Akademia Górniczo-Hutnicza w Krakowie
Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Projekt zaliczeniowy

Analizator EKG

Autorzy:
Mateusz Baran
Krzysztof Bębenek
Bartłomiej Bułat
Szczepan Czaicki
Tomasz Drzewiecki
Krzysztof Farganus
Łukasz Jaromi
Mateusz Krasucki
Łukasz Krzyżek
Łukasz Kutrzuba
Weronika Łabaj
Paweł Maślanka
Piotr Matuszkiewicz
Norbert Pabian
Łukasz Pękala
Krzysztof Piekutowski
Grzegorz Pietrzyk
Łukasz Podolski
Mikołaj Rzepka
Agata Sitnik
Leszek Sosnowski
Aleksander Steliga
Mateusz Ślażyński
Łukasz Zieńkowski

Opiekun:
mgr inż. Tomasz Pięciak

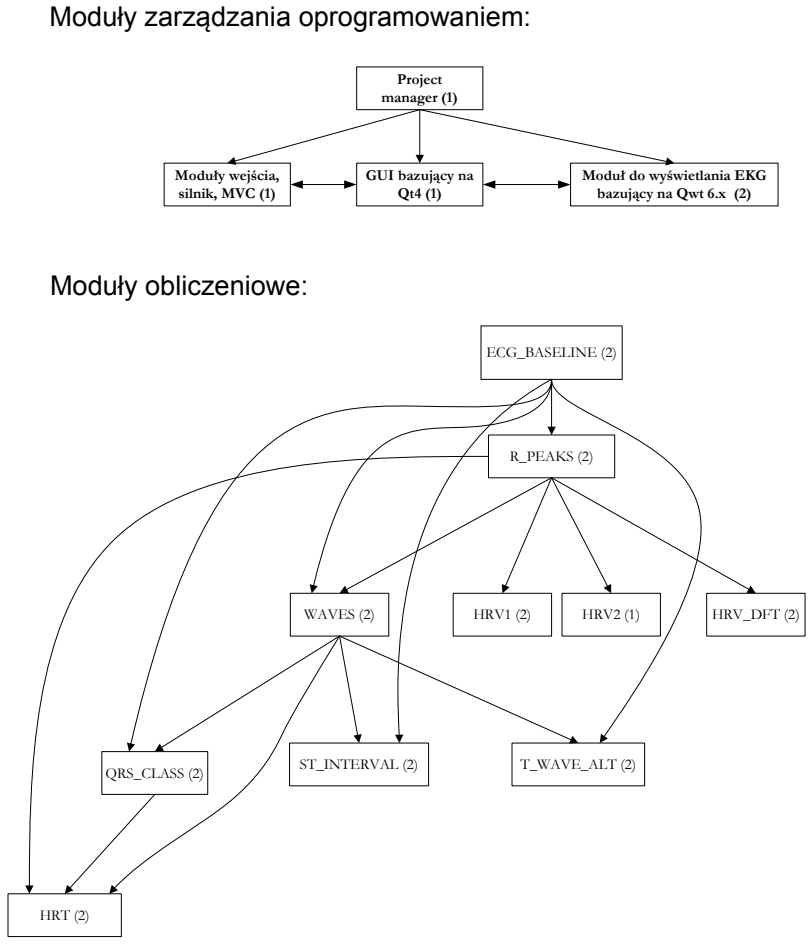
29 stycznia 2013

Spis treści

1 Specyfikacja zadania

Celem projektu jest stworzenie zintegrowanego systemu pozwalającego na przeglądanie i automatyczną analizę sygnału EKG. Sygnał dostarczany jest w formie cyfrowej w standardzie wykorzystywanym w MIT–BIH Arrhythmia Database. Różne etapy przetwarzania, takie jak usuwanie linii bazowej, detekcja załamków R czy klasyfikacja zespołów QRS wykonywana jest przez różne zespoły (szczegółowe opisy specyfikacji modułów: 3), których praca składa się na jeden program.

Moduły przetwarzania integrowane i uzupełniane są modułami kontrolującymi przepływ danych i odpowiadającymi za komunikację z użytkownikiem. Wzajemne zależności pomiędzy modułami prezentuje rys. 1.



Rysunek 1: Zależności pomiędzy modułami projektu.

2 Specyfikacja techniczna rozwiązania

2.1 Wykorzystane narzędzia

Podczas realizacji projektu wykorzystywane były różne narzędzia do tworzenia i prototypowania rozwiązań. Wstępne projekty przygotowywane były w programie Matlab, zaś ostateczny kod powstawał w języku C++ (standard '03 z elementami standardu C++11 obsługiwanymi przez wspierane kompilatory). Minimalne wymagania kompilacji projektu są następujące:

- Jeden z kompilatorów:
  - Microsoft Visual Studio 2010
  - GCC 4.5
- Biblioteki:
  - Boost 1.51
  - Qt 4.8
  - Qwt 6.01
  - gsl 1.15
  - WFDB 10.5.16 (zawarta w źródłach projektu)
  - FFTW 3.3.2
  - KissFFT 1.3.0 (zawarta w źródłach projektu)
  - ALGLIB 3.7.0 (zawarta w źródłach projektu)

Do wersjonowania i śledzenia błędów wykorzystywaliśmy platformę Github wraz z rozproszonym systemem kontroli wersji Git. Posiada on zaawansowane możliwości wspierające pracę grupową nad projektem, co szczególnie przydaje się, gdy liczba osób jest duża.

2.2 Projekt systemu

Program został wykonany w architekturze MVC – istnieje ścisły podział na część wyświetlającą interfejs użytkownika, moduły przetwarzania sygnału oraz kontroler łączący te dwa elementy, co obrazuje diagram pakietów 2.2. Zastosowano obiektowe podejście przy projektowaniu hierarchii klas realizujących przetwarzanie (rys. **??**), umożliwiające bezproblemową wymianę implementacji dowolnego modułu na inną.

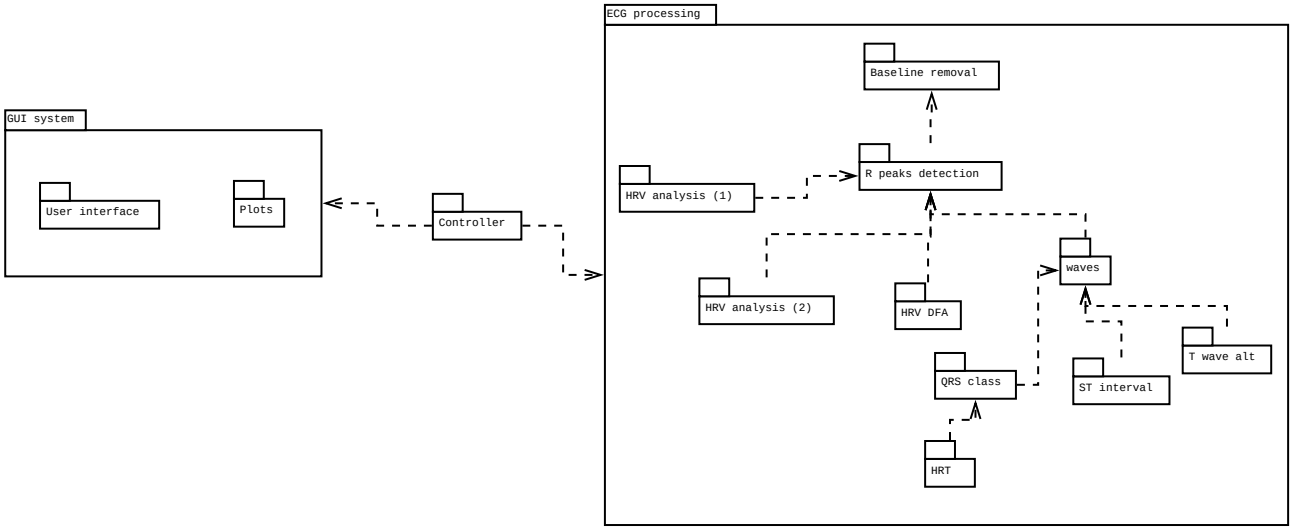
Samo przetwarzanie realizowane jest wielofazowo – moduły przeliczane są sekwencyjnie na osobnym wątku niż wątek zdarzeń GUI, co pozwala na zatrzymanie zbyt długo trwającej operacji. Zaimplementowano także mechanizm buforowania wyników, dzięki czemu przy zmianie parametrów pewnego modułu nie ma konieczności przeliczania wyników modułów wcześniejszych.

Cała aplikacja napisana jest jako samodzielny, wieloplatformowy program (rys. 2.2) – możliwa jest kompilacja pliku binarnego pod systemami Windows, Linux i MacOS X. Całe przetwarzanie wykonywane jest lokalnie, dzięki czemu nie jest wymagane połączenie z Internetem. Dzięki wybraniu licencji GPL v2+ projekt jest wolny i możliwy jest jego dalszy rozwój i kompilacja na nowe platformy.

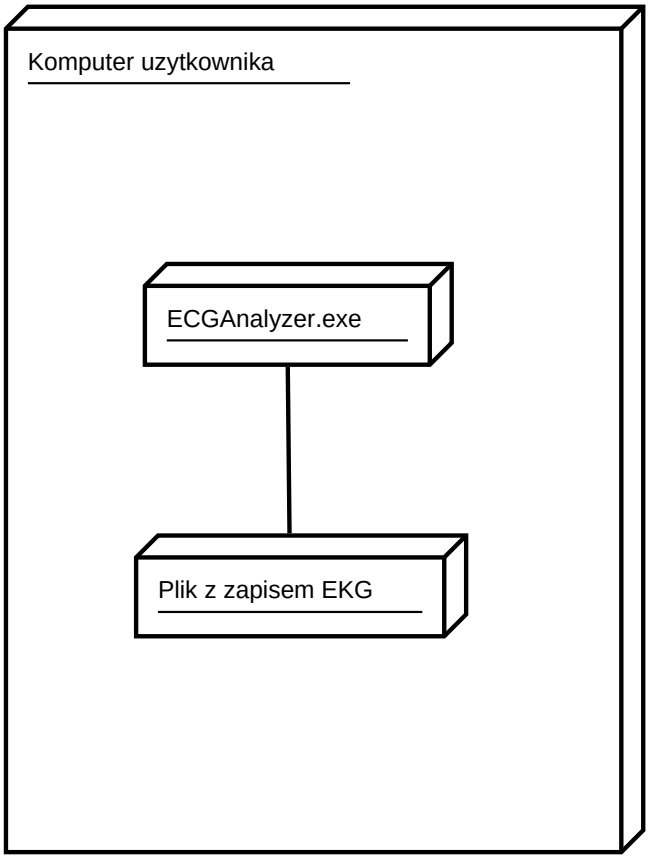
Diagram sekwencji przedstawiający sposób uruchamiania modułów jest przedstawiony na rys. 4. Każdy moduł otrzymuje najpierw zestaw parametrów (o ile posiada jakieś parametry), następnie zaś jest uruchamiany funkcją runModule. Kolejne moduły uruchamiane są sekwencyjnie według identycznego schematu. Wskutek akcji użytkownika możliwe jest przerwanie analizy pomiędzy poszczególnymi modułami.

Diagramy klas zostały przedstawione, z uwagi na wielkość na trzech rysunkach. Rys. 5 obrazuje zależności pomiędzy poszczególnymi klasami zrealizowanymi w ramach modułów, klasami abstrakcyjnymi oraz kontrolerem. Rys. 6 przedstawia zależności pomiędzy klasami modułów a klasami realizującymi model. Rys 7 przedstawia połączenie kontrolera z klasą odpowiadającą za GUI.

Diagram przypadków użycia, przedstawiony na rys. 8 pokazuje różne scenariusze, w których może być używany zrealizowany system. Każdemu przypadkowi odpowiada jeden lub więcej modułów. zachodzące pomiędzy nimi zależności są ukazywane relacjami zawierania.



Rysunek 2: Diagram pakietów.

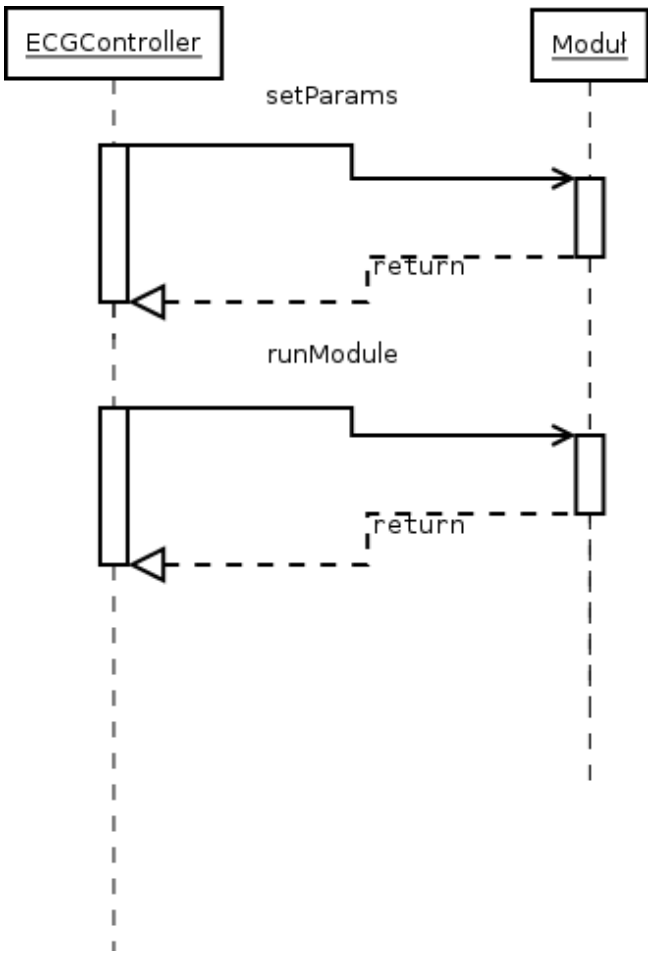


Rysunek 3: Diagram wdrożenia.

2.3 Główne elementy kodu systemu

Głównym elementem systemu jest klasa `ECGController ??`. Metoda `rerunAnalysis`, która się w niej znajduje odpowiada za kilka rzeczy:

1. Upewnianie się, że moduły mają aktualne parametry.
2. Rozpoczynanie analizy zapisu EKG.
3. Przerywanie analizy w przypadku, gdy już ona trwa.



Rysunek 4: Diagram sekwencji.

4. Raportowanie do GUI o kolejnych etapach pracy systemu (który moduł aktualnie się wykonuje).

Ostatnie zadanie wykonywane jest za pomocą dwóch funkcyjnych argumentów wejściowych.

Parametry, ustawiane metodami `setParam*` mają postać mapy przypisującej nazwom poszczególnych parametrów odpowiadające im wartości liczbowe. Wykorzystywane są przede wszystkim w modułach usuwania linii bazowej oraz wykrywania załamków R.

Do klas pomocniczych można zaliczyć między innymi `IntSignal` czy `ECGSignalChannel`. Sygnały, które przechodzą pomiędzy systemami czynią to właśnie poprzez obiekty tych klas. Są one zrealizowane jako sprytne wskaźniki do struktur trzymających wektory w stylu C biblioteki GSL, dzięki czemu możliwe jest pogodzenie szybkości działania z wygodną składnią i semantyką sprytnego wskaźnika.

Poszczególne moduły są tworzone w hierarchii obiektowej (rys. **??**). Umożliwia to podmianę w przyszłości modułu na lepszą implementację bez poprawiania reszty silnika oraz przyczynia się do zmniejszenia współzależności.

3 Opisy modułów

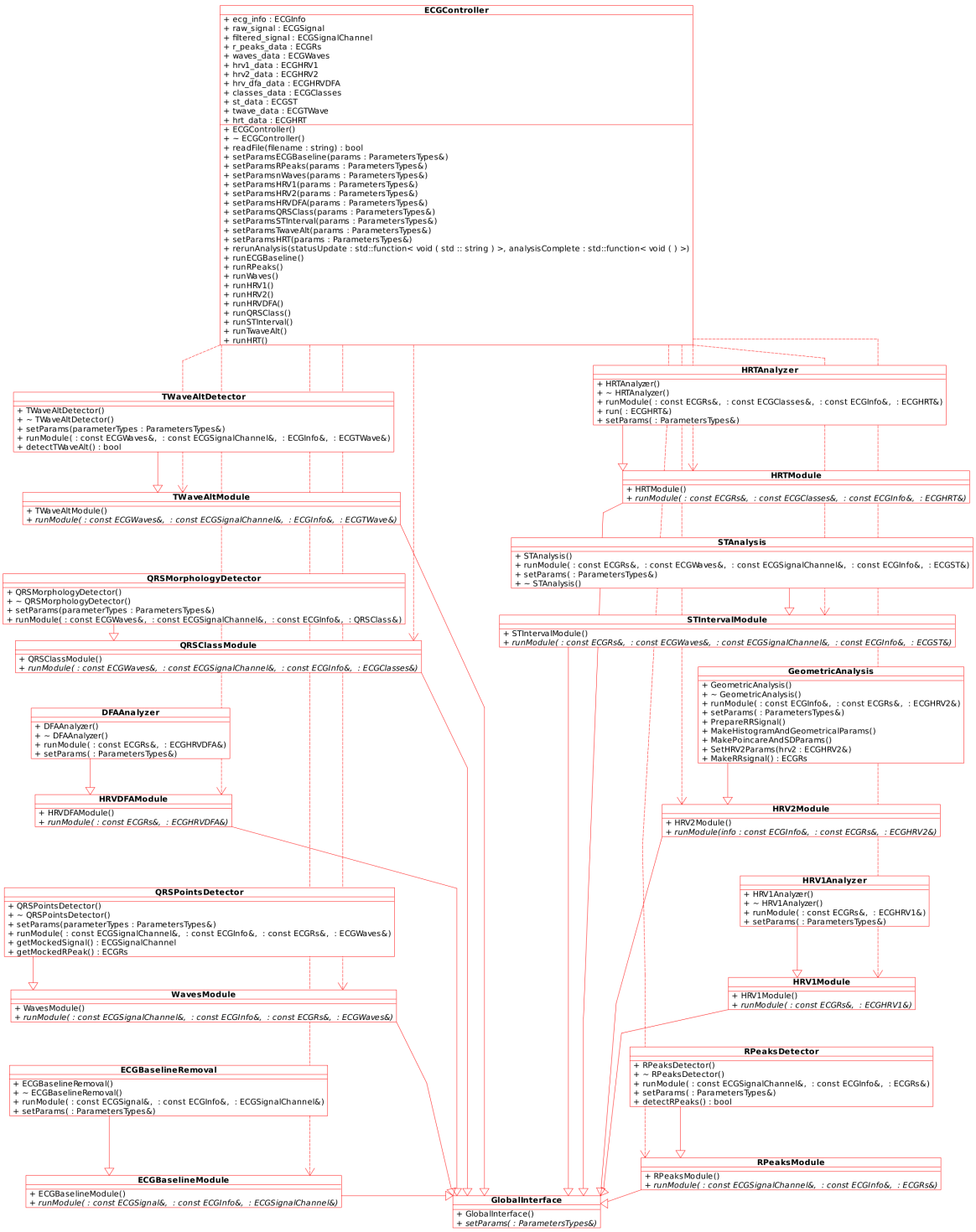
3.1 Usuwanie linii bazowej

Autorzy: Weronika Łabaj i Piotr Matuszkiewicz.

3.1.1 Opis zadania

Temat Metody filtracji i detekcji izolinii w sygnale EKG

Opis Występujące w sygnale EKG zakłócenia sieciowe i mięśniowe, jak również falowanie linii izoelektrycznej niejednokrotnie uniemożliwiają przeprowadzenie właściwej i poprawną analizy sygnału.

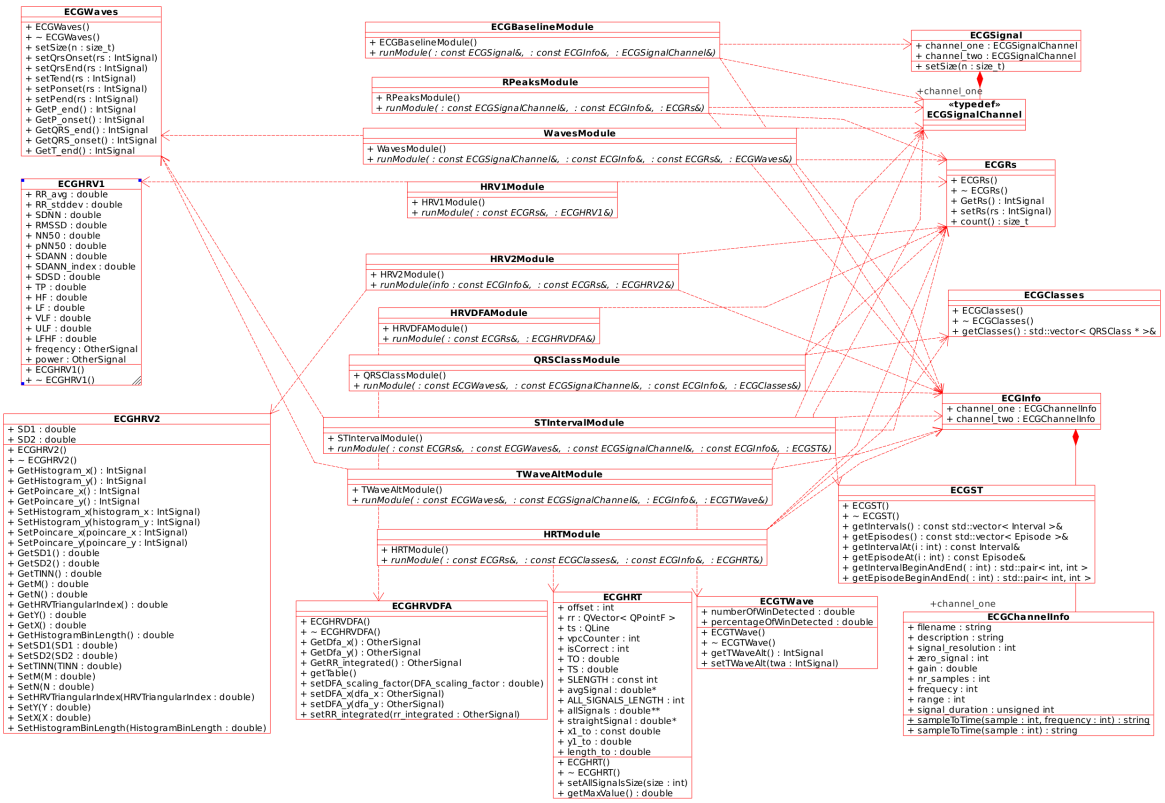


Rysunek 5: Diagram klas kontrolera i modułów.

W szczególności mają one wpływ na detekcję odcinków ST. W związku z tym detekcja i usunięcie izolinii, zwanej również linią bazową (*ang. baseline*) jest standardowym pierwszym elementem w tym procesie. Od tego, w jakim stopniu sygnał oddany do dalszej analizy pozbawiony będzie zakłóceń zależy jej dokładność i powodzenie. Celem niniejszej części projektu jest opracowanie i implementacja metod związanych z detekcją i filtracją linii bazowej w sygnale EKG.

Dane Do przetwarzania w projekcie wykorzystywane są ciągi próbek sygnału EKG z bazy Physionet.org (`MIT-BIH`)

Szukane Efektem pracy ma być moduł programu filtrujący sygnał EKG z zakłóceń sieciowych/mię-



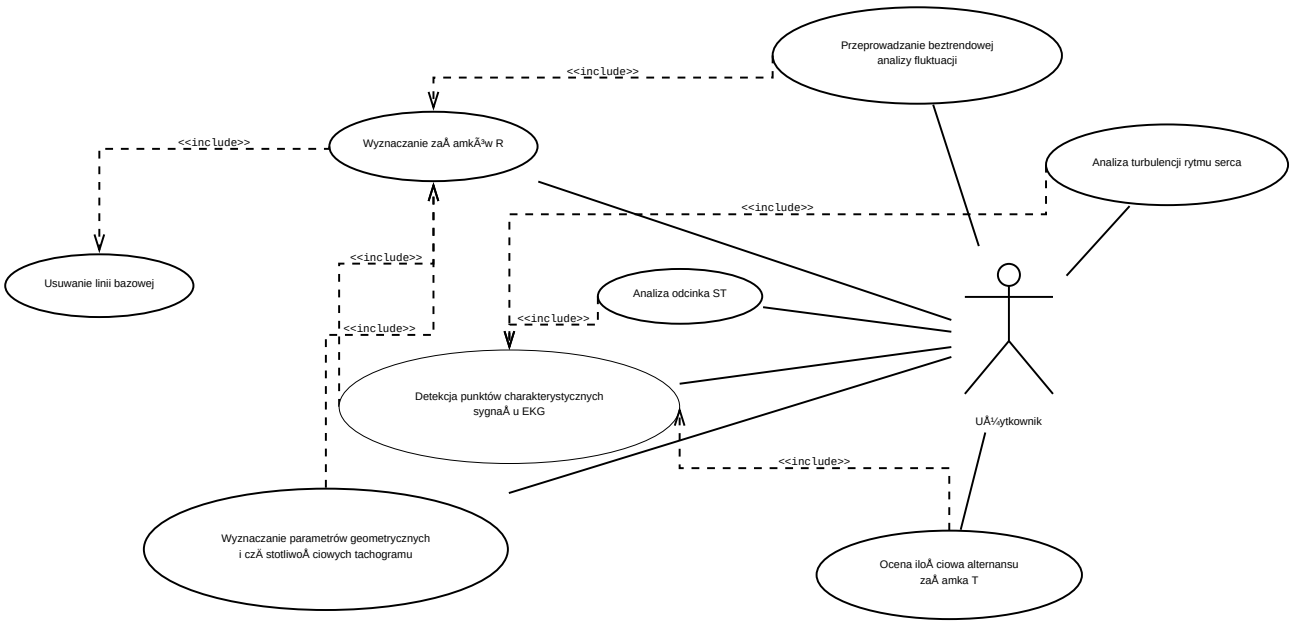
Rysunek 6: Diagram klas modułów i modelu.

Rysunek 7: Diagram klas kontrolera i GUI.

śniowych oraz usuwający z sygnału falowanie linii izoelektrycznej przy wykorzystaniu różnych metod. Moduł powinien umożliwiać wybór metody detekcji i usuwania oraz ustawienie parametrów.

3.1.2 Badania literaturowe

Problem detekcji i usuwania linii bazowej jest szeroko omawiany zarówno w książkach związanych z tematyką przetwarzania sygnałów diagnostycznych [1], jak i w publikacjach i pracach naukowych [3][4]. W kontekście tego klasycznego problemu porównywano już wiele metod – zarówno typowe filtry FIR i IIR, transformację falkową, uśrednianie sygnału, czy też filtry adaptacyjne, biorąc pod uwagę zmiany współczynników jakości w zależności od ustalonych parametrów. Wśród parametrów, jakie stosowane są do oceny jakości działania filtrów można wymienić:



Rysunek 8: Diagram klas.

1. `SNR` – *Signal-to-Noise Ratio* – stosunek sygnału do szumu, jeden z popularniejszych parametrów
2. `PSD` – *Power Spectral Density* – gęstość widmowa mocy
3. `PRD` – *Percent Root Mean Square Difference* – procentowa różnica średnio-kwadratowa
4. `PSNR` – *Peak Signal-to-Noise Ratio* – szczytowy stosunek sygnału do szumu

Każda metoda ma zarówno wady i zalety, dlatego dokonanie wyboru w celu implementacji w projekcie nie było łatwym zadaniem.

Wybór metod

Biorąc pod uwagę fakt, iż nadrzędnym celem projektu jest zbudowanie oprogramowania do analizy sygnału EKG, a nie badanie wydajności metod na poszczególnych etapach przetwarzania, wybór padł na metody sprawdzone. Wybrano jedną metodę uśredniającą – `metodę średniej kroczącej` (*ang. moving average method*) oraz filtry o `nieskończonej odpowiedzi impulsowej` (`IIR`) – `Butterwortha` i `Czebyszewa`. Taki zestaw, bazując na informacjach ze źródeł, powinien spełnić stawianie przed nim wymagania. Do oceny jakości działania filtrów zastosowano `SNR` i `PSNR`.

Metoda średniej kroczącej

Algorytm uśredniania sygnału w ruchomym oknie jest bardzo prostą metodą usuwania zakłóceń sygnału. Sygnał jest wygładzany poprzez zastąpienie wartości średnią wyliczoną wartością sygnału w oknie o ustalonej szerokości (nieparzystej, *2N+1*, gdzie `N` to sąsiedztwo) (1). Po dokonaniu obliczenia okno przesuwane jest o jedną próbkę i kalkulacje są powtarzane.

$$Ys(i) = \frac{1}{2N+1}(Y(i+N) + Y(i+N-1) + ... + Y(i-N) \tag{1}$$

Oczywistym jest, że na krańcach przedziału nie można stosować tej metody, ze względu na brak możliwości obrania odpowiednio dużego sąsiedztwa – w takim przypadku przepisuje się wartości sygnału bazowego do wektora wynikowego.

Aby średnią kroczącą zastosować do usuwania lini bazowej należy wyliczone wartości odejmować od odpowiednich próbek sygnału bazowego. Metoda zachowuje się wtedy jak swego rodzaju filtr górnoprzepustowy [3]. Metoda ta została wybrana do projektu ze względu na prostotę i szybkość implementacji. Nie jest ona jednak pozbawiona wad. Usuwa zakłócenia w sposób zadowalający, niemniej jednak powoduje dość znaczące zatracanie cech charakterystycznych sygnału, na przykład w przypadku EKG

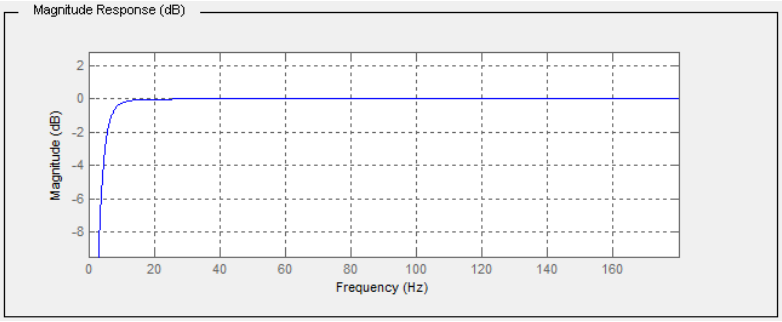
wyraźnych "pików". Jest to łatwe do stwierdzenia empirycznie, zestawiając wykresy sygnału bazowego i przefiltrowanego (11).

Filtr Butterwortha
Według różnych źródeł [6] pasmo informacyjne sygnału EKG zawiera się w przedziale od `0.05Hz-5Hz` do `100Hz`, dlatego do redukcji zakłóceń można stosować filtry zarówno dolno- jak i górnoprzepustowe.

Filtr `Butterwortha` to filtr charakteryzujący się maksymalnie płaską charakterystyką amplitudową w paśmie przenoszenia. Częstotliwość graniczną filtru wyznacza spadek sygnału (*ang. ripple*) o 3 dB. Jest to filtr powszechnie uważany za bardzo skuteczny w filtrowaniu linii bazowej [3].
Na potrzeby projektu proponowane są dwie wersjie filtru `Butterwortha` (IIR):

1. *górnoprzepustowa* drugiego rzędu, o częstotliwości odcięcia (*ang. cut-off frequency*) `5Hz`(9)
2. *dolnoprzepustowa* czwartego rzędu, o częstotliwości odcięcia `100Hz`

O ile w przypadku filtru dolnoprzepustowego dobór częstotliwości nie dziwi – bo jak już wcześniej wspomniano na 100Hz kończy się pasmo informacyjne EKG – o tyle w przypadku górnoprzepustowego może paść pytanie, dlaczego akurat 5Hz. W sygnale EKG, do poziomu 5Hz, można napotkać artefakty wynikające z ruchy pacjenta (*motion artifacts*, a dodatkowo filtr wyeliminuje również zakłócenia na niższych częstotliwościach. Jak już to zostało wspomniane – filtr Butterwortha jest bardzo skuteczny,



Rysunek 9: Odpowiedź amplitudowa filtru Butterwortha (górnoprzepustowego)

niemniej jednak równniez on nie jest pozbawiony wad. Przez to, że jako filtr IIR ma nieliniową odpowiedź fazową, istnieje tendencja do znacznego zniekształcania sygnału bazowego. Aby uniknąć tego zjawiska stosuje się tzw. `filtrowanie o charakterystyce zero-fazowej` *ang. Zero-Phase filtering*, opisan w kolejnych paragrafch.

Filtr Czebyszewa
Jest to rodzaj filtru elektrycznego, którego charakterystyczną cechą jest wykorzystanie *wielomianów Czebyszewa* do aproksymacji charakterystyki częstotliwościowej amplitudowej. Optymalizacja przebiegu charakterystyki częstotliwościowej amplitudowej w filtrach tego typu ma kluczowe znaczenie.
Wyróżnia się dwa typy filtrów:

1. `filtry Czebyszewa typu I` – zafalowania przebiegu wzmocnienia w paśmie przepustowym, oraz płaski przebieg charakterystyki w paśmie zaporowym
2. `filtry Czebyszewa typu II` – zafalowania przebiegu wzmocnienia w paśmie zaporowym, oraz płaski przebieg charakterystyki w paśmie przepustowym, nazywany przez to zwykle *inwersyjnym*

Filtry `Czebyszewa` mają porównywalną skuteczność do filtrów `Butterwortha`. Na ich korzyśc przemawia zdecydowanie lepsze tłumienie za pasmem przepustowym (stroma charakterystyka). Wsród minusów można wymienić falowanie w pasmie przepustowym w przypadku filtru `typu I` oraz dodatkowy parametru – `tętnienie, ang. ripple` – niezbędny do zdefiniowania filtru. Dla porównania w przypadku `Butterwortha` wystarczy częstotliwośc odcięcia.
Na potrzeby projektu proponowane są dwie wersjie filtru `Czebyszewa`:

1. *górnoprzepustowa typu II*, trzeciego rzędu, o częstotliwości odcięcia `0.5Hz` i tętnieniu `-3dB`
2. *dolnoprzepustowa typu I*, piątego rzędu, o częstotliwości odcięcia `100Hz` i tętnieniu `-3dB`

Również w przypadku tego filtru konieczne będzie zastosowanie się tzw. `filtrowania o charakterystyce zero-fazowej` *ang. Zero-Phase filtering.*

Filtrowanie o charakterystyce zero-fazowej
Filtrowanie `Zero-Phase` wykorzystuje w swym działaniu współczynniki filtru bazowego (np. Butterwortha czy też Czebyszewa) i wykonuje filtrowanie dwukrotnie, stosując małe okno, w obie strony sygnału – od początku do końca i od końca do początku. Rezultatem tego filtrowania jest:

1. brak opóźnień w sygnale przefiltrowanym, ze względu na przejście sygnału w obie strony
2. redukcja zaburzeń sygnału na krańcach przedziału – poprzez zachowanie warunków początkowych
3. zachowanie cech charakterystycznych sygnału bazowego (w przeciwieństwie do samego filtru IIR czy też np. średniej kroczącej)
4. rząd filtru zero-fazowego jest równy podwojonemu rzędowi filtru bazowego.

Filtr może być opisany przez poniższe rekurencyjne równanie (2)

$$y[n] = a_0x[n] + a_1x[n+1] + a_2x[n+2] + a_3x[n+3] + b_1y[n+1] + b_2y[n+2] + b_3y[n+3] \tag{2}$$

a nastepnie odwrotne rekurencyjne równanie, do zastosowania przy przejściu od końca sygnału (3)

$$y[n] = a_0x[n] + a_1x[n-1] + a_2x[n-2] + a_3x[n-3] + b_1y[n-1] + b_2y[n-2] + b_3y[n-3] \tag{3}$$

Należy przyjąć, że `b` to współczynniki licznika transmitancji filtru bazowego, a `a` to współczynniki mianownika.

Signal-to-Noise Ratio
Parametr oznaczający stosunek sygnału (użytecznego) do szumu. Jest stosowany jako parametr pomocniczy w określeniu jakości sygnału. Obliczany za pomocą następującej formuły:

$$SNR = 20\log_{10}(\frac{A_{sygnal}}{A_{szum}}) \tag{4}$$

gdzie `A` oznacza amplitudę sygnału. W praktyce `A` wyliczane jest jako średnia kwadratowa amplitud w sygnale.

Peak Signal-to-Noise Ratio
Parametr oznaczający stosunek maksymalnej mocy sygnału do średniej wartości szumu. Również jest stosowany jako parametr pomocniczy w określeniu jakości sygnału. Obliczany za pomocą następującej formuły:

$$PSNR = 10\log_{10}(\frac{k^2}{MSE}) \tag{5}$$

gdzie `MSE` oznacza błąd średniokwadratowy obliczany za pomoca

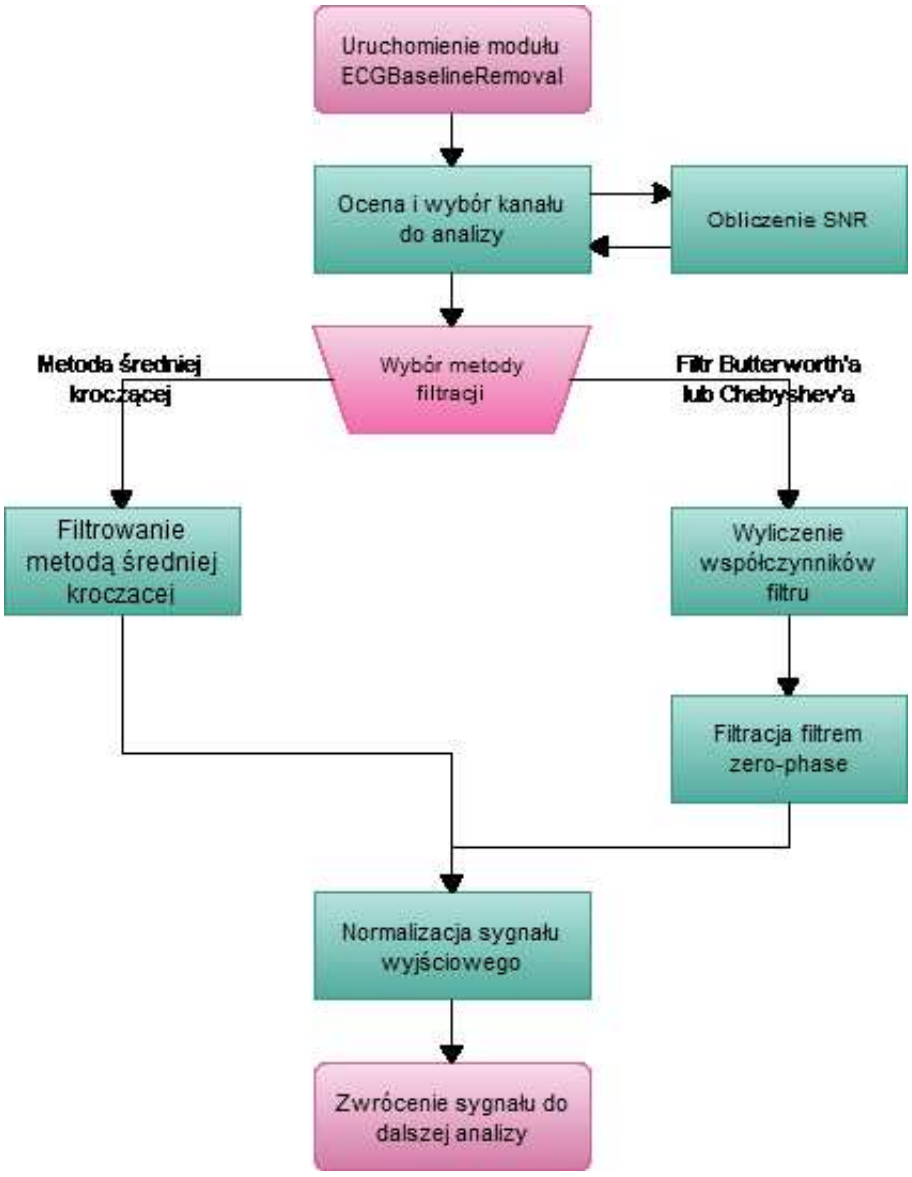
$$MSE = \frac{1}{N}\sum_{i=1}^{N}([f(i) - f'(i)]^2) \tag{6}$$

gdzie `N` oznacza długość sygnału, `f` to sygnał oryginalny, `f'` to sygnał przefiltrowany, a `k` to maksymalna amplituda sygnału bazowego.

3.1.3 Opis funkcjonalności modułu

Nadrzędnym przeznaczeniem modułu jest filtracja i usuwanie lini bazowej. Niemniej jednak moduł bardziej kompleksowo przygotowuje sygnał do dalszej analizy.
Realizowane są następujące funkcjonalności:

1. Wybór kanału, który będzie przetwarzany w kolejnych etapach działania aplikacji. Wybór dokonywany jest poprzez porównanie SNR obu kanałów.
2. Detekcja i usunięcie zakłóceń lini bazowej sygnału – główne zadanie.
3. Normalizacja sygnału przed zwróceniem do dalszej analizy.



Rysunek 10: Diagram prezentujący funkcjonalnosć modułu ECGBaselineRemoval

Konieczność wprowadzenia oceny i wyboru kanału do dalszej analizy został podyktowany nadrzędnymi decyzjami architekturalnymi. W razie potrzeby może zostać pominięta.

3.1.4 Opis procedur i metod

Klasa: ECGBaselineRemoval - Główna klasa modułu, dziedzicząca po wirtualnym ECGBaselineModule. Jest ona odpowiedzialna za przetworzenie sygnału pierwotnego i zwrócenie sygnału przefil-

trowanego (pozbawionego zakłóceń linii bazowej). Pojedyncza instancja tej klasy tworzona jest wraz z uruchomieniem aplikacji.

```
void ECGBaselineRemoval::runModule(const ECGSignal &inputSignal,
const ECGInfo & ecgi, ECGSignalChannel &outputSignal)
```

Podstawowa metoda w module. Jest odpowiedzialna za uruchomienie głównej funkcjonalności – przetworzenia sygnału. Pod hasłem "przetworzenie" kryje się wybranie odpowiedniego kanału, realizacja filtrowania oraz normalizacja.
Parametry:

1. `const ECGSignal &inputSignal` – referencja do sygnału bazowego, zawierającego dwa kanały
2. `const ECGInfo & ecgi` – referencja do obiektu informacyjnego typu `ECGInfo`, przechowującego dane dotyczące sygnału (m.in. częstotliwość próbkowania, ilość próbek)
3. `ECGSignalChannel &outputSignal` – referencja do sygnału wyjściowego – po wykonaniu metody `ECGBaselineRemoval::runModule` odnosi się do sygnału przefiltrowanego; instancja klasy `ECGSignalChannel` – pojedynczy kanał

```
void ECGBaselineRemoval::setParams(ParametersTypes &params)
```

Metoda pozwala na wybór algorytmu filtracji poprzez ustawienie flagi `baselineRemovalMethod`, oraz ustawienie parametrów, korzystając z wartości pochodzących z graficznego interfejsu użytkownika.
Parametry:

1. `ParametersTypes &params` – referencja do obiektu typu `PatametersTypes` – zawierającego parametry przekazane z graficznego interfejsu użytkownika

```
void ECGBaselineRemoval::movingAverageBaselineRemoval(ECGSignalChannel
&inputSignal, ECGSignalChannel &outputSignal, int span)
```

Metoda opakowująca (tzw. *wrapper*) instancję klasy realizującej filtrowanie metodą średniej kroczącej 3.1.2.
Parametry parametry:

1. `ECGSignalChannel &inputSignal` – referencja do obiektu będącego wybranym kanałem z sygnału wejściowego
2. `ECGSignalChannel &outputSignal` – referencja do obiektu wyjściowego
3. `int span` – wartość całkowita – nieparzysta – będąca rozmiarem okna w algorytmie średniej kroczącej

```
void ECGBaselineRemoval::butterworthBaselineRemoval(ECGSignalChannel
&inputSignal, ECGSignalChannel &outputSignal, const ECGInfo &ecgInfo,
int order, double cutoffFrequency, double attenuation)
```

Metoda opakowująca instancję klasy realizującej filtrowanie filtrem Butterwortha 3.1.2.

Parametry:

1. `ECGSignalChannel &inputSignal` – referencja do obiektu będącego wybranym kanałem z sygnału wejściowego
2. `ECGSignalChannel &outputSignal` – referencja do obiektu wyjściowego
3. `const ECGInfo &ecgi` – referencja do obiektu informacyjnego `ECGInfo`, przechowującego dane sygnału (m.in. częstotliwość próbkowania, ilość próbek)

4. `int order` – wartość całkowita dodatnia, odpowiadająca rzędowi filtra
5. `double cutoffFrequency` – częstotliwość odcięcia filtra – wartość zmiennoprzecinkowa (może przyjąć np *0.05[Hz]*)
6. `double attenuation` – tłumienie

```
void ECGBaselineRemoval::chebyshevBaselineRemoval(ECGSignalChannel
&inputSignal, ECGSignalChannel &outputSignal, const ECGInfo &ecgInfo,
int order, double cutoffFrequency, double ripple)
```

Metoda opakowująca instancję klasy realizującej filtrowanie filtrem Czebyszewa 3.1.2.

Parametry:

1. `ECGSignalChannel &inputSignal` – referencja do obiektu będącego wybranym kanałem z sygnału wejściowego
2. `ECGSignalChannel &outputSignal` – referencja do obiektu wyjściowego
3. `const ECGInfo &ecgi` – referencja do obiektu informacyjnego `ECGInfo`, przechowującego dane sygnału (m.in. częstotliwość próbkowania, ilość próbek)
4. `int order` – wartość całkowita dodatnia, odpowiadająca rzędowi filtra
5. `double cutoffFrequency` – częstotliwość odcięcia filtra – wartość zmiennoprzecinkowa (może przyjąć np *0.05[Hz]*)
6. `double ripple` – tętnienie

Klasa: SignalPreprocessor - Klasa pomocnicza odpowiedzialna za wstępną analizę sygnału oraz jego normalizację.

```
int SignalPreprocessor::evaluateSignalChannels(const ECGSignal &inputSignal)
```

Metoda realizująca porównanie kanałów, bazując na wyliczeniu współczynnika `SNR` (`Snr::snr`) tysiąca pierwszych próbek oraz wybór lepszego z nich.

Parametry:

1. `const ECGSignal &inputSignal` – referencja do sygnału bazowego (dwa kanały)

Wartość zwracana:

1. liczba całkowita odpowiadająca numerowi kanału po ewaluacji (1 albo 2)

```
void SignalPreprocessor::normalize(ECGSignalChannel &inputSignal,
ECGSignalChannel &outputSignal)
```

Metoda odpowiedzialna za normalizację sygnału, tj. przeskalowanie do przedziału wartości 0..1. Wywoływana jako ostatni etap realizacji przetwarzania wstępnego – `ECGBaselineRemoval::runModule()`.
Parametry:

1. `ECGSignalChannel &inputSignal` – referencja do obiektu wejściowego
2. `ECGSignalChannel &outputSignal` – referencja do obiektu wyjściowego

Klasa: Filter - Klasa odpowiadająca za realizację filtrowania `IIR` (*ang. infinite impulse response*) na bazie współczynników pozyskanych z metod projektujących filtry `Butterwortha` bądź `Czebyszewa`.

```
void Filter::zeroPhase(std::vector<double> b, std::vector<double> a,
ECGSignalChannel &inputSignal, ECGSignalChannel &outputSignal, int order)
```

Metoda realizująca filtrowanie typu Zero-Phase 3.1.2. Wykorzystuje współczynniki transmitancji filtru bazowego, zgodnie z (2) oraz (3). Metoda została zaimplementowana na wzór funkcji `filtfilt` ze środowiska Matlab.

Parametry:

1. `(std::vector<double> b` – wektor współczynników pochodzących z licznika transmitancji zaprojektowanego filtra
2. `(std::vector<double> a` – wektor współczynników pochodzących z mianownika transmitancji zaprojektowanego filtra
3. `ECGSignalChannel &inputSignal` – referencja do obiektu wejściowego
4. `ECGSignalChannel &outputSignal` – referencja do obiektu wyjściowego
5. `int order` – wartość całkowita dodatnia, odpowiadająca rzędowi filtra

```
void Filter::filter(std::vector<double> b, std::vector<double> a,
ECGSignalChannel &inputSignal, ECGSignalChannel &outputSignal, int order)
```

Metoda realizująca filtr `IIR` na bazie zestawu współczynników transmitancji filtru `Butterwortha` bądź `Czebyszewa`. Funkcja bazuje na swoim odpowiedniku (`filter`) ze środowiska Matlab. Jest uniwersalna i może realizować również inne filtry. `Filter::filter` może być wykorzystywana autonomicznie bądź wywoływana niejawnie w ramach metody `Filter::zeroPhase`.

Parametry:

1. `(std::vector<double> b` – wektor współczynników pochodzących z licznika transmitancji zaprojektowanego filtra
2. `(std::vector<double> a` – wektor współczynników pochodzących z mianownika transmitancji zaprojektowanego filtra
3. `ECGSignalChannel &inputSignal` – referencja do obiektu wejściowego
4. `ECGSignalChannel &outputSignal` – referencja do obiektu wyjściowego
5. `int order` – wartość całkowita dodatnia, odpowiadająca rzędowi filtra

Klasa: MovingAverage - Klasa realizująca filtrowanie metodą średniej kroczącej.

```
void MovingAverage::removeBaseline(ECGSignalChannel &inputSignal,
ECGSignalChannel \&outputSignal, int span)
```

Metoda realizująca uswanie linii bazowej algorytmem średniej kroczącej. Główna pętla metody implementuje (1) wywołując metodę `MovingAverage::calculateAvgValueOfNeighbours`

Parametry:

1. `ECGSignalChannel &inputSignal` – referencja do obiektu będącego wybranym kanałem z sygnału wejściowego
2. `ECGSignalChannel &outputSignal` – referencja do obiektu wyjściowego

3. `int span` – wartość całkowita – nieparzysta – będąca rozmiarem okna w algorytmie średniej kroczącej

```
double MovingAverage::calculateAvgValueOfNeighbours(gsl_vector *signal,
int currentIndex, int span)
```

Metoda obliczająca średnią wartość próbek w sąsiedztwie danej próbki, w oknie o szerokości określonej wartością zmiennej `int span`

Przyjmowane parametry

1. `gsl_vector *signal` – wektor przechowujący próbki przetwarzanego sygnału
2. `int currentIndex` – indeks aktualnie analizowanej próbki
3. `int span` – wartość całkowita – nieparzysta – będąca rozmiarem okna w algorytmie średniej kroczącej

Klasa: Butterworth - Klasa realizująca projektowanie filtra `Butterwortha`.

```
std::vector<std::vector<double>> Butterworth::filterDesign(int order,
double attenuation, double cutoffFreq, int sampleFreq, int type)
```

Metoda odpowiedzialna za zaprojektowanie filtra `Butterwortha`.

Parametry:

1. `int order` – wartość całkowita dodatnia, odpowiadająca rzędowi filtra
2. `double attenuation` – tłumienie
3. `double cutoffFrequency` – częstotliwość odcięcia filtra – wartość zmiennoprzecinkowa (może przyjąć np *0.05[Hz]*)
4. `int sampleFreq` – częstotliwość próbkowania sygnału wejściowego, potrzebną do obliczenia współczynników filtra
5. `int type` – zmienna określająca typ (lowpass -1, highpass - 2) filtra do zaprojektowania

Wartość zwracana:

1. `std::vector<std::vector<double>>` – dwuwymiarowy wektor wartości zmiennoprzecinkowych, zwracający zestaw współczynników z mianownika i licznika transmitancji zaprojektowanego filtra

Klasa: Chebyshev - Klasa realizująca projektowanie filtra `Czebyszewa`.

```
std::vector<std::vector<double>> Chebyshev::filterDesign(int order,
double ripple, double cutoffFreq, int sampleFreq, int type)
```

Metoda odpowiedzialna za zaprojektowanie filtra `Czebyszewa`

Parametry:

1. `int order` – wartość całkowita dodatnia, odpowiadająca rzędowi filtra
2. `double ripple` – tętnienie pasma przepustowego
3. `double cutoffFrequency` – częstotliwość odcięcia filtra – wartość zmiennoprzecinkowa (może przyjąć np *0.05[Hz]*)

4. `int sampleFreq` – częstotliwość próbkowania sygnału wejściowego, potrzebną do obliczenia współczynników filtra
5. `int type` – zmienna określająca typ (*lowpass* - 1, *highpass* - 2) filtra do zaprojektowania

Wartość zwracana:

1. `std::vector<std::vector<double>>` – dwuwymiarowy wektor wartości zmiennoprzecinkowych, zwracający zestaw współczynników z mianownika i licznika transmitancji zaprojektowanego filtra

Klasa: Snr - Klasa realizująca metody ewaluacji filtrów, wraz z metodami pomocniczymi zgodnie ze wzorami (4),(5) i (6). Pozwala na wyliczenie współczynników `SNR` (`Snr::snr`) oraz `PSNR` (`Snr::psnr`) dla zadanych sygnałów. Ze względu na generyczność rozwiązań w niej zaimplementowanych opis interfejsu zostaje pominięty.

Projektowanie filtrów – realizacja
Do projektowania filtrów `Butterwortha` oraz `Czebyszewa` miała zostać wykorzystana biblioteka `SPUC - Signal Processing using C++` (*http://spuc.sourceforge.net/*). Niestety po jej zbudowaniu zgodnie z zaleceniami twórców okazało się, że praktycznie niemożliwe jest linkowanie jej w sposób klasyczny z projektem Visual Studio. Ze względu na ten fakt próby jej wykorzystania zostały porzucone na rzecz implementacji analogicznej z tą, która występuje w pakiecie Matlab.

3.1.5 Warunki testowania

Testy na etapie prototypowania przeprowadzane były w środowisku Matlab. Dało to pogląd na wymagania, jakie niesie za sobą proces usuwania linii bazowej. Na etapie realizacji projektu przeprowadzano równolegle testy zarówno w aplikacji jak i ponownie w Matlab. Dane testowe pochodzą z bazy `MIT-BIH`.

Wartości proponowanych parametrów dla poszczególnych metod zostały dobrane na bazie doświadczeń oraz informacji ze źródeł naukowych, o czym zostało już nadmienione powyżej 3.1.2.

Parametry filtrów

Metoda średniej kroczącej

1. `span = 5`

Filtr Butteworth'a
Pierwszy zestaw:

1. `order = 2`
2. `cutoff_frequency = 5Hz`
3. `sample_frequency = 360Hz`
4. `type = 'high'`

Drugi zestaw:

1. `order = 4`
2. `cutoff_frequency = 100Hz`
3. `sample_frequency = 360Hz`
4. `type = 'low'`

Filtr Czebyszewa
Pierwszy zestaw:

1. `order = 3`
2. `cutoff_frequency = 0.5Hz`
3. `ripple = 3dB`
4. `sample_frequency = 360Hz`
5. `type = 'high'`

Drugi zestaw:

1. `order = 5`
2. `cutoff_frequency = 100Hz`
3. `ripple = 3dB`
4. `sample_frequency = 360Hz`
5. `type = 'low'`

W większości przypadków wyodrębnione w procesie prototypowania skuteczne parametry pokryły się z propozycjami naukowców.

3.1.6 Wyniki

Testy polegały na analizie poglądowej zmiany przebiegu sygnału po zastosowaniu filtra oraz na wyliczeniu parametrów `SNR` oraz `PSNR`.

Poglądowe porównanie metod

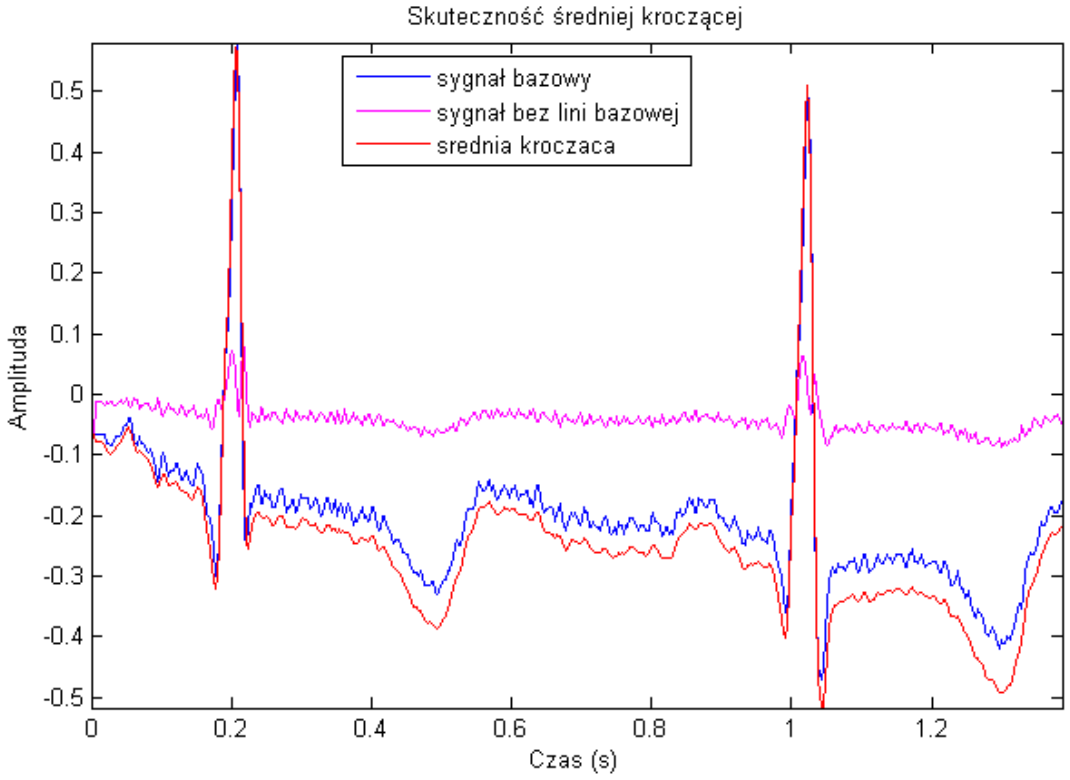
Metoda średniej kroczącej
W przypadku jedynej metody uśredniającej w zestawieniu wynik poglądowy jest zadowalający. Na wykresie 11 widać, iż samo zastosowanie uśredniania wygładziło przebieg sygnału (wykres pomarańczowy), a po odjęciu wartości wyliczonej od sygnału bazowego otrzymano sygnał wynikowy wydaje się być bardziej podatny na skuteczną analizę.

Niestety wyraźnie widać utratę cech charaktetystycznych sygnału bazowego – "piki" dalej występują w tych samych miejscach, niemniej jednak zostały one wyraźnie spłaszczone. Istnieje podejrzenie, że nie każda metoda ekstrakcji cech sygnału poradzi sobie z tak niewyraźnymi "pikami". W tym przypadku należy przeprowadzić testy polegające na stosowaniu różnych metod analizy sygnału, aby ostatecznie ocenić skuteczność średniej kroczącej.

Filtry górnoprzepustowe
Filtry górnoprzepustowe, niezależnie od typu – czy `Butterwortha`, czy też `Czebyszewa` – wykazały się dobrą skutecznoscią w usuwaniu linii bazowej. Jak widać na wykresie 12 falowanie sygnału zostało w znaczącym stopniu zredukowane. Minimalnie lepszy efekt w przypadku filtrów górnoprzepustowych dawał `Butterworth`.

Warto nadmienić, że filtry `IIR` nie powodują aż tak dużej utraty cech charakterystycznych jak średnia krocząca.

Filtry dolnoprzepustowe
Bez względu na to, że wartości filtrów dolnoprzepustowych były bezpośrednio zapożyczone z literatury, ich stosowanie nie przyniosło oczekiwanych rezultatów. Przebieg sygnału po przefiltrowaniu pozostawał praktycznie bez zmian. Jedyną korzyścią, jaką można dostrzec jest nieznaczne wygładzenie wykresu (13).



Rysunek 11: Sygnał bazowy i sygnał przefiltrowany algorytmem średniej kroczącej

Porównanie parametrów SNR i PSNR

Oprócz oceny poglądowej działania filtrów dokonano również pomiarów parametrów `SNR` i `PSNR`. Zestawienie wyników prezentują poniższe tabele:

`SNR`[dB]

sygnał	Butter. (high)	Czeby. (high)	Śr. krocząca	Butter. (low)	Czeby. (low)
100_V5	-6.4236	-3.9277	-12.5362	36.7097	18.4894
100_MII	-6.0413	-4.2806	-12.7211	40.6467	18.1320

`PSNR`[dB]

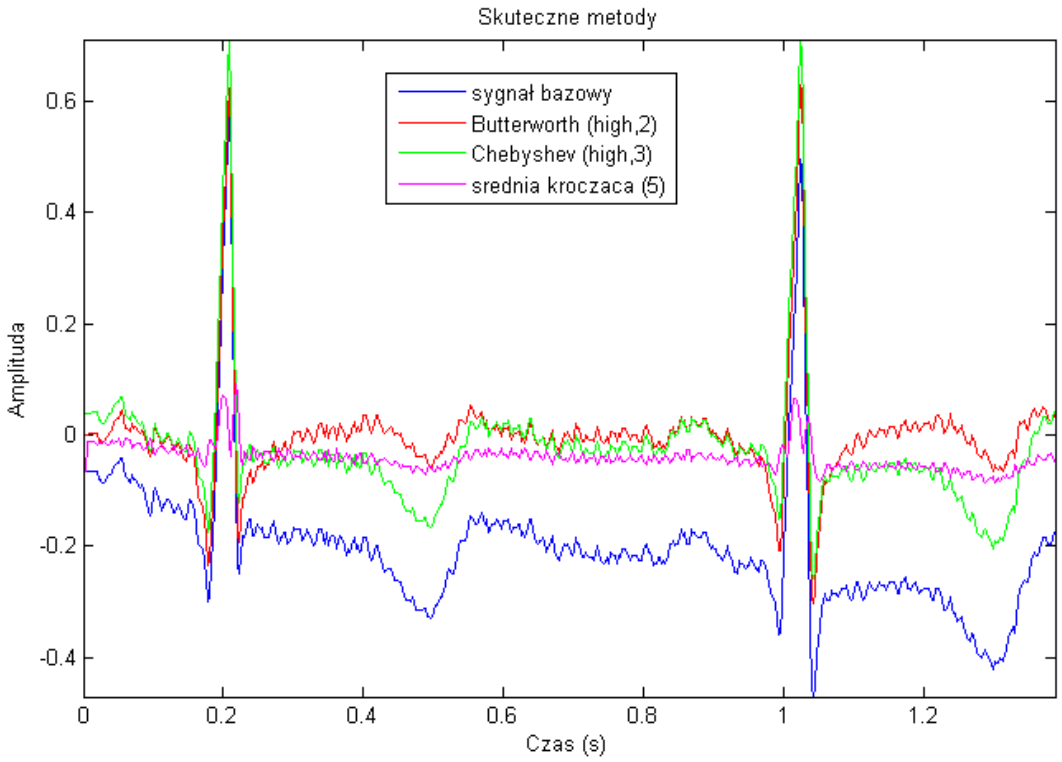
sygnał	Butter. (high)	Czeby. (high)	Śr. krocząca	Butter. (low)	Czeby. (low)
100_V5	-48.1567	-61.0092	-41.1130	-75.2726	-56.3931
100_MII	-50.9663	-76.0654	-43.5110	-72.6385	-57.2145

Niestety oba parametry w przypadku badanych sygnałów okazały się zwracać nieoczekiwane – pozornie błędne – wyniki. W idealnym przypadku zarówno `SNR` jak i `PSNR` powinny zwracać wartości dodatnie. Co więcej – bardziej skuteczne metody (w tym przypadku filtry górnoprzepustowe oraz metoda średniej kroczącej) powinny zwracać wyniki znacząco wyższe od metod nieskutecznych. Otrzymane wyniki mogłyby wskazywać na istnienie błędu w implementacji metody, jednak była ona weryfikowana wielokrotnie.

Dopóki nie zostaną roziwązane zaistniałe problemy nie można traktować parametrów `SNR` oraz `PSNR` jako wiążących w procesie oceny skuteczności filtracji.

Wnioski

Zaproponowane metody w przypadku doboru odpowiednich parametrów oferują skuteczność na dobrym poziomie. Potwierdzają to również lepsze rezultaty osiągane przez zespoły odpowiedzialne za moduły analizujące sygnał EKG po zmianie sygnału źrodłowego na sygnał pozbawiony lini bazowej.



Rysunek 12: Sygnał bazowy i sygnały po skutecznej filtracji

Testy wykazały, że filtry górnoprzepustowe mogą z powodzeniem być stosowane w tym module. Metoda średniej kroczącej do pewnego stopnia również dobrze poradziła sobie z zadaniem. Warto jednak zaznaczyć, że w przypadku wyboru między tymi dwoma typami filtracji i usuwania lini bazowej użytkownik musi zastanowić się, czy ważniejsze w dalszym etapie analizy będzie zachowanie wyraźnych cech sygnału czy też jak najskuteczniejsza redukcja falowania linii izoelektrycznej.

Zalecane są dalsze testy, w celu ustalenia przyczyn niskiej skuteczności filtrów dolnoprzepustowych.

3.2 Wykrywanie załamków R

Autorzy: Paweł Maślanka i Norbert Pabian

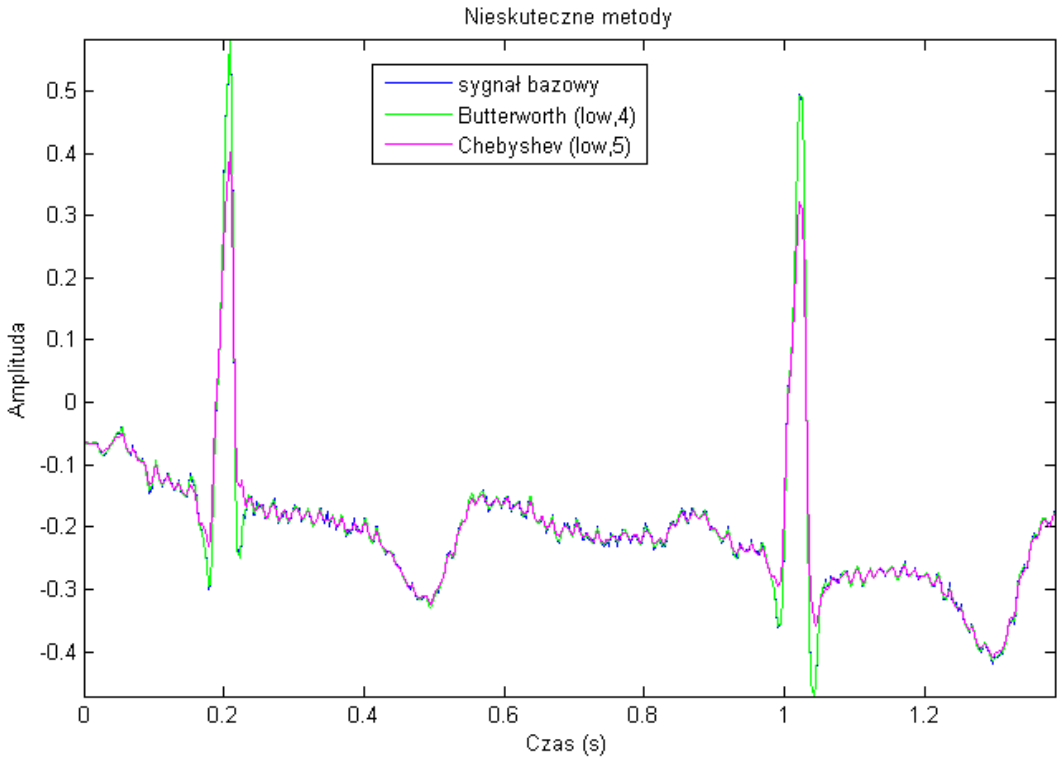
3.2.1 Opis zadania

Temat Detekcja załamka R

Opis Detekcja załamków R w przefiltrowanym sygnale cyfrowym EKG, mimo iż jest tematem przebadanym i dobrze znanym, to wciąż nie istnieje złoty standard postępowania. Alternatywą dla powszechnie znanego algorytmu "Pan-Tompkins", opartego o podstawowe przekształcenia matematyczne potęgowania i różniczkowania może okazać się transformata Hilberta odpowiednio przygotowanego sygnału. Celem projektu jest opracowanie i implementacja metod służących do detekcji załamków R w sygnale EKG w oparciu o algorytm "Pan-Tompkins" oraz zespolone przekształcenie Hilberta. Podejmując projekt należy zwrócić uwagę na problematykę wystąpienia szumu i analizy sygnału na końcach przedziału, gdzie rozważania teoretyczne transformaty Hilberta nieco odbiegają od praktycznych.

Dane przefiltrowany ciąg próbek z modułu ECG_BASELINE

Szukane moduł programu wyznaczający numery próbek załamków R oraz zaznaczający na wykresie wykryte załamki R. Numery próbek pozwolą na dalszą analizę ilościową i jakościową sygnału



Rysunek 13: Sygnał bazowy i sygnały po nieskutecznej filtracji

EKG.

3.2.2 Badania literaturowe

Usuwanie stałej DC i normalizacja sygnału Często w sygnałach EKG można spotkać dodatkową stałą DC w sygnale którą można skutecznie usunąć stosując wzór (7). Obliczanie średniej sygnału i odejmowanie jej od każdej próbki sygnału jest przybliżoną metodą usuwania składowej stałej.

$$y(n) = x(n) - \frac{x(1) + x(2) + ... + x(n)}{n} \tag{7}$$

Aby ułatwić wszystkie obliczenia dokonywane na wybranym sygnale można znormalizować sygnał tak by wartości amplitudy były w zakresie od -1 do 1. Normalizację sygnału można uzyskać stosując wzór (8).

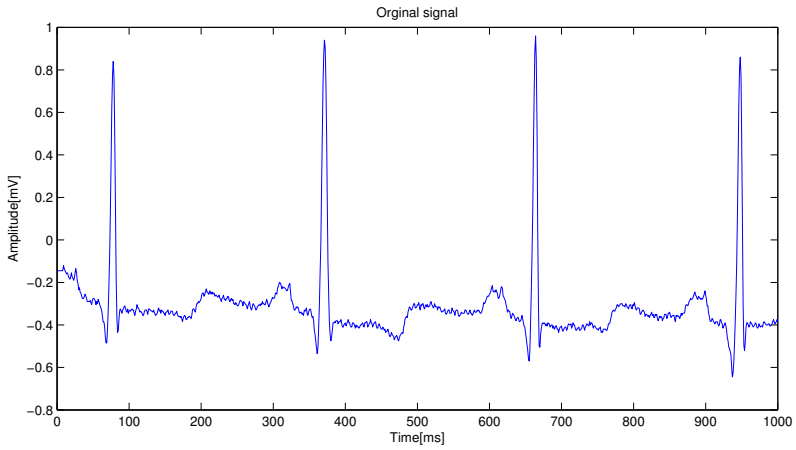
$$y(n) = \frac{x(n)}{\max(|x(n)|)} \tag{8}$$

Rezultaty operacji usuwania składowej DC oraz normalizacji sygnału możemy zobaczyć na rysunku (15).

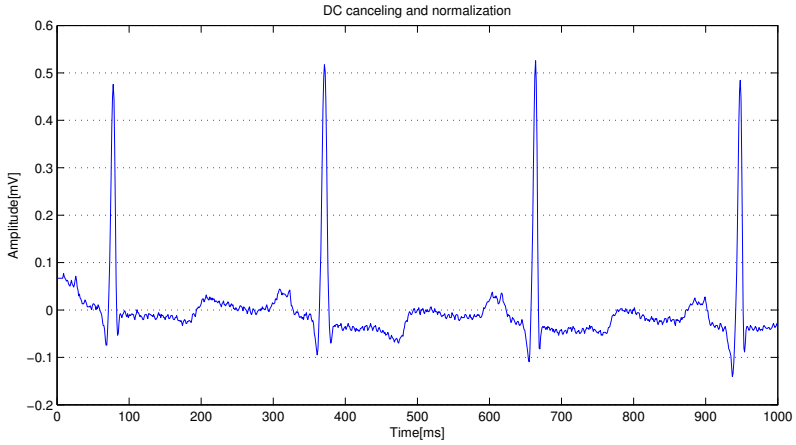
Filtr dolnoprzepustowy Filtracja dolnoprzepustowa $0Hz - 15Hz$ ma na celu usunięcie zakłóceń sygnału powstających pod wpływem drżenia mięśni ($35Hz$), wpływu zakłóceń sieci elektroenergetycznej $50/60Hz$, wpływu załamka T oraz pływającej izolinii elektrycznej. Powszechnie stosowany jest filtr o transmitancji (9).

$$H(z) = \frac{1}{32}\frac{(1 - z^{-6})^2}{(1 - z^1)^2} \tag{9}$$

Dla częstotliwości próbkowania $fs = 200Hz$, częstotliwość odcięcia dla filtru (9) wynosi $fc = 11Hz$. Filtr wprowadza opóźnienie 6 próbek ($30ms$). Dla $60Hz$ filtr tłumi sygnał na poziomie większym niż $35dB$.



Rysunek 14: Sygnał EKG oryginalny(100_ MLII.dat)



Rysunek 15: Sygnał EKG znormalizowany bez składowej stałej(100_ MLII.dat)

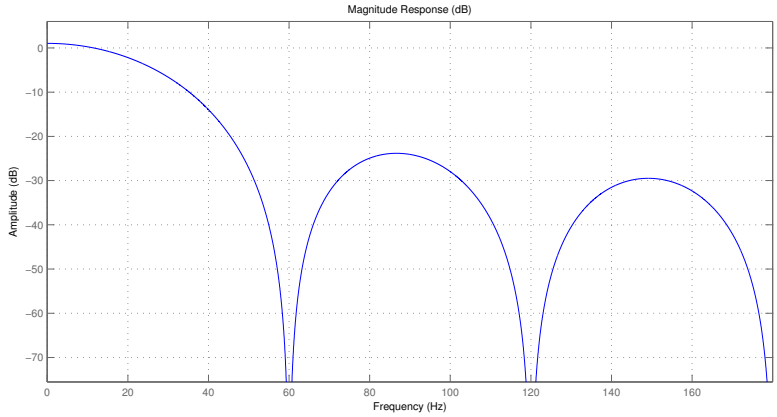
Rysunek (16) oraz (17) przedstawiają dokładne charakterystyki filtru dolnoprzepustowego. Filtr ten ma liniową odpowiedź fazową. Bardzo dobrze tłumi zakłócenia z linii energetycznych($50/60Hz$) oraz pozostałe zakłócenia na poziomie powyżej $25dB$.

Rysunek (18) przedstawia sygnał EKG po filtracji filtrem dolnoprzepustowym. Rezultat można porównać z oryginalnym sygnałem znajdującym się na rysunku (14), możemy dostrzec znaczną eliminację zakłóceń pochodzących od wyższych częstotliwości.

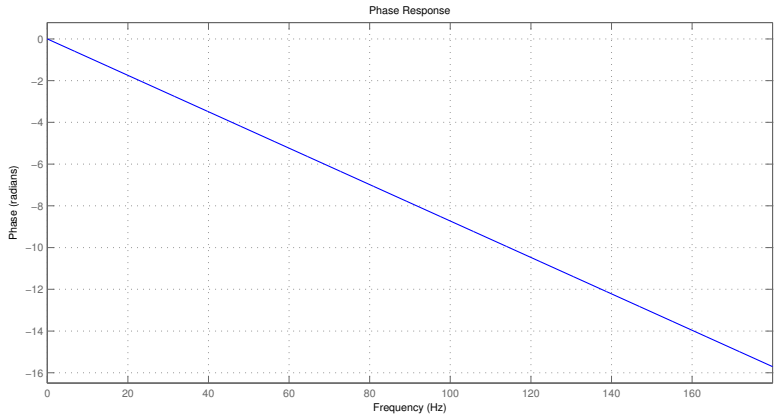
Filtr górnoprzepustowy Filtracja górnoprzepustowa ma na celu wyeliminowanie wszystkich niepożądanych sygnałów o niskich częstotliwościach. Transmitancja użytego filtru jest przedstawiona równaniem (10)

$$H(z) = 32z^{-16} - \frac{1 - z^{-32}}{1 - z^{-1}} \tag{10}$$

Dla częstotliwości próbkowania $fs = 200Hz$, częstotliwość odcięcia wynosi $fc = 5Hz$. Filtr wprowadza opóźnienie 16 próbek ($80ms$).



Rysunek 16: Odpowiedź amplitudowa filtru dolnoprzepustowego



Rysunek 17: Odpowiedź fazowa filtru dolnoprzepustowego

Rysunek (19) oraz (20) przedstawiają dokładne charakterystyki filtru górnoprzepustowego.

Różniczkowanie sygnału Po filtracji sygnału EKG, następnym krokiem jest różniczkowanie sygnału. Efektem takiego różniczkowania jest podkreślenie w sygnale nachyleń zespołów QRS oraz stłumienie załamków P i T. Przykładowa transmitancja do wykonania tej operacji jest przedstawiona równaniem (11).

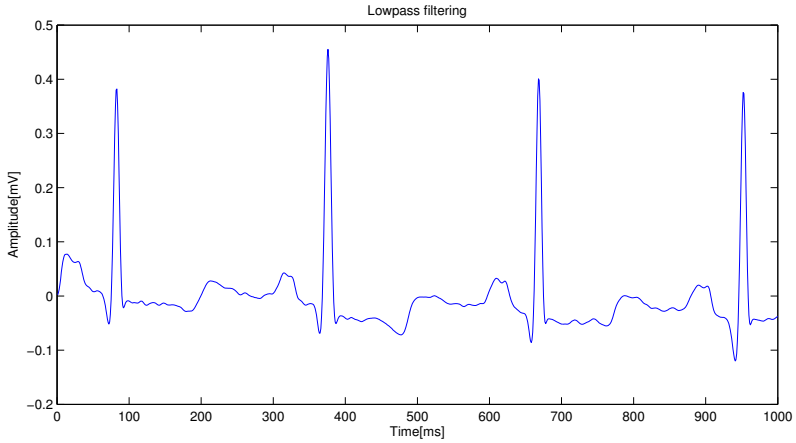
$$H(z) = 0.1(2 + z^{-1} - z^{-3} - 2z^{-4}) \tag{11}$$

Na rysunku (22) można dostrzec efekt różniczkowania sygnału.

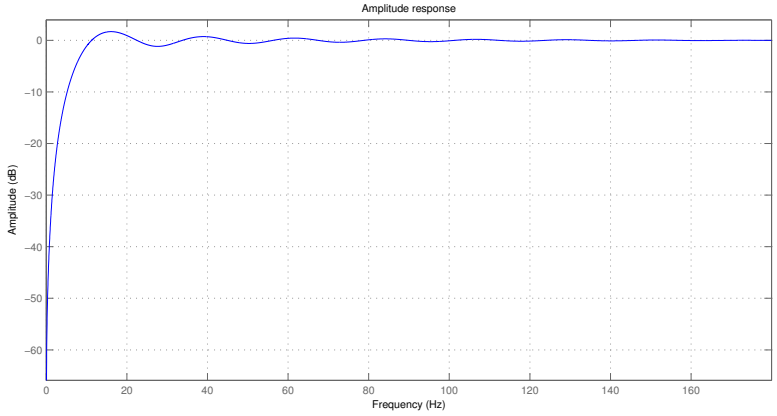
Potęgowanie sygnału Potęgowanie sygnału ma na celu jeszcze bardziej stłumić załamki P i T oraz wzmocnić próbki sygnału reprezentujące zespół QRS. Dodatkowo odwraca ujemną część sygnału i dzięki temu ujemne zespoły QRS również zostaną poddane analizie.

$$y(n) = (x(n))^2 \tag{12}$$

Efekt potęgowania sygnału możemy zaobserwować na rysunku (23).



Rysunek 18: Sygnał EKG po filtracji dolnoprzepustowej (100_ MLII.dat)

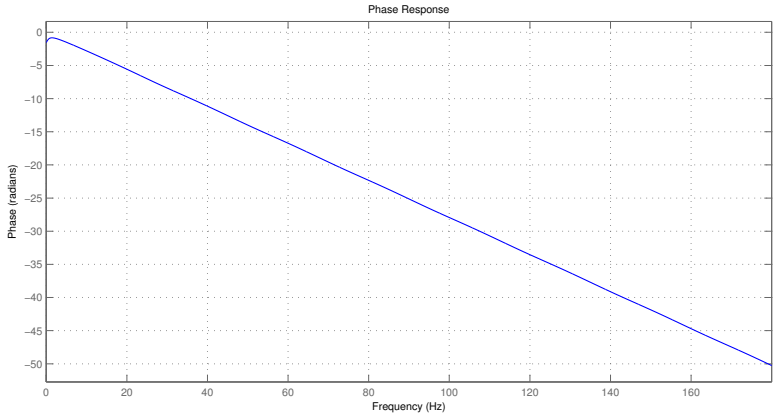


Rysunek 19: Odpowiedź amplitudowa filtru górnoprzepustowego

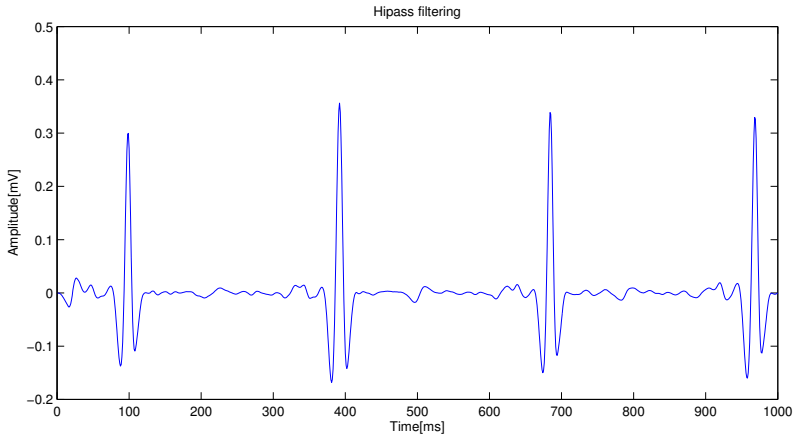
Całkowanie sygnału w ruchomym oknie Całkowanie sygnału w ruchomym oknie ma na celu uzyskanie pojedynczej "fali" w obrębie zespołu QRS. Bardzo ważną sprawą jest poprawne dobranie długości okna całkowania. Całkowanie sygnału w ruchomym oknie można uzyskać stosując wzór (13).

$$y(n) = \frac{1}{N}[x(n-(N-1)+)x(n-(N-2)+...+x(n)] \tag{13}$$

Na rysunku (24) możemy zobaczyć rezultat prawidłowego doboru długości okna, cały zespół QRS jest widoczny bez dodatkowych zbędnych informacji, na rysunku (25) widać efekt zbyt krótkiego okna całkującego, kompleks QRS kończy się już przed załamkiem R, na rysunku (26) widać efekt zbyt długiego okna całkowania, oprócz zespołu QRS widać jeszcze część sygnału poza nim.



Rysunek 20: Odpowiedź fazowa filtru górnoprzepustowego



Rysunek 21: Sygnał EKG po filtracji górnoprzepustowej(100_ MLII.dat)

Postać sygnału otrzymanego po całkowaniu ruchomym oknem jest przedstawiona na rysunku (27).

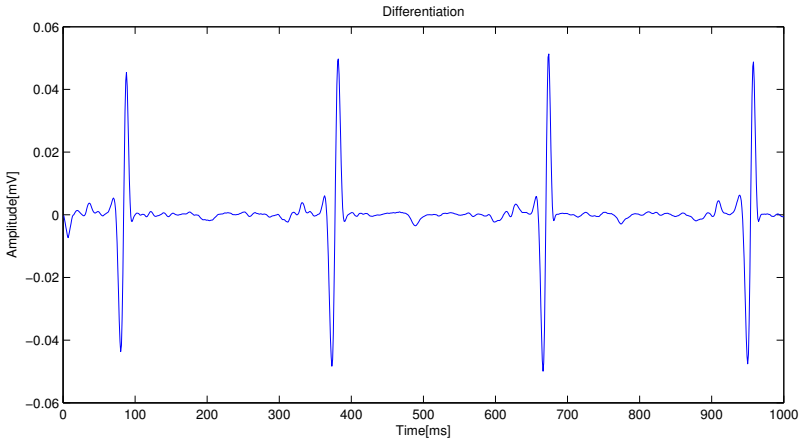
Progowanie i detekcja zespołów QRS Ostatnim krokiem metody —verb—PanTompkins jest detekcja progu detekcji oraz sama detekcja zespołu QRS. Próg detekcji może być ustalony na stałe bądź wyliczony według własnych wzorów.

Wykrywanie załamka R Aby odnaleźć załamek R należy teraz odnaleźć początek i koniec każdego kompleksu QRS w sygnale oryginalnym, używamy do tego punktów zapisanych w odnalezionym wektorze po operacji progowania. Mając taki wycinek oryginalnego sygnału, wyszukujemy wartości maksymalnej sygnału i oznaczamy ten punkt jako załamek R.
Na rysunku (28) można zobaczyć przykładową detekcję załamków R.

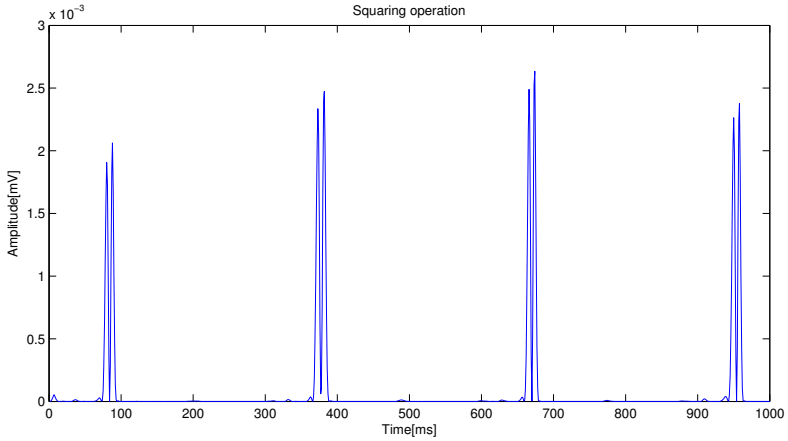
Hilbert

Usuwanie stałej DC i normalizacja sygnału Procedura operacji usuwania stałej DC i normalizacji sygnału przebiega w identyczny sposób jak opisano w procedurze "Pan-Thompkins"

Filtr dolnoprzepustowy Filtracja dolnoprzepustowa przebiega w identyczny sposób jak opisano w procedurze "Pan - Thompkins"



Rysunek 22: Sygnał EKG po różniczkowaniu (100_ MLII.dat)



Rysunek 23: Sygnał EKG po potęgowaniu (100_ MLII.dat)

Filtr górnoprzepustowy Filtracja górnoprzepustowa przebiega w identyczny sposób jak opisano w procedurze "Pan - Thompkins"

Różniczkowanie sygnału Po filtracji sygnału EKG, następnym krokiem jest różniczkowanie sygnału. Efektem takiego różniczkowania jest podkreślenie w sygnale nachyleń zespołów QRS oraz stłumienie załamków P i T. Pierwszą pochodną dla danego sygnału EKG można uzyskać za pomocą równania:

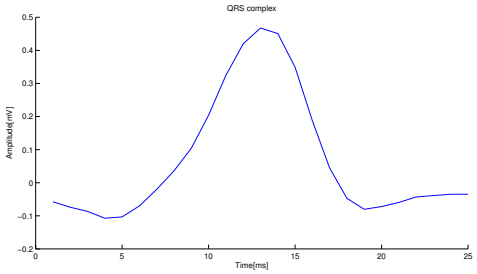
$$y(n) = \frac{1}{2\Delta t}[x(n+1) - x(n-1)], dlan = 0, 1, 2, 3..., m-1 \tag{14}$$

gdzie m jest całkowita liczbą próbek w sygnale, a Δt jest częstotliwością próbkowania. Początkowy stan jest określony przez $x(-1)$ kiedy n = 0, a końcowy $x(m)$, kiedy n=m-1. Warunki te minimalizują błąd na granicach.

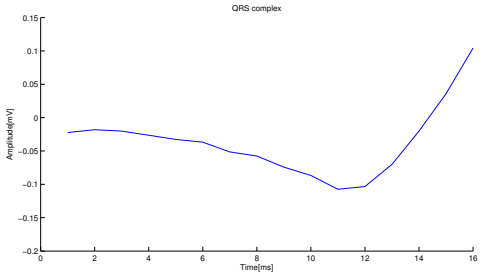
Podział sygnału w przesuwnym oknie Opisany poniżej algorytm transformaty Hilberta działa dobrze z krótkimi ciągami. Dlatego używa się okna o szerokości 1024 próbek do podziału zróżniczkowanego sygnału $y(n)$ przed uzyskaniem jego transformaty Hilberta. DO optymalizacji dokładności, początkowy punkt kolejnego okna powinien być zlokalizowany w punkcie ostatniego załamka R w poprzednim oknie.

Transformata Hilberta Transformata Hilberta rzeczywistej funkcji x(t) jest zdefiniowana jako:

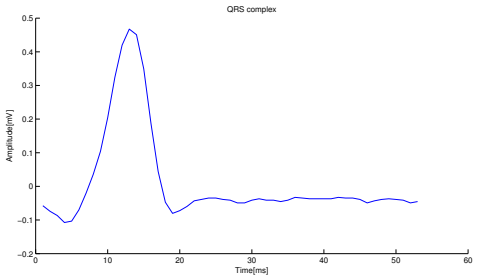
$$\hat{x} = H[x(t)] = \frac{1}{\pi}\int_{-\infty}^{\infty} x(\tau)\frac{1}{t-\tau}d\tau \tag{15}$$



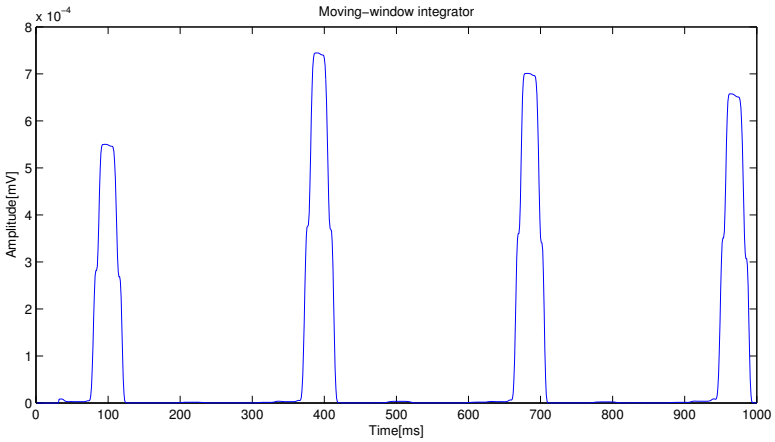
Rysunek 24: Prawidłowo wykryty kompleks QRS (100_ MLII.dat)



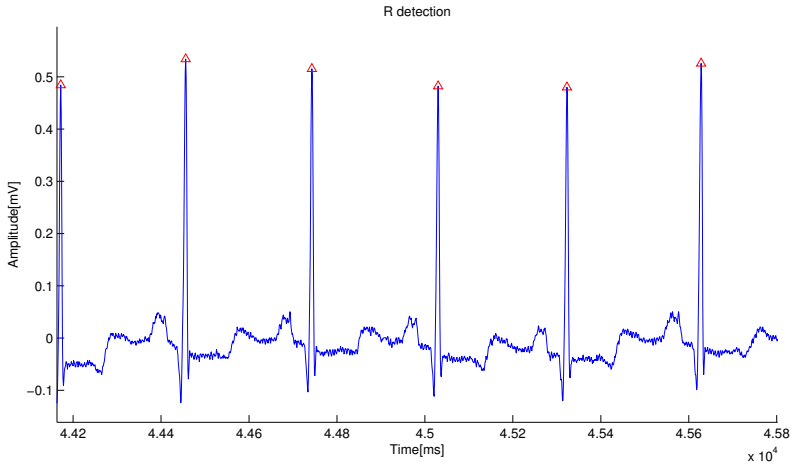
Rysunek 25: Za krótki wykryty kompleks QRS (100_ MLII.dat)



Rysunek 26: Za długi wykryty kompleks QRS (100_ MLII.dat)



Rysunek 27: Sygnał EKG scałkowany ruchomym oknem (100_ MLII.dat)



Rysunek 28: Detekcja załamków R (100_ MLII.dat)

Transformata Hilberta może być zinterpretowana jako konwolucja pomiędzy $x(t)$ i $\frac{1}{\pi t}$
Stosując transformatę Fouriera do (15), otrzymujemy:

$$F\{\hat{x}(t)\} = \frac{1}{\pi} F\{\frac{1}{t}\} F\{x(t)\} \tag{16}$$

Ponieważ

$$F\{\frac{1}{t}\} = \int_{-\infty}^{+\infty} \frac{1}{x} e^{-j2\pi f x dx} = -j\pi sgnf \tag{17}$$

gdzie $sgnf$ równe jest $+1$ dla $f > 0$, 0 dla $f = 0$ i -1 dla $f < 0$,
wtedy transformacja Fouriera transformaty Hilberta $x(t)$ jest podana przez (16) jako:

$$F\{\hat{x}(t)\} = -jsgnf F\{x(t)\} \tag{18}$$

W dziedzinie częstotliwości, wynik jest wtedy uzyskiwany przez przemnożenie widma $x(t)$ przez $j(+90^o)$ dla ujemnych częstotliwości i $-j(-90^o)$ dla dodatnich częstotliwości. Dziedzina czasowa może zostać uzyskana przez wyliczenie odwrotnej transformaty Fouriera. Dlatego, transformata Hilberta oryginalnej funkcji $x(t)$ reprezentuje jesgo harmoniczną koniugację.
Transformatę Hilberta $h(n)$ ciągu $y(n)$, który reprezentuje pierwszą pochodną sygnału EKG, jest uzyskiwana w według następujących kroków :

- Uzyskaj transformatę Fouriera $F(n)$ wejściowego sygnału $y(n)$,
- Przemnóż dodatnie oraz ujemne harmoniczne odpowiednio przez $-j$ i j,
- Wyzeruj stałą składową DC,
- Wylicz odwrotną transformatę Fouriera poprzedniego ciągu.

Adaptacyjny próg Szczyty w przetransformowanym sygnał metodą Hilberta $h(n)$ reprezentują obszary o wysokim prawdopodobieństwie wystąpienia prawdziwych załamków QRS. W praktyce, te szczyty różnią się od prawdziwych szczytów R o kilka milisekund. W celu zagwarantowania dokładności detekcji załamków R, detektor drugiego poziomu jest wymagany. Ponieważ fale P i T są zminimalizowane w relacji do odpowiednich załamków w zespole QRS w ciągu Hilberta, detekcja prostym progiem jest używana do zlokalizowania załamków w ciągu $h(n)$. Próg musi być adaptacyjny w celu zagwarantowania dokładności detekcji załamków R.
poziom progu jest ustalony według poniższych kryteriów:

- Początkowo poziom szumu w analizowanym podciągu jest określany równoważną wartością $RMS(RootMeanSquare)$ ciągu $h(n)$ i jego maksymalnej amplitudy w oknie,

- Jeżeli wartość RMS jest równa lub większa 18% maksymalnej wartości ciągu $h(n)$, poziom szumu w segmencie uważany jest za wysoki i dlatego poziom progu ustawiany jest na 39% maksymalnej amplitudy ciągu $h(n)$,
- Jeżeli maksymalna wartość rozważanego ciągu jest dwukrotnie większa od maksymalnej wartości amplitudy poprzedniego okna 1024 próbek, wtedy próg jest powiększany do 39% maksymalnej amplitudy poprzedniego okna $h(n)$,
- Kiedy wartość RMS jest niższa niż 18% maksymalnej wartości ciągu $h(n)$, poziom szumu uważany jest za niski i poziom progu zmniejszany o 1.6 raza wartości RMS,
- Jeżeli dwa załamki R w ciągu Hilberta $h(n)$ są zlokalizowane blisko siebie(w mniejszym odstępie czasowym niż $200ms$), tylko jeden załamek jest prawdopodobnie rzeczywistym załamkiem R,
- Decyzja jest podejmowana na podstawie amplitudy załamka i ich pozycji w stosunku do ostatniego załamka R zlokalizowanego za pomocą adaptacyjnego progu czasowego średniej długości interwału $R - R$ od poprzedniego zlokalizowanego załamka R.

3.2.3 Opis procedur i metod

Implementacja modułu wykrywania załamków `R` znajduje się w klasie `RPeaksDetector` która rozszerza abstrakcyjny moduł wykrywania załamków `R`. Klasa ta posiada implementację dwóch metod wykrywania załamków `R`:

- Pan-Tompkins
- Hilbert

Moduł na podstawie parametrów jakie otrzymuje z `GUI` wybiera odpowiednią metodę detekcji. Oprócz metody detekcji możliwe jest również ustawienie ręczne parametrów takich jak:

- dla metody Pan-Tompkins
  - próg detekcji
  - szerokość okna całkowania
- dla metody Hilbert
  - TODO JAKIE PARAMETRY

Użytkownik może też skorzystać z automatycznej detekcji powyższych parametrów. Automatyczna detekcja jest włączona jako standardowe parametry. Opis implementacji metod wykrywania załamków:

```
bool panTompkinsRPeaksDetection(ECGSignalChannel *signal);
```

Funkcja otrzymuje na wejściu przefiltrowany sygnał `ECGSignalChannel *signal` z modułu `ECGBaseLine`. Z założenia sygnał jest pozbawiony składowej stałej, szumów pochodzących z mięśni orz znormalizowany w zakresie od -1 do 1. Detekcja pomija więc wstępną filtracje która została wykonana w poprzednim module. Zadania wykonywane wewnątrz można podzielić na następujące etapy:

- różniczkowanie sygnału
- potęgowanie sygnału
- obliczanie szerokości okna całkowania
- całkowanie sygnału ruchomym oknem
- obliczanie progu detekcji

- wykrywanie zespołów QRS
- wykrywanie załamków R

Jeśli podczas wykrywania nie wystąpi żaden problem zostanie stworzony wektor z punktami w jakich zostały wykryte załamki `R`. Natomiast jeśli z jakiegoś powodu wykrycie załamków nie powiedzie się zostanie rzucony wyjątek `RPeaksDetectionExveption`.

```
bool hilbertRPeaksDetection(ECGSignalChannel *signal);
```

Funkcja również na wejściu otrzymuje sygnał z modułu `ECGBaseLine`. Zadania wykonywane podczas detekcji można podzielić następująco:

- różniczkowanie sygnału
- podział sygnału ruchomym oknem
- transformacja Hilberta
- wyznaczanie adaptacyjnego progu
- wykrywanie załamków R

Jeśli podczas wykrywania nie wystąpi żaden problem zostanie stworzony wektor z punktami w jakich zostały wykryte załamki `R`. Natomiast jeśli z jakiegoś powodu wykrycie załamków nie powiedzie się zostanie rzucony wyjątek `RPeaksDetectionExveption`

3.2.4 Warunki testowania

Początkowo moduł był testowany przy użyciu własnego przefiltrowanego sygnału testowego utworzonego w Matlabie. Testowy sygnał został pozbawiony składowej stałej, zostały usunięte zakłócenia filtrami: dolnoprzepustowym oraz górnoprzepustowym. Cały sygnał został znormalizowany do zakresu od -1 do 1.

Sygnał można wczytać przy pomocy funkcji jaka znajduje się w klasie `RPeaksDetector` o nazwie `getMockedSignal`. W pliku `RPeaksDetector.h` znajdują się makrodefinicje które w prosty sposób pozwalają włączyć debugowanie procesu wykrywania sygnału oraz użycie testowego sygnału.

Makrodefinicja `USE_MOCKED_SIGNAL` włącza użycie testowego sygnału. Sygnał jaki dostarcza moduł `ECGBaseLine` jest ignorowany. Dodatkowo makrodefinicja `DEBUG` pozwala na wypisanie podstawowych informacji z przebiegu wykrywania załamków `R`. Poniżej przedstawiamy przykładowy log zarówno dla metody `PanTompkins` jak i `Hilbert`:

```
Thersold size not found, use automatic calculated value
Input parameters for R peaks module:
Detection method: PanTompkins
Moving window size: 0
Thersold size: 0
R peaks module started
Use mocked signal for R-peaks module.
Running module with custom parameters
Convolution [-0.125 -0.25 0.25 0.125]
Orginal signal size: 600000
Exponentiation ^2
Signal size after convolution: 599996
Moving window integration
Calculating moving window size
```

Moving window size: 24
Signal size after exponentiation: 599996
Calculating detection thersold
After moving window integration signal size: 599972
Final max value: 0.0438604
Final mean value: 0.00221016
Current thresold value: 0.0109651
Looking for points over thersold
Detect begin and end of QRS complex
Number of left points: 2093
Number of right points: 2093
Final R peaks detection
Number of detected R-peaks: 2093
Done

R peaks module started
Running module with default parameters
Window size not found, use automatic calculated value
Thersold size not found, use automatic calculated value
Input parameters for R peaks module:
Detection method: Hilbert
R peaks module started
Running module with custom parameters
Number of detected R-peaks: 2272

Kiedy moduł `ECGBaseLine` dostarczył nam prawidłowo przygotowany sygnał, rozpoczęliśmy testowanie na plikach pochodzących z bazy `MIT_BIH`. Wyniki jakie otrzymaliśmy można znaleźć w kolejnym rozdziale.

3.2.5 Wyniki

Uzyskane wyniki wykrywania załamków `R` dla sygnałów pochodzących z bazy `MIT_BIH`.

sygnał	ilość R	wykryte PanTompkins	wykryte Hilbert	skuteczność PanTompkins	skuteczność Hilbert
100	2273	3001	2272	91,7%	99,9%
105	2572	2982	2559	85,1%	99,5%
107	2137	3803	2125	33,1%	99,4%
109	2532	2904	2492	85,4%	98,4%
111	1774	3535	2089	12,2%	83,3%
200	2601	3716	2593	66,8%	99,7%
202	2136	3711	2116	36,3%	99%
220	2955	2055	2047	70%	69,2%
223	2332	3187	2501	63,2%	93,2%
test100	2273	2274	2272	99,9%	99,9%

Uzyskane wyniki pomiarów wyraźnie pokazują przewagę metody `Hilberta`. W większości plików wynik skuteczności wahał się w około 98%. Metoda detekcji `PanTompkins` sprawia problemy głównie w sygnałach o dużych zakłóceniach oraz tam gdzie morfologia pochodzenia sygnału uderzenia jest inna niż zatokowo-przedsionkowa.

Ostatni sygnał 'test100' został wygenerowany z Matlaba, wczytaliśmy sygnał 100.dat, przefiltrowaliśmy, usunęliśmy składową stałą oraz znormalizowaliśmy. Tak przygotowany sygnał został wczytany

i przeprowadziliśmy detekcję. Jak widać skuteczność jest znacznie większa niż w przypadku oryginalnego sygnału gdzie przygotowany sygnał dostajemy z modułu `BaseLIne`.

Próbowaliśmy również testować wykrywanie zmieniając ostawienia modułu `BaseLine` wybierając inne filtracje. Wyniki jakie uzyskaliśmy były praktycznie identyczne jak powyżej dlatego nie zamieszczamy ich tutaj.

3.3 Waves

Autorzy: Agata Sitnik i Łukasz Zieńkowski.

3.3.1 Opis zadania

Celem modułu było wykrycie zespołów QRS oraz załamków P i T. Zespół QRS to największy zespół załamków EKG. Opisuje depolaryzację (pobudzenie) mięśni komór serca i składa się z jednego lub kilku załamków określanych kolejno jako Q, R i S, w zależności od miejsca wystąpienia i kierunku wychylenia. Czas trwania zespołu prawidłowo wynosi od 0,6-0,11 s i składa się z następujących składowych:

- `Załamek Q` – pierwsze ujemne wychylenie zespołu QRS
- `Załamek R` – pierwsze dodatnie wychylenie zespołu QRS
- `Załamek S` – każde ujemne wychylenie za załamkiem R

Poza zespołem QRS możemy wyróżnić również załamki P i T. Załamek P odpowiada depolaryzacji przedsionków. W warunkach prawidłowych jest on dodatni w odprowadzeniach I, II, aVF i ujemny w aVR; czas jego trwania jest krótszy od 0.12 s, a amplituda nie przekracza 2.5 mm w odprowadzeniach kończynowych. Odcinek PQ odpowiada repolaryzacji przedsionków. W warunkach prawidłowych repolaryzacja przedsionków nie powoduje przemieszczenia odcinka PQ w stosunku do odcinka TP i odcinek PQ przebiega w linii izoelektrycznej. Odstęp PQ jest elektrokardiograficznym odpowiednikiem czasu wędrowania bodźca z węzła zatokowego przez prawy przedsionek, węzeł przedsionkowo - komorowy, pęczek Hisa i włókna Purkinjego aż do mięśnia komór. W warunkach prawidłowych czas jego trwania zależy od wieku badanej osoby i częstotliwości rytmu serca mieszcząc się w granicach od 0.12 s do 0.20 s; u osób w starszym wieku za górną granicę normy można przyjąć wartość 0.23 s. Odcinek ST jest elektrokardiograficznym odpowiednikiem czasu wędrowania bodźca z węzła zatokowego przez prawy przedsionek, węzeł przedsionkowo - komorowy, pęczek Hisa i włókna Purkinjego aż do mięśnia komór. W warunkach prawidłowych czas jego trwania zależy od wieku badanej osoby i częstotliwości rytmu serca mieszcząc się w granicach od 0.12 s do 0.20 s; u osób w starszym wieku za górną granicę normy można przyjąć wartość 0.23 s. Załamek T odpowiada fazie szybkiej repolaryzacji mięśnia komór. Prawidłowy załamek T jest dodatni w odprowadzeniach I, II, aVL, V3 - V6 a ujemny w aVR.

3.3.2 Badania literaturowe

W literaturze można znaleźć kilka rozwiązań problemu odnalezienia zespołów QRS i załamków P i T. Większość z nich opiera się na dyskretnej transformacie falkowej, która polega na splocie sygnału pochodzącego z zapisu badania EKG z wybraną falką, pobraną z bazy. Baza jest stale rozwijana o nowe falki. Filtry cyfrowe analizy falkowej odpowiadają falce i funkcji skalującej w postaci tzw. spline'ów bramkowych drugiego stopnia o zwartym i krótkim nośniku. Dzięki temu podczas analizy sygnału i detekcji osobliwości możemy dokładniej kontrolować parametry procesu separacji wybranych częstotliwości. Dzięki analizie wieloskalowej możliwe jest zlokalizowanie miejsca gwałtownej zmiany sygnału, a tym samym lokalizacji zespołu QRS. Metoda posiada mniejszą wrażliwość na zmiany morfologii kolejnych zespołów QRS, minimalizuje problemy związane z występowaniem składowej wolnozmiennej, artefaktów ruchu i napięcia mięśni oraz pozwala na łatwiejszą separację załamka R w stosunku do załamków P i T.

Wiele rozwiązań zagadnienia detekcji zespołów QRS opiera się na filtracji adaptacyjnej, zastosowaniu ukrytych modeli Markowa, algorytmów genetycznych lub transformacie Hilberta. Wszystkie opierają się na pomyśle utworzenia detektora zespołów QRS, jednak pozwalają jedynie na drobną poprawę skuteczności lub szybkości działania, gdyż tak naprawdę nie znaleziono jeszcze w 100% skutecznej funkcji detekcyjnej. Istnieją dane literaturowe, w których detekcja zespołów QRS opiera się na stosowaniu filtrów wysokopasmowych. Filtry pełnią rolę narzędzia różniczkującego, a wykorzystywane są pierwsze i drugie pochodne sygnału. Różniczkowanie w tych metodach wykonuje się w celu uwypuklenia cech charakterystycznych sygnału. Zróżniczkowany sygnał poddaje się progowaniu. Pozwala to na wykrycie potencjalnych zespołów QRS, a dodatkowe progi pozwalają eliminować fałszywych kandydatów. Próg składa się z trzech parametrów: adaptacyjnej wartości slew-rate, drugiej wartości, która rośnie gdy w sygnale obserwuje się wzrost częstotliwości i trzeciej mającej na celu uniknięcie braku niskich wartości amplitudy. Zaletą tych metod jest ich prostota, a wadą niska efektywność i skomplikowane ciągi decyzyjne. Istnieją również metody łączące tosowanie filtrów wysokoprzepustowych z niskoprzepustowymi, pozwala to zwiększyć efektywność wykrywania zespołów QRS.

Jedną z najbardziej innowacyjnych metod automatycznej detekcji zespołów QRS jest stosowanie do tego celu sieci neuronowych typu wielowarstwowego perceptronu lub sieci LVQ. Budowa sieci MLP opiera się na konstrukcji z prostych neuronów o funkcji przejścia w postaci tangensa hiperbolicznego lub sigmoidy. Liczba warstw sieci nie powinna przekraczać 3, a liczba neuronów również powinna być ograniczona. Sieć neuronowa MPL działa na zasadzie próby przewidzenia stanu przyszłego w oparciu o stan poprzedni. Wadą tego rozwiązania jest to, że najpierw należy przeprowadzić uczenie sieci oraz duża wrażliwość na zaszumienie sygnału. Sieci typu LVQ działają inaczej: neurony środkowej warstwy uczą się rozpoznawania sygnałów określonego typu. Ich liczba określa liczbę grup sygnałów. Wadą sieci LVQ jest to, że jest bardzo rozbudowana oraz wymaga dosyć dużego zbioru testowego. Jednak dobrze nauczona sieć typu LVQ działa w zasadzie dla każdego zapisu EKG. Obie metody MLP i LVQ wymagają jednak dużego nakładu pracy przy tworzeniu sieci oraz ich uczeniu. Ważne jest również to, ze błędnie przeprowadzony sposób uczenia, może znacząco pogorszyć wykrywalność zespołów QRS.

W przypadku tego projektu zdecydowano się na wykorzystanie algorytmu treshold based detection.

3.3.3 Opis procedur i metod

Detekcja opiera się na algorytmie `treshold based detection`.
zastosowane funkcje:

```
bool QRSPointsDetector::detectQRS()
```

Główna funkcja odpowiedzialna za detekcję zespołów QRS. Funkcja korzysta ze znajomości położenia załamków R, uzyskanych z modułu Rpeaks. Określa ona minima po obu stronach tych załamków, następnie wykonuje normalizację sygnału, na którą składa się:

- odjęcie średniej z całego sygnału
- spotęgowanie (wzmocnienie)
- podzielenie przez maksymalną wartość występującą w sygnale (otrzymujemy sygnał w zakresie 0-1)

Po normalizacji w funkcji jest wydzielana część sygnału o intensywności powyżej 5%, które ma na celu wyzerowanie sygnału poza obszarem QRS. Funkcja służy do obliczenia parametrów `QRS_onset` i `QRS_end`.

```
bool QRSPointsDetector::detectPT()
```

Główna funkcja odpowiedzialna za detekcję załamków P i T. Funkcja najpierw wstępnie przetwarza otrzymany sygnał, wraz z oznaczonymi miejscami występowania zespołów QRS. Pierwszym etapem jest wyzerowanie sygnału w miejscach występowania QRS. Następnie wykonywana jest normalizacja sygnału składająca się z następujących etapów:

- Każdą część sygnału znajdującą się w okresie podzielić przez maksymalną wartość w danym okresie
- Filtracja sygnału
- Odjęcie mediany dla danego okresu od wszystkich wartości w okresie
- wydzielenie części sygnału o intensywności powyżej 7% (usunięcie sygnału poza QRS)

Funkcja następnie wykonuje właściwą detekcję załamków P i T. Ich wykrycie opiera się na następujących założeniach:

- Załamek T zawsze jest dłuższy niż 90 próbek (najmniejszy zanotowany to 95, a największy 208).
- Długość załamka jest większa niż 30% długości okresu.
- Odległość między końcem załamka T, a początkiem następnego QRS jest zawsze większa niż 25% długości okresu.
- Załamek P zawsze jest dłuższy niż 9 próbek (najmniejszy zanotowany to 10).
- Odległość między początkiem załamka P, a początkiem następnego QRS jest zawsze mniejsza niż 23% długości okresu.

W celu wyznaczenia załamków P i T w funkcji są najpierw wyznaczane początki załamków T jako końce poprzedzającego go zespołu QRS . W danym okresie sygnału wynikowego, funkcja znajduje pierwszą próbkę, której wartość jest różna od zera (dalej pod nazwą próbka pierwsza). Następnie znajduje pierwszą próbkę po tej poprzedniej, której wartość jest równa zero (dalej pod nazwą próbka druga). Jeżeli odległość pomiędzy początkiem następnego zespołu QRS, a tą pierwszą próbką jest większa niż 25% długości okresu i jednocześnie odległość między drugą próbką, a końcem poprzedzającego ją QRS jest większa niż 90, oznacza to, że druga próbka jest końcem załamka T. Pierwsze z tych założeń zapobiega zidentyfikowaniu załamka P jako T, w przypadku gdy T jest odwrócone i w sygnale wynikowym nie występuje. Drugie z założeń mówiące o szerokości załamka T, zapobiega zaklasyfikowaniu szumów o małej długości jako załamka T.

```
double findMinimum (ECGSignalChannel *signal, int forBegin, int forEnd)
```

Funkcja znajdująca minimum lokalne w danym zakresie sygnału, parametry forBegin i forEnd określają początek i koniec analizowanego okna.

```
double findMaximum (ECGSignalChannel *signal, int forBegin, int forEnd)
```

Funkcja znajdująca maksimum lokalne w danym zakresie sygnału, parametry forBegin i forEnd określają początek i koniec analizowanego okna.

```
void QRSPointsDetector::runModule(const ECGSignalChannel &filteredSignal,
 const ECGInfo &ecgi, const ECGRs &ecgRs, ECGWaves & ecgWaves)
```

Funkcja uruchamiająca moduł

Parametry: `&filteredSignal` – przefiltrowany sygnał
`& ecgi` – informacje o sygnale
`&ecgRs` – położenie załamków R
`& ecgWaves` – informacja zwrotna – położenie zespołów QRS oraz załamków P i T

```
ECGSignalChannel QRSPointsDetector::gradient(ECGSignalChannel * signal)
```

```
ECGSignalChannel QRSPointsDetector::averageFilter(ECGSignalChannel * signal)
```

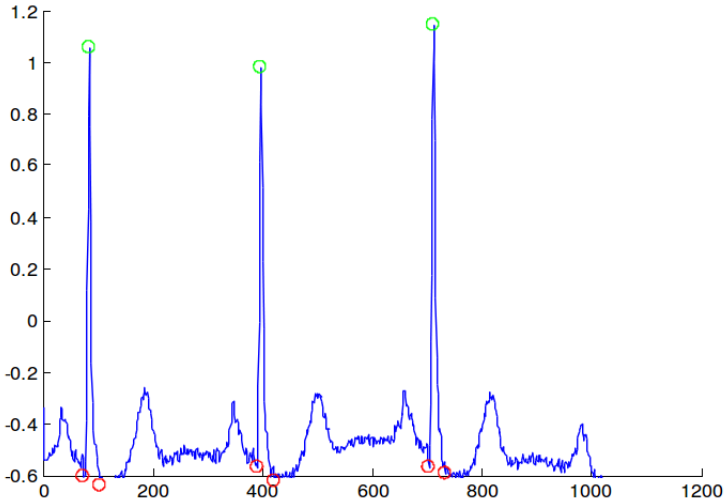
Funkcja realizująca filtr uśredniający sygnał

```
ECGRs QRSPointsDetector::getMockedRPeak()
```

Funkcja ustawiająca wartości załamków R, używana do testowania modułu.

```
ECGSignalChannel QRSPointsDetector::getMockedSignal()
```

Funkcja ustawiająca wartości sygnału, używana do testowania modułu.



Rysunek 29: Wyniki modułu Waves – Matlab, oznaczone zespoły QRS i załamki R

3.3.4 Warunki testowania

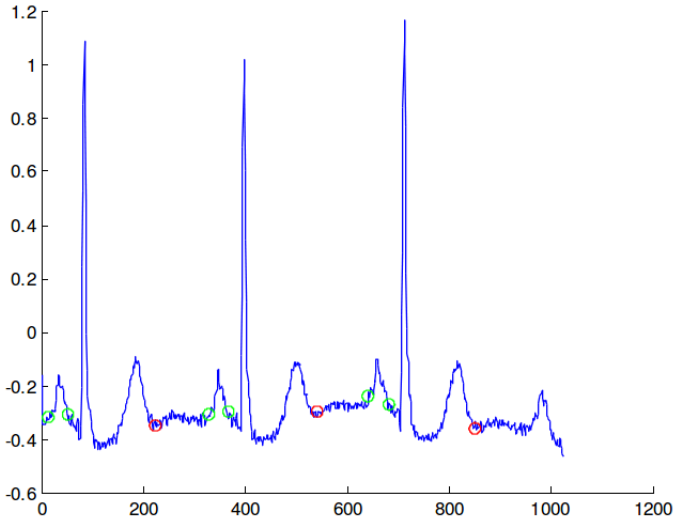
W celu przetestowania algorytmu najpierw wykonano implementację w programie Matlab 2009, sprawdzenie skuteczności algorytmu wykonano na sygnałach ściągniętych z bazy wykorzystywanej w poprzednim semestrze na zajęciach z przedmiotu Przetwarzanie sygnałów w systemach diagnostyki medycznej. Następnie kod przepisano w C++ w środowisku VisualStudio 2010 i tam testowano go za pomocą tych samych sygnałów. Posłużono się jako danymi wejściowymi również wygenerowanym wektorem położenia załamków R.

3.3.5 Wyniki

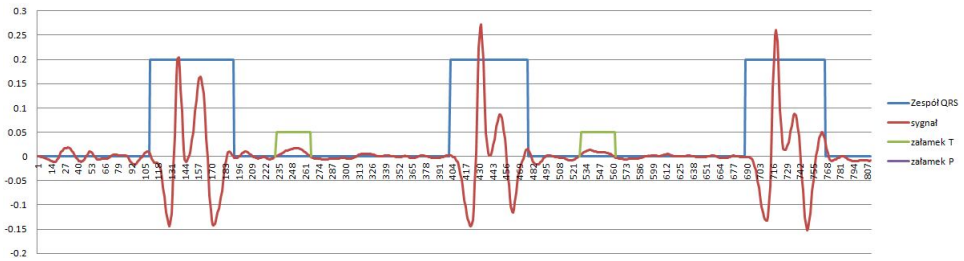
Udało się osiągnąć zadowalające wyniki, które nieznacznie odbiegają od danych przewidywanych. Wyniki osiągnięte podczas wstępnej implementacji w Matlabie, gdzie wykonano oznaczenia początków i końców zespołów QRS 29 oraz następnie oznaczono położenie załamków P i końców załamków T 30 są nieznacznie gorszej jakości niż uzyskane później w wynikowym programie 31 . Jest to spowodowane między innymi tym, że program w Matlabie był przygotowany jako prototyp, który później został dopracowany i rozszerzony o dodatkowe warunki klasyfikacji sygnału jako załamki oraz o dodatkową filtrację.

3.4 HRV1

Autorzy: Łukasz Jaromir i Leszek Sosnowski



Rysunek 30: Wyniki modułu Waves – Matlab, oznaczone załamki P, T i R



Rysunek 31: Wyniki modułu Waves – Visual Studio 2010

3.4.1 Opis zadania

Temat Analiza zmienności rytmu serca (HRV) #1

Opis Zmienność rytmu zatokowego opisuje różnice w długościach interwałów RR wyznaczanych przez kolejne szczyty zespołów QRS. Występowanie tych różnic świadczy o zdolności serca do adaptacji względem zewnętrznych bodźców i przedstawia informacje o pracy autonomicznego systemu nerwowego. Najprostsze ze stosowanych metod to analiza statystyczna w dziedzinie czasu oraz analiza częstotliwościowa. Celem projektu jest opracowanie i implementacja metod czasowych oraz częstotliwościowych analizy zmienności rytmu serca.

Dane ciąg próbek załamków R z modułu RPEAKS

Szukane moduł programu wyznaczający, wyświetlający parametry analizy czasowej i częstotliwościowej, a także rysujący postać częstotliwościową tachogramu wraz z naniesionymi zakresami parametrów HF, LF, VLF, ULF

3.4.2 Badania literaturowe

Ocena zmienności rytmu serca jest podstawową techniką umozliwiającą ocenę wpływu układu współczulnego (pobudzanie) i przywspółczulnego (hamowanie) na akcję serca. Ponieważ rola tych układów silnie zależy od stanu człowieka (sen, czuwanie) dopiero analiza całodobowych zmian rytmu serca pozwala diagnozować poprawność tego współdziałania.

Analiza zmienności rytmu serca opiera się na bieżących wartościach interwałów międzyuderzeniowych (odstępów RR) i może być prowadzona:

- w dziedzinie czasu (metodami statycznymi),
- metodami geometrycznymi (aproksymacja histogramu trójkątem),
- metodami częstotliwościowymi (z wykorzystaniem funkcji autokorelacji lub przekształcenia Fouriera).

Tachogram (po uzupełnieniu luk po zespołach QRS innych niż nadkomorowe) może być podstawą wyznaczania statycznych i geometrycznych współczynników zmienności akcji serca:

Oznaczenie:	Znaczenie wskaźnika
Średni RR	Wartość średnia ze wszystkich odstępów RR rytmu zatokowego
SDNN	odchylenie standardowe interwałów RR (ang. standard deviation normal-to-normal)
rMSSD	pierwiastek kwadratowy ze średniej kwadratów różnic pomiędzy kolejnymi dwoma interwałami RR
NN50	liczba interwałów RR, których różnica przekracza 50 ms
pNN50	odsetek różnic pomiędzy interwałami RR, które przekraczają 50 ms [%]
SDANN	odchylenie standardowe ze wszystkich średnich interwałów RR w 5 minutowych segmentach czasu całego zapisu [ms]
SDANNindex	średnia z odchyleń standardowych interwałów RR w 5 minutowych segmentach czasu całego zapisu [ms]
SDSD	odchylenie standardowe różnic pomiędzy dwoma sąsiadującymi interwałami RR [ms]

Analiza zmienności rytmu serca w dziedzinie częstotliwości wymaga rozwiązania problemu dotyczącego natury sygnału: tachogram jest dyskretną funkcją czasu próbkowaną niejednorodnie (interwał próbkowania nie jest stały). Konieczne jest więc odtworzenie ciągłej funkcji czasu na podstawie posiadanego tachogramu, a następnie ponowna jej dyskretyzacja w sposób jednorodny. Algorytm interpolacji, który wybraliśmy to algorytm interpolacji funkcjami sklejanymi trzeciego stopnia.

Kolejnym krokiem jest stosowanie przekształcenia Fouriera w celu otrzymania widma sygnału. Wykres widma dzielony jest zwykle na pasma, w których określana jest całkowita moc wyrażana

następnie przy użyciu współczynników:

Oznaczenie:	Znaczenie wskaźnika
TP	całkowita moc widma (większe lub równe 0.4 Hz)
HF	moc widma w zakresie wysokich częstotliwości (0.15 - 0.4 Hz)
LF	moc widma w zakresie niskich częstotliwości (0.04 - 0.15 Hz)
VLF	moc widma w zakresie bardzo niskich częstotliwości (0.003 - 0.04 Hz)
ULF	moc widma w zakresie ultra niskich częstotliwości (0.003 Hz)
LFHF	stosunek mocy widm w zakresie niskich częstotliwości do wysokich częstotliwości

Analiza częstotliwościowa dostarcza informacji o dynamicznej równowadze układu autonomicznego umożliwiając ocenę współpracy układu współczulnego i przywspółczulnego. Współczynniki analizy częstotliwościowej HRV dobrze reprezentują zjawiska zmienności [1].

3.4.3 Opis procedur i metod

Klasą realizującą obliczenia parametrów analizy czasowej i częstotliwościowej jest klasa HRV1Analyzer. Metody w klasie HRV1Analyzer:

- `HRV1Analyzer::HRV1Analyzer()` – konstruktor klasy
- `HRV1Analyzer::~HRV1Analyzer()` – destruktor klasy
- `void HRV1Analyzer::runModule(const ECGRs & r_peaks_data, ECGHRV1 & hrv1_data)` – metoda wywoływana przez kontroler służąca do uruchomienia modułu i realizacji wyliczenia współczynników
- `void HRV1Analyzer::prepareSignal()` – funkcja zajmująca się przygotowaniem sygnału – metoda przekształca tablicę z numerami próbek na tablicę zawierającą czas pomiędzy 2 załamkami
- `void HRV1Analyzer::prepareSigAbsolute()` – funkcja zajmująca się dalszym przygotowaniem sygnału – tworzy sygnał zawierający bezwzględny czas wystąpienia kolejnych załamków RR
- `void HRV1Analyzer::calculateParameters()` – właściwa metoda odpowiadająca za wyliczenie wszystkich współczynników ilościowych oraz częstotliwościowych, które składają się na właściwą analizę, za którą odpowiada moduł HRV1
- `void HRV1Analyzer::setParams(ParametersTypes &parameterTypes)` – metoda odpowiedzialna za ustawianie parametrów (w naszym module nie wykorzystywana)
- `ouble* HRV1Analyzer::doFFT(double* sigAfterSpline)` – metoda przeprowadzająca analizę FFT zinterpolowanego sygnału z wykorzystaniem biblioteki KISS FFT
- `kiss_fft_cpx* HRV1Analyzer::copycpx(double *mat, int nframe)` – funkcja alokująca klasę z danymi potrzebnymi do realizacji transformaty Fouriera
- `double* HRV1Analyzer::cubicSpline(double* x, double* y, int nframe)` – funkcja odpowiedzialna za interpolowanie próbek (wykonywana jest interpolacja funkcjami sklejanymi trzeciego stopnia). Zachodzi interpolacja funkcji o wektorach x i y – długość wektorów x i y jest taka sama i wynosi nframe
- `double HRV1Analyzer::mean(double *tab, int start, int end)` – funkcja obliczająca średnią arytmetyczną elementów z tablicy tab
- `double HRV1Analyzer::std(double *tab, int start, int end)` – funkcja obliczająca odchylenie standardowe

Klasą przechowującą współczynniki analizy częstotliwościowej i czasowej jest klasa ECGHRV1.

Szukając biblioteki odpowiedzialnej za realizację transformaty Fouriera skupiliśmy się na znalezieniu narzędzia maksymalnie prostego i szybkiego aby kod pozostał zoptymalizowany. Zdecydowaliśmy się na darmową bibliotekę KISS (Keep it Simple, Stupid).

Tymi samymi przesłankami kierowaliśmy się szukając biblioteki odpowiedzialnej za interpolowanie sygnału. Nasz wybór padł na bibliotekę algib.

3.4.4 Warunki testowania

Jako, że realizacja projektu odbywała się w grupach i tempo pracy było różne musieliśmy sobie poradzić i rozpocząć testowanie nie mając R_PEAKów. Do tego celu stworzyliśmy prosty algorytm, który emulował pracę całego kontrolera. Program ten na wejście podawał zmienne, które w przyszłości otrzymalibyśmy z grupy R_PEAKs – korzystaliśmy z materiałów, które otrzymaliśmy w poprzednim semestrze realizując w Matlabie laboratorium HRV1. Program zwracał natomiast policzone współczynniki analizy czasowej i częstotliwościowej.

W ten sposób, mogliśmy skutecznie pracować nad optymalizacją algorytmów porównując otrzymane wyniki z tymi, które pochodziły z prototypu napisanego w Matlabie, czekając na sprzężenie naszego modułu z modułem R_PEAKS.

3.4.5 Wyniki

Pierwszy etap prac, który zakończyliśmy implementując metody odpowiedzialne za obliczanie poszczególnych współczynników, pozwolił nam na ich porównanie do tych otrzymywanych z prototypu napisanego w środowisku Matlab. Były one zbliżone, ale nie satysfakcjonujące: brak algorytmu aproksymacji w sposób istotny zwiększał błędy.

Dopiero dalsze poprawki i optymalizacja, a przede wszystkim implementacja algorytmu aproksymacji skutecznie zminimalizował błędy, tak że wyniki przez nas otrzymane były zadowalające.

3.5 HRV2

Autor: Krzysztof Farganus

3.5.1 Opis zadania

Celem projektu jest opracowanie i implementacja metod geometrycznych analizy HRV.

Dane przyjmowane przez moduł:

- ciąg próbek załamków R z modułu R_PEAKS.

Dane zwracane przez moduł:

- wykres histogramu
- wskaźnik TINN
- Indeks Trójkątny
- wykres Poincare wraz z parametrami SD1 i SD2

3.5.2 Badania literaturowe

Techniki geometryczne służą do przedstawienia długookresowej zmienności rytmu serca [2]. Są łatwe do uzyskania, ponieważ bazują na aproksymacji histogramu trójkątem. Szerokość przedziałów klasowych histogramu ma tutaj kluczowe znaczenie, gdyż jej wartość wpływa na rezultaty metod geometrycznych. Głównie stosowany jest zakres wynoszący 7.8125 ms (1/128 s). Powodem jest częstotliwość próbkowania sygnału o najczęściej występującej wartości 128 Hz.

Cechy metod geometrycznych:

- odporność na zakłócenia ze względu na zastosowanie technik aproksymacyjnych
- eliminacja artefaktów zlokalizowanych poza trójkątem
- wyniki niezależne od jakości zapisu sygnału
- rezultaty zależne od czasu rejestracji
- wymagana duża liczba odstępów RR dla poprawnej analizy (minimalny czas zapisu – 20 minut)

Do najbardziej rozpowszechnionych metod należą:

- Wykres histogramu przedstawiający rozkład interwałów RR
- Indeks trójkątny (HRV triangular index) – całkowita liczba wszystkich odstępów RR podzielona przez liczbę odstępów RR o najczęściej spotykanym czasie trwania.
- Trójkątna interpolacja odstępów RR (TINN) – długość podstawy trójkąta aproksymującego histogram kolejnych odstępów interwałów RR rytmu zatokowego wyrażana w milisekundach.
- Wykres Poincare – graficzna reprezentacja korelacji pomiędzy kolejnymi interwałami , gdzie każdy odstęp RR jest opisany funkcją RR+1. Do analizy rozproszenia punktów na wykresie stosuje się dwa deskryptory: SD1 oraz SD2, odpowiadające odchyleniom standardowym. Pierwszy charakteryzuje rozkład punktów w poprzek linii identyczności, natomiast drugi wzdłuż tej linii.

Algorytm obliczania wskaźnika TINN [5]

Aby obliczyć parametr TINN, czyli wyznaczyć wartości punktów N i M należy opracować funkcję multiliniową q(t) o postaci:

- $q(t) = 0$ dla $t \leqslant N$
- $q(t) = 0$ dla $t \geqslant M$
- $q(X) = Y$ w pozostałych przypadkach

a następnie znaleźć minimum z całki o wzorze:

$$\int_0^{+\infty} (\ D(t)) - q(t)\ )^2 dt \tag{19}$$

spośród wszystkich kombinacji punktów (N,M) , gdzie D(t) to wartość histogramu. W układzie dyskretnym poprzedni wzór wygląda następująco:

$$\sum (\ D(t) - q(t)\ )^2 \rightarrow \text{minimum} \tag{20}$$

przy czym dla $t\,\epsilon(0, N)$ oraz $t\,\epsilon(M, \infty)$ ma postać:

$$D(t)^2 \tag{21}$$

natomiast dla $t\,\epsilon\,\langle 0, N\rangle$ wygląda następująco:

$$(D(t) - q(t))^2 \tag{22}$$

Algorytm obliczania parametrów SD1 i SD2
Parametry SD1 i SD2 są wyznaczane według poniższych wzorów:

$$SD1 = \sqrt{\frac{1}{2} \cdot SDSD^2} \tag{23}$$

$$SD2 = \sqrt{2 \cdot SDNN^2 - \frac{1}{2} \cdot SDSD^2} \tag{24}$$

gdzie:

- SDNN to odchylenie standardowe interwałów RR: $SDNN = \sqrt{\frac{1}{N-1} \cdot \sum_{i=1}^{N} \left(\bar{RR} - RR_i\right)^2}$
- SDSD to odchylenie standardowe różnic pomiędzy dwoma sąsiadującymi interwałami: $SDSD = \sqrt{E\left\{\triangle RR_j^2\right\} - E\left\{\triangle RR_j\right\}^2}$

3.5.3 Opis procedur i metod

`GeometricAnalysis::runModule` – wirtualna funkcja uruchamiająca moduł HRV2.

Argumenty funkcji:

- `ECGInfo info` – dane o wczytanym sygnale
- `ECGRs ecgRs` – wektor numerów próbek zawierający załamki R
- `ECGHRV2 ecgHRV2` – instancja klasy przechowującej wyniki analizy zmienności rytmu serca metodami geometrycznymi

Funkcja zwraca:
Funkcja przekazuje wyniki metod geometrycznych analizy HRV do instancji klasy `ECGHRV2`.

Używane funkcje:

- `PrepareRRSignal`
- `MakeHistogramAndGeometricalParams`
- `MakePoincareAndSDParams`
- `SetHRV2Params`

Używane zmienne:

- `ECGRs rpeaks` – atrybut klasy `GeometricAnalysis` przechowujący wektor numerów próbek z załamkami R
- `double SamplingInterval` – atrybut klasy `GeometricAnalysis` przechowujący częstotliwość wczytanego sygnału

`GeometricAnalysis::PrepareRRSignal` – funkcja przekształca wektor numerów próbek z załamkami R na wektor interwałów RR w milisekundach.

Argumenty funkcji:

- `rpeaks` – wektor numerów próbek z załamkami R

Funkcja zwraca:
Funkcja zapisuje wektor interwałów RR w atrybucie klasy `RR_intervals`.

Używane funkcje:

- `gsl_vector_int_get` – metoda GSL pobierająca całkowitą wartość danego elementu z wektora
- `gsl_vector_set` – metoda GSL zapisująca zmiennoprzecinkową wartość w danym elemencie wektora
- `gsl_vector_scale` – metoda GSL mnożąca każdy element wektoru przez liczbę zmiennoprzecinkową

Używane zmienne:

- `unsigned int rpeaks_size` – długość wektora z numerami próbek załamków R
- `double SamplingInterval` – częstotliwość wczytanego sygnału

`GeometricAnalysis::MakeHistogramAndGeometricalParams` – funkcja tworzy wektor wartości histogramu, oblicza wysokość i pozycję kolumny histogramu reprezentującą najczęściej powtarzający się interwał RR, wylicza długość oraz pozycję podstawy trójkąta aproksymującego, oszacowuje wartość indeksu trójkątnego.

Argumenty funkcji:

- `RR_intervals` – wektor interwałów RR w milisekundach

Funkcja zwraca:

- histogram_x – pozycje kolumn histogramu
- histogram_y – wysokości kolumn histogramu
- X – pozycja najwyższej kolumny histogramu
- Y – wysokość najwyższej kolumny histogramu
- N – początek podstawy trójkąta aproksymującego
- M – koniec podstawy trójkąta aproksymującego
- HRVTriangularIndex – indeks trójkątny
- TINN – wskaźnik TINN

Używane funkcje:

- `gsl_vector_max` – metoda GSL zwracająca największy element danego wektora
- `gsl_vector_min` – metoda GSL zwracająca najmniejszy element danego wektora
- `gsl_vector_int_max_index` – metoda GSL zwracająca pozycję największego elementu danego wektora

- `gsl_vector_int_get` – metoda GSL pobierająca całkowitą wartość danego elementu z wektora
- `gsl_vector_get` – metoda GSL pobierająca zmiennoprzecinkową wartość danego elementu z wektora
- `gsl_vector_set` – metoda GSL zapisująca zmiennoprzecinkową wartość w danym elemencie wektora
- `gsl_vector_int_set` – metoda GSL zapisująca całkowitą wartość w danym elemencie wektora
- `gsl_interp_alloc` – metoda GSL tworząca wskaźnik do nowo utworzonego obiektu interpolacji
- `gsl_interp_accel_alloc` – metoda GSL tworząca wskaźnik na obiekt iteratora do wyszukiwania interpolacji
- `gsl_interp_init` – metoda GSL wyliczająca funkcję interpolującą
- `gsl_interp_eval` – metoda GSL zwracająca punkt funkcji interpolującej

Używane zmienne:

- `double RRmax` – najdłuższy interwał RR
- `double RRmin` – najkrótszy interwał RR
- `IntSignal Histogram` – tymczasowy wektor punktów histogramu
- `unsigned Histogram_size` – liczba wszystkich kolumn histogramu
- `unsigned int RR_intervals_size` – liczba wszystkich interwałów RR
- `double minimum` – zmienna przechowująca minimalną wartość całki z algorytmu wyznaczania TINN
- `double x[3], y[3]` – tablica współrzędnych trójkąta aproksymującego

`GeometricAnalysis::MakePoincareAndSDParams` – funkcja wylicza punkty wykresu Poincare oraz parametry: SD1 i SD2.

Argumenty funkcji:

- `RR_intervals` – wektor interwałów RR w milisekundach

Funkcja zwraca:

- poincare_x – współrzędne osi OX wykresu Poincare
- poincare_y – współrzędne osi OY wykresu Poincare
- parametr SD1
- parametr SD2

Używane funkcje:

- `gsl_vector_get` – metoda GSL pobierająca zmiennoprzecinkową wartość danego elementu z wektora
- `gsl_vector_int_set` – metoda GSL zapisująca całkowitą wartość w danym elemencie wektora
- `gsl_stats_sd` – metoda GSL wyliczająca odchylenie standardowe
- `gsl_vector_int_sub` – metoda GSL odejmująca dwa wektory

Używane zmienne:

- `unsigned int RR_intervals_size` – liczba wszystkich interwałów RR
- `IntSignal diff` – wektor różnic pomiędzy sąsiednimi interwałami RR

`GeometricAnalysis::SetHRV2Params` – funkcja przekazuje wyniki analizy HRV metodami geometrycznymi do instancji klasy ECGHRV2.

Argumenty funkcji:

- `ECGHRV2 &hrv2`

Funkcja zwraca:
Funkcja zwraca wyniki analizy geometrycznej do `ECGHRV2 &hrv2`.

Używane funkcje:

- wszystkie funkcje pozwalające na zapis wyników do atrybutów klasy ECGHRV2.

Używane zmienne:
Funkcja nie używa dodatkowych zmiennych.

`GeometricAnalysis::MakeRRsignal` – funkcja tworząca testowy wektor interwałów RR w milisekundach.

Argumenty funkcji:
Nie przyjmuje argumentów.

Funkcja zwraca:
Funkcja zwraca testowy wektor interwałów RR jako instancję klasy ECGRs.

Używane funkcje:

- `gsl_vector_int_set` – metoda GSL zapisująca całkowitą wartość w danym elemencie wektora
- `setRs` – zapisuje wektor w instancji klasy ECGRs

Używane zmienne:

- `int RRsignal_length` – długość wektora interwałów RR
- `int RRsignal[]` – tablica wartości interwałów RR

`GeometricAnalysis::setParams` – funkcja modyfikująca szerokość kolumny histogramu.

Argumenty funkcji:

- `ParametersTypes &parameterTypes` – zawiera dane ustawień poszczególnych modułów

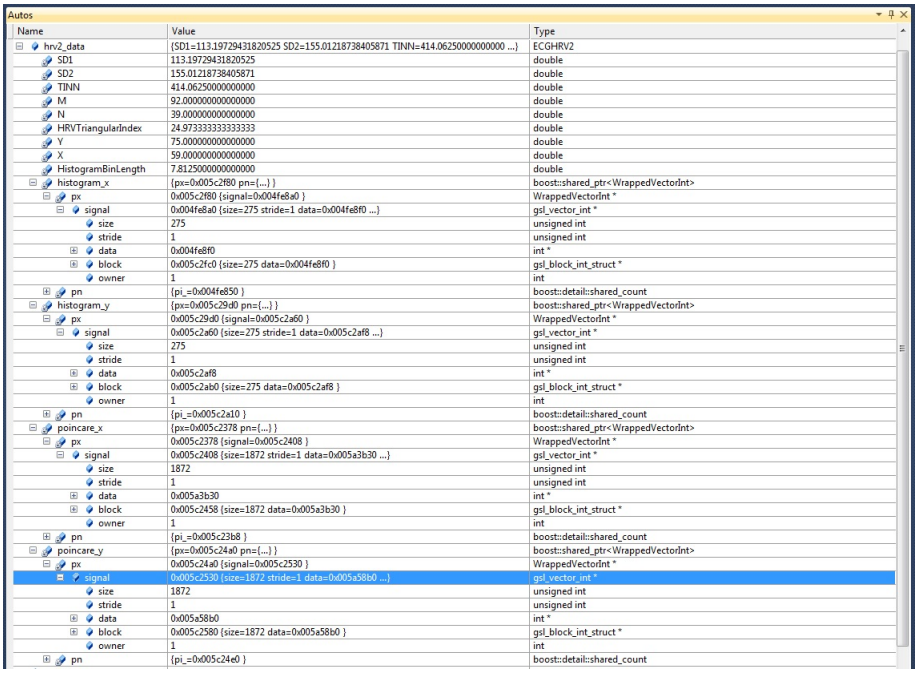
Funkcja zwraca:
Nie zwraca niczego.

Używane funkcje:

- `parameterTypes.find(histogram_bin_length)` - wyszukuje parametr modyfikujący szerokość kolumny histogramu

Używane zmienne:

- `double HistogramParameter` – zmienna przechowująca rezultat funkcji `parameterTypes.find`.
- `double HistogramBinLength` – atrybut klasy `GeometricAnalysis` przechowujący szerokość kolumny histogramu.



Rysunek 32: Wyniki modułu HRV2 – Visual Studio Ultimate 2010

3.5.4 Warunki Testowania

Podczas procesu implementacji, moduł HRV2 poddawany był dwóm rodzajom testów:

- Pierwszy test polegał na porównaniu wyników działania modułu zaimplementowanego w Matlabie oraz Visual Studio Ultimate 2010. W tym celu jako dane wejściowe wykorzystano wektor interwałów RR wczytywany z pliku tekstowego dla Matlaba z tablicy jednowymiarowej dla Visual Studio.
- Drugi test sprawdzał działanie programu, gdy ten korzystał z testowych wyników modułu R_PEAKS.

3.5.5 Wyniki

Rysunki 32, 33 oraz 34 prezentują wyniki pierwszego testu opisanego w poprzedniej podsekcji:

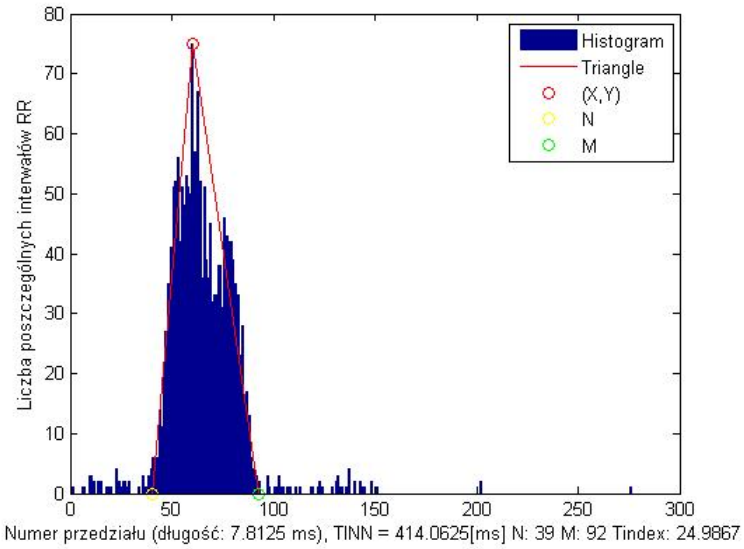
3.6 HRV DFA

Autorzy: Mikołaj Rzepka i Szczepan Czaicki

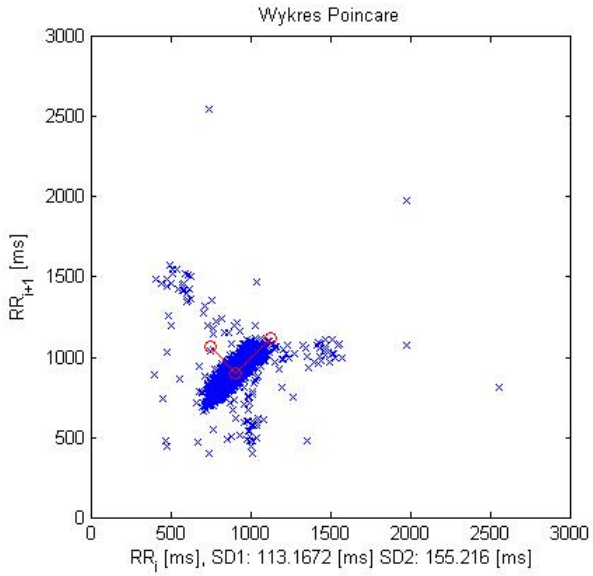
3.6.1 Opis zadania

Temat Beztrendowa analiza fluktuacji tachogramu (ang. Detrended Fluctuation Analysis)

Opis Poszukiwanie wciąż nowych rozwiązań oceny ilościowej tachogramu doprowadziło do opracowania nowej metody — beztrendowej analizy fluktuacji, która to jest rozszerzeniem zwyczajnej analizy fluktuacji. Podejście to sprawdziło się w „odkrywaniu" rozległych i długotrwających korelacji w sygnale EKG. Zasadniczą zaletą tego podejścia jest możliwość zastosowania algorytmu



Rysunek 33: Wykres histogramu – Matlab



Rysunek 34: Wykres Poincare – Matlab

dla szeregów czasowych, których podstawowe statystyki mają charakter niestacjonarny (zmieniają się w czasie). Celem projektu jest opracowanie i implementacja metody wyznaczającej parametry DFA.

Dane ciąg próbek załamków R z modułu R_PEAKS

Szukane moduł programu wyznaczający, wyświetlający parametry analizy DFA w postaci tabel oraz rysujący wykresy analizy DFA

3.6.2 Badania literaturowe

Beztrendowa analiza fluktuacyjna polega na określeniu zależności błędu liniowej interpolacji $F(n)$ dla odpowiednio zdefiniowanego szeregu od wielkości okna interpolacji n. Metoda ta bierze pod uwagę różnice w lokalnej zawartości danych i może być stosowana dla jednoczesnej analizy całego szeregu danych zawierającego wiele niejednolitych obszarów. W przeciwieństwie do funkcji f(n) analizy fluktuacyjnej, mającej zmienne nachylenie nawet dla n mniejszego od wielkości typowego obszaru stacjonarnego, funkcja beztrendowa F(n) wykazuje nachylenie stałe przez kilka rzędów wielkości. Metoda DFA znakomicie wykazała różnice między sekwencjami kodującymi a niekodującymi DNA. Użyto więc metody również do wykrywania korelacji w zjawiskach pogodowych, ekonofizyce (np. fluktuacje wartości indeksów giełdowych) oraz do sygnałów fizjologicznych, przede wszystkim zmienności rytmu serca. Metoda beztrendowej analizy fluktuacji (ang. Detrended Fluctuation Analysis, DFA) polega na powtórnej analizie sygnału w blokach o długościach kolejno rosnących, aż do określonej wartości N. W pierwszej kolejności tachogram jest całkowany zgodnie ze wzorem:

$$\sum_{i=1}^{N}(RR(i) - RR_{avg}) \tag{25}$$

gdzie $RR(i)$ to i-ty interwał, zaś RR_{avg} to średni interwał między kolejnymi uderzeniami (w dzień średnio około 0.7 sekundy). Następnie tak scałkowany sygnał podzielony zostaje na bloki o określonych długościach n i w każdym z bloków następuje wyznaczenie trendu lokalnego $y_n(k)$ metodą najmniejszych kwadratów. Skorzystano w tym celu ze wzorów na współczynniki linii trendu a i b:

$$\frac{S \cdot S_{xy} - S_x \cdot S_y}{\Delta} \tag{26}$$

$$b = \frac{S_{xx} \cdot S_y - S_x \cdot S_{xy}}{\Delta} \tag{27}$$

Wykorzystane we wzorach na współczynniki prostej oznaczenia rozpisano poniżej:

$$S = \sum_{i=1}^{n} 1 = n \tag{28}$$

$$S_x = \sum_{i=1}^{n} x_i \tag{29}$$

$$S_y = \sum_{i=1}^{n} y_i \tag{30}$$

$$S_{xx} = \sum_{i=1}^{n} x_i^2 \tag{31}$$

$$S_{xy} = \sum_{i=1}^{n} x_i y_i \tag{32}$$

$$S_{yy} = \sum_{i=1}^{n} y_i^2 \tag{33}$$

$$\Delta = S \cdot S_{xx} - (S_x)^2 \tag{34}$$

Proces jest powtarzany dla coraz większych bloków, aż do pełnej długości N . Wówczas można wyznaczyć zależność fluktuacji od rozmiaru bloku jako

$$F(n) = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y(k) - y_n(k))^2} \tag{35}$$

Zależność ta może zostać umieszczona na wykresie z logarytmiczno-logarytmicznym. Można zdefiniować współczynnik skalowania

$$\alpha = \lim_{n\to\infty} \frac{\log F(n)}{\log n} \tag{36}$$

W zależności od wartości wyliczonego współczynnika możemy wyciągnąć różne wnioski odnoszące się do sygnału wejściowego:

- Dla $0 < \alpha < 0.5$ obserwuje się długozasięgowe antykorelacje (po krótszym interwale bardziej prawdopodobne jest wystąpienie dłuższego), natomiast dla $0.5 < \alpha < 1$ dodatnie korelacje długozasięgowe,
- Przypadek $\alpha = 1$ odpowiada szumowi $1/f$
- Dla $\alpha \geqslant 1$ korelacje istnieją, lecz przestają mieć charakter potęgowy,
- Wartość $\alpha = 1.5$ odpowiada scałkowanemu sygnałowi widma białego, czyli błądzeniu przypadkowemu

3.6.3 Opis procedur i metod

Główna funkcjonalność modułu znajduje się w klasie `DFAAnalyzer`. Klasa ta, według przyjętego schematu rozszerza abstrakcyjny moduł analizy beztrendowej analizy fluktuacji tachogramu.

Lista i opis najważniejszych funkcji:

```
void DFAAnalyzer::runModule (const ECGRs &ecgRs, ECGHRVDFA &ecghrvdfa)
```

Wirtualna funkcja uruchamiająca moduł HRV DFA. Ustawia instancje klas, w których przechowuje dane i uruchamia fukcję `calcDFA` obliczającą podstawowe parametry związane z beztrendową analiza fluktuacji tachogramu.

Argumenty funkcji:

- `ECGRs &ecgRs` – wektor numerów próbek zawierający załamki R
- `ECGHRVDFA &ecghrvdfa` – instancja klasy przechowującej wyniki analizy HRV DFA

```
void DFAAnalyzer::calcDFA()
```

Funkcja obliczająca podstawowe parametry związane z beztrendową analiza fluktuacji tachogramu. Najpierw wylicza z sygnału długość interwałów RR, następnie całkuje otrzymany sygnał i wywołuje funkcję `calc_all_trends` wyliczającą parametry linii trendów dla kolejnych wielkości okien. Następnie funkcja przelicza wartości fluktuacji w kolejnych oknach, wpisuje je do instancji klasy `ECGHRVDFA` w celu ich późniejszego wyrysowania. Na końcu obliczana jest wartość współczynnika skalowania α, również wpisywana do instancji klasy `ECGHRVDFA`.

```
trend_coefs** DFAAnalyzer::calc_all_trends(double* rr_integrated,
        int length)
```

Funkcja wyliczająca parametry linii trendów dla kolejnych wielkości okien sygnału `rr_integrated`. Dzieli sygnał na okna o wielkości od 2 próbek do całej długości sygnału i dla każdego okna wywołuje funkcję `calc_trend` obliczającą parametry a i b linii trendu metodą najmniejszych kwadratów.

Argumenty funkcji:

- `double* rr_integrated` – sygnał (tablica) interwałów RR
- `int length` – długość tablicy interwałów RR

```
trend_coefs* DFAAnalyzer::calc_trend(double* rr_integrated, int start,
        int w_length)
```

Funkcja obliczająca parametry a i b linii trendu metodą najmniejszych kwadratów dla podanego sygnału od próbki `start` w oknie długości `w_length`.

Argumenty funkcji:

- `double* rr_integrated` – sygnał (tablica) interwałów RR
- `int start` – numer próbki sygnału interwałów RR od którego funkcja zaczyna przeliczenia
- `int w_length` – ilość próbek sygnału interwałów RR brana do przeliczeń począwszy od próbki `start`

3.6.4 Warunki Testowania

Grupa wykonująca zadania były uzależniona od wyników grupy implementującej moduł R_PEAKS. W związku z tym, aby uniezależnić się od pracy tej grupy, do testowania algorytmu używano sygnałów zawierających piki R pobranych z internetu. Wyniki porównywano z wynikami uzyskiwanymi z wyliczeń skryptu napisanego w Matlabie w poprzednim semestrze.

3.6.5 Wyniki

W pierwszym etapie prac porównywano wyniki pracy algorytmu do wyników uzyskiwanych z wyliczeń skryptu napisanego w Matlabie, udało się osiągnąć takie same rezultaty. Na efekty wizualizacji wyników czekaliśmy do samego końca, z powodu braku czasu zaimplementowano wizualizację jednego wykresu: zależności logarytmu fluktuacji od logarytmu z wielkości okna.

3.7 Klasyfikacja zespołów QRS

Autorzy: Krzysztof Bębenek

3.7.1 Opis zadania

Temat Metody detekcji morfologicznego pochodzenia zespołu QRS.

Opis Proces automatycznego klasyfikowania zespołów QRS należy do jednego z trudniejszych procesów podczas przetwarzania sygnałów EKG. Jest jednak on niezbędny w dalszych etapach analizy podczas których brane są pod uwagę tylko niepoprawne pobudzenia. Do prostych sposób oceny morfologi pobudzenia służą:

- współczynnik kształtu Malinowskiej,
- stosunek części ujemnej do dodatniej sygnału,

Dane ciąg próbek przefiltrowanego sygnału EKG, wektory QRS_onset oraz QRS_end określające początek oraz koniec zespołu QRS

Szukane moduł programu klasyfikujący zespołu QRS na podstawie ich morfologii

3.7.2 Badania literaturowe

3.7.3 Opis procedur i metod

3.7.4 Warunki testowania

3.7.5 Wyniki

3.8 ST interval

Autorzy: Bartłomiej Bułat i Krzysztof Piekutowski

3.8.1 Opis zadania

Celem projektu jest wyznaczenie odcinków ST, pomiar poziomu odcinka ST względem linii izoelektrycznej i jego nachylenia, a także detekcja epizodów ST oraz parametry ilościowe i jakościowe epizodów ST. Celowość analizy odcinka ST względem linii izoelektrycznej związana jest z predykcją choroby wieńcowej, miażdżycy oraz niedotlenieniem mięśnia serca.

Dane przyjmowane przez moduł:

- sygnał ECG_BASELINE;
- wektor numerów próbek załamków R z modułu R PEAKS;
- wektor numerów próbek punktów charakterystycznych z modułu WAVES:
  - QRS_{onset}
  - QRS_{end}
  - T_{end}

Dane zwracane przez moduł:

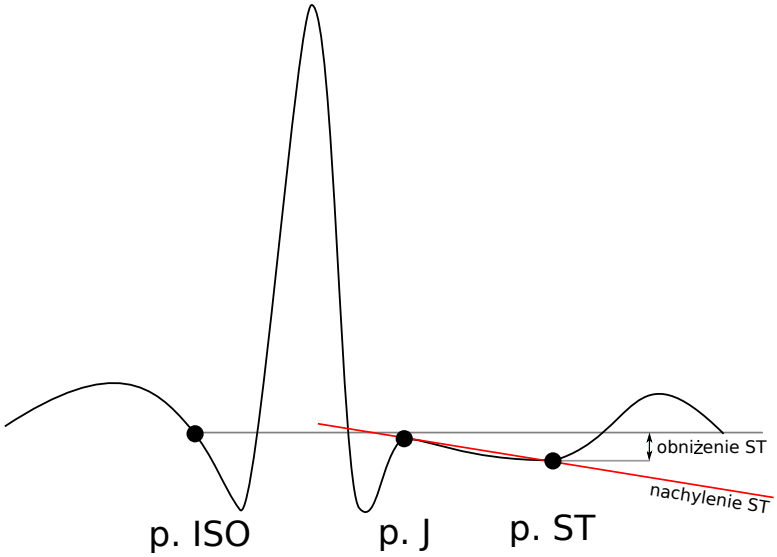
- wektor wykrytych odcinków ST zawierających informacje:
  - początek i koniec odcinka;
  - poziom izolinii;
  - pomiar poziomu względem izolinii;
  - pomiar nachylenia odcinka ST;
  - opis słowny interwału;
- wektor wykrytych epizodów wraz z parametrami:
  - numer próbki początku epizodu;
  - numer próbki końca epizodu.

3.8.2 Badania literaturowe

Analiza odcinka ST w sygnale EKG jest kluczowa we wczesnym wykrywaniu chorób wieńcowych, chorobach niedokrwiennych i miażdżycy. Badając uniesienie i nachylenie odcinka względem izolini można uzyskać informacje na temat zagrożenia wymienionymi chorobami i podjąć wczesną interwencję. Dość sporym mankamentem tych parametrów diagnostycznych jest ich charakter amplitudowy, gdyż wiadomo, że wielkość amplitudy zależy często od wielu pozakardiologicznych przyczyn, m.in. jakości kontaktu elektrod ze skórą.

Książka [1] prezentuje bardzo prosty algorytm diagnostyki odcinka ST. Algorytm składa się z trzech kroków: lokalizacji odcina, obliczenia uniesienia i nachylenia oraz wykrywanie epizodów.

Na rysunku 35 przedstawiono kluczowe punkty, które należy wyznaczyć przed wyznaczaniem parametrów diagnostycznych odcinka ST. Według wyżej wymienionego algorytmu, punktem referencyjnym w założeniu znajdującym się na izolinii jest początek zespołu QRS i koniec odcinka P. Punkt J, który



Rysunek 35: Kluczowe punkty analizy odcinka ST. ISO – punk referencyjny leżący na izolinii, J – początek odcinka, ST – koniec odcinka. Opracowanie własne.

jest początkiem badanego odcinka i znajduje się $45ms$ za szczytem załamka R. Punkt ST będący końcem badanego odcinka znajduje się $60ms$ później. Tych stałych wartości używa się w prostych algorytmach, dodatkowo można uwzględnić zmienność rytmu serca i uzależnić przesunięcie tych wartości od aktualnego odstępu RR. Uniesieniem lub obniżeniem odcinka jest różnica wartości miedzy punktem ISO, a punktem ST. Nachyleniem prostej przechodzącej przez punkty J i ST nazywamy nachyleniem odcinka. Prawidłowy odcinek ST powinien być poziomy i nie być uniesiony ani obniżony. Algorytm wykrywania epizodów polega na wykryciu takich okresów zapisu EKG w którym przez czas dłuższy niż 60 sekund występują ciągle nieprawidłowe odcinki ST.

W artykule [7] przedstawiono bardziej złożoną procedurę wyznaczania punktów charakterystycznych odcinka. W oparciu o znalezione wcześniej punkty ISO, J i punkt końca odcinka T (T_{end}) znajduje szczyt załamka T z wykorzystaniem dekompozycji falkowej. następnie wyznacza punkt ST jako najbardziej odległą wartość sygnału od prostej łączącej punkt J i punkt szczytowy załamka T.

Obniżenie załamka jest wyznaczane metodą J+X, który polega na wyznaczeniu, podobnie jak w pierwszym algorytmie, różnicy miedzy punktem J+X (punkt X milisekund za punktem J), a punktem ISO. X jest wyznaczany na podstawie aktualnego HRV. Ocena przesunięcia odcina ST względem izolinii odbywa się z użyciem progu K, dla bazy MIT-BIH użyta wartością może być 0.01. Odcinek uznany jest za podniesiony gdy przesunięcie jest większe od K, obniżony gdy jest mniejszy od -K i normalny, prawidłowy w pozostałych przypadkach.

Następnie w zalezności od oceny przesunięcia odcinka wyznaczany jest referencyjny punkt końcowy. Dla odcinka uniesionego jest to punkt szczytowy fali T, dla normalnego i obniżonego – punkt ST. Kolejno, na podstawie charakteru odcinka do J20 (punkt J przesunięty o 20ms) do końcowego punktu referencyjnego okreslane jest czy odcinek jest prosty czy zakrzywiony, a następnie określany jest charakter zakrzywienia (wklęsły/wypukły) lub charakter monotoniczny odcinka prostego.

Rodzaj zakrzywienia odcinka ST wyznaczany jest na podstawie stosunku liczby punktów badanego fragmentu sygnału które są ponad/poniżej liniowej interpolacji odcinka do liczby wszystkich punktów odcinka. Jeżeli ten stosunek dla punków ponad prostą jest większy od 70% to odcinek jest wypukły, jeżeli 70% punktów jest poniżej prostej, odcinek jest wklęsły.

Charakter monotoniczny prostego odcinka ST jest określany na podstawie nachylenia interpolacji liniowej badanego fragmentu sygnału.

Epizody ST są wyznaczane podobnie jak w pierwszym algorytmie.

Podczas implementacji wprowadzono kilka poprawek, z których należy wyszczególnić:

Zmiana sposobu wyznaczania szczytu fali T. Zamiast dekompozycji falkowej użyto pochodnej sygnału. Ostatni punkt przejścia pochodnej przez zero jest szczytem fali.

Porzucono branie pod uwagę HRV, przy wyznaczaniu kolejnych punktów.

3.8.3 Opis procedur i metod

Główna funkcjonalność modułu znajduje się w klasie `STAnalysis`. Klasa ta, według przyjętego schematu rozszerza abstrakcyjny moduł analizy odcinka ST. W tej klasie znajdują się drzewo klas prywatnych reprezentujących poszczególne algorytmy analizy. W szczególności dwie klasy `SimpleAnalizer` oraz `ComplexAnalizer`. Pierwsza klasa reprezentuje najprostszy algorytm zaprezentowany w [1, p. 155], druga zaś implementuje zmodyfikowany algorytm opisany w [7].

Lista i opis najważniejszych funkcji:

```
void STAnalysis::SimpleAnalizer::analyse(const int it, const ECGRs& rpeaks,
  const ECGWaves& waves, const ECGSignalChannel& signal,
  const  ECGInfo& info, ECGST& output);
```

Funkcja analizująca załamek ST w zespole QRS numer `it`. Wykorzystując punkty sygnału obliczone we wcześniejszych modułach (tj. QRS_{onset}, QRS_{end} oraz T_{end}) oblicza koniec odcinka ST i wylicza jego parametry:

- przesunięcie względem izolinii
- wartość nachylenia w stosunku do izolini
- długość odcinka ST

Używając wcześniej ustawionego parametru `simple_thresh`, odcinek ST określany jest jako „normalny", „uniesiony" lub „obniżony".

Parametry:

- `const int it` – numer badanego zespołu QRS, jako liczba porządkowa numerów próbek z tablicy załamków R.
- `const ECGR& rpeaks` – struktura zawierająca tablicę numerów próbek kolejnych załamków R
- `const ECGEaves& waves` – struktura zawierająca tablice przechowujące numery próbek punktów charakterystycznych kolejnych zespołów QRS: P_{onset}, QRS_{onset}, QRS_{end} oraz T_{end}.
- `const ECGSignalChanel& signal` – jeden kanał odfiltrowanego sygnału z usuniętym przesunięciem izolinii.
- `const ECGInfo& info` – struktura zawierająca informacje o badanym sygnale EKG, m.in. częstotliwość.
- `ECGST& output` – parametr wyjściowy, struktura zawierająca tablice wszystkich interwałów ST wraz z ich parametrami, oraz tablice zarejestrowanych epizodów ST wraz z ich parametrami.

```
void STAnalysis::ComplexAnalizer::analyse(const int it, const ECGRs& rpeaks,
  const ECGWaves& waves, const ECGSignalChannel& signal,
  const  ECGInfo& info, ECGST& output);
```

Funkcja analizująca załamek ST w zespole QRS numer `it`. Wykorzystując punkty sygnału obliczone we wcześniejszych modułach (tj. QRS_{onset}, QRS_{end} oraz T_{end}) oblicza koniec odcinka ST (wykorzystując bardziej zaawansowane algorytmy od poprzedniej funkcji), a następnie oblicza jego parametry:

- przesunięcie względem izolinii,
- wartość nachylenia w stosunku do izolinii,
- długość odcinka ST,
- klasyfikacja kształtu.

Używając wcześniej ustawionego parametru `complex_thresh`, odcinek ST określany jest jako „normalny", „uniesiony" lub „obniżony". Algorytm ocenia również to, czy załamek jest prosty czy zakrzywiony w oparciu o wcześniej ustawiony parametr `type_thresh`. Dla prostych odcinków określa, czy kierunek narasta, opada czy jest poziomy. Do określenia tej cechy wykorzystywany jest parametr `slope_thresh`. Dla zakrzywionych odcinków ST oceniana jest wypukłość krzywej. Szczegółowy opis działania znajduje się w poprzednim rozdziale.

Parametry:

- `const int it` – numer badanego zespołu QRS, jako liczba porządkowa numerów próbek z tablicy załamków R.
- `const ECGR& rpeaks` – struktura zawierająca tablicę numerów próbek kolejnych załamków R
- `const ECGEaves& waves` – struktura zawierające tablice przechowujące numery próbek punktów charakterystycznych kolejnych zespołów QRS: P_{onset}, QRS_{onset}, QRS_{end} oraz T_{end}.
- `const ECGSignalChanel& signal` – jeden kanał odfiltrowanego sygnału z usuniętym przesunięciem izolinii.
- `const ECGInfo& info` – struktura zawierająca informacje o badanym sygnale EKG, m.in. częstotliwość.
- `ECGST& output` – parametr wyjściowy, struktura zawierająca tablice wszystkich interwałów ST wraz z ich parametrami, oraz tablice zarejestrowanych epizodów ST wraz z ich parametrami.

Funkcje pomocnicze klasy `ComplexAnalizer`:

```
int STAnalysis::ComplexAnalizer::getTPeak(const OtherSignal& sig,
  int from, int to);
```

Funkcja wyszukująca położenia punktu T_{peak} na zadanym odcinku sygnału. Początkiem wyszukiwania szczytu fali T jest zwykle punkt 20ms za punktem QRS_{end}, a punktem końcowym jest koniec fali T. Funkcja wykorzystuje dyskretną pochodną sygnału.

Parametry:

- `const OtherSignal& sig` – cały, odfiltrowany sygnał EKG
- `int from` – numer próbki, w której należy zacząć poszukiwania
- `int to` – numer próbki, w której należy skończyć poszukiwania

```
std::pair<int, double> maxDistanceSample(const OtherSignal& signal,
  int from, int to);
```

Funkcja szukająca numeru próbki najbardziej oddalonego od liniowej interpolacji fragmentu sygnału miedzy dwoma punktami. Oprócz numeru próbki od początku sygnału, zwracana jest wartość największej różnicy miedzy punktem sygnału, a prostą interpolacji.

Parametry:

- `const OtherSignal& signal` – badany sygnał
- `int from` – numer próbki będącej początkiem interesującego fragmentu sygnału, równocześnie, pierwszy punkt interpolacji liniowej.
- `int to` – numer próbki będącej końcem interesującego fragmentu sygnału, równocześnie, drugi punkt interpolacji

```
std::pair<int, int> overBelowSamples(const OtherSignal& signal,
  int from, int to);
```

Funkcja obliczająca ilość próbek ponad i poniżej prostej interpolującej sygnał miedzy dwoma punktami. Funkcja wykorzystywana jest do określenia wypukłości odcinka ST.

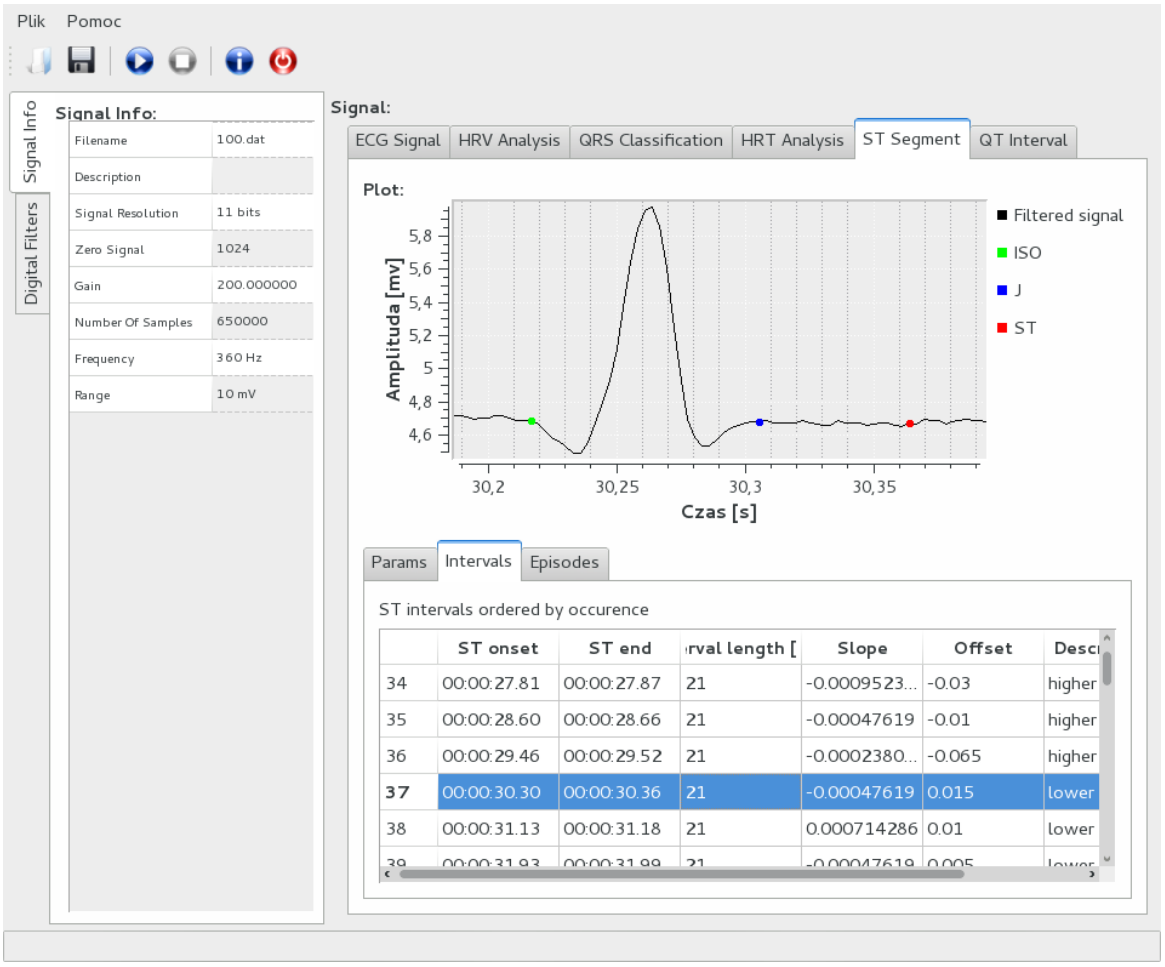
Parametry:

- `const OtherSignal& signal` – badany sygnał
- `int from` – numer próbki będącej początkiem interesującego fragmentu sygnału, równocześnie, pierwszy punkt interpolacji liniowej.
- `int to` – numer próbki będącej końcem interesującego fragmentu sygnału, równocześnie, drugi punkt interpolacji

3.8.4 Warunki testowania

W trakcie implementacji do testowania algorytmów używaliśmy surowego sygnału MIT BIH oraz wzorcowych załamków R z pliku z rozszerzeniem atr dostępnego razem z sygnałem. Testy polegały na wizualnej ocenie poprawności wykrytych punków przedstawionych na wykresie. Kolejnym krokiem było testowanie algorytmu na przefiltrowanym sygnale oraz załamkach R zwracanych przez moduł RPeaks, również na bazie oceny wizualnej wyznaczonych właściwości odcinka ST. Flaga `DEVELOPMENT` pozwala na przełączanie się między użyciem załamków wzorcowych (gdy jest ustawiona) i wykrywanych przez moduł RPeaks (gdy nie jest ustawiona).

3.8.5 Wyniki



Rysunek 36: Zakładka modułu w GUI przedstawiająca przykładowy zespół QRS z zaznaczonymi punktami charakterystycznymi analizy odcinka ST.

Resultatem projektu są dwa algorytmy wyznaczania i analizy parametrów odcinków ST. Prostrzy algorytm opisany w [1] został w całości zaimplementowany i przetestowany. Drugi, rozszerzony algorytm z [7] również został zrealizowany, jednak ze względu na złożoność matematyczną niezbędne okazało się uproszczenie jednego z kroków. Algorytm ten zakłada wykorzystanie 5-warstwowej dekompozycji falkowej splinów 4-stopnia do detekcji punktu T_{peak}, co okazało się być niewykonalne technicznie. Wykrycie tego punktu zostało zrealizowane przy użyciu filtru różnicowego wyszukującego maximum lokalne sygnału. Poprawność wyników z obu algorytmów w dużym stopniu zależy od jakości wyników działania modułów nadrzędnych, dostarczających przefiltrowany sygnał EKG oraz zespoły QRS. Na rysunku 36 przedstawiono przykładowy wynik modułu analizy ST.

3.9 T wave alt

3.9.1 Opis zadania

Temat Ocena ilościowa alternansu załamka T

Opis W ostatnich latach alternans załamka T stał się elementem szczególnych zainteresowań klinistów w środowisku kardiologów. Dowiedziono, że pojawienie się naprzemienności kształtu, amplitudy czy polarności załamka T jest zapowiedzią zaburzeń komorowych rytmu serca (arytmii komorowych). Wykazano również korelację amplitudy alternansu załamka T i spadku progu trzepotania przedsionków i komór. Celem projektu jest oznaczenie załamków T, w których wykryto alternans oraz ich ilościowa analiza.

Dane ciąg punktów charakterystycznych z modułu WAVES i sygnał

Szukane moduł programu oznaczający załamki T, w których wykryto alternans oraz dokonujący ich ilościowej analizy. Wszystkie dane należy odpowiednio nanieść na wykresy i przedstawić w postaci graficznej (parametry alternansu).

3.9.2 Badania literaturowe

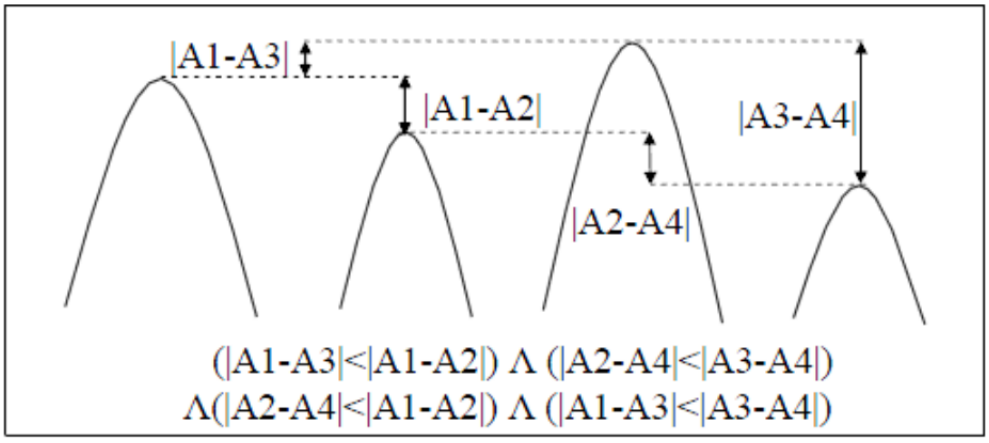
Alternans załamka T to naprzemienna zmiana wektora oraz amplitudy załamka T w sygnale EKG. Zmiany wielkości są mierzone w mikrowoltach, dlatego do analizy sygnału i detekcji TWA wykorzystywany jest sprzęt o bardzo dużej czułości.

Detekcja alternansu załamka T jest obecnie przedmiotem wielu badań oraz analiz prowadzonych przez ośrodki kardiologiczne, laboratoria w ośrodkach akademickich oraz centra badawcze na całym świecie. Nie istnieje obecnie jedna uniwersalna metoda dająca obiektywne wyniki i mogącą z 100% pewnością stwierdzić, że dany przebieg EKG zawiera alternans. Dodatkowo testy skuteczności różnych metod bardzo często dają sprzeczne ze sobą wyniki. W związku z faktem istnienia bardzo dużej różnorodności metod, narzędzi oraz algorytmów mających na celu wykrycie alternansu załamka T postanowiono przeprowadzić badania umożliwiające ocenę różnych metod.

Na podstawie przeanalizowanej literatury oraz algorytmów biorących udział w konkursie `Physionet Challenge 2008`, zostały wyodrębnione trzy główne metody, które zostały ostatecznie wzięte pod uwagę przy wyborze algorytmu detekcji oraz projektowaniu modułu:

- Moving four-beat window techniqure
- Fast Fourier Transform
- Spectral analysis PCA

Po przeprowadzeniu testów oraz analizie wniosków z pracy `T-Wave Alternans: A Comparison of Different Measurement Techniques` zdecydowano wybrać podejście `Moving four-beat window` jako najbardziej odpowiednie do stawianego przed autorami problemu i dającego względnie najlepsze wyniki. O złożoności problemu detekcji TWA świadczy między innymi fakt, że podczas przeprowadzania testów trzech wyżej wymienionych algorytmów jedynie 7 razy wszystkie metody były zgodne co do występowania w sygnale alternansu załamka T.



Rysunek 37: Algorytm Movign four-beat window techniqure

Algorytm `Moving four-beat window techniqure` analizuje jednocześnie 4 występujące obok siebie załamki T (okno o rozmiarze 4). W przypadku, gdy zajdą w takim oknie następujące zależności:

- A2-A4 <A1-A2
- A2-A4 <A3-A4
- A2-A4 <A1-A2
- A1-A3 <A3-A4

dane okno brane jest pod uwagę jako posiadające alternans. Okno jest przesuwane, aż do napotkania końca sygnału. W przypadku gdy 5% wszystkich okien spełnia wyżej wymieniony warunek, dany sygnał kwalifikowany jest jako zawierający alternans załamka T.

3.9.3 Opis procedur i metod

Pełna funkcjonalność modułu została za implementowana w klasie `TWaveAltDetector`. Klasa ta dziedziczy po abstrakcyjnej klasie `TWaveAltModule`. Moduł uruchamiany jest za pomocą funkcji:

`TWaveAltDetector::runModule(const ECGWaves, const ECGSignalChannel, ECGInfo, ECGTWave)`
Argumenty funkcji:

- `ECGWaves ecgWaves` – instancja klasy przechowującej wyniki detekcji punktów charakterystycznych sygnału EKG takich jak QRS-onset, QRS-end, T-end, P-onset, P-end;
- `ECGSignalChannel ecgSignal` – instancja klasy zawierajacej badany sygnał EKG;
- `ECGInfo ecgInfo` – informacje na temat wczytanego sygnału;
- `ECGTWave ecgTWave` – iinstancja klasy przechowującej wyniki detekcji oraz wektor numerów próbek sygnału zawierających alternans załamka T;

Używane funkcje:

- `detectTWaveAlt()` –główna funkcja klasy odpowiedzialna za całą logikę modułu;

Używane zmienne:

- `filteredSignal` – badany sygnał

- `wavesDatal` – wektor numerów próbek zawierających QRS-onset, QRS-end, T-end, P-onset, P-end
- `tWaveAltData` – wektor numerów próbek zawierających alternans

Detekcja oraz obliczenie parametrów oceny ilościowej alternansu wykonywane są w metodzie `detectTWaveAlt()`. Cały algorytm przetwarzania sygnału, detekcji alternansu (zgodnie z algorytmem `Moving four-beat window techniqure`) oraz obliczenia ilościowe związane z analizowanym sygnałem zostały zawarte właśnie w tej funkcji. Funkcja nie przyjmuje żadnych parametrów wejściowych. Funkcja zwraca wartość logiczną 1 w przypadku wykrycia w sygnale alternansu oraz wartość 0 w przypadku braku alternansu w sygnale. Funkcja oblicza następujące parametry ilościowe alternansu:

- `ilość okien w sygnale zawierających alternans`
- `procentowy udział wszystkich okien o rozmiarze 4 zawierających alternans w sygnale`

`bool TWaveAltDetector::detectTWaveAlt()` Argumenty funkcji:

- `brak`

Używane funkcje:

- `gsl_vector_int_get()` –funkcja GSL pobierająca całkowitą wartość danego elementu z wektora
- `gsl_vector_int_alloc()` –funkcja GSL tworząca wskaźnik do nowo utworzonego obiektu
- `gsl_vector_int_set()` –funkcja GSL zapisująca całkowitą wartość danego elementu wektora
- `GetT_end()` –funkcja zwracająca wektor numerów próbek zawierających załamek T
- `setTWaveAlt()` –funkcja ustawiająca wektor numerów próbek zawierających alternans załamka T

Używane zmienne:

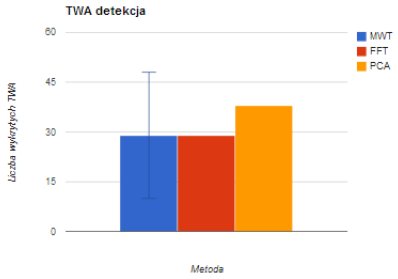
- `filteredSignal` – badany sygnał
- `wavesDatal` – wektor numerów próbek zawierających QRS-onset, QRS-end, T-end, P-onset, P-end
- `tWaveAltData` – wektor numerów próbek zawierających alternans

3.9.4 Warunki testowania

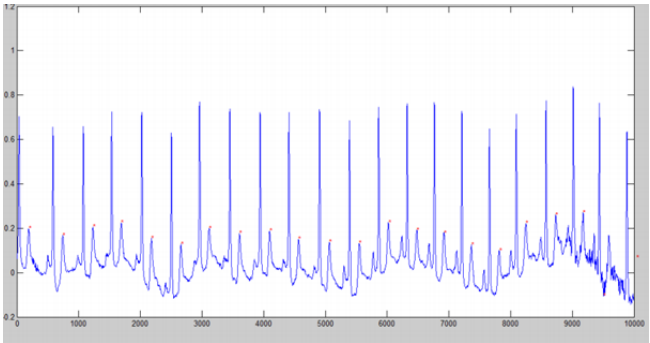
Wszystkie algorytmy brane pod uwagę przy wyborze docelowego rozwiązania zostały wstępnie zaimplementowane w środowisku Matlab. Po wybraniu odpowiedniego i najbardziej skutecznego rozwiązania (`Moving four-beat window techniqure`) moduł `TWaveAlt` wraz z detekcją załamków T został zaimplementowany w Matlab, a następnie w C++.

Zostały przeprowadzone następujące testy:

- Testy różnych algorytmów detekcji załamka T: Zostały opracowane prototypy różnych alogtytmów detekcji alternansu w Matlab.
- Testy poprawności działania wybranego algorytmu: Porównanie działania modułu zimplementowanego w Matlab z algorytmem w Visual Studio.
- Testy skuteczności wykrywania alternansu T: Zostały przetestowanie sygnały zawierające alternans T, które przedmiotem badań podczas konkursu "Physionet Challenge 2008". Uzyskane wyniki zostały porównane z oficjalnymi rezultatami prac konkursowych.



Rysunek 38: Test 1 – TWA algorytmy



Rysunek 39: Test 2 – TWA skuteczność

3.9.5 Wyniki

Przeprowadzone rysunki przedstawiają testy różnych algorytmów detekcji załamka T.

3.10 HRT

3.10.1 Opis zadania

Turbulencja rytmu serca jest nową, jeszcze nie do końca poznaną techniką służącą do oceny odpowiedzi węzła zatokowego na przedwczesny skurcz komorowy. Podejście okazało się trafnym narzędziem predykcji nagłych zgonów pacjenta. Celem projektu była implementacja modułu programu wyznaczającego podstawowe parametry turbulencji rytmu serca (m. in. początek turbulencji, nachylenie turbulencji, itp. . .).

Autorzy: Łukasz Kutrzuba, Mateusz Krasucki

3.10.2 Badania literaturowe

Przedwczesne pobudzenie komorowe
Przedwczesny skurcz komorowy (ang. Premature Ventricular Contraction) powstaje na skutek przedwczesnego rozprzestrzenienia się impulsu w komorach powodującego skurcz mięśnia komór (impuls w normalnym przypadku powinien pochodzić z węzła zatokowo-przedsionkowego). Jest to jedna z najczęstszych arytmii. **Turbulencja rytmu serca**
Turbulencja rytmu serca (ang. Heart Rate Turbulence) jest to reakcja fizjologiczna będąca powrotem do stanu równowagi po przedwczesnym skurczu komorowym. Analiza sygnałów HRT pozwala na ocenienie jak szybko i jak energicznie serce reaguje na przedwczesny skurcz komorowy. Duże znaczenie

ma fakt, że występowanie tego ostatniego zjawiska fizjologicznego jest naturalne dla ogółu osób dorosłych, co pozwala na wykorzystanie oceny HRT dla diagnostyki stanu zdrowia. Analiza HRT jest zwykle wykonywana na długich, 24 godzinnych zapisach EKG.

W zapisie reakcji organizmu na PVC można wskazać charakterystyczne etapy. Po przedwczesnym skurczu komorowym następuje przerwa wyrównawcza, następnie faza akceleracji, deceleracja i ostatecznie powrót do normalnego rytmu. Nie zdołano dotychczas określić przyczyn zjawiska turbulencji rytmu serca, istnieją przypuszczenia, że powoduje je odruch z baroreceptorów.

Wystąpienie turbulencji rytmu serca świadczy o prawidłowej reakcji układu przywspółczulnego i zdolności serca do reagowania na przedwczesne pobudzenie – z tego powodu wystąpienie skróconych interwałów po VPC i ich stopniowe wydłużanie jest uznawane za prawidłową reakcję. Z kolei pojawienie się interwałów dłuższych niż przed przedwczesnym pobudzeniem jest negatywnym prognostykiem, świadczy o braku reakcji organów kontrolujących pracę serca.

Rysunek 40: Wykres przedstawiający wykresy HRT dla zdrowego i chorego człowieka. *Źródło: http://www.h-r-t.org, Working Group of Biological Signal Analyses, Technische Universität München Źródło: Axel Bauer, Marek Malik, Georg Schmidt: „Heart Rate Turbulence: Standards of Measurement, Physiological Interpretation, and Clinical Use", Journal of the American College of Cardiology 2008, tom 52, nr 17*

Znaczenie diagnostyczne turbulencji rytmu serca

Analiza turbulencji rytmu serca jest jedną z najważniejszych metod pozwalającą na ocenę ryzyka zgonu pacjentów po przebytym zawale, obok frakcji wyrzutowej lewej komory, ilości przedwczesnych skurczów komorowych oraz analizy rytmu serca. Dowiedziono wysokiej wartości predykcyjnej tej metody badania. Przy analizie wyników pacjentów zostaje dokonane przyporządkowanie do jednej trzech grup – oba parametry poprawne, jeden parametr negatywny, oba parametry negatywne. Na przykład w przypadku obu parametrów negatywnych ryzyko zgonu w ciągu roku od przebytego zawału wynosi powyżej 32 procent.

Parametry turbulencji rytmu serca

W przypadku turbulencji rytmu serca wykorzystuje się zwykle dwa parametry – początek turbulencji (turbulence onset) oraz nachylenie turbulencji (turbulence slope). Turbulence Onset (TO) określa procentową różnicę między częstością serca przedwczesnym skurczu komorowym i częstością serca przed tym skurczem. Obliczanie TO opiera się na różnicy pomiędzy sumą długości dwóch interwałów RR występujących bezpośrednio po pauzie wyrównawczej a sumą dwóch interwałów występujących przed pauzą wyrównawczą. Wartości dodatnie TO oznaczają zwolnienie, natomiast ujemne przyspieszenie rytmu zatokowego. Pozytywnym prognostykiem są wartości ujemne.

Rysunek 41: Przykładowy schemat ilustrujący wyznaczanie parametru Turbulence Onset. *Źródło: http://www.h-r-t.org, Working Group of Biological Signal Analyses, Technische Universität München*

Turbulence Slope (TS) odpowiada najbardziej stromej linii regresji przeprowadzona przez każde 5 kolejnych odstępów prawidłowego rytmu w tachogramie (spośród 20 kolejnych interwałów występujących zaraz po przerwie wyrónawczej). Obliczenia Turbulence Slope opierają się na tachogramie uśrednionym z wszystkich fragmentów rytmu zatokowego po skurczu komorowym. Warości TS wyrażone w ms/odstęp RR. Prawidłowa wartość parametru to 2.5ms/RR.

Rysunek 42: Przykładowy schemat ilustrujący wyznaczanie parametru Turbulence Slope. *Źródło: http://www.h-r-t.org, Working Group of Biological Signal Analyses, Technische Universität München*

3.10.3 Koncepcja rozwiązania

Detekcja PVC

Przy wyliczeniu parametrów TO i TS brane są pod uwagę niektóre z fragmentów zapisu interwałów RR – zwykle jest to fragment zapisu liczący 27 kolejnych interwałów (5 interwałów przed VPC, interwał z VPC, pauza kompensacyjna, około 20 interwałów po pauzie). Dodatkowo, wybierane fragmenty zapisu muszą charakteryzować się się brakiem występowania nierównego rytmu serca przed i po PVC oraz brakiem zbyt krótkich oraz zbyt długich interwałów RR.

W celu selekcji odpowiednich odcinków zastosowano następujący algorytm:

- Pobranie sygnału z długościami poszczególnych interwałów
- Każdy kolejny interwał RR jest traktowany jako potencjalny interwał VPC (z kolei 5 interwałów poprzedzających oraz 20 interwałów występujących po tym interwale będzie wykorzystanych do oceny sygnału jako poprawny VPC). W przypadku, jeśli fragment sygnału zostanie uznany jako poprawny interwał VPC, będzie uwzględniony w obliczaniu TO i TS. Warunki pozwalające na uznanie fragmentu jako poprawny VPC określono poniżej.
- Dla potencjalnego interwału VPC zostaje wykonane obliczenie długości przedziału referencyjnego (średnia z 5 interwałów poprzedzających VPC).
- Potencjalny RR z VPC musi być krótszy o przynajmniej 20 procent względem przedziału referencyjnego
- Interwał występujący po potencjalnym interwale z VPC musi spełniać warunki pauzy kompensacyjnej (jego długość musi być większa o przynajmniej 10 procent od przedziału referencyjnego).
- Pośród 5 interwałów przed VPC oraz 20 interwałów po pauzie kompensacyjnej nie mogą znajdować się interwały o długościach mniejszych od 300 ms oraz większych od 2000 ms.
- W ciągu 5 interwałów przed VPC oraz 20 interwałów po pauzie kompensacyjnej musi występować równy rytm – sąsiadujące interwały muszą różnić się o mniej niż 200 ms oraz ich różnica względem przedziału referencyjnego musi być mniejsza od 20 procent.
- Jeśli interwał spełnia założone warunki, zostanie wykorzystany do obliczania parametrów TS oraz TO.
- Na koniec sprawdzana jest liczba wykrytych PVC – jeśli wykryto mniej niż 5, sygnał nie może być efektywnie wykorzystany, ponieważ nie zawiera odpowiedniej ilości danych diagnostycznych.

Obliczanie TO

Dla poprawnie zidentyfikowanych VPC wyznaczany jest początek turbulencji. Wykorzystywany jest

Rysunek 43: Schemat prezentujący umiejscowienie interwałów wykorzystywanych przy wyliczeniu parametru TS. *Źródło: http://www.h-r-t.org, Working Group of Biological Signal Analyses, Technische Universität München*

przy tym wzór: RR-2 i RR-1 to długości interwałów występujących bezpośrednio przed VPC, a RR1

Rysunek 44: Wzór pozwalający na wyliczenie TO.

i RR2 to długości interwałów występujących bezpośrednio po pauzie kompensacyjnej. Wstępnie TO określa się dla każdego pojedynczego VPC a następnie uśrednia się wszystkie wartości TO z analizowanego zapisu EKG. **Obliczanie TS**

Algorytm obliczania turbulence slope:

- Na podstawie wyznaczonych odcinków zawierających poprawne VPC wyznaczany jest uśredniony tachogram.

- Poddanych analizie zostaje 20 interwałów uśrednionego tachogramu występujących po pauzie kompensacyjnej.
- Spośród pierwszych dwudziestu interwałów po pauzie kompensacyjnej dla każdych kolejnych 5 interwałów wyznaczana jest prosa regresji (brane są kolejno interwały 1-5, 2-6, 3-7 itd).
- Spośród wyznaczonych prostych regresji zostaje wybrana ta ze współczynnikiem regresji o największej dodatniej wartości.

Prostą regresji obliczamy wykorzystując współrzędne x, y pięciu kolejnych punktów należących do uśrednionego tachogramu. Przy wykorzystaniu współrzędnych punktów tworzone są macierze A i Y.

$$A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ x_4 & 1 \\ x_5 & 1 \end{bmatrix} \tag{37}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} \tag{38}$$

Następnie, korzystając z równania $(A'A)C = (A'Y)$ wyliczana jest macierz C. Pierwszy element wynikowej macierzy C to współczynnik kierunkowy prostej **a** (dla równania $y = ax + b$).

3.10.4 Opis wykorzystanych w programie klas i funkcji

Poniżej opisano najważniejsze klasy, funkcje i zmienne wykorzystane w module:

Dołączona do projektu klasa **Matrix** jest klasą wykorzystywaną do wykonywania obliczeń. Dzięki niej można definiować obiekty klasy Matrix reprezentujące dwuwymiarowe macierze oraz wykonywać na nich najważniejsze obliczenia. Klasa Matrix została wykorzystana przy obliczaniu Turbulence Slope.

Klasa **HRTAnalyzer** jest główną klasą modułu, dziedziczy po klasie HRTModule. Metody klasy HRTAnalyzer pozwalają na wyliczenie parametrów HRT.

Klasa **ECGHRT** jest klasą wykorzystywaną do przechowywania danych uzyskanych podczas pracy – przechowuje najważniejsze obiekty będące wynikiem pracy HRTAnalyzera – parametry Turbulence Slope i Turbulence Onset, liczbę wykrytych VPC, oraz inne parametry wykorzystywane do rysowania.

Najważniejsze zmienne klasy ECGHRT:

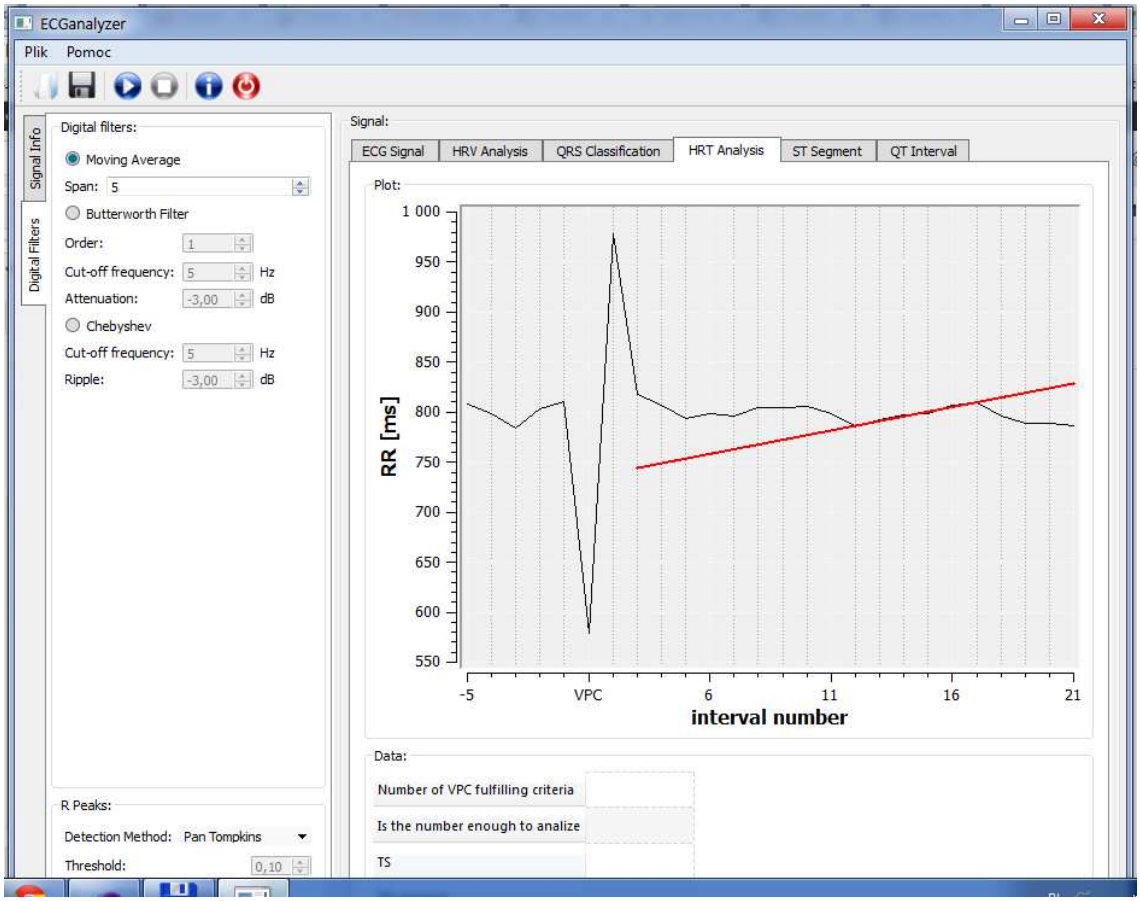
- Dane wykorzystywane przy rysowaniu wykresu: **QVector¡QPointF¿ rr** wektor reprezentujący uśredniony tachogram, **QLine ts** – linia reprezentująca Turbulence Slope.
- Wyświetlane wartości:**int vpcCounter** – ilość znalezionych VPC, **double TO** – średnia wartość Turbulence Onset,**double TS** – średnia wartość Turbulence Slope.

Funkcje i zmienne klasy HRTAnalyzer:

- **double* RRs** – zmienna wykorzystywana do tymczasowego przechowywania sygnału.
- **void calculateHrtParams(double *signal, int size, ECGHRT &)** – funkcja, która zleca obliczenie parametrów HRT dla sygnału wejściowego i zapisuje je w wynikowym obiekcie. Parametry: signal to wskaźnik do tablicy interwałów RR, size to rozmiar tablicy, trzeci argument to referencja do obiektu przechowującego wyniki.
- **vector¡int¿ findVpcOnsets(double *signal, int size)** – funkcja wyszukująca VPC w sygnale, zwracająca wektor z numerami próbek uznanymi za poprawne VPC, które mogą być wykorzystane przy działaniu programu. Parametry: signal to wskaźnik do tablicy interwałów RR, size to rozmiar tablicy.

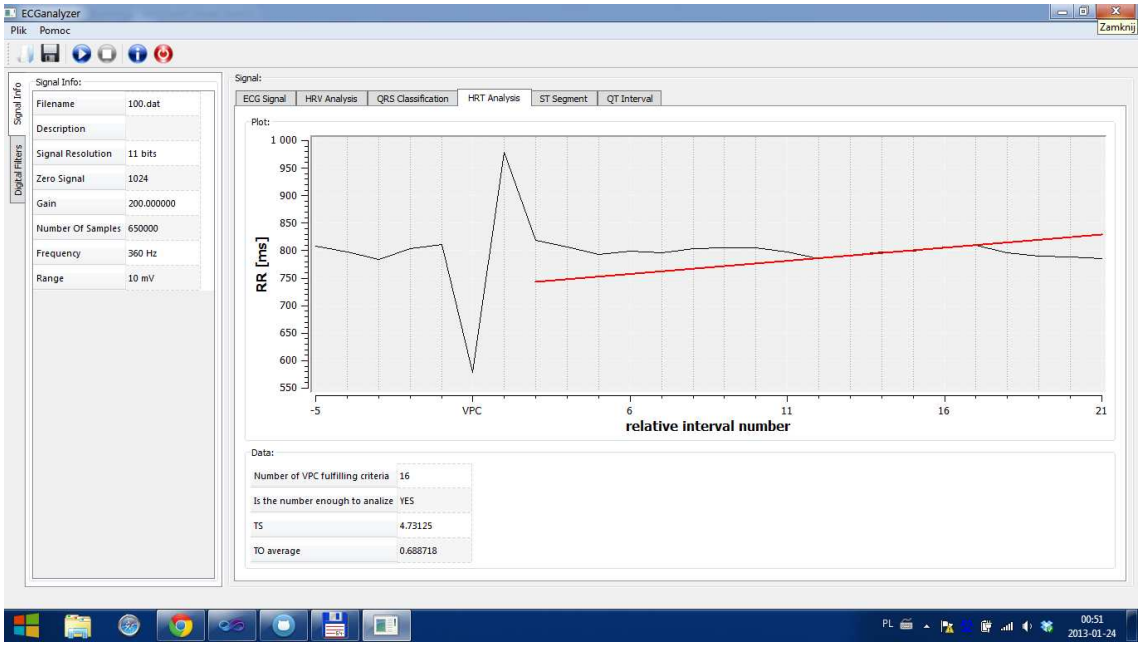
- **double* calculateAvgTach(double *signal, vector¡int¿ vpc_list)** – funkcja obliczająca uśredniony tachogram. Parametry: signal to wskaźnik do tablicy interwałów RR, vpc_list to wektor indeksów oznaczajacych wystąpienie VPC.
- **double calculateTO(double * signal, int size, vector¡int¿ vpc_list)** – funkcja obliczająca parametr TO, Parametry: signal to wskaźnik do tablicy interwałów RR, size to rozmiar tablicy, vpc_list to wektor indeksów oznaczających wystąpienie VPC.
- **void calculateTS(double * signal, int size, vector¡int¿ vpc_list, double* avgTach, double to, ECGHRT&)** – funkcja obliczająca parametr TS. Parametry: signal to wskaźnik do tablicy interwałów RR, size to rozmiar tablicy, vpc_list to wektor indeksów oznaczających wystąpienie VPC.

3.10.5 Testy

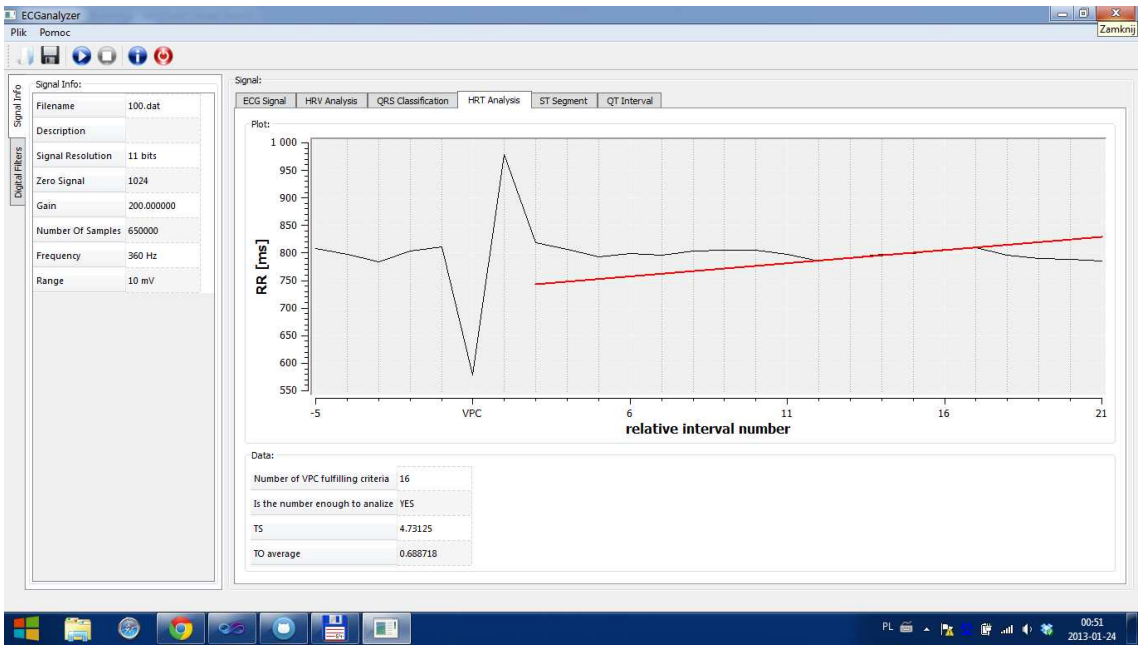


Rysunek 45: Zrzut ekranu prezentujący ostateczny wygląd modułu

Dane do pierwszego etapu testów zostały uzyskane z bazy European ST-T Database (link: `http://www.physionet.org/`). Początkowo testy wykonywano za pomocą prototypu aplikacji w Matlabie – testy działania aplikacji zakończyły się sukcesem (na wykresie wyświetlały się wartości TO i TS, wartości parametrów były zgodne z oczekiwaniami opartymi o analizę wyglądu wykresów). Dla sygnału e0113 otrzymano wartości: turbulence slope -1.962, turbulence onset 6.03. Dla sygnału e0123 otrzymano wartości: turbulence slope -0.957, turbulence onset 3.82. Dla sygnału e0104 otrzymano wartości: turbulence slope -2.891, turbulence onset 8.272. Niestety, w związku z faktem, że niektóre moduły, od których moduł HRT był zależny nie funkcjonowały poprawnie na 13 godzin przed terminem oddania, poniechano dalszych bezowocnych prób dokonania bardziej miarodajnych rodzajów testów na aplikacji. Nie mniej jednak, wcześniej dokonano podstawowej analizy działania modułu. Testy polegały na ocenieniu poprawności rozwiązania w oparciu o analizę wyglądu wykresów. Dzięki wykresowi możliwa była wzrokowa ocena poprawności otrzymywanych wyników (np. sprawdzenie, czy prosta regresji



Rysunek 46: Zrzut ekranu prezentujący ostateczny wygląd modułu



Rysunek 47: Zrzut ekranu prezentujący ostateczny wygląd modułu

pokrywa się z najbardziej stromym zboczem na wykresie, ocena wartości TO w odniesieniu do różnic długości). Testy nie wykazały uchybień.

Literatura

[1] Piotr Augustyniak. *Przetwarzanie sygnałów elektrodiagnostycznych.* Uczelniane wydawnictwa naukowo-dydaktyczne AGH, 2001.

[2] T. Krauze, P. Guzik, and H. Wysocki. Zmiennosc rytmu serca: aspekty techniczne. *Nowiny Lekarskie*, 9(ISSN 0860-7397):973 – 984, 2001.

[3] Seema M. Kaur, B. Singh. Comparisons of different approaches for removal of baseline wander from ecg signal. *Proceedings of the ICWET '11 International Conference & Workshop on Emerging Trends in Technology*, (ISBN 978-1-4503-0449-8):1290–1294, 2011.

[4] Vineet Kumar Mukamia. Baseline wander estimation for ecg characterization. Master's thesis, Electrical and Instrumentation Engineering Department, Thapar University, 2010.

[5] Task Force of The European Society of Cardiology, The North American Society of Pacing, and Electrophysiology (Membership of the Task Force listed in the Appendix). Heart rate variability standards of measurement, physiological interpretation, and clinical use. *European Heart Journal*, 17:354–381, 1996.

[6] M. D. Uplane S. K. Jagtap. A real time approach: Ecg noise reduction in chebyshev type ii digital filter. *International Journal of Computer Applications*, 49:52–53, 2012.

[7] Zhao Shen, Chao Hu, and Jingsheng Liao. An algorithm of st segment classification and detection. *Proceedings of the 2010 IEEE International Conference on Automation and Logistics*, 2010.