AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

Biblioteka do obsługi filtrów cyfrowych w języku C++

z przedmiotu

Języki Programowania Obiektowego

Elektronika i Telekomunikacja, Rok III

Michał Dobrzycki

Środa, 9:45

prowadzący: mgr inż. Jakub Zimnol

03.01.2026

Spis treści

1. Opis projektu

Projekt ma na celu stworzenie biblioteki w języku C++, która umożliwia implementację i obsługę filtrów cyfrowych. Biblioteka zawiera klasy i funkcje do tworzenia, konfigurowania oraz stosowania filtrów takich jak FIR i IIR. Dodatkowo zaimplementowano klasy Signal oraz Window, które ułatwiają operacje na sygnałach. Przykład użycia biblioteki znajduje się w pliku `main.cpp`.

1.1. Zastosowania

Biblioteka znajduje zastosowanie w:

- Przetwarzaniu sygnałów audio (equalizery, redukcja szumów)
- Systemach embedded wymagających filtracji sygnałów w czasie rzeczywistym
- Analizie sygnałów telekomunikacyjnych
- Aplikacjach pomiarowych i akwizycji danych

2. Struktura projektu

Namespace: `md`

Hierarchia klas:

- **SignalProcessor**: Abstrakcyjna klasa bazowa dla procesorów sygnałów.
  - **Filter**: Klasa bazowa dla filtrów cyfrowych (dziedziczy po SignalProcessor).
    - **FirFilter**: Implementacja filtru o skończonej odpowiedzi impulsowej.
    - **IirFilter**: Implementacja filtru o nieskończonej odpowiedzi impulsowej.
  - **Window**: Klasa do tworzenia funkcji okienkowych (Hamming, Hann, Blackman).
- **Signal**: Klasa reprezentująca sygnał cyfrowy z operacjami I/O i obliczeniami parametrów.

3. Opis klas

3.1. SignalProcessor

Abstrakcyjna klasa bazowa dla wszystkich procesorów sygnałów w bibliotece. Przechowuje tablicę współczynników przetwarzania i definiuje wspólny interfejs dla klas potomnych. Klasa jest szablonowa - parametryzowana typem danych oraz rozmiarem. Implementuje walidację w czasie kompilacji.

3.2. Filter

Abstrakcyjna klasa bazowa dla wszystkich filtrów cyfrowych, dziedzicząca po `SignalProcessor`. Rozszerza funkcjonalność klasy bazowej o metody specyficzne dla filtrów.

3.3. FirFilter

Klasa implementująca filtry FIR. Klasa udostępnia metody do projektowania filtrów dolnoprzepustowych, górnoprzepustowych oraz pasmowoprzepustowych.

3.4. IirFilter

Klasa implementująca filtry IIR (Infinite Impulse Response). Implementacja wykorzystuje dwa zestawy współczynników: `b` (współczynniki wejścia) oraz `a` (współczynniki sprzężenia zwrotnego).

3.5. Window
Klasa do tworzenia i stosowania funkcji okienkowych w analizie sygnałów.

3.6. Signal
Klasa reprezentująca sygnał cyfrowy jako tablicę próbek. Wszystkie próbki są przechowywane jako tablica `std::array`, co zapewnia bezpieczeństwo typów i brak alokacji dynamicznej.

4. Cechy implementacji

4.1. Programowanie generyczne
Wszystkie klasy wykorzystują szablony C++, co pozwala na:
- Elastyczny dobór typu danych (float, double, long double) w zależności od wymaganej precyzji lub zasobów MCU
- Określenie rozmiaru filtru/sygnału w czasie kompilacji

4.2. Polimorfizm
Biblioteka wykorzystuje polimorfizm:
- Klasa `Filter` może być używana przez wskaźnik bazowy, co umożliwia dynamiczną zmianę typu filtru w czasie wykonania
- Czysto wirtualne metody wymuszają implementację kluczowej funkcjonalności w klasach pochodnych
- Wirtualny destruktor zapewnia poprawne zwalnianie zasobów

4.3. Bezpieczeństwo typów
Kompilator wymusza:
- Używanie wyłącznie typów zmiennoprzecinkowych (`static_assert`)
- Niezerowy rozmiar tablicy współczynników
- Zgodność rozmiarów podczas przetwarzania sygnałów

5. Kompilacja i uruchomienie

5.1. Wymagania
- Kompilator C++ z obsługą C++17 (np. GCC ≥ 7.0)
- CMake ≥ 3.10

5.2. Kompilacja
W głównym katalogu projektu:

```
mkdir build
cd build
cmake ..
cmake --build .
```

Na Windows z MinGW:

```
cmake .. -G "MinGW Makefiles"
```

5.3. Uruchomienie
```
.\DSP_App.exe
```

Projekt automatycznie kopiuje katalog `test_data/` z przykładowymi danymi do katalogu `build/`.