranych

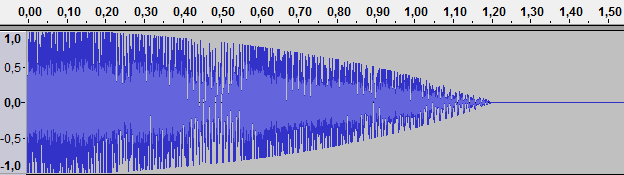
[Kilka funkcji przedstawionych poprzez wzory matematyczne, przebieg wartości funkcji w dziedzinie oraz wykresy wyników fft sygnału]

## 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*). W zależności od preferencji słuchowych i przeznaczenia wybiera się odpowiedni przebieg krzywej, według której przebiega zmiana natężenia dźwięku nagrania. Rysunek 4 przedstawia efekt zastosowania wzrastającego przebiegu eksponenciajnym, a Rysunek 5 przedstawia wynik zastosowania cieniowania opadającego o charakterze logarytmicznym.

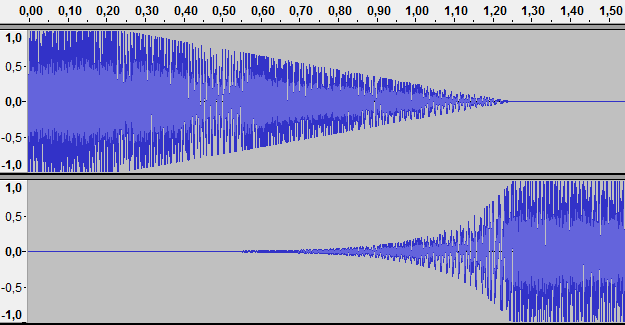


Rysunek Sygnał z eksponencjalnym efektem fade in



Rysunek 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 6 przedstawia przykład zastosowanie różnych fadingów – fade out liniowy, fade in eksponencjalny.



Rysunek Asymetryczny cross-fade dwóch ścieżek dźwiękowych

## Dyskretna transformata Fouriera

[Czym jest, splot w dziedzinie czasu zamienia się w mnożenie w dziedzinie częstotliwości, interpretacja wyniku – tablica *n* wartości reprezentujących widma o szerokości *fmax/n* a *fmax = fs/2* ]

*[Rozwój nauki związany z pojęciem przetwarzania sygnałów w znacznym stopniu przypada na okres przed nadejściem epoki komputerów. ]*

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.

Matematycznie operacja transformaty polega na zastosowaniu wzoru asd dla funkcji reprezentującej przebieg analizowanego sygnału. Musi spełniać on jednak szereg warunków niezbędnych dla poprawności rezultatów.

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.

### Szybka transformata Fouriera (FFT)

[zasada działaia -> wymóg liczby próbek wielkości potęgi dwójki]

### Odwrotna transformata Fouriera

[czym jest]

# Aplikacja realizująca modulowanie głosu

## Zdefiniowane modulacje

Przedmiotem tej pracy jest implementacja i zastosowanie do przetwarzania sygnału dźwiękowego trzech modulacji. Poniżej przedstawiam opis działania każdego z nich, sposób reprezentacji dla 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

[sposób reprezentacji przepustowości filtru, słowny opis implementacji, listing implementacji, przykład modulacji]

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 całego sygnału. 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.

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 xxx 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; } |

Wykresy – widmo + wykresik filtru

### Skalowanie sygnału w dziedzinie częstotliwości

[znaczenie magicznego współczynnika skalowania, słowny opis implementacji, listing implementacji, przykład modulacji]

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 kontekście fizycznym 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 dać 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 1 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 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 7 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 Skalowanie w dziedzinie częstotliwości

### Rozmycie charakterystyk częstotliwościowych

[znaczenie szerokości rozmycia, słowny opis implementacji, listing implementacji, przykład modulacji]

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 xxx przedstawia metodę realizującą modulację uśredniania. Po walidacji parametrów wejściowych algorytm przystępuje do obliczenia nowych wartości widma. Wykorzystuje do tego funkcję przedstawioną na listingu xxx. Realizuje 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; } |

|  |
| --- |
| /\*\*  \* **@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; } |

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

## 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):

(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 8Rysunek 8 Przebieg okna Hanninga

Rysunek Przebieg okna Hanninga

Zastosowanie funkcji okna na oknie czasowym sygnału zostało zaimplementowane w sposób przedstawiony w *Listing 2.* 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 Implementacja funkcji okna Hanninga

Efekt działania funkcji okna na sygnale okna czasowego przedstawia Rysunek 9***Błąd! Nie można odnaleźć źródła odwołania.****.* 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 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 5.5 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 3* 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 3 Implementacja mechanizmu cross-fadingu

### Cross-fading liniowy

Liniowy przebieg cross-fadingu można nazwać 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 4* 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 4 Implementacja getCrossFadeFactor dla liniowego cross-fadingu

Poniżej przedstawiony rysunek zx 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 xxx pozwala zaobserwować jak w praktyce zachodzi ujednolicenie wyników modulacji na wspólnym fragmencie okna czasowego.

### Cross-fading sinusowo – cosinusowy

[Wykres przebiegu, opis implementacji]

Sinusowo-cosinusowa metoda cross-fadingu jest autorską odpowiedzią na opisywany w literaturze sinusowy fading. 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 5* 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 5 Implementacja getCrossFadeFactor dla sinusowo-cosinusowego cross-fadingu

Podobnie jak w poprzednim przypadku *cross-fading­­* 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.

## 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 przyjęto dzielić na dwa 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 aplikacji, 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,

### Struktura projektu

[diagram klas, użyte wzorce i dlaczego]

### 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 (adres do źródeł: http://www.netlib.org/fftpack/) .** 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.

Wykres xxx 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).

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 zxc. Zastosowano skalę logarytmiczną dla dokładniejszego przedstawienia małych wartości.

Można zauważyć, że suma kroków określonego scenariusza użycia jest mniejsza niż czas średni czas takiego scenariusza na rysunku zxc. Wynika to z wykonania dodatkowych operacji przekazujących wyniki jednego kroku do następnego. Analiza rysunku tego tutaj 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

[screeny poszczególnych aktywności, działanie przycisków, w ustawieniach znaczenie poszczególnych opcji]

### Pliki konfiguracyjne

[co jest czym w plikach, co się dzieje kiedy ich nie ma, jak powinien wyglądać pusty plik profili]

## Przykłady działania

[czy ten punkt ma sens? Projekt opiera się o modulacje sygnałów, a działanie modulacji pokazane jest już w 4.1]

# Wnioski i podsumowanie

# Literatura

# Spis rysunków

[Rysunek 1 Diagram przepływu danych w procesie modulacji sygnału 9](#_Toc437789655)

[Rysunek 2 Podział sygnału na okna czasowe. fs = 44,1kHz, n = 512, czas okna = 11,6 ms 10](#_Toc437789656)

[Rysunek 3 Zjawisko aliasingu, źródło: https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon\_sampling\_theorem#/media/File:AliasedSpectrum.png 12](#_Toc437789657)

[Rysunek 4 Sygnał z eksponencjalnym efektem fade in 13](#_Toc437789658)

[Rysunek 5 Sygnał z logarytmicznym efektem fade out 13](#_Toc437789659)

[Rysunek 6 Asymetryczny cross-fade dwóch ścieżek dźwiękowych 14](#_Toc437789660)

[Rysunek 7 Skalowanie w dziedzinie częstotliwości 18](#_Toc437789661)

[Rysunek 8 Przebieg okna Hanninga 21](#_Toc437789662)

[Rysunek 9 Działanie funkcji okna Hanninga 22](#_Toc437789663)

# Spis listingów