|  |
| --- |
| agh_nzw_s_pl_1w_wbr_rgb_150ppi |
| **WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI**  INSTYTUT ELEKTRONIKI |
| **PROJEKT DYPLOMOWY** |
| **Aplikacja do analizy częstotliwościowej sygnału** |
| *Application for the frequency signal analysis* |
| Autor: **Dominik Duchnik**  Kierunek studiów: Elektronika  Opiekun pracy: dr inż. Rafał Frączek |
| Kraków, 2024 |

Spis treści

[1. Wstęp 3](#_Toc158419888)

[1.1. Wprowadzenie 3](#_Toc158419889)

[1.2. Geneza 3](#_Toc158419890)

[1.3. Cel pracy 4](#_Toc158419891)

[1.4. Przegląd istniejących rozwiązań 4](#_Toc158419892)

[1.4.1. Wolfram Alpha 4](#_Toc158419893)

[1.4.2. Audacity 6](#_Toc158419894)

[2. Analiza częstotliwościowa sygnałów jednowymiarowych 9](#_Toc158419895)

[2.1. Próbkowanie sygnału 9](#_Toc158419896)

[2.1.1. Twierdzenie o próbkowaniu 9](#_Toc158419897)

[2.1.2. Próbkowanie krytyczne 10](#_Toc158419898)

[2.2. Dyskretne przekształcenie Fouriera 11](#_Toc158419899)

[2.2.1. Przeciek widma 11](#_Toc158419900)

[2.2.2. Efekt Gibbs’a 11](#_Toc158419901)

[2.3. Dyskretne przekształcenie kosinusowe 11](#_Toc158419902)

[3. Interfejs użytkownika 12](#_Toc158419903)

[3.1. Sygnał jednowymiarowy 12](#_Toc158419904)

[3.1.1. Wybór parametrów 13](#_Toc158419905)

[3.1.2. Przyciski 14](#_Toc158419906)

[3.1.3. Opcje wykresu 16](#_Toc158419907)

[3.2. Sygnał dwuwymiarowy 18](#_Toc158419908)

[4. Projekt 19](#_Toc158419909)

[4.1. Wymagania 19](#_Toc158419910)

[4.2. Zasady SOLID 19](#_Toc158419911)

[4.2.1. Single Responsibility Principle 19](#_Toc158419912)

[4.2.2. Open/Closed Principle 19](#_Toc158419913)

[4.2.3. Liskov Substitution Principle 19](#_Toc158419914)

[4.2.4. Interface Segregation 19](#_Toc158419915)

[4.2.5. Dependency Inversion 19](#_Toc158419916)

[4.3. Wzorzec projektowy Command 20](#_Toc158419917)

[4.4. Biblioteki 20](#_Toc158419918)

[4.4.1. NumPy 20](#_Toc158419919)

[4.4.2. SciPy 20](#_Toc158419920)

[4.4.3. Matplotlib 20](#_Toc158419921)

[4.4.4. PyQt5 20](#_Toc158419922)

[5. Implementacja 21](#_Toc158419923)

[5.1. Język programowania Python 21](#_Toc158419924)

[6. Podsumowanie 22](#_Toc158419925)

[6.1. Podsumowanie pracy 22](#_Toc158419926)

[6.2. Plany rozwoju 22](#_Toc158419927)

[Bibliografia 23](#_Toc158419928)

# Wstęp

## Wprowadzenie

Cyfrowe przetwarzanie sygnałów jest jedną z najważniejszych gałęzi współczesnej inżynierii. Obszary takie jak telekomunikacja, medycyna, akustyka zostały zrewolucjonizowane przez własne, skomplikowane matematycznie algorytmy dotyczące przetwarzania sygnałów. Głównym jego celem jest efektywne manipulowanie sygnałami wykorzystując algorytmy numeryczne, które korzystają z cyfrowych układów elektronicznych. W przeciwieństwie do sygnałów analogowych, które są ciągłe w czasie, sygnał cyfrowy charakteryzuje się dyskretyzacją w dziedzinie czasu – jest reprezentowany przez skończony ciąg liczb. Sygnały w postaci cyfrowej posiadają wiele zalet takich jak odporność na zakłócenia i straty informacji w porównaniu do sygnałów analogowych co przekłada się na poprawę jakości i niezawodności ich przetwarzania. Ponadto dane w postaci cyfrowej są łatwiejsze w przechowywaniu i przesyłaniu. Jednym z kluczowych zagadnień w aspekcie cyfrowego przetwarzania sygnałów jest analiza częstotliwościowa, zwana również analizą fourierowską. Jest to matematyczne narzędzie umożliwiające przekształcenie sygnału z dziedziny czasu do częstotliwości co znacznie ułatwia analizę sygnału i znajduje swoje zastosowanie m.in. w kompresji danych. Poza transformacją Fouriera w praktyce wykorzystywana jest również transformacja cosinusowa, która przekształca sygnał na sumę funkcji cosinus o współczynnikach rzeczywistych. Swoje zastosowanie znalazła w szczególności w kompresji obrazów JPEG.

## Geneza

Genezą powstania aplikacji, która będzie opisywana w pracy inżynierskiej był powszechnie znany problem zrozumienia przez uczniów oraz studentów podstawowych zagadnień związanych z przetwarzaniem sygnałów. Z obserwacji przeprowadzonych podczas studiów można wywnioskować, iż abstrakcja oraz skomplikowany aparat matematyczny analizy sygnałów stanowią dużą przeszkodę w zrozumieniu materiału, który ma duże znaczenie w elektronice. Interaktywne narzędzia do wizualizacji mają potencjał do znacznego ułatwiania procesu nauki poprzez możliwość eksperymentowania z różnymi parametrami oraz obserwowania natychmiastowych efektów. Skłoniło mnie to do opracowania innowacyjnego narzędzia edukacyjnego w postaci aplikacji, które ma służyć do ilustracji procesów zachodzących w obszarze jakim jest analiza częstotliwościowa.

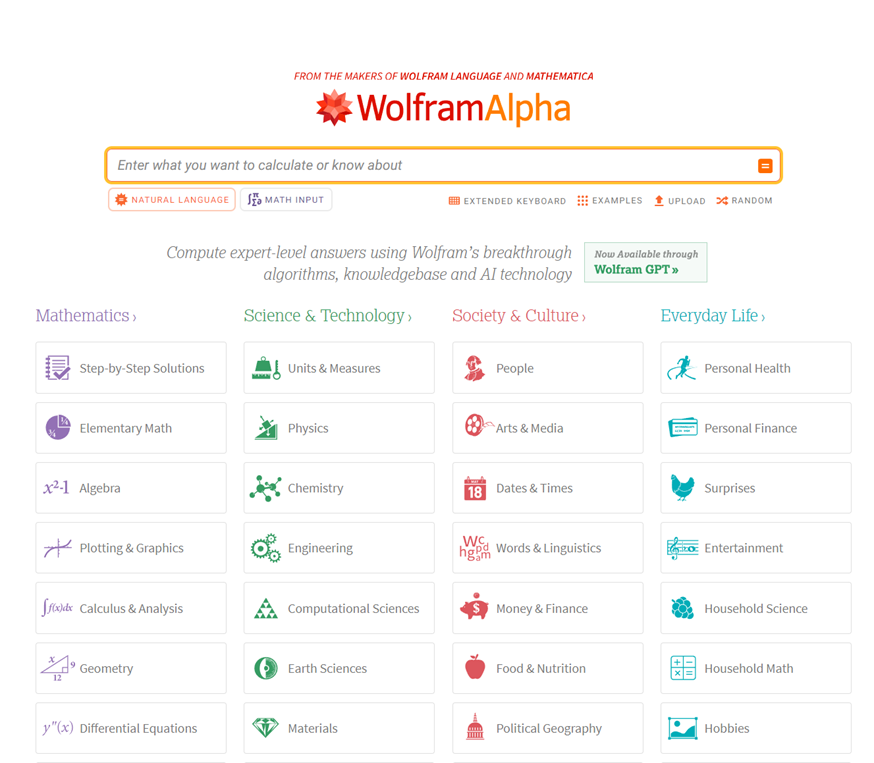
## Cel pracy

Celem pracy jest opracowanie, zaprojektowanie oraz implementacja aplikacji desktopowej, którą można uruchomić na każdym komputerze z zainstalowanym środowiskiem języka programowania Python. Zakres prac obejmuje cały proces od projektowania interfejsu użytkownika po implementację algorytmów analizy sygnałów. Głównym elementem aplikacji jest graficzny interfejs użytkownika stworzony za pomocą biblioteki PyQt5. Elementy interaktywne, takie jak przyciski czy pola tekstowe, pozwalają na dynamiczną modyfikacje wybranych parametrów w czasie rzeczywistym, co pozwala na lepsze zrozumienie wpływu różnych ustawień na wynik. Aplikacja zawiera implementację algorytmów teorii sygnałów obejmujących obszar analizy częstotliwościowej. W tym celu zostały wykorzystane popularne biblioteki, takie jak NumPy w celu manipulacji danymi, SciPy, która dostarcza gotowe algorytmy do obliczania transformat oraz Matplotlib służąca do wizualizacji wyników analizy. Uniwersalność oraz prostota obsługi sprawią, że aplikacja będzie cennym wsparciem w procesie nauczania.

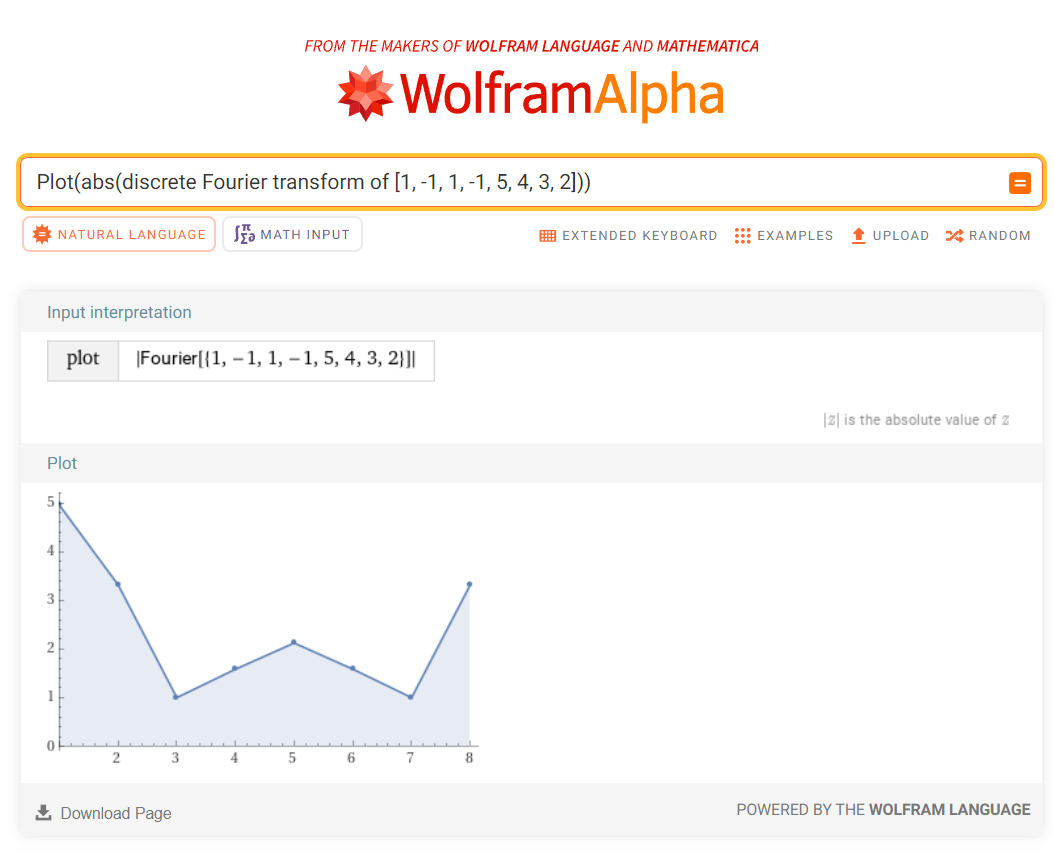
## Przegląd istniejących rozwiązań

### Wolfram Alpha

Wolfram Alpha(Rys 1.1, Rys 1.2) to program obliczeniowy służący do analizy matematycznej oraz obliczeń numerycznych. Dostępny jest zarówno jako strona internetowa oraz aplikacja mobilna.



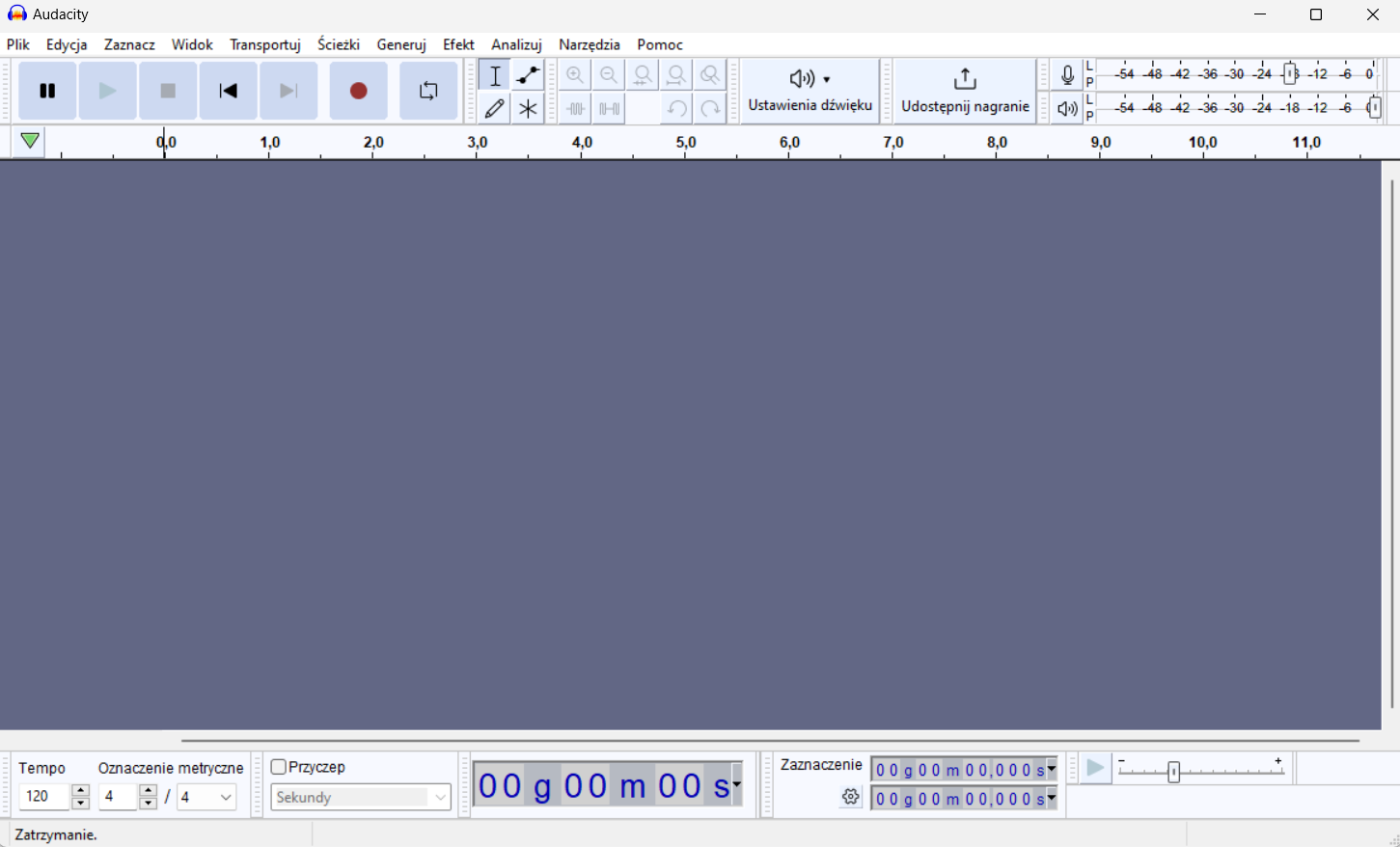
*Rys 1.1 Widok główny strony Wolfram Alpha*

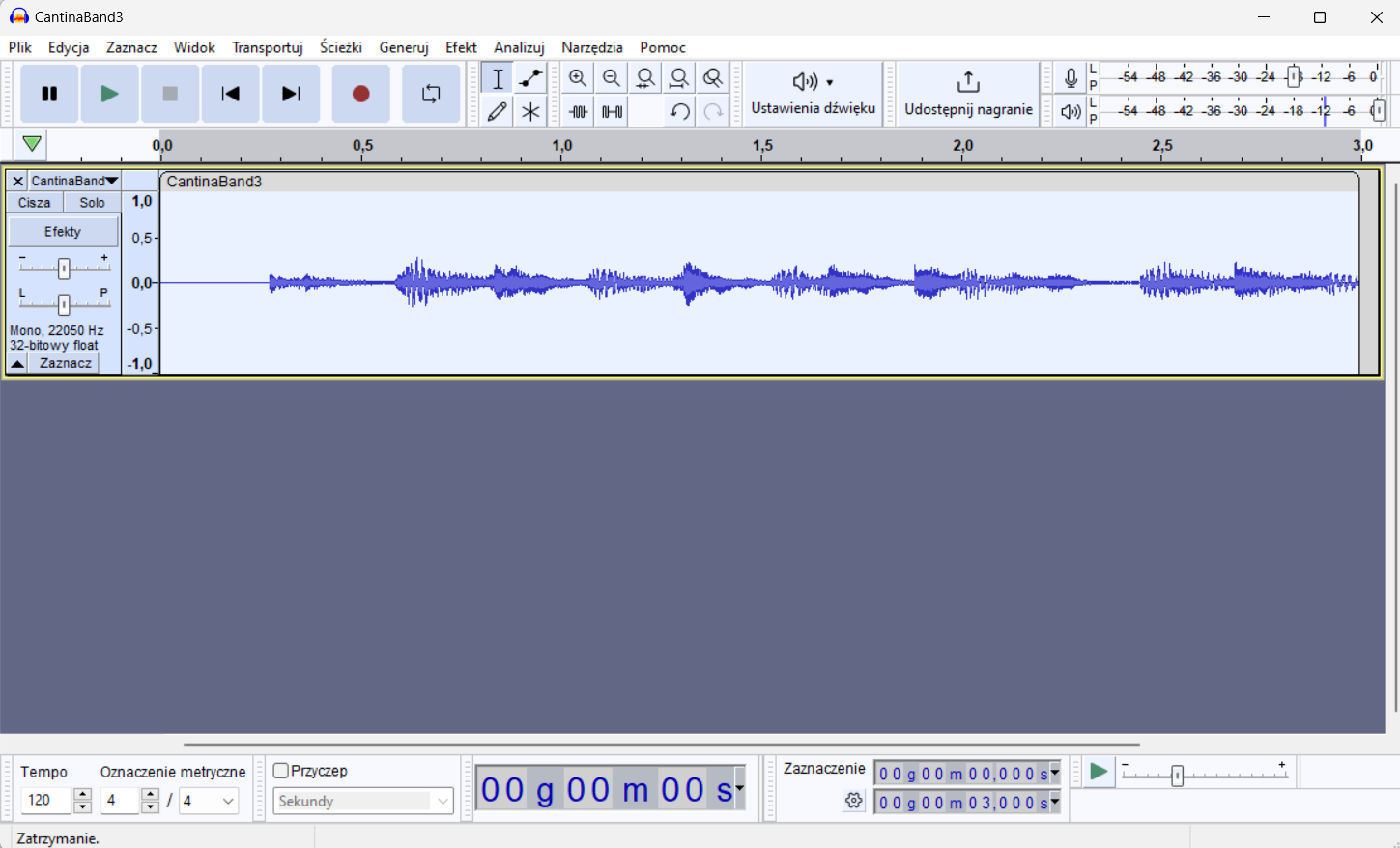


*Rys 1.2 Przykładowy wykres DFT*

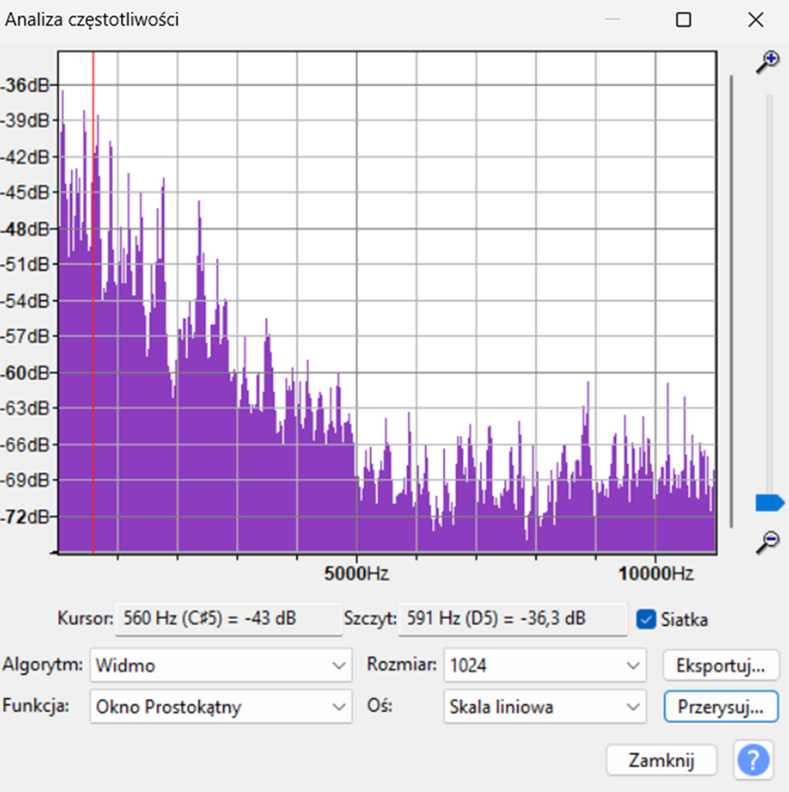
### Audacity

Aplikacja Audacity(Rys 1.3, Rys 1.4) to darmowe oprogramowanie służące do edycji oraz nagrywania dźwięku. Jest dostępny dla użytkowników systemów operacyjnych takich jak Window, macOS oraz Linux. Oferuje on szereg funkcji takich jak analiza widma sygnału(Rys 1.5).

*Rys. 1.3 Główny panel aplikacji*



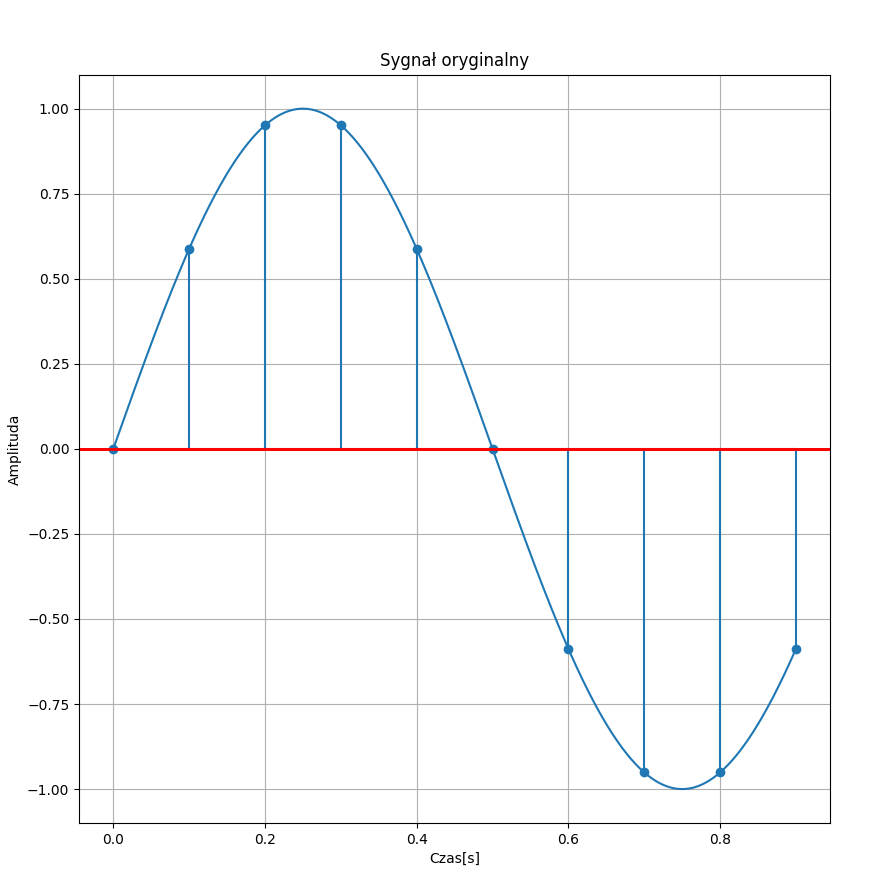
*Rys 1.4 Przykładowa analiza sygnału dźwiękowego*



*Rys 1.5 Przykładowa transformata sygnału dźwiękowego*

# Analiza częstotliwościowa sygnałów jednowymiarowych

## Próbkowanie sygnału

Próbkowanie to operacja, która polega na pobieraniu wartości sygnału w określonych momentach czasowych – tak zwanych próbek. Odstęp czasowy pomiędzy próbkami nazywany jest okresem próbkowania, zaś jego odwrotność częstotliwością próbkowania. Celem takiego procesu jest przeniesienie sygnału z dziedziny analogowej do cyfrowej. 

*Rys 2.1 Sygnał spróbkowany*

### Twierdzenie o próbkowaniu

Twierdzenie o próbkowaniu(również twierdzenie Nyquista) to jedna z fundamentalnych zależności w cyfrowym przetwarzaniu sygnałów wyrażona wzorem:

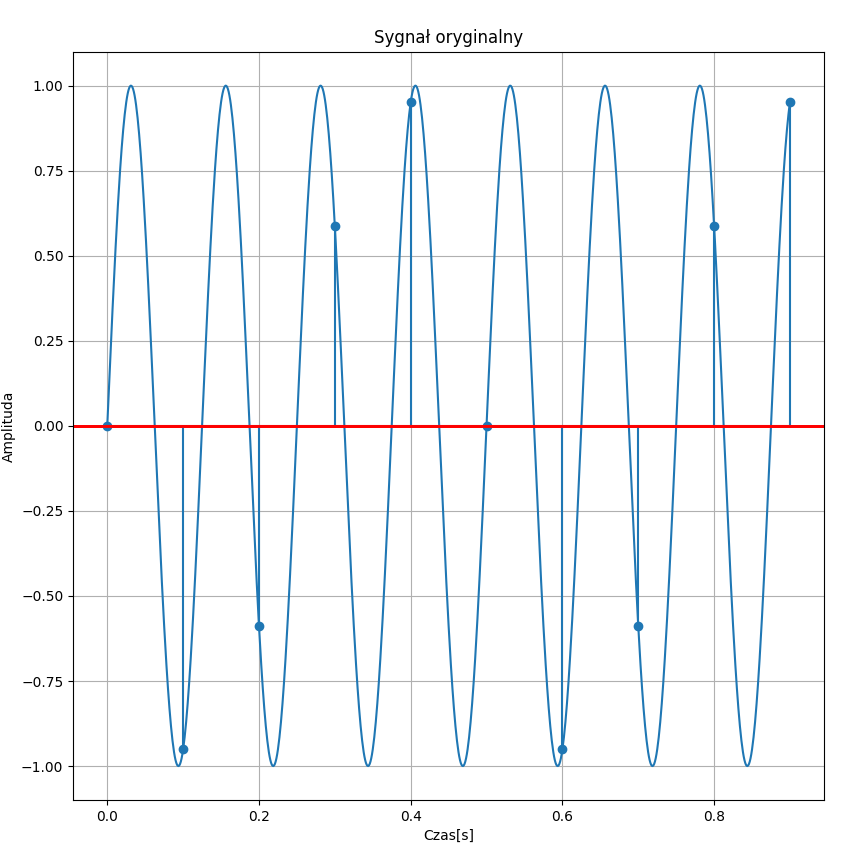
, (1)

gdzie:

– częstotliwość próbkowania,

– najwyższa częstotliwość w sygnale.

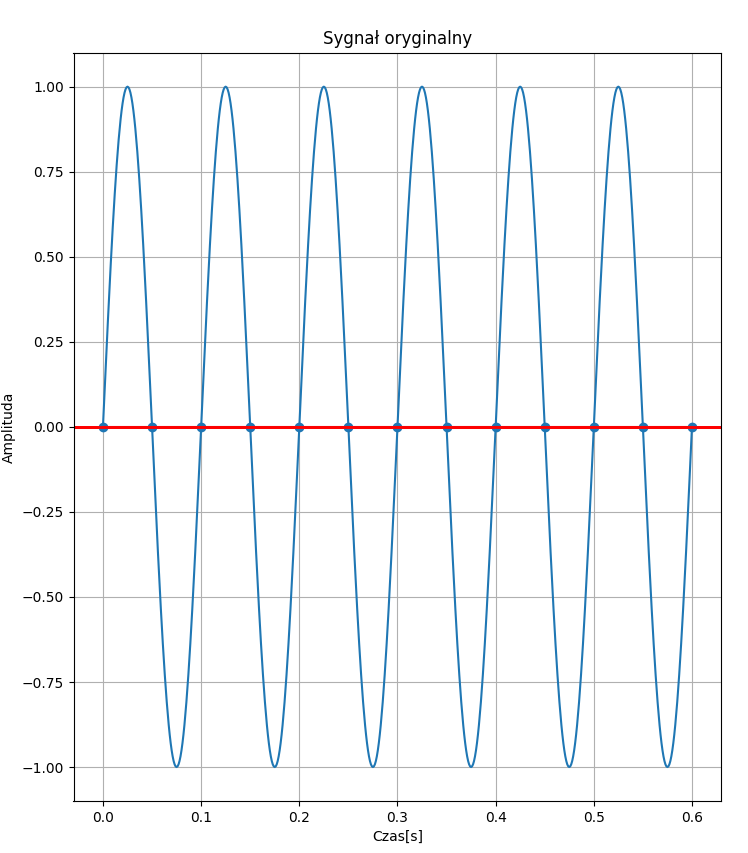
Twierdzenie to mówi o tym, że aby poprawnie zrekonstruować sygnał analogowy z jego próbek, próbkowanie należy przeprowadzić z częstotliwością próbkowania będącą przynajmniej dwukrotnością najwyższej częstotliwości harmonicznej sygnału. W przeciwnym wypadku wystąpi efekt aliasingu, który spowoduje, że częstotliwości wyższe niż częstotliwość Nyquista[[1]](#footnote-1) zostaną fałszywie zinterpretowane jako częstotliwości niższe niż w rzeczywistości, co prowadzi do zniekształceń sygnału.



*Rys 2.2 Sygnał spróbkowany bez spełnienia twierdzenia o próbkowaniu*

### Próbkowanie krytyczne

Próbkowaniem krytycznym nazywamy przypadek, gdy próbkujemy z częstotliwością próbkowania równą dokładnie dwukrotności częstotliwości harmonicznej sygnału. W tym wypadku, mimo spełnienia warunków twierdzenia Nyquista, nie zawsze jesteśmy w stanie poprawnie odtworzyć sygnał z jego próbek. Rozważając przypadek sygnału sinusoidalnego o częstotliwości równej 10Hz, którego próbki zostały pobrane z częstotliwością próbkowania równą 20Hz(Rys 2.3) możemy zauważyć, że widoczne są w miejscach, gdy wartość sygnału wynosi 0.



*Rys 2.3 Wizualizacja próbkowania krytycznego*

W tym przypadku poprawna rekonstrukcja sygnału jest niemożliwa, pomimo spełnienia kryteriów twierdzenia o próbkowaniu.

## Dyskretne przekształcenie Fouriera

Po poprawnie przeprowadzonym procesie próbkowania, kolejnym krokiem w analizie częstotliwościowej sygnału jest poddanie go transformacji Fouriera. Przekształcenie Fouriera to koncept matematyczny, który pozwala na przedstawienie dowolnej funkcji jako sumę składowych sinusoid o różnych częstotliwościach. Jego nazwa pochodzi od francuskiego matematyka oraz fizyka Jean Baptiste Joseph’a Fouriera, który pracując nad propagacją ciepła odkrył, że każdy ciągły w czasie, okresowy sygnał, może zostać przedstawiony jako suma dobranych fal sinusoidalnych. Dla sygnałów w postaci cyfrowej czyli takich poddanych procesowi próbkowania możemy zastosować dyskretną wersję przekształcenia.

### Własności DFT[[2]](#footnote-2)

### Przeciek widma

### Efekt Gibbs’a

### Zastosowanie DFT

## Dyskretne przekształcenie kosinusowe

### Własności DCT

### Zastosowanie DCT[[3]](#footnote-3)

# Interfejs użytkownika

## Sygnał jednowymiarowy

*Rys 2.1 Zakładka dla sygnałów jednowymiarowych*

Po uruchomieniu aplikacji(Rys 2.1) pierwszą rzeczą, którą zobaczymy będzie zakładka stworzona do analizy sygnałów jednowymiarowych. Składa się ona z panelu do wyboru parametrów sygnału oraz okna z miejscem, w którym rysowany będzie wykres stworzonego sygnału. W górnej części okna znajdują się zakładki przełączające pomiędzy sygnałem jednowymiarowym a dwuwymiarowym.

### Wybór parametrów

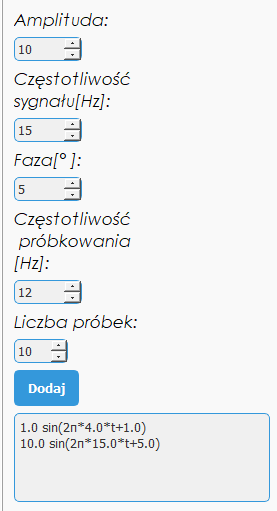
*Rys. 2.2 Panel wyboru parametrów sygnału*

Kluczowym elementem w interakcji z użytkownikiem aplikacji jest panel(Rys 2.2) z elementami, które można na bieżąco edytować. Składa się on z pól tekstowych, które następnie przez program zamieniane są na liczbę o typie *float* lub *int* takich jak:

* Amplituda – największa wartość danej składowej sygnału
* Częstotliwość[Hz] – ilość okresów w czasie jednej sekundy trwania sygnału
* Faza[°] – przesunięcie sygnału względem osi OX
* Częstotliwość próbkowania[Hz] – liczba próbek pobranych z sygnału w czasie jednej sekundy
* Liczba próbek – ilość próbek danego sygnału, która ma zostać narysowana na wykresie
* Krok czasowy[s] – odstęp pomiędzy kolejnymi wartościami na osi czasu w przypadku sygnału ciągłego
* Numer składowej – po dodaniu składowej sygnału, korzystając z tego pola możemy wybrać, na której składowej chcemy dokonywać operacji takich jak zmiana amplitudy czy częstotliwości i obserwować zmiany w czasie rzeczywistym lub też ją usunąć.

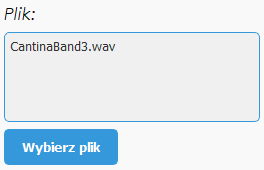
Powyższe pola zawierają również przyciski mające na celu płynną zmianę parametrów, która od razu wyświetli jak wpływa dana wielkość na wynik analizy.

### Przyciski

Kolejną rzeczą, która wchodzi w skład panelu są przyciski, każdy z nich ma inne funkcje, takie jak:

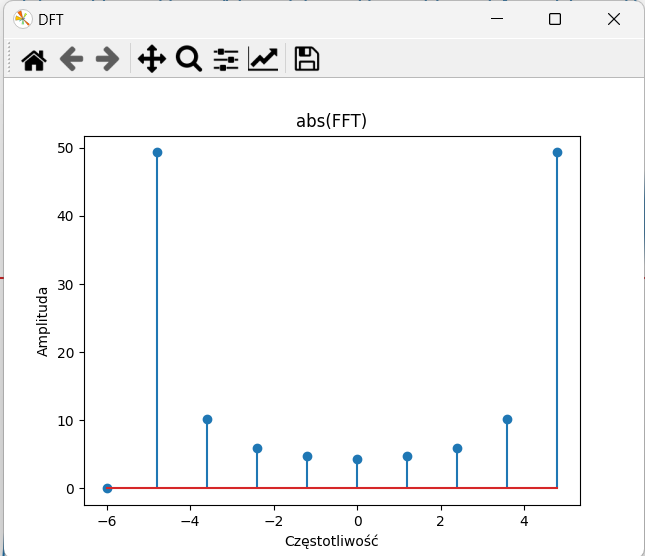
*Rys 2.2 Pole z sygnałem*

* Przycisk „dodaj” ma za zadanie odczytać dane z wszystkich pól tekstowych i na ich podstawie dodaje je do pozostałych składowych tworząc jeden sygnał(Rys 2.2).
* Przycisk „usuń” służy do odczytania wartości numeru składowej oraz następnie jej usunięciu z sygnału.



*Rys 2.3 Pole ze ścieżką do pliku*

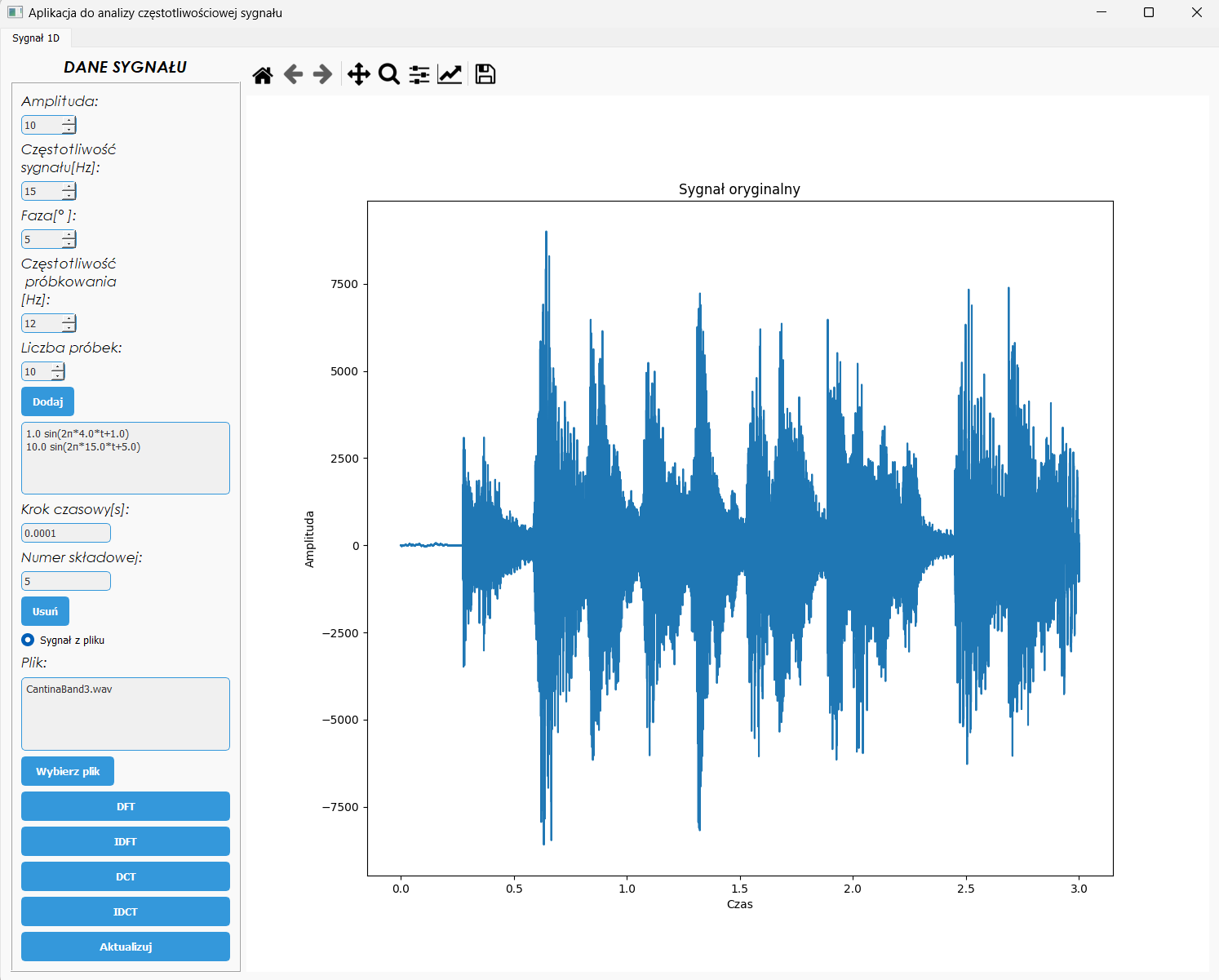
* Przycisk „wybierz plik” otwiera okno z wyborem pliku dźwiękowego za pomocą którego możemy przekazać aplikacji ścieżkę do pliku w formacie *wav.* Ścieżka zostanie przedstawiona w polu tekstowym znajdującym się pod przyciskiem(Rys 2.3).



*Rys 2.4 Okno z wykresem DFT*

* Przyciski „DFT”, „IDFT”, „DCT”,”IDCT” służą kolejno do otwierania okna na którym wykreślane są wykresy odpowiednich transformat(Rys 2.4)
* Przycisk „Aktualizuj” służy do ręcznej aktualizacji wszystkich wykresów. Pomimo tego, że zmiana parametrów automatycznie zaktualizuje wszystkie okna, po zamknięciu i ponownym otworzeniu, początkowo ujrzymy puste okno. W tej sytuacji, jeżeli nie chcemy ingerować w parametry, możemy skorzystać z opisanego przycisku.

Ostatnim elementem tej części interfejsu jest możliwość zaznaczenia pola „*Sygnał z pliku”*. Opcja ta pozwala na zignorowanie sygnału zgromadzonego przez aplikacje na rzecz odczytania danych z poprzednio wybranego pliku dźwiękowego. Następnie po wciśnięciu przycisku „*Aktualizuj”* we wszystkich oknach ukażą się nam wyniki analizy wybranego pliku(Rys 2.5)

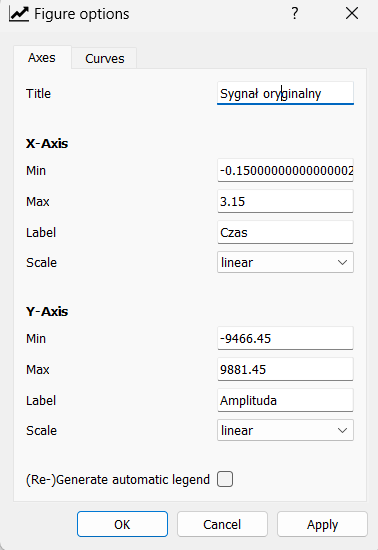


*Rys 2.5 Wykres sygnału odczytanego z pliku CantinaBand3.wav*

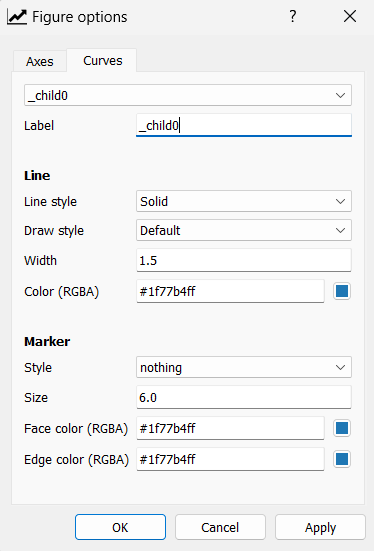
### Opcje wykresu

Każdy wykres niezależnie od tego, czy jest zintegrowany z głównym oknem lub jako osobne okno posiada ustawienia dostarczone przez bibliotekę Matplotlib, takie jak:

* Możliwość zapisu do pliku



*Rys 2.6 Edycja parametrów osi*



*Rys 2.7 Edycja parametrów krzywych*

* Edycja tytułu wykresu oraz ustawień osi OX i OY(Rys 2.6)
* Zmiana stylu wyświetlania wykresu(Rys 2.7)
* Powiększanie lub pomniejszanie wykresu w wybranym miejscu, po którym możemy powrócić do poprzedniego stanu.

## Sygnał dwuwymiarowy

# Projekt

## Wymagania

## Zasady SOLID

W miarę rozwoju aplikacji, stopień komplikacji kodu zwiększa się. Aby utrzymać czytelność należy stosować dobre praktyki programowania obiektowego. Jedną z takich praktyk są zasady SOLID[1].

### Single Responsibility Principle

Zasada pojedynczej odpowiedzialności mówi o tym, że każda klasa w kodzie programu powinna być odpowiedzialna za jedną, konkretną rzecz. Zwiększa to ilość klas w kodzie, jednak stosowanie tej zasady znacznie upraszcza rozumienie kodu.

### Open/Closed Principle

Zasada otwarte/zamknięte oznacza, że klasa powinna być otwarta na rozbudowę, ale zamknięta na modyfikacje. Dodanie nowych funkcjonalności powinno być możliwe bez edycji kodu, ale poprzez dodawanie nowych fragmentów. W implementacji tej zasady często używany jest polimorfizm. Jest to jeden z fundamentów programowania obiektowego, który umożliwia rozbudowę klasy w przyszłości. [przykład klasa Signal, opisać?]

### Liskov Substitution Principle

Nazwa tej zasady pochodzi od amerykańskiej programistki Barbary Liskov. Polega ona na tym, że musi zostać zachowana zgodność interfejsu oraz wszystkich metod w klasie. Przykład: klasa(interfejs) *Pojazd*, która posiada metodę *uruchom\_silnik()*, w takim wypadku jeżeli pojazdem będzie rower, ta metoda będzie bezużyteczna – niespełniona jest zasada podstawienia.

### Interface Segregation

Zasada segregacji interfejsów to najprostsza w zrozumieniu zasada, mówi o tym, że nie należy tworzyć interfejsów z niepotrzebnymi metodami – powinny być one jak najmniejsze i spełniać konkretne funkcje.

### Dependency Inversion

Zasada odwrócenia zależności polega na tym, że wysokopoziomowe moduły nie powinny zależeć bezpośrednio od modułów niskopoziomowych. Obydwa powinny zależeć od interfejsów lub klas abstrakcyjnych.

## Wzorzec projektowy Command

Polecenie(Command) jest wzorcem projektowym z kategorii wzorców behawioralnych.

## Biblioteki

### NumPy

### SciPy

### Matplotlib

### PyQt5

Jako narzędzie służące do zaprojektowania graficznego interfejsu użytkownika(GUI) wykorzystana została biblioteka PyQt5. Jest to zbiór modułów umożliwiający integrację interfejsu graficznego z aplikacjami w języku Python. Oparta jest ona na technologii Qt wykorzystywanej do projektowania aplikacji desktopowych oraz mobilnych. Swoją popularność zawdzięcza wszechstronności oraz prostocie użytkowania. Niewątpliwym atutem jest integracja z narzędziem Qt Designer, który umożliwia w szybki sposób zaprojektowanie całego interfejsu w sposób graficzny, bez konieczności tworzenia kodu w sposób manualny.

# Implementacja

## Język programowania Python

Język programowania Python jest jednym z najbardziej wszechstronnych oraz powszechnie używanych narzędzi programistycznych. Prostota, czytelność kodu oraz ogromna społeczność programistyczna sprawiają, że jest popularnym wyborem wśród wielu projektów. Jest on językiem jest wysokopoziomowym i wieloparadygmatowym językiem programowania. Posiada bogatą kolekcję bibliotek, które znajdują zastosowanie niemal w każdej dziedzinie. Jednym z kluczowych atutów języka Python jest dynamiczne typowanie, co oznacza, że nie musimy deklarować typu zmiennej, sprawiając, że kod staje się bardziej elastyczny. Niewątpliwie jego zaletą jest również automatyczne zarządzanie pamięcią, które zwalnia programistę z konieczności ręcznego jej alokowania, co zwiększa wygodę oraz skraca czas poświęcony rozwojowi aplikacji.

# Podsumowanie

## Podsumowanie pracy

## Plany rozwoju

# Bibliografia

1. <https://forreya.medium.com/the-solid-principles-writing-scalable-maintainable-code-13040ada3bca> dostęp 10.01.2024
2. <https://thepythoncodingbook.com/2021/08/30/2d-fourier-transform-in-python-and-fourier-synthesis-of-images/> dostęp 15.01.2024
3. Zieliński T., *Cyfrowe Przetwarzanie Sygnałów – od teorii do zastosowań*, Warszawa, Wydawnictwo Komunikacji i Łączności, 2005
4. Wzorce projektowe - <https://www.tutorialspoint.com/design_pattern/command_pattern.htm>
5. Steven W. Smith, *The Scientist and Engineer’s Guide to Digital Signal Processing*, California, California Technical Publishing, 1999

1. Częstotliwość Nyquista – połowa częstotliwości próbkowania [↑](#footnote-ref-1)
2. DFT (ang. discrete fourier transform) – dyskretna transformata fouriera [↑](#footnote-ref-2)
3. DCT (ang. discrete cosine transform) – dyskretna transformata kosinusowa [↑](#footnote-ref-3)