Jakub Janiszewski, Bartosz Golis

Politechnika Wrocławska,

Equalizer akustyczny

Wstęp

Tematem projektu, który wybraliśmy w ramach realizowanych zajęć z cyfrowego przetwarzania sygnałów było zaprojektowanie equalizera akustycznego. Korektor, czyli inaczej equalizer, to zespół filtrów, które służą do podbijania lub tłumienia określonego zakresu częstotliwości zmieniając tym samym barwę dźwięku.

Zastosowanie korektora jest integralną częścią procesu obróbki i miksowania dźwięku podczas rejestracji żywych muzyków, ścieżek akustycznych, dźwięków syntetycznych czy samplowanych. Korektory są stosowane do kształtowania ogólnego brzmienia, korygowania specyficznych problemów częstotliwościowych oraz mieszania brzmień różnych dźwięków czy instrumentów. Istnieje kilka typów korektorów, najpowszechniejszymi są korektory graficzne i parametryczne. Projekt equalizera akustycznego w tym wypadku miał za zadanie filtrować sygnał lub np. przetwarzać muzykę na potrzeby użytkownika.

Większość typowych korektorów używa układów, które aktywnie podbijają lub tłumią dźwięk w różnych pasmach przez zastosowanie technik elektronicznego sprzężenia zwrotnego. Mogą one wprowadzać słyszalne „dzwonienie" w układach, wynikające z nieuchronnych w takich sytuacjach przesunięć fazowych. Inaczej działają korektory pasywne. Korekcja charakterystyki odbywa się tam przy użyciu elementów pasywnych (pracujących bez zasilania), takich jak rezystory, kondensatory oraz cewki. Znajdujący się za tym układem pojedynczy stopień wzmacniacza o prostej konstrukcji kompensuje obniżenie poziomu sygnału, co powoduje, że kiedy wszystkie regulatory są w centralnym położeniu – charakterystyka jest płaska.
Miał on za zadanie ustalanie odpowiednich amplitud sygnału w różnych pasmach częstotliwości. Projekt składał się z trzech części:

- Equalizera akustycznego wykonanego w programie Code Composer Studio przy użyciu języka C.

- Plików tworzących nastawy equalizera napisanych w programie MatLab. Na potrzeby projektu zaprojektowano trzy osobne mskrypty w których użytkownik łatwo mógł zmienić nastawy każdego z trzech wykorzystanych filtrów.

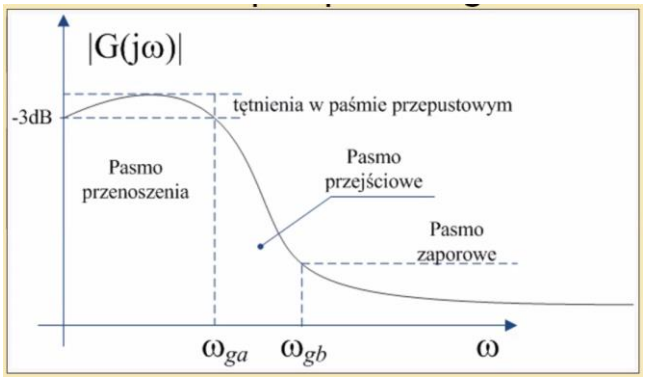
- GUI(Graphic User Interface), czyli menu stworzonego za pomocą Windows Forms, wraz z wykorzystanymi elementami języka C# i C++. Jest to element dodatkowy umożliwiający prostą obsługę całego projektu

Opis projektu

I)

Zaprojektowany w ramach projektu korektor składał się z trzech filtrów odpowiedzialnych za tłumienie odpowiedniego pasma częstotliwości.

1. Pierwszy z filtrów to filtr dolnoprzepustowy mający na celu tłumienie basów, czyli częstotliwości od zera do 200 Hz. Filtru dolnoprzepustowe znalazły wiele zastosowań w przetwarzaniu sygnałów. Przykładowo usuwanie zjawiska aliasingu, czyli nakładania się próbek odbywa się właśnie za pomocą filtra dolnoprzepustowego.



Rys.1 Charakterystyka amplitudowa filtru dolnoprzepustowego.

Wielkością charakteryzującą taki układ jest transmitancja określana jako stosunek napięcia wyjściowego do wejściowego i często zapisuje się ją w postaci operatorowej:

$$G(s) = \frac{K}{Ts + 1}$$

Gdzie:
K – wzmocnienie w paśmie przenoszenia
T – stała czasowa
S - pulsacja zespolona, $s = j2\pi f$.

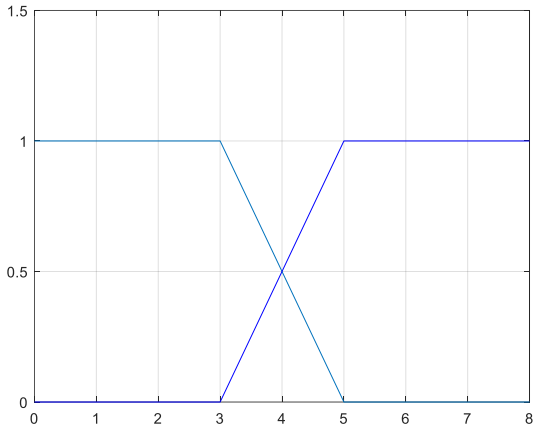
2. Drugi z filtrów to filtr środkowoprzepustowy nazywany również pasmowoprzepustowym. Ten filtr przepuszcza w equalizerze częstotliwości w przedziale od 200 Hz do 2000 Hz. Taki filtr o szerokim paśmie często realizuje się poprzez zbudowanie układu z filtra dolnoprzepustowego do określonej środkowej wartości i górnoprzepustowego przepuszczającego częstotliwości powyżej środkowej częstotliwości przenoszenia, aż do zakładanego końca pasma przepustowego projektowanego filtru. W projekcie zastosowano filtry stworzone za pomocą algorytmu, co ułatwia znacznie realizację tłumienia sygnałów, a co za tym idzie nie trzeba stosować takich zastępstw filtru innymi. Jeżeli odpowiednie nastawy filtrów będą nieoptymalne projekt zakłada możliwość zmiany parametrów. W kodzie matlabowskim znajdują się dwie linie kodu odpowiedzialne za generowanie wartości na których bazują filtry. Można je dowolnie zmieniać pamiętając o zasadzie, że obydwie tablice z wartościami muszą mieć tyle samo składników.

```
fHz=[0 50 100 150 200 250 300 1250 1350 1950 2050 3000 fNq]; %wektor f
amp=[1 1 1 1 1 0.5 0 0 0 0 0 0 0];% i odpowiadające im pożądane amplitudy
```

Rys.2 Fragment kodu z mskryptu filtru dolnoprzepustowego

Jak można zauważyć wartości tablicy amp, czyli wartości czynnika odpowiedzialnego za tłumienie są ustawione na jedynki przy częstotliwościach tylko do 200 Hz. Dzięki temu zostały ustawione pasma przenoszenia i tłumienia filtru. Tam, gdzie zostaną ustawione współczynniki na 0 sygnał jest tłumiony i filtr go nie przepuszcza. Jest to już obszar działania filtru środkowoprzepustowego i to on jest odpowiedzialny za przepuszczanie(bądź tłumienie) wartości sygnałów w tym przedziale. Jak można zauważyć pojawiła się również jedna wartość 0.5. Powodem tego jest zwiększenie jakości naszego equalizera. Filtry swoim działaniem delikatnie na

siebie nachodzą, by nie stworzyć sztywnej granicy, a dźwięki będące na granicy przy kładowo basu również były delikatnie tłumione.



Rys.3 Schematyczny rysunek przedstawiający nachodzące na siebie pasma przenoszenia dwóch filtrów

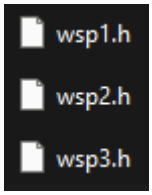
3. Ostatnim z filtrów jest filtr górnoprzepustowy przepuszczającym tylko wartości powyżej wartości granicznej, czyli w naszym wypadku częstotliwości od 2000 Hz do 4000 Hz, czyli połowy naszej wartości częstotliwości próbkowania. Equalizer został zaprojektowany dla częstotliwości próbkowania równej 8 kHz. W idealnym filtrze, w paśmie przepustowym współczynnik tłumienia powinien być równy zero, natomiast w paśmie zaporowym powinien być duży. Znajomość charakterystyki częstotliwościowej współczynnika fazowego pozwala na określenie zmiany fazy napięcia i prądu przy przejściu sygnału przez filtr. W przypadku tego filtru również w plikach programu mamy mskrypt odpowiedzialny za nastawy filtru i jego współczynniki tłumienia.

```
fHz=[0 50 100 150 200 250 300 1250 1350 1950 2050 2500 2700 3000 fNq]; %wektor
amp=[0 0 0 0 0 0 0 0 0 0 1 1 1 1 1];% i odpowiadające im pożądane amplitudy
```

Rys.4 Linie kodu z nastawami filtra górnoprzepustowego

Jak można zauważyć w tym wypadku nie ma wartości tłumienia 0.5 na granicy pasma przepustowego. Wynika to z faktu, że wartości przy granicy przenoszenia są blisko siebie (1950 Hz i 2050 Hz), więc jeżeli dla odpowiednio pierwszej z nich jest ustawione 0, a dla drugiej 1, to między nimi charakterystyka będzie rosnąć od 0 do 1 tworząc w ten sposób pasmo przejściowe również w kształcie funkcji rosnącej, a dzięki temu również charakterystyka przypomina kształtem trapez.

Każdy mplik po wgraniu go tworzy w plikach equalizera biblioteki z wygenerowanymi nastawami nastawami każdego z filtrów.



Rys.5 Biblioteki z nastawami wszystkich trzech filtrów

II)

Kolejnym elementem projektu jest sam kod equalizera łączący filtry i wykorzystujący jego działania.

```
//Fir.c
#include "dsk6713_aic23.h"  //plik narz

Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;  //u


#include "wsp1.h" //wspó³czynniki filtr
#include "wsp2.h" //wspó³czynniki filtr
#include "wsp3.h" //wspó³czynniki filtr
int yn = 0,yn1=0,yn2=0,yn3=0;
float bass_gain,mid_gain,treble_gain;
short x[N];              // tablica prób
```

Rys.6. Początek kodu equalizera w programie Code Composer Studio

Analizując kod możemy zauważyć, że pierwsze linijki to oczywiście zaimplementowane biblioteki, zarówno samych sterowników dotyczących procesora na którym pracujemy, jak i wcześniej wspomnianych współczynników wszystkich trzech filtrów. Trzy ostatnie linie to dodanie potrzebnych zmiennych. W pierwszej z nich widzimy zmienne, które na początku wynoszą zero. Są to zmienne, które będziemy później inkrementować. To w tych zmiennych będzie przechowywana wartość każdej kolejnej próbki. Pierwsza z nich, czyli yn odpowiada za ostateczną wartość próbki. Pozostałe trzy to zmienne przechowujące jedynie próbkę po filtracji jednego z filtrów.

```
interrupt void c_int11()        //funkcja obs³ugi przerwania
{
    short i;

    x[0] = input_sample();    //czytanie próbki z przetwornika A
        yn = 0;

    for (i = 0; i< N; i++)
    {
        yn1 += (h1[i] * x[i]);
        yn2 += (h2[i] * x[i]);
        yn3 += (h3[i] * x[i]);  // tutaj robimy sumê iloczynów(
    }
    for (i = N-1; i > 0; i--)
       x[i] = x[i-1];       //przesuniêcie o jedn¹ próbkê zawart

    yn=bass_gain*yn1*1/3+mid_gain*yn2*1/3+treble_gain*yn3*1/3;
    output_sample(yn);    //wys³anie próbki wyjœciowej do przetw
    return;
}
```

Rys. 7. Dalsza część kodu equalizera

Kod przedstawiony na powyższym zdjęciu obrazuje nam kolejno implementacje biblioteki x[0] odpowiedzialnej za przechowywanie próbek sygnału wejściowego. Następnie mamy pętlę for realizującą przetwarzanie kolejnych próbek przez wszystkie trzy filtry. Próbka wchodząca jest mnożona razy współczynnik każdego filtru z bibliotek. Gdy już dokonamy filtracji kolejna pętla wraca z powrotem do początkowej wartości tablicy x[i]. Jak można zauważyć to teraz wszystkie próbki są mnożone razy odpowiedni spółczynnik suwaka i dzielony przez trzy. Dzielenie odbywa się ze względu na to, że zrealizowany program zakłada połączenie filtrów równolegle, nie szeregowo. Umożliwia to działanie filtrów osobno, niezależnie od siebie. Drugi ze współczynników, o który mnożymy próbki to współczynniki odpowiedzialne za ustawienia suwaków.

III)

Suwaki zostały dodane do programu ze względu na możliwość regulacji działania każdego z poszczególnych

filtrów. Do programu został zaimplementowany plik z rozszerzeniem .gel.

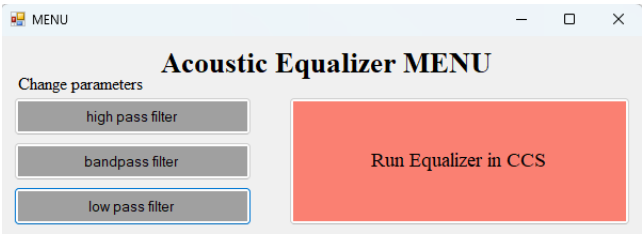
```
slider Bass(0,10,1,1,bass_slider) {       /* From 0 to 10
      bass_gain = ((float)bass_slider)/10;  }   /* ->From

slider Middle(0,10,1,1,mid_slider) {            /* From 0
      mid_gain = ((float)mid_slider)/10;  }     /* ->From

slider Treble(0,10,1,1,treble_slider) {         /* From 0
      treble_gain = ((float)treble_slider)/10;  }   /* ->F
```

Rys.8 Przedstawienie kodu odpowiedzialnego za dodanie suwaków

Takie rozwiązanie pozwala manipulować stopniem tłumienia każdego z filtrów osobno. To tu w zależności od ustawienia suwaka jest ustawiana w czasie rzeczywistym wartość zmiennej o którą jest mnożona w głównym programie ostateczna wartość próbek sygnału.
Suwak ma możliwość regulacji od 0 do 1 o krok równy 0.1. Dzięki temu ustawienie suwaka na sam dół(wartość zero) tłumi w pełni sygnał i go nie przepuszcza. Umożliwia nam to zmianę dźwięku przykładowo by przepuszczał tylko częstotliwości środka.

IV)

Ostatnim elementem projektu jest menu ułatwiające poruszanie się po plikach. Aby wszystko sprawnie działało pliki equalizera zostały umieszczone w plikach menu. Takie rozwiązanie umożliwiło dynamiczne ścieżki, przez co każdy kto pobiera projekt może je uruchomić. Menu jednak powstało za pośrednictwem już nowszych rozwiązań Visual Studio i wewnętrznej struktury Windows Forms, co utrudnia włączenie go na starszych urządzeniach.

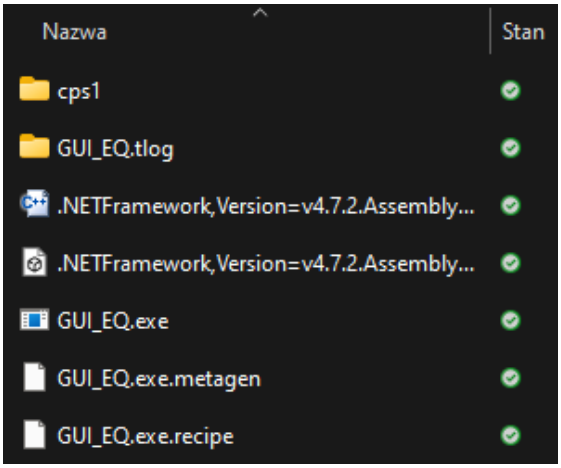


Rys.8 Widok zaprojektowanego menu.

Menu posiada cztery przyciski istotne dla działania projektu. Trzy szare, po lewej stronie podpisane nazwami filtrów to przyciski uruchamiające mskrypty w MatLabie opisane wyżej jako odpowiedzialne za współczynniki tłumienia poszczególnych plików. Działanie menu odbywa się w tle dzięki czemu jest ono aktywne i czeka na zakończenie pracy w MatLabie. Ostatni przycisk odpowiedzialny jest za wywołanie Code Composer Studio, w którym mamy cały projekt equalizera wraz z suwakami. Dzięki takiemu rozwiązaniu użytkowanie jest prostsze i nie wymaga zaglądania w pliki projektu.

Instrukcja użytkowania

Aby uruchomić działanie projektu należy otworzyć plik o nazwie Project_EQ, następnie GIU_EQ i wejść w folder Debug



Rys.8 Widok folderu Debug

Menu całego projektu to aplikacja znajdująca się w tym folderze o nazwie GUI_EQ. Po włączeniu menu już swobodnie możemy się poruszać po projekcie. Uruchamianie mskryptów zostało już opisane wyżej(opis menu IV), natomiast uruchamianie samego equalizera jest bardziej złożone. Aby tego dokonać należy kliknąć w przycisk „Run Equalizer in CCS", następnie wybrać opcję „load project" i wybrać plik z rozszerzeniem .pjt w plikach projektu. Gdy tego dokonamy należy skompilować program i w folderze „Debug" equalizera(nie w folderze o tej samej nazwie, jednak w projekcie GUI) wybrać plik z rozszerzeniem .out. Po tym należy już tylko wybrać opcję „connect" w rozwinięciu opcji na pasku narzędzi(u góry okna Code Composer Studio), a następnie włączyć przycisk „run". Program powinien już działać. By dodać przyciski należy przed kompilacją najpierw dodać plik gel(powinien być dodany na początku, gdyż znajduje się w plikach projektu), a następnie na pasku narzedzi wybrać opcję „GEL" i włączyć wszystkie suwaki klikając na ich nazwy w rozszerzeniu przycisku „graphicEQ".

Podsumowanie

Program został napisany tak by był czytelny i łatwy w użytkowaniu. Dzięki pozyskanej wiedzy podczas zajęć z przetwarzania sygnałów cyfrowych, zajęć wprowadzających i wykładów, pozyskano wiedzę na temat środowiska MatLab i jego funkcji niezbędnych do określania danych transmisji, próbek, projektowania filtrów, sprawdzania ich aspektów i wielu innych przydatnych rzeczy, które zostały uwzględnione w danym projekcie. Na początku zaprojektowano wzmacniacz równoważący, który składał się z trzech filtrów: dolnoprzepustowego, górnoprzepustowego i pasmowoprzepustowego. Każdy z nich miał swoją rolę w programie.

- Filtr dolnoprzepustowy miał na celu tłumienie basów do 200Hz. Działanie filtra i tłumienie basów zostało przetestowane i zweryfikowane w laboratorium. Podobny filtr istnieje również do tłumienia niechcianych superpozycji spektów zwanych aliasingiem (opisane w opisie projektu).
- Drugi filtr pasmowoprzepustowy przepuszczał częstotliwości od 200Hz do 2000Hz. Jego celem było tłumienie lub przepuszczanie sygnałów danych na częstotliwości środka, a dokładne parametry filtru mogą być dowolnie zmieniane przez użytkownika.
- Ostatni filtr opisany krótko w podsumowaniu to filtr górnoprzepustowy. Jego wartość graniczna wynosiła od 2000Hz do 4000Hz, czyli połowy wartości częstotliwości próbkowania. Stosując wiedzę o filtrach zdecydowano jakich filtrów i częstotliwości granicznych należy użyć w projekcie, aby spełnić wymagania tworzenia wzmacniacza

równoważącego. Każdy z filtrów został napisany w kodzie Matlabowskim oraz próbki zostały dla nich wygenerowane od specjalnych bibliotek. Gdy zaprojektowano filtry oraz kod Equalizera napisany w języku C, naniesiono poprawki tak, aby dźwięk był idealny oraz każdy rodzaj filtru spełniał w pełni swoje zadanie. Kod opracowano na podstawie kodu zawartego w małym projekcie na temat projektowania filtrów, informacji dostarczonych przez prowadzącego, a także materiałów pomocniczych dostarczonych w ramach kursu bądź znalezionych w internecie. Equalizer, (a dokładniej filtry go tworzące) zaprojektowano w sposób równoległy, co było o wiele łatwiejsze, ponieważ nie trzeba było filtrować każdej próbki osobno przez wszystkie trzy filtry, tylko przez jeden o odpowiedniej częstotliwości, a następnie jako składnik próbki wyjściowej przemnożyć przez 1/3. Kod został opisany krok po kroku w instrukcji, co robi każda dana linijka kodu.
Dzięki danemu projektowi zapoznano się również z zasadą działania suwaków, jak i sposobie ich dodania do projektu przez rozszerzenie gel, które przedstawił nam prowadzący. Zrozumiano ich zadanie, jak je znaleźć oraz zaprojektować aby w łatwy sposób można było operować danymi operacjami, takimi jak regulacja basu(czyli regulacja filtrem dolnoprzepustowym). Ważnym elementem projektu jest również stworzenie okna menu. Okno menu zostało stworzone w celu łatwego poruszania po plikach Equalizera. Zrozumiano zasadę tworzenia menu w programie Visual Studio, a dokładniej funkcji WindowsForms. Dzięki większej chęci rozwinięcia programu postawiono również na dopełnienia programu nie tylko o optymalny i przejrzysty kod napisany w języku C i dokształcania się w dziedzinie programowania oraz poznawania świata cyfrowego przetwarzania sygnałów, ale również ważną rolę w całości danego zadania miało rozwinięcie go pod względem estetyki. Założono odrębność projektu equalizera od innych(wykonywane przez pozostałe grupy) wykonywanych na płytkach w laboratoriach by ten projekt nie był jedynie kodem źródłowym do przeanalizowania, lecz miał również oprawę graficzną, która pozwoli każdemu użytkownikowi na przejrzyste instrukcje dotyczące korzystania z wzmacniacza równoważącego. Każdy z przycisków w naszym Menu(oknie startowym) został krótko sklasyfikowany w opisie programu gdzie wszystko zostało szczegółowo oraz przejrzyście opisane. Projekt jest funkcjonalny, jednak wnioskiem, który można wyciągnąć na przyszłość, jest fakt, iż nie spodziewano się braku kompatybilności plików między nowym a starym systemem operacyjnym. Menu equalizera niestety jak się okazało nie jest obsługiwane przez starsze komputery, jakich używano podczas laboratorium, co wiążę się niestety z brakiem wygodnej i przejrzystej oprawy graficznej, co skutkuje problematycznym poruszaniem się po projekcie, którego chciano uniknąć. Z kolei nowsze komputery napotkają problem z programem wykorzystywanym głównie w całym projekcie, czyli Code Composer Studio. Nowszy system operacyjny nie chce uruchamiać aplikacji, przez co część projektu wymagała pracy zdalnej, a pozostała część pisania kodu w laboratorium. Optymalne działanie programu należało by uzyskać poprzez stworzenie menu w starszych programach, bądź zaprojektowania GUI w innym języku pozwalającym na sposobniejszą interakcję z innymi systemami (na przykład Python, czy JavaSwing). Projektowanie Equalizera pozwala na zapoznanie się z podstawową wiedzą obsługi procesora, programowania go tak, by wykorzystywał dźwięki, jak i inne próbki sygnałów. Wiedza ta jest istotna w cyfrowym przetwarzaniu sygnałów, gdyż zapewnia podstawy do tworzenia projektów komercyjnych sprzedawanych na rynku przez firmy, które następnie trafiają do konsumentów. Nabywcami projektów pokrewnych mogą być choćby bary, kluby czy sceny koncertowe mające zawsze zapotrzebowanie na dobry sprzęt przetwarzający dźwięk.

Literatura

[1] *https://pl.wikipedia.org/wiki/Korektor_graficzny*
[2] *https://www.youtube.com/watch?v=_zSZQo7FY84&ab_channel=ONTMATLAB*
[3] *https://upload.wikimedia.org/wikibooks/pl/6/6a/C.pdf*
[4] *https://pl.wikipedia.org/wiki/Filtr_dolnoprzepustowy*
[5] *https://pl.wikipedia.org/wiki/Filtr_g%C3%B3rnoprzepustowy*
[6] *https://pl.wikipedia.org/wiki/Filtr_%C5%9Brodkowozaporowy*
[7] *https://pl.wikipedia.org/wiki/Filtr_%C5%9Brodkowoprzepustowy*
[8] *https://visualstudio.microsoft.com/pl/*
[9] *https://www.youtube.com/watch?v=v40x3EeLUZE&ab_channel=Obud%C5%BAwsobieGeeka-GeekON*
[10] *https://www.youtube.com/watch?v=_JBGV8jVcbw&ab_channel=MMCSchool*
[11] Chassaing Rulph *Digital Signal Processing and Applcatios with the C6713 and C6416 DSK*
[12] *Wyklad_01_CPS_analizawczasie.pdf*
[13] *Wyklad_02_CPS_probkowanie pdf*
[14] *Wyklad_03_CPS_transformataZ.pdf*
[15] *Wyklad_04_CPS_Analizawidmowa.pdf*
[16] *Wyklad_05_CPS_filtry.pdf*
[17] *Wyklad_06_CPS_algorytmy.pdf*
[18] Bejmert Daniel *Wykład z MIASC_filtry.pdf ~*
[19] *https://pl.wikipedia.org/wiki/Aliasing_(przetwarzanie_sygna%C5%82%C3%B3w)*
[20] *https://edu.pjwstk.edu.pl/wyklady/poj/scb/PrgGui1/PrgGui1.html*
[21] *https://pl.wikipedia.org/wiki/Cyfrowe_przetwarzanie_sygna%C5%82%C3%B3w*
[22] Zieliński T., *Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań*
[23] *https://www.ti.com/tool/CCSTUDIO*
[24] *https://www.youtube.com/watch?v=zMKX5Ci3vRg&ab_channel=drselim*
[25] *https://pl.wikipedia.org/wiki/Transmitancja_operatorowa*
[26] *https://sound.eti.pg.gda.pl/~greg/dsp/07-SpecjalneFiltry.html*
[27] *https://estradaistudio.pl/technologia/30777-wszystko-o-filtrach*

Autorzy:
Bartosz Golis, 259833, Politechnika Wrocławska Email: 259833@student.pwr.edu.pl

Jakub Janiszewski, 259831, Politechnika Wrocławska, Email: 259831@student.pwr.edu.pl