



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 PEKALA
Krzysztof PIEKUTOWSKI
Grzegorz PIETRZYK
Łukasz PODOLSKI
Mikołaj RZEPKA
Agata SITNIK
Leszek SOSNOWSKI
Aleksander STELIGA
Mateusz ŚLAŻYŃSKI
Łukasz ZIEŃKOWSKI

Opiekun:

mgr inż. Tomasz PIĘCIAK

24 stycznia 2013

Spis treści

1	Specyfikacja zadania	3
2	Specyfikacja techniczna rozwiązania	4
2.1	Wykorzystane narzędzia	4
2.2	Projekt systemu	4
2.3	Główne elementy kodu systemu	5
3	Opisy modułów	6
3.1	Usuwanie linii bazowej	6
3.1.1	Opis zadania	6
3.1.2	Badania literaturowe	9
3.1.3	Opis procedur i metod	9
3.1.4	Warunki testowania	9
3.1.5	Wyniki	9
3.2	Wykrywanie załamków R	9
3.2.1	Opis zadania	9
3.2.2	Badania literaturowe	9
3.2.3	Opis procedur i metod	14
3.2.4	Warunki testowania	17
3.2.5	Wyniki	18
3.3	Waves	18
3.3.1	Opis zadania	19
3.3.2	Badania literaturowe	19
3.3.3	Opis procedur i metod	20
3.3.4	Warunki testowania	21
3.3.5	Wyniki	22
3.3.6	Opis zadania	23
3.3.7	Badania literaturowe	23
3.3.8	Opis procedur i metod	24
3.3.9	Warunki testowania	25
3.3.10	Wyniki	25
3.4	HRV2	25
3.4.1	Opis zadania	25
3.4.2	Badania literaturowe	25
3.4.3	Opis procedur i metod	27
3.4.4	Warunki Testowania	30
3.4.5	Wyniki	30
3.5	HRV DFA	32
3.6	Klasyfikacja zespołów QRS	32
3.6.1	Opis zadania	32
3.6.2	Badania literaturowe	32
3.6.3	Opis procedur i metod	32
3.6.4	Warunki testowania	32
3.6.5	Wyniki	32
3.7	ST interval	32
3.7.1	Opis zadania	32
3.7.2	Badania literaturowe	33
3.7.3	Opis procedur i metod	34
3.7.4	Warunki testowania	37
3.7.5	Wyniki	37
3.8	T wave alt	37
3.9	HRT	37

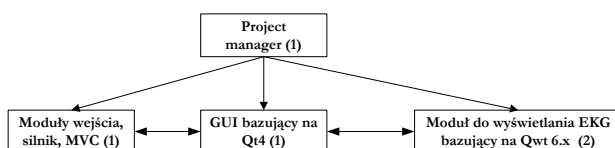
3.9.1	Opis zadania	37
3.9.2	Badania literaturowe	37
3.9.3	Koncepcja rozwiązania	39
3.9.4	Opis wykorzystanych w programie klas i funkcji	40
3.9.5	Testy	41

1 Specyfikacja zadania

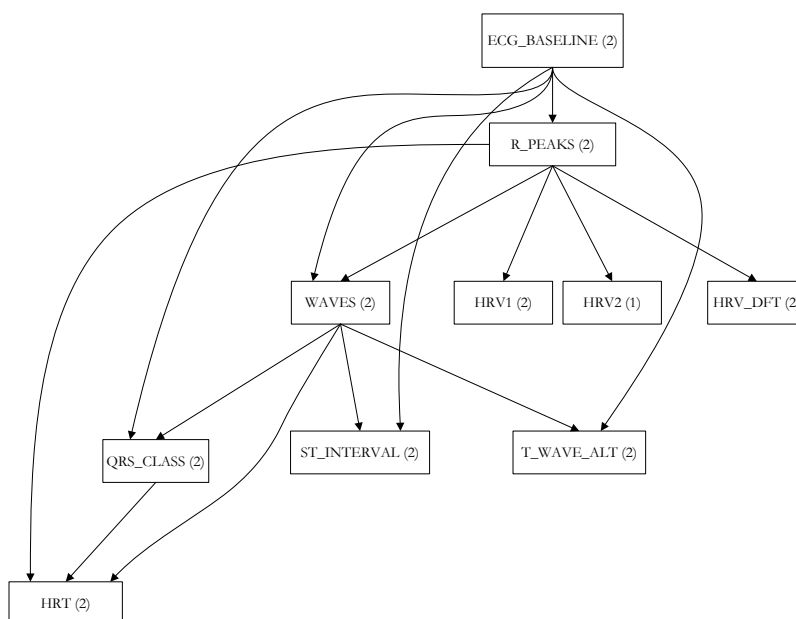
Celem projektu jest stworzenie zintegrowanego systemu pozwalającego na przeglądanie i automatyczną analizę sygnału EKG. Sygnał dostarczany jest w formie cyfrowej w standardzie wykorzystywanym w MIT-BIH Arrhythmia Database. Różne etapy przetwarzania, takie jak usuwanie linii bazowej, detekcja załamków 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.

Moduły zarządzania oprogramowaniem:



Moduły obliczeniowe:



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.

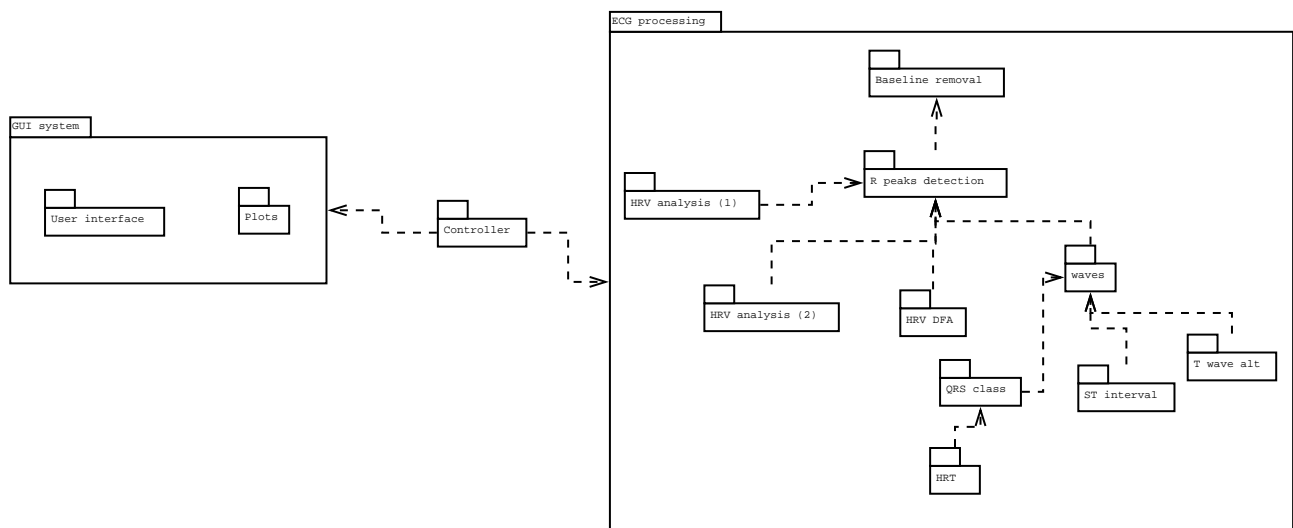
Samo przetwarzanie realizowane jest wielofazowo – moduły przeliczane są sekwencyjnie na osobnym wątku niż wątek zdarzeń GUI, co pozwala na zatrzymanie zbyt długo trwającej operacji. Zaimplementowano także mechanizm buforowania wyników, dzięki czemu przy zmianie parametrów pewnego modułu nie ma konieczności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

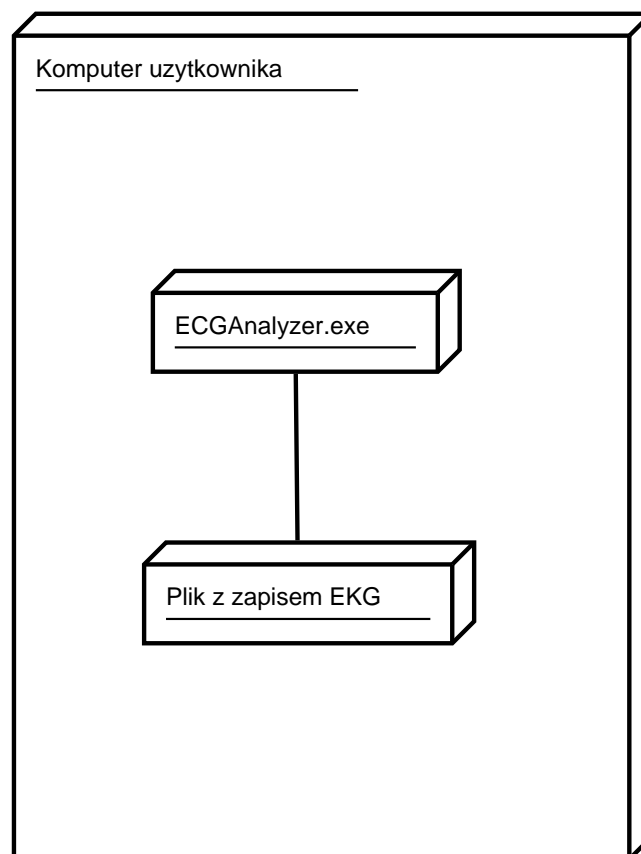
Diagram sekwencji przedstawiający sposób uruchamiania modułów jest przedstawiony na rys. 4. Każdy moduł otrzymuje najpierw zestaw parametrów (o ile posiada jakieś parametry), następnie zaś jest uruchamiany funkcją runModule. Kolejne moduły uruchamiane są sekwencyjnie według identycznego schematu. Wskutek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

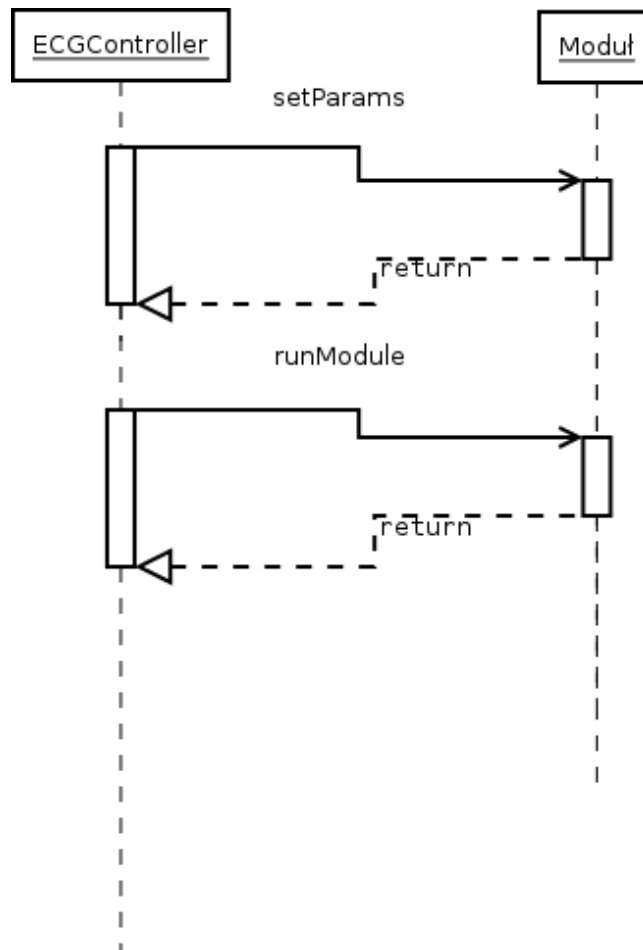


Rysunek 3: Diagram wdrożenia.

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 sprytnie 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

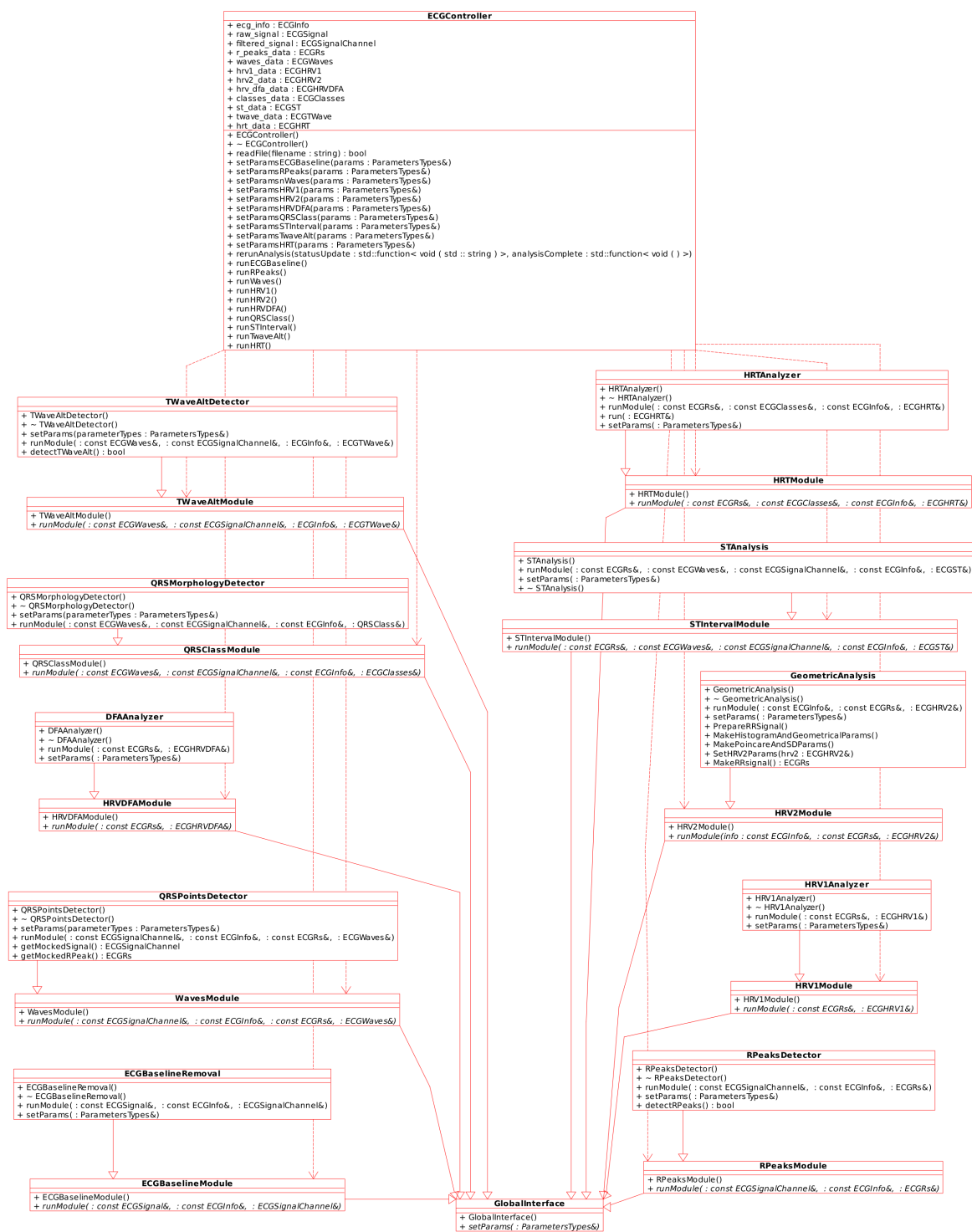
3.1 Usuwanie linii bazowej

Autorzy: Weronika Łabaj i Piotr Matuszkiewicz.

3.1.1 Opis zadania

Temat Metody filtracji i detekcji izolinii w sygnale EKG

Opis Występujące zakłócenia sieciowe i mięśniowe, jak również falowanie linii izoelektrycznej w sygnale EKG niejednokrotnie uniemożliwiają właściwą i poprawną analizę sygnału. Celem projektu

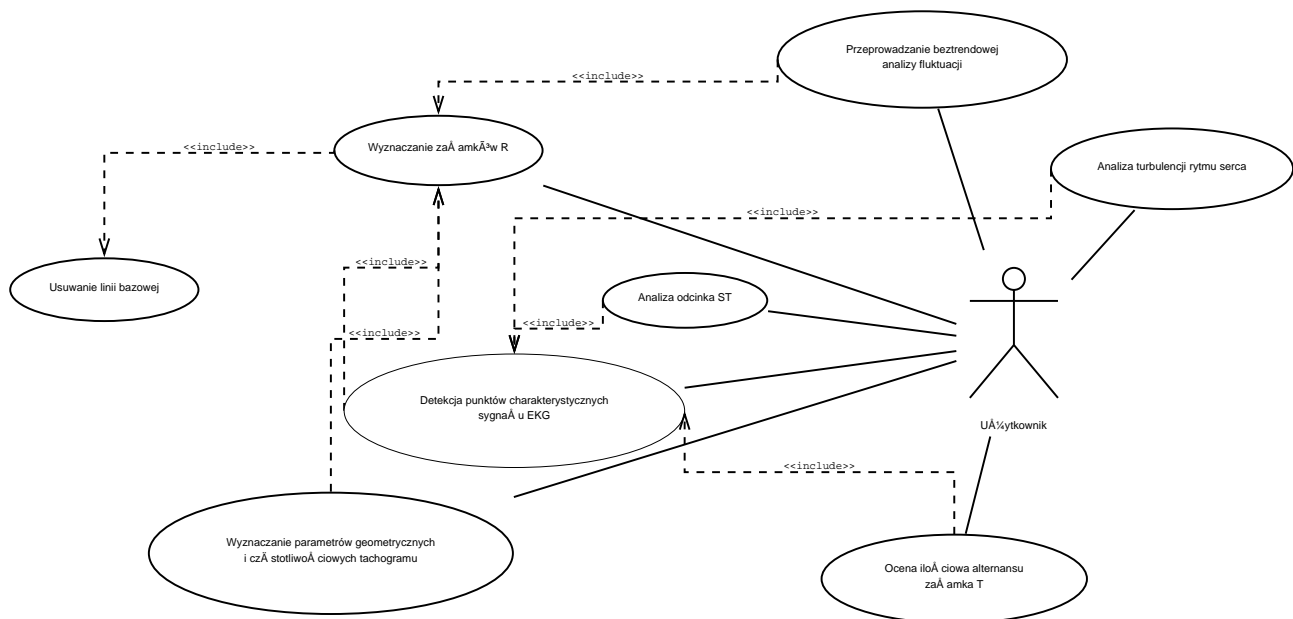


Rysunek 5: Diagram klas kontrolera i modułów.

jest opracowanie i implementacja metod związanych filtracją i detekcją linii izoelektrycznej w sygnale EKG. W szczególności należy rozważyć:

- filtr Butterwortha,
- średnią kroczącą,
- metody nieadaptacyjne: filtr Savitzky-Golay'a
- metody adaptacyjne np. filtr Wienera, LMS

Dane ciąg próbek sygnału EKG z bazy Physionet.org



Rysunek 8: Diagram klas.

3.1.2 Badania literaturowe

3.1.3 Opis procedur i metod

3.1.4 Warunki testowania

3.1.5 Wyniki

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 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 (1). Obliczanie średniej sygnału

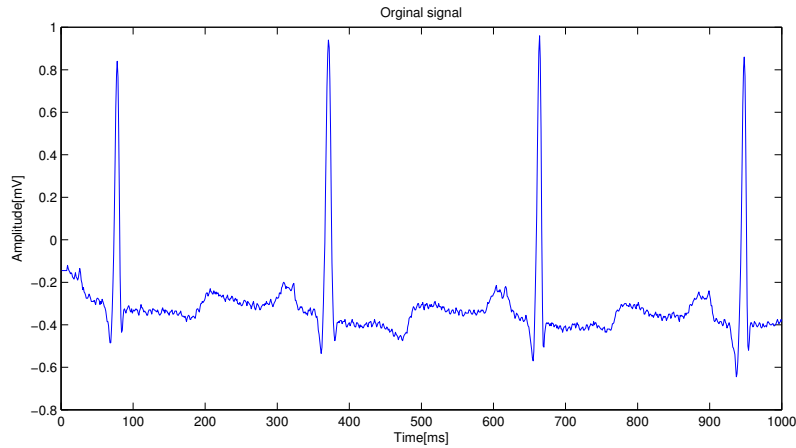
i odejmowanie jej od każdej próbki sygnału jest przybliżoną metodą usuwania składowej stałej.

$$y(n) = x(n) - \frac{x(1) + x(2) + \dots + x(n)}{n} \quad (1)$$

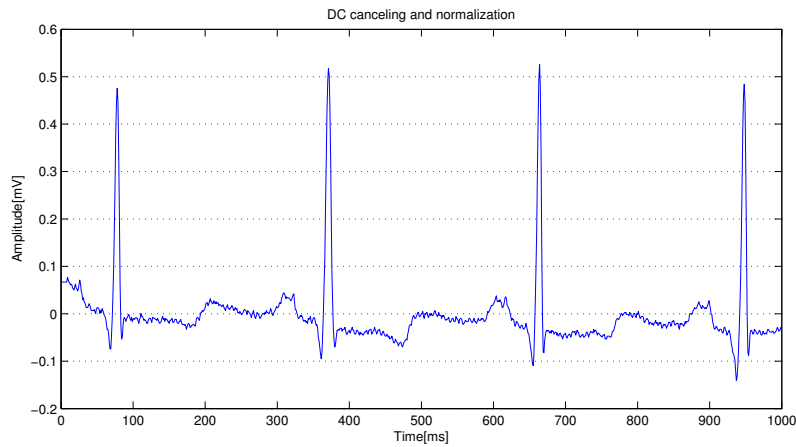
Aby ułatwić wszystkie obliczenia dokonywane na wybranym sygnale można znormalizować sygnał tak by wartości amplitudy były w zakresie od -1 do 1 . Normalizację sygnału można uzyskać stosując wzór (2).

$$y(n) = \frac{x(n)}{\max(|x(n)|)} \quad (2)$$

Rezultaty operacji usuwania składowej DC oraz normalizacji sygnału możemy zobaczyć na rysunku (10).



Rysunek 9: Sygnał EKG oryginalny(100_ MLII.dat)

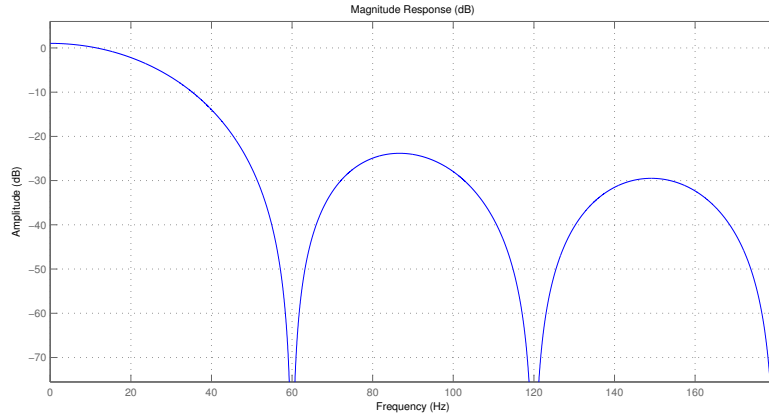


Rysunek 10: Sygnał EKG znormalizowany bez składowej stałej(100_ MLII.dat)

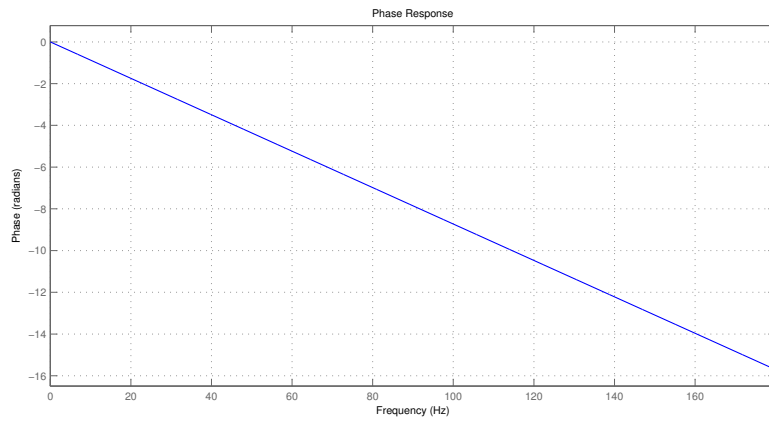
Filtr dolnoprzepustowy Filtracja dolnoprzepustowa $0Hz - 15Hz$ ma na celu usunięcie zakłóceń sygnału powstających pod wpływem drżenia mięśni ($35Hz$), wpływu zakłóceń sieci elektroenergetycznej $50/60Hz$, wpływu załamka T oraz pływającej izoliny elektrycznej. Powszechnie stosowany jest filtr o tranzytancji(3).

$$H(z) = \frac{1}{32} \frac{(1 - z^{-6})^2}{(1 - z^{-1})^2} \quad (3)$$

Dla częstotliwości próbkowania $fs = 200Hz$, częstotliwość odcięcia dla filtru (3) wynosi $fc = 11Hz$. Filtr wprowadza opóźnienie 6 próbek ($30ms$). Dla $60Hz$ filtr tłumi sygnał na poziomie większym niż $35dB$.



Rysunek 11: Odpowiedź amplitudowa filtra dolnoprzepustowego



Rysunek 12: Odpowiedź fazowa filtra dolnoprzepustowego

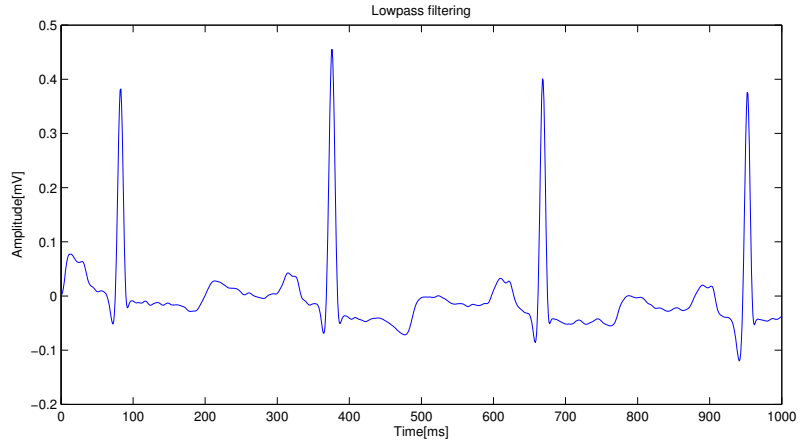
Rysunek (11) oraz (12) przedstawiają dokładne charakterystyki filtra dolnoprzepustowego. Filtr ten ma liniową odpowiedź fazową. Bardzo dobrze tłumi zakłócenia z linii energetycznych ($50/60\text{Hz}$) oraz pozostałe zakłócenia na poziomie powyżej 25dB .

Rysunek (13) przedstawia sygnał EKG po filtracji filtrem dolnoprzepustowym. Rezultat można porównać z oryginalnym sygnałem znajdującym się na rysunku (9), możemy dostrzec znaczną eliminację zakłóceń pochodzących od wyższych częstotliwości.

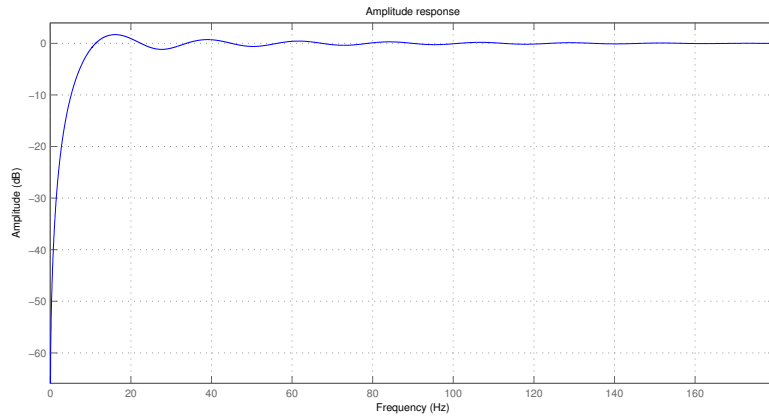
Filtr górnoprzepustowy Filtracja górnoprzepustowa ma na celu wyeliminowanie wszystkich niepożądanych sygnałów o niskich częstotliwościach. Transmitancja użytego filtra jest przedstawiona równaniem (4)

$$H(z) = 32z^{-16} - \frac{1 - z^{-32}}{1 - z^{-1}} \quad (4)$$

Dla częstotliwości próbkowania $f_s = 200\text{Hz}$, częstotliwość odcięcia wynosi $f_c = 5\text{Hz}$. Filtr wprowadza opóźnienie 16 próbek (80ms).



Rysunek 13: Sygnał EKG po filtracji dolnoprzepustowej (100_ MLII.dat)



Rysunek 14: Odpowiedź amplitudowa filtra górnoprzepustowego

Rysunek (14) oraz (15) przedstawiają dokładne charakterystyki filtra 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 (5).

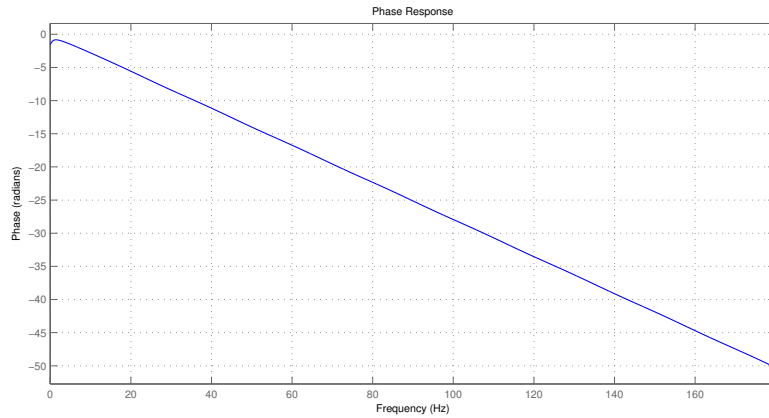
$$H(z) = 0.1(2 + z^{-1} - z^{-3} - 2z^{-4}) \quad (5)$$

Na rysunku (17) można dostrzec efekt różniczkowania sygnału.

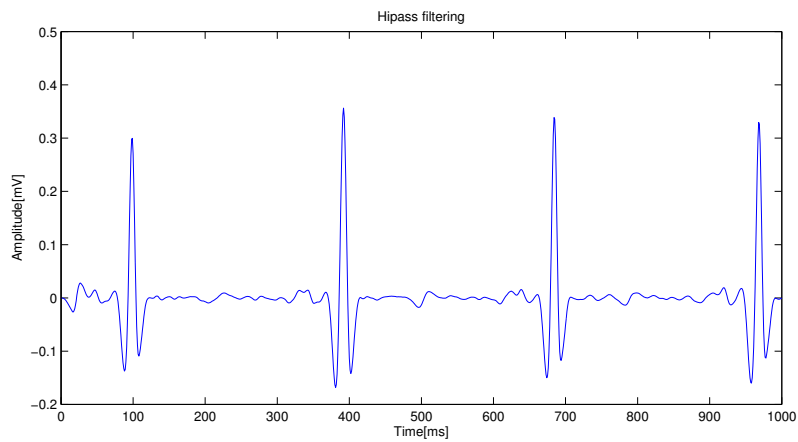
Potęgowanie sygnału Potęgowanie sygnału ma na celu jeszcze bardziej stłumić załamki *P* i *T* oraz wzmocnić próbki sygnału reprezentujące zespół *QRS*. Dodatkowo odwraca ujemną część sygnału i dzięki temu ujemne zespoły *QRS* również zostaną poddane analizie.

$$y(n) = (x(n))^2 \quad (6)$$

Efekt potęgowania sygnału możemy zaobserwować na rysunku (18).



Rysunek 15: Odpowiedź fazowa filtru górnoprzepustowego

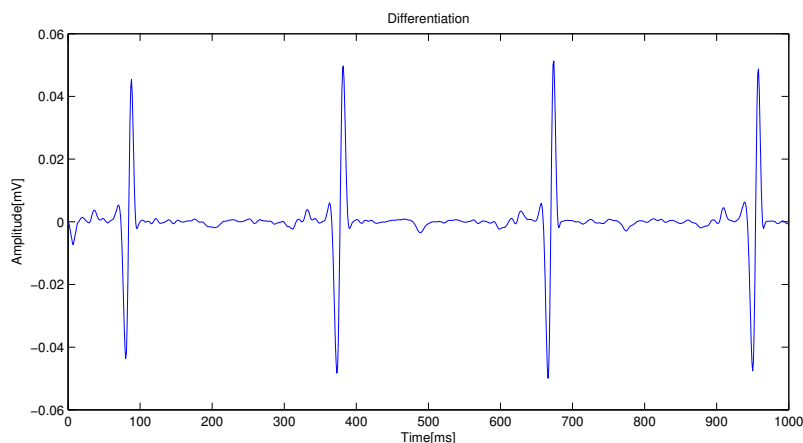


Rysunek 16: Sygnał EKG po filtracji górnoprzepustowej(100_ MLII.dat)

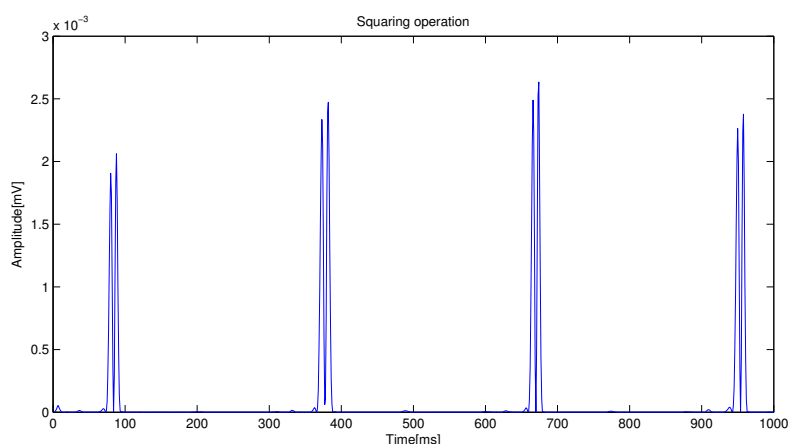
Całkowanie sygnału w ruchomym oknie Całkowanie sygnału w ruchomym oknie ma na celu uzyskanie pojedynczej “fali” w obrębie zespołu *QRS*. Bardzo ważną sprawą jest poprawne dobranie długości okna całkowania. Całkowanie sygnału w ruchomym oknie można uzyskać stosując wzór (7).

$$y(n) = \frac{1}{N} [x(n - (N - 1)) + x(n - (N - 2)) + \dots + x(n)] \quad (7)$$

Na rysunku (19) możemy zobaczyć rezultat prawidłowego doboru długości okna, cały zespół *QRS* jest widoczny bez dodatkowych zbędnych informacji, na rysunku (20) widać efekt zbyt krótkiego okna całkującego, kompleks *QRS* kończy się już przed załamkiem *R*, na rysunku (21) widać efekt zbyt długiego okna całkowania, oprócz zespołu *QRS* widać jeszcze część sygnału poza nim.



Rysunek 17: Sygnał EKG po różniczkowaniu (100_ MLII.dat)



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

Postać sygnału otrzymanego po całkowaniu ruchomym oknem jest przedstawiona na rysunku (22).

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

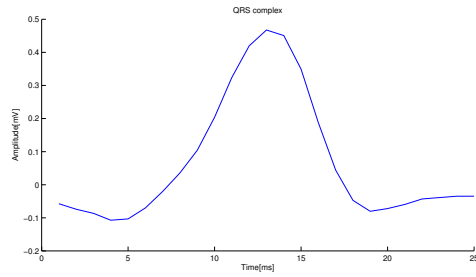
Wykrywanie załamka R Aby odnaleźć załamek R należy teraz odnaleźć początek i koniec każdego kompleksu QRS w sygnale oryginalnym, używamy do tego punktów zapisanych w odnalezionym wektorze po operacji progowania. Mając taki wycinek oryginalnego sygnału, wyszukujemy wartości maksymalnej sygnału i oznaczamy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

Na rysunku (23) można zobaczyć przykładową detekcję załamków R.

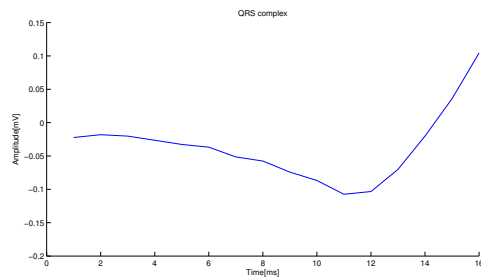
3.2.3 Opis procedur i metod

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:

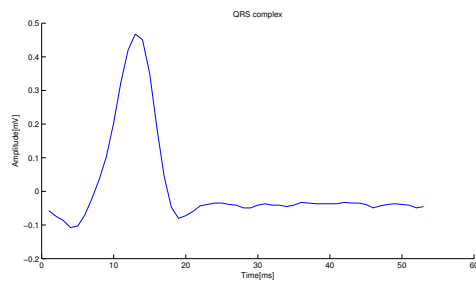
- Pant-Pompkins



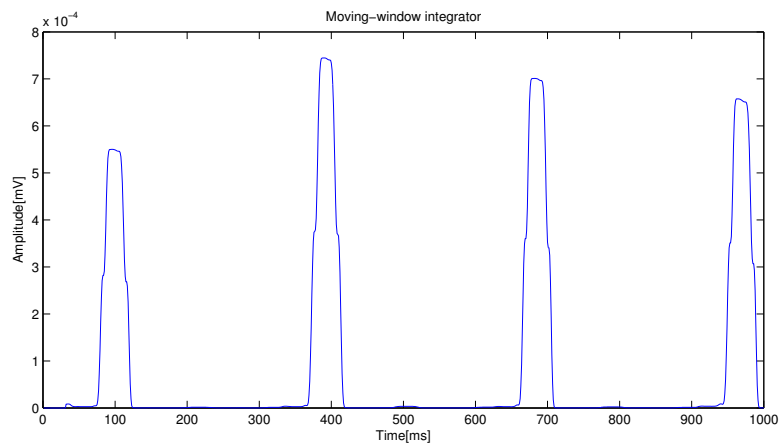
Rysunek 19: Prawidłowo wykryty kompleks QRS (100_ MLII.dat)



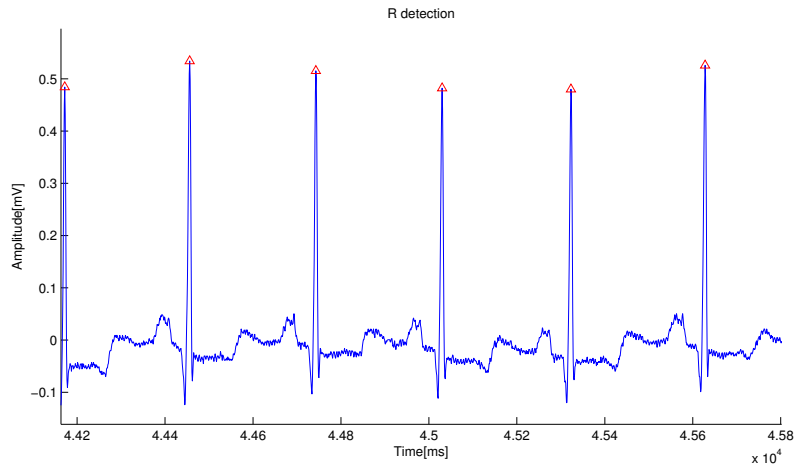
Rysunek 20: Za krótki wykryty kompleks QRS (100_ MLII.dat)



Rysunek 21: Za długi wykryty kompleks QRS (100_ MLII.dat)



Rysunek 22: Sygnał EKG scałkowany ruchomym oknem (100_ MLII.dat)



Rysunek 23: Detekcja załamków R (100_ MLII.dat)

- 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 Pant-Pomkins
 - 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 oraz znormalizowany w zakresie od -1 do 1. Detekcja pomija więc wstępną filtrację 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:

- TODO
- TODO
- TODO
- TODO
- TODO

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 ustawienia 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 Purkiniego 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 Purkiniego 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 transformacji 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 fałce 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 wolnozmiennnej, 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 Markova, algorytmów genetycznych lub transformacji Hilberta. Wszystkie opierają się na pomysłe utworzenia detektora zespołów QRS, jednak pozwalają jedynie na dropną 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 dość 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, że błędnie przeprowadzony sposób uczenia, może znacząco pogorszyć wykrywalność zespołów QRS.

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

3.3.3 Opis procedur i metod

Detekcja opiera się na algorytmie **threshold based detection**.
zastosowane funkcje:

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

Główna funkcja odpowiedzialna za detekcję zespołów QRS. Funkcja korzysta ze znajomości położenia załamek R, uzyskanych z modułu Rpeaks. Określa ona minima po obu stronach tych załamek, 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 załamek 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 załamek 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 ECGSignalChannel &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 i załamków P i T

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

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

Funkcja realizująca filtr uśredniający sygnał

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

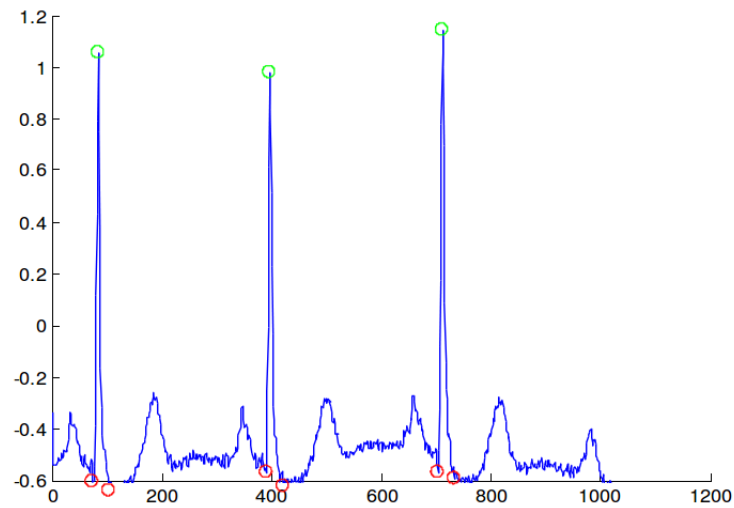
Funkcja ustawiająca wartości załamków R, używana do testowania modułu.

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

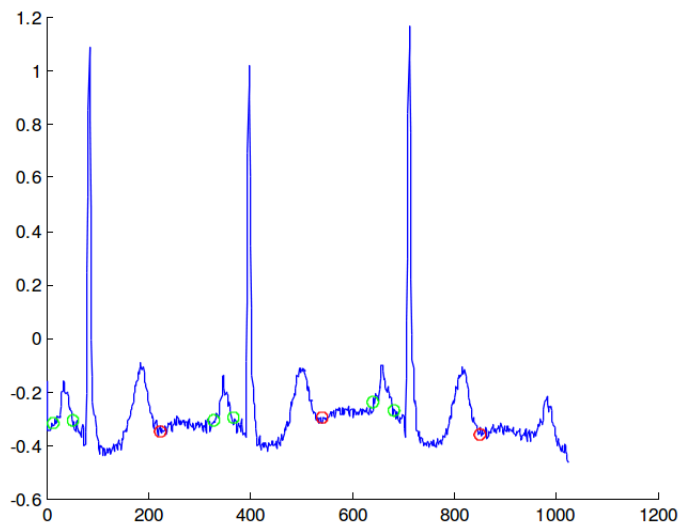
Funkcja ustawiająca wartości sygnału, używana do testowania modułu.

3.3.4 Warunki testowania

W celu przetestowania algorytmu najpierw wykonano implementację w programie Matlab 2009, sprawdzenie akutechności 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.



Rysunek 24: Wyniki modułu Waves – Matlab, oznaczone zespoły QRS i załamki R



Rysunek 25: Wyniki modułu Waves – Matlab, oznaczone załamki P, T i 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 24 oraz następnie oznaczono położenie załamków P i końców załamków T 25 są nieznacznie gorszej jakości niż uzyskane później w wynikowym programie 26 . Jest to spowodowane



Rysunek 26: Wyniki modułu Waves – Visual Studio 2010

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 HRV2

Autorzy: Łukasz Jaromir i Leszek Sosnowski

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 umożliwiającą 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
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
SDANNindex	średnia z odchyleń standardowych interwałów RR w 5 minutowych segmentach czasu całego zapisu
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. (za: Piotr Augustyniak *Przetwarzanie sygnałów elektrodagnostycznych*).

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
- void HRV1Analyzer::run() - metoda odpowiedzialna za uruchomienie algorytmów liczenia współczynników
- void HRV1Analyzer::setParams(ParametersTypes ¶meterTypes) - metoda odpowiedzialna za ustawianie parametrów (w naszym module nie wykorzystywana)
- double* HRV1Analyzer::cubicSpline(double* x, double* y, int nframe) - funkcja odpowiedzialna za interpolowanie próbek (wykonywana jest interpolacja funkcjami sklejanymi trzeciego stopnia)
- kiss_fft_cpx* HRV1Analyzer::copycpx(double *mat, int nframe) - funkcja alokująca klasę z danymi potrzebnymi do realizacji transformaty Fouriera

- `double HRV1Analyzer::mean(double *tab, int start, int end)` - funkcja obliczająca średnią arytmetyczną
- `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_PEAK`ów. 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 modulem `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 załamek `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 [?]. 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 [?]

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

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

a następnie znaleźć minimum z całki o wzorze:

$$\int_0^{+\infty} (D(t) - q(t))^2 dt \quad (8)$$

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

$$\sum (D(t) - q(t))^2 \rightarrow \text{minimum} \quad (9)$$

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

$$D(t)^2 \quad (10)$$

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

$$(D(t) - q(t))^2 \quad (11)$$

Algorytm obliczania parametrów SD1 i SD2

Parametry SD1 i SD2 są wyznaczone według poniższych wzorów:

$$SD1 = \sqrt{\frac{1}{2} \cdot SDSD^2} \quad (12)$$

$$SD2 = \sqrt{2 \cdot SDNN^2 - \frac{1}{2} \cdot SDSD^2} \quad (13)$$

gdzie:

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

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 załawkami R
- `double SamplingInterval` – atrybut klasy `GeometricAnalysis` przechowujący częstotliwość wczytanego sygnału

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

Argumenty funkcji:

- **rpeaks** – wektor numerów próbek z załawkami 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 wektora przez liczbę zmiennoprzecinkową

Używane zmienne:

- **unsigned int rpeaks_size** – długość wektora z numerami próbek załawków 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 ¶meterTypes` – 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.

Name	Value	Type
hrv2_data	{SD1=113.19729431820525 SD2=155.01218738405871 TINN=414.06250000000000 ...}	ECG_HRV2
SD1	113.19729431820525	double
SD2	155.01218738405871	double
TINN	414.06250000000000	double
M	92.00000000000000	double
N	39.00000000000000	double
HRV_TriangularIndex	24.973333333333333	double
Y	75.00000000000000	double
X	59.00000000000000	double
HistogramBinLength	7.812500000000000	double
histogram_x	{px=0x005c2f80 pn={...}}	boost::shared_ptr<WrappedVectorInt>
px	0x005c2f80 {signal=0x004fe8a0 }	WrappedVectorInt *
signal	0x004fe8a0 {size=275 stride=1 data=0x004fe8f0 ...}	gsl_vector_int *
size	275	unsigned int
stride	1	unsigned int
data	0x004fe8f0	int *
block	0x005c2fc0 {size=275 data=0x004fe8f0 }	gsl_block_int_struct *
owner	1	int
pn	{pi_=0x004fe850 }	boost::detail::shared_count
histogram_y	{py=0x005c29d0 pn={...}}	boost::shared_ptr<WrappedVectorInt>
py	0x005c29d0 {signal=0x005c2af8 }	WrappedVectorInt *
signal	0x005c2af8 {size=275 stride=1 data=0x005c2af8 ...}	gsl_vector_int *
size	275	unsigned int
stride	1	unsigned int
data	0x005c2af8	int *
block	0x005c2ab0 {size=275 data=0x005c2af8 }	gsl_block_int_struct *
owner	1	int
pi_	{pi_=0x005c2a10 }	boost::detail::shared_count
poincare_x	{px=0x005c2378 pn={...}}	boost::shared_ptr<WrappedVectorInt>
px	0x005c2378 {signal=0x005c2408 }	WrappedVectorInt *
signal	0x005c2408 {size=1872 stride=1 data=0x005a3b30 ...}	gsl_vector_int *
size	1872	unsigned int
stride	1	unsigned int
data	0x005a3b30	int *
block	0x005c2458 {size=1872 data=0x005a3b30 }	gsl_block_int_struct *
owner	1	int
pn	{pi_=0x005c23b8 }	boost::detail::shared_count
poincare_y	{py=0x005c24a0 pn={...}}	boost::shared_ptr<WrappedVectorInt>
py	0x005c24a0 {signal=0x005c2530 }	WrappedVectorInt *
signal	0x005c2530 {size=1872 stride=1 data=0x005a3b60 ...}	gsl_vector_int *
size	1872	unsigned int
stride	1	unsigned int
data	0x005a3b60	int *
block	0x005c2580 {size=1872 data=0x005a3b60 }	gsl_block_int_struct *
owner	1	int
pi_	{pi_=0x005c24e0 }	boost::detail::shared_count

Rysunek 27: 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 Matlab'a 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 27, 28 oraz 29 prezentują wyniki pierwszego testu opisanego w poprzedniej podsekcji:

3.6 HRV DFA

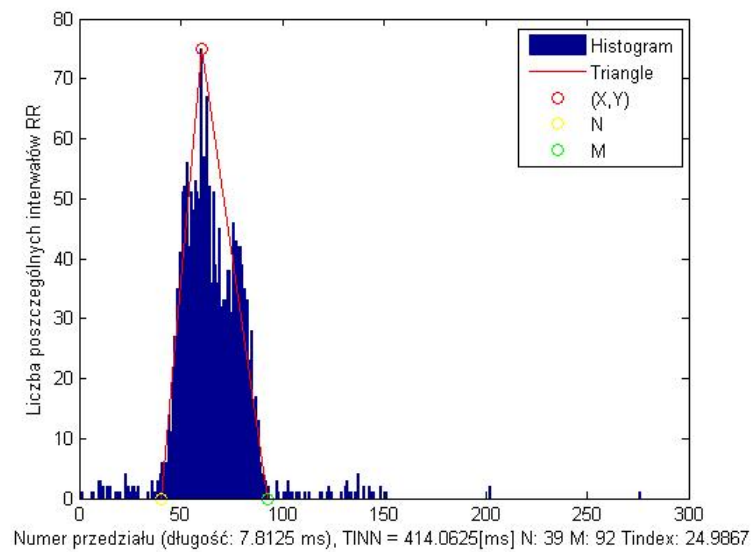
3.7 Klasyfikacja zespołów QRS

Autorzy: Krzysztof Bębenek

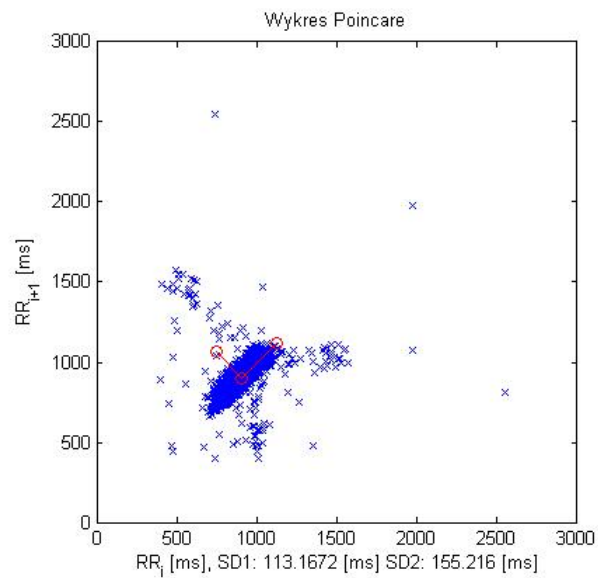
3.7.1 Opis zadania

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 morfologii pobudzenia służą:



Rysunek 28: Wykres histogramu – Matlab



Rysunek 29: Wykres Poincare – Matlab

- 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 zespół 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}

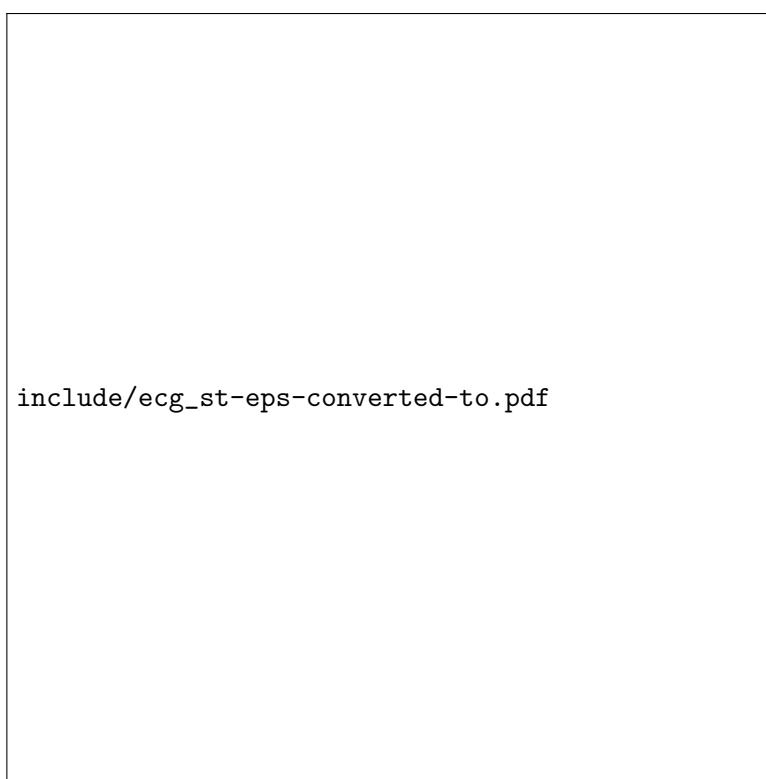
Dane zwracane przez moduł:

- 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 izoliny 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 [?] prezentuje bardzo prosty algorytm diagnostyki odcinka ST. Algorytm składa się z 3 kroków: lokalizacji odcinka, obliczenia uniesienia i nachylenia oraz wykrywanie epizodów.



Rysunek 30: Kluczowe punkty analizy odcinka ST. ISO - punkt referencyjny leżący na izoliny, J - początek odcinka, ST - koniec odcinka. Opracowanie własne.

Na rysunku 30 przedstawiono kluczowe punkty, które należy wyznaczyć przed wyznaczaniem parametrów diagnostycznych odcinka ST. Według wyżej wymienionego algorytmu, punktem referencyjnym w założeniu znajdującym się na izoliny jest początek zespołu QRS i koniec odcinka P. Punkt J, który 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 odstępów RR. Uniesieniem lub obniżeniem odcinka jest różnica wartości między 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 [?] 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 między punktem J+X (punkt X milisekund za punktem J), a punktem ISO. X jest wyznaczany na podstawie aktualnego HRV. Ocena przesunięcia odcinka ST względem izoliny

odbywa się z użyciem progu K , dla bazy MIT-BIH użyta wartość 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 zależności 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 określa się czy odcinek jest prosty czy zakrzywiony, a następnie określa się charakter zakrzywienia (wkłęsły/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 punktów ponad prostą jest większy od 70% to odcinek jest wypukły, jeżeli 70% punktów jest poniżej prostej, odcinek jest wkłęsły.

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 barię 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 `SimpleAnalyzer` oraz `ComplexAnalyzer`. Pierwsza klasa reprezentuje najprostszy algorytm zaprezentowany w [?, p. 155], druga zaś implementuje zmodyfikowany algorytm opisany w [?].

Lista i opis najważniejszych funkcji:

```
void STAnalysis::SimpleAnalyzer::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 załamek R.
- `const ECGRs& rpeaks` – struktura zawierająca tablicę numerów próbek kolejnych załamek R
- `const ECGWaves& waves` – struktura zawierająca tablice przechowujące numery próbek punktów charakterystycznych kolejnych zespołów QRS: P_{onset} , QRS_{onset} , QRS_{end} oraz T_{end} .
- `const ECGSignalChannel& 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::ComplexAnalyzer::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 załamek R.
- `const ECGR& rpeaks` – struktura zawierająca tablicę numerów próbek kolejnych załamek 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 `ComplexAnalyzer`:

```
int STAnalysis::ComplexAnalyzer::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 między dwoma punktami. Oprócz numeru próbki od początku sygnału, zwracana jest wartość największej różnicy między 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 między 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 załamek R z pliku z rozszerzeniem `atr` dostępnego razem z sygnałem. Testy polegały na wizualnej ocenie poprawności wykrytych punktów przedstawionych na wykresie. Kolejnym krokiem było testowanie algorytmu na przefiltrowanym sygnale oraz załamek R zwracanych przez moduł `RPeaks`, również na bazie oceny wizualnej wyznaczonych właściwości odcinka ST. Flaga `DEVELOPMENT` pozwala na przełączanie się między użyciem załamek wzorcowych (gdy jest ustawiona) i wykrywanych przez moduł `RPeaks` (gdy nie jest ustawiona).

3.8.5 Wyniki

Resultatem projektu są dwa algorytmy wyznaczania i analizy parametrów odcinków ST. Prostrzy algorytm opisany w [?] został w całości zaimplementowany i przetestowany. Drugi, rozszerzony algorytm z [?] 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 filtra 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.

3.9 T wave alt

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 31: 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 parameter negatywny, oba parametry negatywne. Na przykład w przypadku obu parametrów negatywnych ryzyko zgonu w ciągu roku od przebitego 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 32: 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 wyrównawczej). Obliczenia Turbulence Slope opierają się na tachogramie uśrednionym z wszystkich fragmentów rytmu zatokowego po skurczu komorowym. Wartości TS wyrażone w ms/odstęp RR. Prawidłowa wartość parametru to 2.5ms/RR.

Rysunek 33: 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 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 bliczenie 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 34: 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 35: 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 każdego kolejnych 5 interwałów wyznaczana jest prosta 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.

$$A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ x_4 & 1 \\ x_5 & 1 \end{bmatrix} \quad (14)$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} \quad (15)$$

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 paramterów HRT.

Klasa **ECGHRT** jest klasą wykorzystywaną do przechowywania danych uzyskanych podczas pracy - przechowuje najważniejsze obiekty będące wynikiem pracy HRTAnalyzeera - 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 oznaczających 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

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

Rysunek 36: Screen prezentujący ostateczny wygląd modułu

Rysunek 37: Screen prezentujący ostateczny wygląd modułu

Rysunek 38: Screen prezentujący ostateczny wygląd modułu