



Wydział Fizyki i Informatyki Stosowanej

Michał Piotrowski

Kierunek: Informatyka

Specjalność: Informatyka Stosowana

Specjalizacja: Algorytmy i programowanie

Numer albumu: 325585

Wizualizacja przebiegów EKG

Praca dyplomowa

wykonana pod kierunkiem

dr Pawła Kowalczyka

w Katedrze Fizyki Stosowanej

WFiIS UŁ

Łódź 2015

Streszczenie

Badanie elektrokardiograficzne to podstawowe badanie służące diagnostyce chorób serca. Tradycyjne urządzenia pomiarowe umożliwiają rejestrację krzywych elektrokardiograficznych tylko fizycznie na papierze. Celem pracy był projekt aplikacji graficznej umożliwiającej wyświetlanie krzywych elektrokardiograficznych oraz stworzenie narzędzi umożliwiających pomiar badanych krzywych. Podstawą aplikacji jest element, dzięki któremu możliwe jest wyświetlanie wykresów, ich skalowanie, pomiar. Istotnym elementem pracy jest również możliwość eksportu danych do pliku *.pdf, który umożliwi późniejszy fizyczny wydruk badania elektrokardiograficznego.

Słowa kluczowe: *Interakcja człowieka z komputerem , programowanie, wizualizacja danych, analiza danych*

Abstract

Title: Visualization of ECG waveforms

ECG is the basic test for the diagnosis of heart diseases. Traditional measuring devices allow registration of electrocardiographic curves only physically - on the paper. The aim of the study was to design graphic application which allows to display of electrocardiographic waveforms and creation of tools that allow to measure waveforms. The basis of the application is graphic component whereby it is possible to display charts, scaling charts and measurement of them. Other important element of thesis is also possibility to export data to *.pdf file, which allows to print out data physically on the paper.

Keywords: *Human-computer interaction, programming, data visualization, data analysis*

Spis treści

1. Wstęp	5
1.1 Fizyczne podstawy elektrokardiografii.....	5
1.2 Standardowy elektrokardiogram	6
1.3 Technika badania	7
1.4 Rejestracja elektrokardiogramu	8
2. Specyfikacja projektu	9
2.1 Cel i struktura pracy	9
2.2 Wymagania funkcjonalne	9
2.3 Wymagania нефunkcjonalne	10
3. Charakterystyka aplikacji	11
3.1 Narzędzia i technologie użyte w pracy	11
3.1.1 Zestaw bibliotek Qt	11
3.1.2 Środowisko programistyczne	11
3.1.3 Języki programowania.....	12
3.2 Opis klas	13
3.2.1 Klasa MainWindow	13
3.2.3 Klasa ScrollArea	15
3.2.4 Klasa Waveform	18
3.2.5 Klasa Plotter	20
3.2.6 Klasa PdfPlotter	21
3.3 Algorytm najbliższego sąsiedztwa	22
3.3.1 Opis algorytmu.....	22
3.3.2 Implementacja algorytmu	23
3.3.3 Złożoność czasowa oraz złożoność obliczeniowa	25
3.4 Interfejs graficzny użytkownika	26
3.4.1 Główne okno aplikacji.....	26
3.4.2 Okno generowania pliku *.pdf	34
4. Opis korzystania z programu	42
4.1 Wczytywanie pliku z danymi	42
4.2 Korzystanie z narzędzi	43
4.2.1 Zmiana skali.....	43
4.2.2 Powiększanie.....	44
4.2.3 Pomniejszanie	45
4.2.4 Przesuń w lewo.....	45
4.2.5 Przesuń w prawo	45
4.2.6 Pomiar horyzontalny wykresu	45

4.2.7 Pomiar wertykalny wykresu.....	47
4.3 Tworzenie pliku *.pdf.....	47
5. Podsumowanie	49
Bibliografia.....	50

1. Wstęp

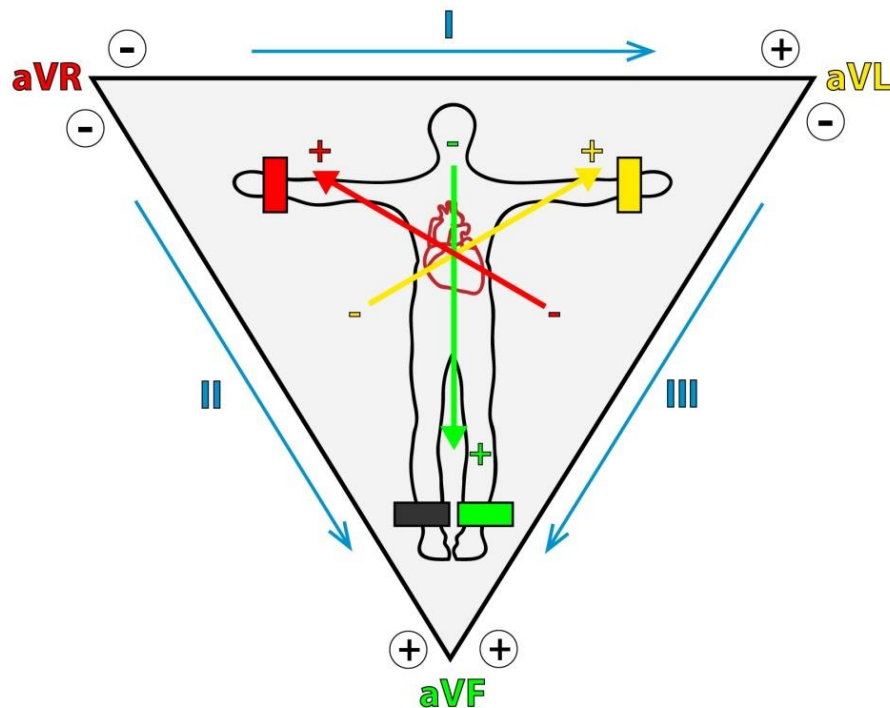
1.1 Fizyczne podstawy elektrokardiografii

Badanie elektrokardiograficzne polega na pomiarze różnicy potencjałów pomiędzy dwoma różnymi elektrodami. Zmiany napięcia między elektrodami odczytujemy w postaci krzywej elektrokardiograficznej. Różnice potencjałów mierzone są za pomocą galwanometru, który pełni najważniejszą rolę podczas przeprowadzonego badania. Elektrody znajdujące się na powierzchni ciała łączą się z galwanometrem, dzięki czemu powstaje obwód elektryczny zwany odprowadzeniem. Odprowadzenia możemy podzielić na odprowadzenia "jednobiegunowe" oraz "dwubiegunowe". Elektroda badająca w odprowadzeniach jednobiegunowych jest elektrodą o dodatnim napięciu. Umieszcza się ją w miejscu, w których chcemy zbadać potencjał. Druga elektroda jest elektrodą o zerowym napięciu. Obojętna elektroda zapewnia nam pomiar bezwzględny napięcia w konkretnym, jednym punkcie. Elektrody w odprowadzeniach dwubiegunowych umieszcza się na ciele pacjenta w punktach o różnym potencjale, dzięki czemu rejestrują różnice potencjałów pomiędzy dwoma konkretnymi punktami. Prąd elektryczny, który płynie od strony wyższego napięcia do niższego, powoduje wychylenie wskazówki galwanometru, które jest rejestrowane, a następnie widoczne na wykresie w postaci krzywej elektrokardiograficznej.

Podczas oceny pola elektrycznego wytwarzanego przez ludzkie serce wykorzystywana jest koncepcja Einthovena, która oparta jest na trzech założeniach:

1. Serce można porównać do dipola, czyli układu pary biegunów - dodatniego i ujemnego tworzących najprostszy generator energii elektrycznej. [1]
2. Serce znajduje się w środku geometrycznym tkanek, w których panują identyczne warunki przewodzenia prądu. [1]
3. Punkty połączeń obu kończyn górnych i lewej kończyny dolnej z tułowiem są wierzchołkami trójkąta równobocznego, w środku którego znajduje się serce. [1]

Wynika z tego, iż umieszczone na lewym podudziu oraz na obu przedramionach elektrody tworzą wierzchołki trójkąta równobocznego, w środku którego umieszczone jest serce. Taki układ odprowadzeń nazywany jest trójkątem Einthovena. Obrazuje on układ trzech dwubiegunowych odprowadzeń kończynowych (I, II, III).



Rys.1 Odprowadzenia kończynowe górne (zgodnie z teorią Einthovena) [3]

1.2 Standardowy elektrokardiogram

Standardowy elektrokardiogram obejmuje dwanaście odprowadzeń, których pomiar wykonuje się podczas rutynowego, spoczynkowego badania EKG. Odprowadzenia obejmują:

1. trzy odprowadzenia kończynowe dwubiegunowe (I, II, III)
2. trzy odprowadzenia kończynowe jednobiegunowe (aVR, aVL, aVF)
3. sześć odprowadzeń przedsercowych jednobiegunowych (V1,V2,V3,V4,V5,V6)

Odprowadzenie I ukazuje różnice potencjałów pomiędzy lewym przedramieniem pacjenta (wartość dodatnia), a prawym przedramieniem (wartość ujemna). Odprowadzenie II uzyskuje się z pomiaru napięcia pomiędzy lewym podudziem badanego (wartość dodatnia), a prawym przedramieniem (wartość ujemna). Rejestracja różnicy potencjałów odprowadzenia III odbywa się pomiędzy lewym podudziem pacjenta oraz lewym przedramieniem (oba o wartościach ujemnych napięcia). Odprowadzenia kończynowe jednobiegunowe uzyskuje się po podłączeniu elektrod odpowiednio:

1. aVR - do prawej kończyny górnej
2. aVL - do lewej kończyny górnej
3. aVF - do lewej kończyny dolnej

Odprowadzenia przedsercowe (V1,V2,V3,V4,V5,V6) służą rejestrowaniu bezwzględnej wartości potencjału w standardowych punktach klatki piersiowej. Zdarza się, iż rutynowy elektrokardiogram nie jest wystarczający do ustalenia rozpoznania elektrokardiograficznego. Wtedy należy wykonywać pomiary w dodatkowych miejscach. W ramach tej pracy inżynierskiej zostało stworzone uniwersalne narzędzie do wizualizacji tych danych, dzięki czemu bez trudu będzie można wyświetlić dodatkowe informacje, potrzebne do rozpoznania elektrokardiograficznego.

1.3 Technika badania

Badanie elektrokardiograficzne powinno się przeprowadzać w odpowiednich warunkach, które zapewniają dobrą jakość zapisu. Należy przede wszystkim zwrócić uwagę na ułożenie pacjenta podczas badania oraz na odpowiednią izolację elektryczną. Badany pacjent powinien być poddawany zabiegowi w pozycji leżącej. Górne kończyny nie powinny stykać się z tułowiem, a dolne kończyny same ze sobą. Dopuszcza się przeprowadzenie badania w pozycji siedzącej, ale tylko w wyjątkowych przypadkach. Elektrody mocowane do kończyn umieszcza się na zewnętrznych częściach podudzi i przedramion, zazwyczaj powyżej odpowiednio kostki i nadgarstka. Dla ułatwienia przewody oznacza się kolorami. Przewód o kolorze czerwonym podłącza się do prawej

ręki. Do lewej ręki - przewód żółty. Do kończyn dolnych przyczepia się odpowiednio: prawa noga - przewód czarny (uziemiaenie), lewa noga - przewód zielony. Elektrody przedsercowe mocowane są do klatki piersiowej za pomocą gumowych pasków lub stosuje się elektrody przyssawkowe. Aby uzyskać wiarygodne wyniki badania, elektrody nie powinny stykać się pomiędzy sobą, oraz ściśle przylegać do powierzchni ciała. Zmniejszenie oporu elektrycznego między skórą i elektrodami uzyskać można dzięki odtłuszczeniu skóry alkoholem, użyciu żelu do EKG, dzięki gazikom zwilżonym roztworem soli fizjologicznej lub zwykłą wodą.

1.4 Rejestracja elektrokardiogramu

Przebiegi elektrokardiograficzne rejestruje się dzisiaj na ruchomej taśmie papieru lub elektronicznie. Ważnym elementem badania EKG jest ustawienie szybkości przesuwu papieru oraz sprawdzenie czułości urządzenia poprzez ustawienie cechy. Cechę zaznacza się na każdym odcinku badania. Dzięki niej możemy obiektywnie pomierzyć amplitudy poszczególnych załamków.

Uniwersalna wartość przesuwu papieru to 25 mm/s i 50 mm/s. Stosowany papier do wydruku posiada podziałkę milimetrową, ułatwiającą odczyt i analizę przebiegu. Dzięki znajomości szybkości przesuwu papieru, możliwe jest obliczenie czasu trwania odstępów, odcinków, załamków. Dane te umożliwiają też obliczenie rytmu serca na minutę.

2. Specyfikacja projektu

2.1 Cel i struktura pracy

Założeniem niniejszej pracy inżynierskiej jest stworzenie graficznej aplikacji, która umożliwi wyświetlanie dwunastu lub więcej wykresów (przebiegów) rejestrowanych podczas wykonywania badania EKG, oraz ułatwi ich wstępną analizę. Podstawą tej pracy będzie element umożliwiający wyświetlanie wielu krzywych kardiograficznych, ich odpowiednie skalowanie, pomiar. Jedną z przyczyn powstania tej pracy inżynierskiej jest współpraca Uniwersytetu Łódzkiego oraz Uniwersytetu Medycznego w Łodzi, w celu opracowania nowoczesnej, tańszej i alternatywnej metody wykonywania zabiegu elektrokardiograficznego.

Oprogramowanie powinno działać na wielu systemach operacyjnych, tak aby zapewnić jak największą możliwość używania go w praktyce.

Praca została podzielona na cztery rozdziały, opisujące zadania potrzebne do wykonania zadanego problemu oraz proponowane rozwiązania poszczególnych kwestii związanych z implementacją aplikacji. Praca pokazuje drogę jaką wykonał autor od zadanego problemu do końcowej implementacji, wraz z opisem działań i aktywności wykonanych podczas tworzenia projektu.

2.2 Wymagania funkcjonalne

Wymagania funkcjonalne zostały sformułowane na podstawie wywiadu z klientem (w tym wypadku z lekarzami Uniwersytetu Medycznego). Ze względu na złożoność wymagań oraz biorąc pod uwagę wymagania pracy dyplomowej inżynierskiej, wybrano najpotrzebniejsze cechy konieczne do wprowadzenia aplikacji na rynek komercyjny. Wymagania te są następujące:

1. Dowolna długość czasu badania - aplikacja powinna umożliwiać wizualizację dużych, złożonych danych, niezależnie od długości czasu przeprowadzanego badania.
2. Wybór fragmentów badania i zapis ich do pliku *.pdf - umożliwienie lekarzowi wycięcie pewnej długości zapisu badania i zapisanie ich w pliku *.pdf.

3. Szybkość przesuwu papieru 25 mm/s - dzięki tej informacji możliwe jest obliczenie czasu trwania poszczególnych załamków, odcinków, częstość rytmu serca w ciągu jednej minuty.
4. Powiększanie lub pomniejszanie zaznaczonych fragmentów obrazu.
5. Linijka lub inne narzędzie ułatwiające pomiar.

2.3 Wymagania нефunkcjonalne

Wymagania нефunkcjonalne są bardzo często pomijane podczas uzgadniania specyfikacji z klientem. Są one bardzo trudne do opisanja, przez co uniemożliwiają poprawne testowanie oprogramowania. Z tego powodu wymagania нефunkcjonalne powinny być jasno sprecyzowane. Podczas projektowania tej aplikacji, starano się uzyskać jak największą zgodność z wymaganiami нефunkcjonalnymi, co powodowało wydłużenie prac oraz dbałość o każdy szczegół kodu. Wymagania нефunkcjonalne przedstawiają się następująco:

1. Wczytywanie pliku z danymi powinno być proporcjonalnie długie do ilości danych w pliku.
2. Idealnie, jeżeli aplikacja będzie wyświetlała wykresy poniżej 1 sekundy.
3. Nieakceptowalne są bardzo widoczne opóźnienia w generowaniu obrazu.
4. Wieloplatformowość - stworzenie aplikacji, którą będzie można uruchomić na wielu systemach operacyjnych.

3. Charakterystyka aplikacji

3.1 Narzędzia i technologie użyte w pracy

3.1.1 Zestaw bibliotek Qt

Qt to nic innego jak zestaw przenośnych bibliotek i zbiór narzędzi programistycznych dedykowanych dla takich języków programowania jak C++, Java czy QML. Podstawowym składnikiem biblioteki są klasy zapewniające możliwość budowy interfejsu graficznego aplikacji.

Wersja Qt 1.0 ukazała się dnia 24 września 1996 roku. Od tamtej pory wprowadzono wiele usprawnień i nowych wersji biblioteki. Najnowsza wersja, Qt 5.0, została opublikowana 19 grudnia 2012 roku. Właśnie tą wersją oprogramowania autor posługuje się pisząc ten projekt dyplomowy. W skład biblioteki Qt wchodzi wiele narzędzi, m. in. :

- qmake - program zarządzający procesem kompilacji; tworzy i aktualizuje plik Makefile na podstawie definicji projektu, pliku *.pro.
- Qt Designer - aplikacja umożliwiająca definiowanie interfejsu graficznego, np. przycisków, okien dialogowych, itp.
- Qt Creator - zintegrowane środowisko programistyczne

3.1.2 Środowisko programistyczne

W niniejszej pracy dyplomowej wykorzystano darmową wersję środowiska Qt Creator. Środowisko Qt jest dostępne dla platform m. in. Linux, Windows, Mac OS X. Za pomocą środowiska bibliotek Qt zostały przygotowane takie aplikacje jak Skype czy Google Earth.

Środowisko zawiera w sobie edytor kodu źródłowego obsługujący podświetlanie składni czy autouzupełnianie, oraz narzędzie zwane Qt Designer, które ułatwia projektowanie graficznego interfejsu użytkownika (GUI).

Qt Creator nie posiada własnego debuggera. Zawiera jedynie plugin, który umożliwia działanie jednego z natywnych debuggerów C++, są to m. in.:

- GNU Debugger (GDB)
- Microsoft Console Debugger (CDB)
- LLVM debugger (LLDB)

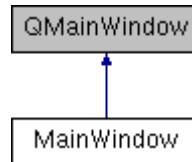
Środowisko zapewnia obsługę systemów kontroli wersji takich jak: ClearCase, Git, Subversion oraz innych.

3.1.3 Języki programowania

Aplikacja będąca częścią niniejszego projektu inżynierskiego, została w całości napisana w języku C++ z użyciem biblioteki Qt oraz środowiska Qt Creator. Zapewnia to wieloplatformowość i uniwersalność napisanego oprogramowania.

3.2 Opis klas

3.2.1 Klasa MainWindow



Główne okno aplikacji jest reprezentowane przez tę klasę. Zapewnia szkielet do dalszego budowania aplikacji, reprezentuje graficzny interfejs użytkownika. Jest to najbardziej rozbudowana klasa tej aplikacji, jednocześnie będąc klasą o największym znaczeniu dla projektu. Klasa MainWindow zawiera wiele funkcji służących do obsługi aplikacji, np. wczytywanie pliku czy obsługę narzędzi aplikacji. *Tabela 1* przedstawia funkcje i atrybuty składowe klasy MainWindow.

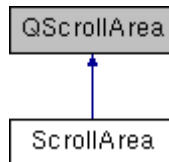
Nazwa funkcji / atrybutu	Opis
<code>MainWindow(QWidget *parent = 0)</code>	Konstruktor klasy. Tworzy szkielet wizualny, tworzy menu narzędziowe, obiekty reprezentujące akcje, wątki potrzebne podczas działania aplikacji itp.
<code>~MainWindow()</code>	Destruktor klasy MainWindow.
<code>createActions() : void</code>	Tworzy akcje, czyli obiekty reprezentujące działania przycisków.
<code>loadFile() : bool</code>	Funkcja wczytująca plik z danymi. Zwraca <code>true</code> jeżeli dane zostały wczytane, <code>false</code> jeżeli wczytywanie pliku się nie powiodło.
<code>calculateAmplitude() : void</code>	Funkcja licząca amplitudę wszystkich wczytywanych wykresów łącznie.
<code>addWaves(_nrOfWaves : int, *_data : QVector<QPointF>, _size : QSize) : void</code>	Tworzy obiekty klasy Waveform, podłącza odpowiednie sygnały i gniazda do obiektów, oraz uruchamia wątki obiektów typu Waveform.
<code>setToolBar() : void</code>	Tworzy pasek narzędzi.
<code>zoomIn() : void</code>	Funkcja obsługuje zdarzenie powiększenia wykresu.
<code>zoomOut() : void</code>	Funkcja obsługuje zdarzenie pomniejszenia wykresu.

exitApplicationSlot() : void	Gniazdo służące do zamykania aplikacji.
loadFileSlot() : void	Gniazdo uruchamiające funkcję do wczytywania pliku.
receiveBgHeightSlot(float) : void	Gniazdo odbierające aktualną wysokość tła aplikacji – obiektu typu Background.
disablePaperSlot() : void	Gniazdo wyłączające papier milimetrowy.
horizontalMeasurerSlot() : void	Gniazdo włączające poziome linie pomiarowe.
verticalMeasurerSlot() : void	Gniazdo ustawiające pionowe linie pomiarowe.
closeWaveformsSlot() : void	Gniazdo zamykające aktualnie wczytany plik.
zoomInSlot() : void	Gniazdo uruchamia funkcję do zwiększania wykresów.
zoomOutSlot() : void	Gniazdo uruchamia funkcję do zmniejszania wykresów.
getHorizontalDiffSlot(float) : void	Gniazdo pobiera odległość pomiędzy dwoma poziomymi liniami pomiarowymi i ustawia wartość w miejscu widocznym dla użytkownika.
getVerticalDiffSlot(float) : void	Gniazdo pobiera odległość pomiędzy dwoma pionowymi liniami pomiarowymi i ustawia wartość w miejscu widocznym dla użytkownika.
setNormalCursorSlot() : void	Gniazdo wyłącza narzędzia pomiarowe jeżeli są włączone i ustawia normalne działanie kursora.
goLeftSlot() : void	Gniazdo obsługujące przesuwanie wykresu w lewo.
goRightSlot() : void	Gniazdo obsługujące przesuwanie wykresu w prawo.
oneToOneSlot() : void	Gniazdo ustawiające stosunek wartości X wczytanych danych do wartości X wyświetlanych danych na 1 : 1
oneToTwoSlot() : void	Gniazdo ustawiające stosunek wartości X wczytanych danych do wartości X wyświetlanych danych na 1 : 2
oneToFourSlot() : void	Gniazdo ustawiające stosunek wartości X wczytanych danych do wartości X wyświetlanych danych na 1 : 4
oneToEightSlot() : void	Gniazdo ustawiające stosunek wartości X wczytanych danych do wartości X wyświetlanych danych na 1 : 8
oneToSixteenSlot() : void	Gniazdo ustawiające stosunek wartości X wczytanych danych do wartości X wyświetlanych danych na 1 : 16
getIdxOfActiveWidgetSlot(int) : void	Gniazdo pobiera numer aktualnie aktywnego wykresu.
createPDFSlot() : void	Gniazdo uruchamiające okno do tworzenia pliku *.pdf.

sendBgStatusSignal(bool) : void	Sygnal wysyłający informację do obiektu tła z informacją o statusie papieru milimetrowego.
--	--

Tabela 1. Funkcje składowe klasy MainWindow

3.2.3 Klasa ScrollArea

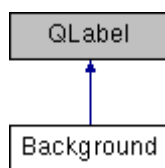


Niniejsza klasa zapewnia rozszerzalną przestrzeń dla głównego okna. Dzięki tej klasie możliwe jest umieszczanie nieskończenie wielu wykresów w dowolnej skali bez konieczności zwiększania rozmiarów głównego okna aplikacji. Kiedy ScrollArea osiąga odpowiedni rozmiar, na boku ekranu pojawia się pasek, dzięki któremu użytkownik może przewinąć zawartość do odpowiedniego miejsca. *Tabela 2* przedstawia funkcję klasy ScrollArea.

Nazwa funkcji	Opis
<code>ScrollArea(QWidget *parent = 0)</code>	Konstruktor klasy ScrollArea.
<code>~ScrollArea()</code>	Destruktor klasy ScrollArea
<code>addWaveform(* _wf : Waveform) : void</code>	Funkcja dodająca obiekt klasy Waveform do obiektu klasy Background
<code>resizeBackground(_f : float) : void</code>	Funkcja zmieniająca wysokość tła do wartości _f
<code>clear() : void</code>	Funkcja wywołująca funkcję czyszczenia tła na obiekcie typu Background.
<code>removeWaveforms() : void</code>	Funkcja usuwająca wykresy z obiektu typu Background
<code>getBgHeightSlot(float) : void</code>	Gniazdo odbiera wysokość tła oraz wysyła sygnał z wartością do obiektu MainWindow.
<code>getBgStatusSlot(bool): void</code>	Gniazdo odbiera wartość oznaczająca status tła.
<code>getScaleSlot(float) : void</code>	Gniazdo odbiera skale oraz wysyła ją do obiektu klasy MainWindow.
<code>resizeBgSignal(float) : void</code>	Sygnał wysyła wysokość do jakiej tło powinno zostać przekonwertowane.
<code>sendBgHeightSignal(float) : void</code>	Sygnał wysyła informację do obiektu MainWindow o wartości wysokości tła.
<code>sendBgStatusSignal(bool) : void</code>	Sygnał wysyła informację o statusie tła.

Tabela 2. Funkcje składowe klasy ScrollArea.

3.2.3 Klasa Background

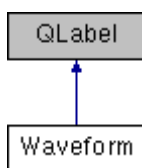


Dzięki tej klasie możliwe jest wyświetlenie tła aplikacji, a konkretniej tła dla wyświetlanych przebiegów elektrokardiograficznych. Zapewnia wyświetlanie papieru milimetrowego w odpowiedniej skali, dopasowując się automatycznie do wielkości wykresów. *Tabela 3* zawiera funkcje składowe klasy Background.

Nazwa funkcji	Opis
<code>Background(QWidget *parent = 0)</code>	Konstruktor klasy Background.
<code>~Background()</code>	Destruktor klasy. Background
<code>addWaveform(*_wave : Waveform) : void</code>	Dzięki tej funkcji możliwe jest dodawanie obiektów klasy Waveform do układu klasy Background.
<code>clearBackground() : void</code>	Czyści tło i jego układ z wszystkich obiektów klasy Waveform.
<code>resizeSlot(float) : void</code>	Gniazdo odbiera wartość zmiennoprzecinkową i ustawia wysokość obiektu Background do tej wartości.
<code>repaintSlot() : void</code>	Gniazdo odbiera żądanie odmalowania tła.
<code>setStatusSlot(bool) : void</code>	Gniazdo otrzymuje wartość typu bool i ustawia wewnętrzną flagę isPaperNeeded na zadaną wartość.
<code>resizeParentSignal(s : QSize) : void</code>	Sygnał wysyła obiekt s, zawierający informację o rozmiarze obiektu Background.
<code>sendHeightSignal(float) : void</code>	Sygnał wysyła wysokość obiektu Background.
<code>paintEvent(*_event : QPaintEvent) : void</code>	Przeddefiniowane zdarzenie umożliwiające malowanie po układzie klasy Background.

Tabela 3. Funkcje składowe klasy Background.

3.2.4 Klasa Waveform



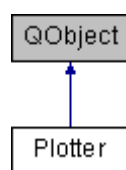
Klasa reprezentująca jedno, konkretne odprowadzenie (przebieg) pomiaru badania elektrokardiograficznego. Tabela 4 zawiera funkcję składowe klasy Waveform.

Nazwa funkcji	Opis
<code>Waveform(* data: QVector<QPointF>, size: QSize)</code>	Konstruktor klasy Waveform.
<code>~Waveform()</code>	Destruktor klasy Waveform.
<code>replot() : void</code>	Funkcja uruchamia odpowiednie funkcje na obiekcie <code>plotter : Plotter</code> , aby odmalować zawartość klasy Waveform.
<code>next() : void</code>	Umożliwia przejście do następnej części wyświetlanego wykresu.
<code>prev() : void</code>	Umożliwia przejście do poprzedniej części wyświetlanego wykresu.
<code>updateSize() : void</code>	Funkcja wysyła sygnał z wysokością obiektu klasy Waveform do obiektu klasy Background.
<code>changeRatioX(float _ratio) : void</code>	Ustawia wartość <code>ratioX</code> : float obiektu <code>plotter : Plotter</code> na wartość <code>_ratio</code> .
<code>changeRatioXScale(float _scale) : void</code>	Ustawia wartość <code>ratioXScale</code> : float obiektu <code>plotter : Plotter</code> na wartość <code>_scale</code> .
<code>isActive() : bool</code>	Zwraca wartość typu bool parametru <code>isActive</code> : bool.
<code>setActive(bool _active) : void</code>	Ustawia wartość <code>isActive</code> : bool na wartość <code>_active</code> .
<code>emitMeasures() : void</code>	Funkcja wysyła sygnały z wartością pomiaru.
<code>paintEvent(QPaintEvent * _event) : void</code>	Przeddefiniowane zdarzenie umożliwiające odmalowanie klasy Waveform.
<code>mousePressEvent(QMouseEvent * _event) : void</code>	Przeddefiniowane zdarzenie wykrywające naciśnięcie przycisku.
<code>mouseMoveEvent(QMouseEvent * _event) : void</code>	Przeddefiniowane zdarzenie wykrywające przesunięcie myszy.

<code>mouseReleaseEvent(QMouseEvent * _event) : void</code>	Przedefiniowane zdarzenie wykrywające uwolnienie przycisku myszy.
<code>updatePlotSlot(p : QImage) : void</code>	Gniazdo odbierające obiekt p : QImage, u ustawiające plot : QImage na p.
<code>changeParentSizeSignal(float) : void</code>	Sygnał wysyła informację o zmianie wielkości widżetu rodzica.
<code>plotReqSignal() : void</code>	Sygnał wysyłający żądanie odmalowania obiektu.
<code>plotHeightSignal(float) : void</code>	Sygnał wysyła informację z wysokością obiektu plot : QImage.
<code>horizontalDiffSignal(float) : void</code>	Sygnał wysyła informację z pomiarem horyzontalnym.
<code>verticalDiffSignal(float) : void</code>	Sygnał wysyła informację z pomiarem wertykalnym.
<code>sendIdOfActiveWidgetSignal(int) : void</code>	Sygnał wysyła id : int aktywnego widżetu.

Tabela 4 zawiera funkcje składowe klasy Waveform

3.2.5 Klasa Plotter



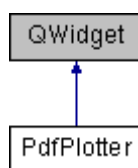
Klasa Plotter zapewnia szybkie odmalowanie konkretnej części wykresu wyświetlanego na ekranie. Tabela 5 zawiera funkcje składowe klasy Plotter.

Nazwa funkcji	Opis
<code>Plotter(_size : QSize, _data : QVector<QPointF>);</code>	Konstruktor klasy Plotter.
<code>~Plotter()</code>	Destruktor klasy Plotter.
<code>invertData(* _vector QVector<QPointF>) : void</code>	Funkcja służy do odwracania danych względem rozmiaru obiektu klasy Plotter.
<code>next() : void</code>	Funkcja umożliwia przejście do następnej części wykresu.
<code>prev() : void</code>	Funkcja umożliwia przejście do poprzedniej części wykresu.
<code>getRange() : int</code>	Pobiera wartość range : int.
<code>replot() : void</code>	Funkcja używa sygnału finishSignal() aby zażądać odmalowania wykresu.
<code>getScale() : float</code>	Funkcja pobiera skalę.
<code>scaleWave(QVector<QPointF> * _data) : void</code>	Funkcja służy do skalowania danych.
<code>getAmplitude() : float</code>	Funkcja pobiera amplitudę wykresu.
<code>getAmplitude(QVector<QPointF> * _tmp) : float</code>	Funkcją pobiera łączną amplitudę wykresów znajdujących się na wektorze _tmp.
<code>setMarkers() : void</code>	Funkcja ustawia znaczniki.
<code>calculateResamplingFactors() : void</code>	Funkcja przelicza wartości względem których skalowane są dane.
<code>changeRatioX(float _ratio) : void</code>	Funkcja zmienia stosunek zmiennej X danych wejściowych do wyjściowych.
<code>doneSignal(QImage) : void</code>	Sygnał wysyła obrazek do odmalowania.
<code>finishSignal() : void</code>	Sygnał informuje o możliwości odmalowania obrazka.

resizeSignal(QSize) : void	Sygnał wysyła informację o rozmiarze obrazka.
plotSlot() : void	Gniazdo odbiera informację o możliwości odmalowania obrazka.
updateHeightSlot(float) : void	Gniazdo odbiera wysokość obrazka, jaka ma być odmalowana przy najbliższym żądaniu.

Tabela 5. Zawiera funkcję składowe klasy Plotter.

3.2.6 Klasa PdfPlotter



Klasa PdfPlotter umożliwia utworzenie pliku *.pdf, służącego do wydruku fizycznego. Tabela 6 zawiera funkcję składowe klasy PdfPlotter.

Nazwa funkcji	Opis
PdfPlotter(* _v : QVector<QPointF>, _counter : int, _amplitude : float)	Konstruktor klasy PdfPlotter.
~PdfPlotter()	Destruktor klasy PdfPlotter.
fillBackground(* _pix : QPixmap) : void	Funkcja wypełnia przestrzeń na wykresy siatką papieru milimetrowego.
drawWaveform(* _pix : QPixmap, _currentPage : int) : void	Funkcja maluje wykres na obiekcie _pix.
prepareData() : void	Funkcja przygotowuje dane do wydruku do pliku.
saveSlot() : void	Gniazdo odbiera żądanie zapisu pliku na dysk.

Tabela 6. Zawiera funkcję składowe klasy PdfPlotter.

3.3 Algorytm najbliższego sąsiedztwa

3.3.1 Opis algorytmu

Algorytm najbliższego sąsiedztwa (ang. *Nearest Neighbour*) to najprostszy algorytm służący do zmiany wielkości wyświetlanego obrazu. W grafice komputerowej proces polegający na zmianie właściwości obrazu (zmniejszenie, powiększenie, obrót itp.) nazywa się resamplingiem. Każdy pojedynczy piksel obrazu wyjściowego przyjmuje wartość piksela obrazu źródłowego położonego w najbliższym sąsiedztwie aktualnie rozważanego punktu. Matematycznie algorytm można opisać następująco :

- $width_{src}$ - długość obrazu źródłowego
- $width_{dst}$ - długość obrazu wyjściowego
- $height_{src}$ - wysokość obrazu źródłowego
- $height_{dst}$ - wysokość obrazu wyjściowego
- $P(i,j)$ – punkt wejściowy o współrzędnych (i,j)
- $P'(x,y)$ – punkt wyjściowy o współrzędnych (x,y)

Możemy przyjąć iż :

- $ratio_x = width_{src} / width_{dst}$
- $ratio_y = height_{src} / height_{dst}$

Wówczas dla każdego pojedynczego punktu obrazu możemy przeprowadzić następujące obliczenia:

$$x = i * ratio_x$$

$$y = j * ratio_y$$

Oznacza to, iż aby uzyskać punkt $P'(x,y)$, należy pomnożyć współrzędne punktu $P(i,j)$ przez odpowiednie wartości $ratio_x$ oraz $ratio_y$.

Przykład:

Należy przekształcić obraz 4x12 w obraz 2x10.

Obliczmy współrzędne powiększenia obrazu:

$$ratio_x = width_{src} / width_{dst} = 4/2 = 2$$

$$\text{ratioy} = \text{height}_{\text{src}} / \text{height}_{\text{dst}} = 12/10 = 1.2$$

Punkt (0,0)

$$x = i * \text{ratiox} = 0 * 2 = 0$$

$$y = j * \text{ratioy} = 0 * 1.2 = 0$$

Punkt (0,0) obrazu wyjściowego przyjmuje wartość punktu (0,0) obrazu źródłowego.

Punkt (1,1)

$$x = i * \text{ratiox} = 1 * 2 = 2$$

$$y = j * \text{ratioy} = 1 * 1.2 = 1.2$$

Punkt (1,1) obrazu wyjściowego przyjmuje wartość punktu (1,1.2) obrazu źródłowego.

Kolejne punkty oblicza się analogicznie do podanego procesu.

3.3.2 Implementacja algorytmu

Algorytm najbliższego sąsiedztwa został zaimplementowany w tym projekcie, aby w szybki i prosty sposób uzyskiwać powiększone lub zmniejszone wykresy przebiegów elektrokardiograficznych. Implementacja wygląda następująco:

```
void Plotter::plot()
{
    widthDest = widthSrc / skala;
    heightDest = heightSrc / skala;

    ratioX = widthSrc/widthDest;
    ratioY = heightSrc/heightDest;
    // Dalsza część kodu
}
```

Na początku obliczane są wartości długości i wysokości obrazu wyjściowego, który chcemy uzyskać. Liczone są na podstawie wielkości obrazu źródłowego, które są stałymi wartościami po wczytaniu pliku. Kluczowym elementem jest tutaj

skala, która pozwala na dopasowanie obrazu do pożądanej wielkości. Następnie obliczane są wartości współczynników ratioX oraz ratioY, które wykorzystane są w dalszej części algorytmu.

```
void Plotter::plot()
{
    // Wcześniejsza część kodu

    // deklaracja wektora punktów, który zostanie użyty w algorytmie
    QVector<QPointF> tmp;

    // część kodu odpowiedzialna za określenie ilości punktów do skopiowania i
    umieszczenia ich na wektorze tmp

    // Użycie funkcji scale(QVector<QPointF> * data)
    scale(&tmp);
}
```

Definicja funkcji scale() jest następująca:

```
void Plotter::scale(QVector<QPointF> *data)
{
    for(QVector<QPointF>::iterator it = data->begin(); it != data->end(); ++it){
        it->setY(it->y()*ratioY);
        it->setX(it->x()*ratioX);
    }
}
```

Zauważmy, że powyższa funkcja odpowiada za proces przeliczania punktów z obrazu źródłowego do wyjściowego.

3.3.3 Złożoność czasowa oraz złożoność obliczeniowa

Algorytm najbliższego sąsiedztwa jest algorytmem szybkim, ponieważ nie wymaga wielu obliczeń. Niestety przy dużych powiększeniach zauważalne są pojedyncze piksele z obrazu źródłowego. Jest to spowodowane tym, iż dla wielu punktów obrazu wyjściowego brany do obliczeń jest dokładnie ten sam punkt obrazu źródłowego. Przy pomniejszeniu natomiast w obrazie wyjściowym możemy stracić całe linie punktów obrazu źródłowego.

Złożoność czasowa

Złożoność czasowa algorytmu jest liniowa. Algorytm określa położenie punktu P' na podstawie współrzędnych punktu P . Czas wykonania jest zależny tylko i wyłącznie od ilości elementów, punktów, które należy obliczyć. Zatem złożoność czasową możemy oznaczyć następująco:

$$O(n)$$

Złożoność obliczeniowa

Złożoność obliczeniowa algorytmu najbliższego sąsiedztwa jest stała, ponieważ algorytm wykonuje dokładnie taką samą liczbę operacji dominujących, niezależnie od rozmiaru danych wejściowych.

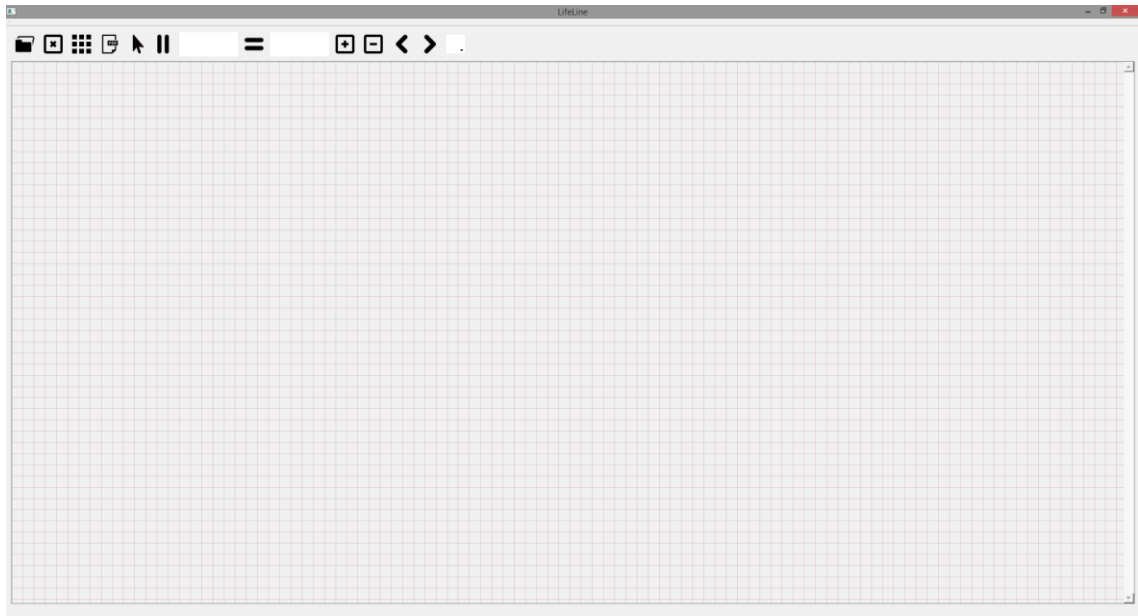
$$O(1)$$

3.4 Interfejs graficzny użytkownika

3.4.1 Główne okno aplikacji

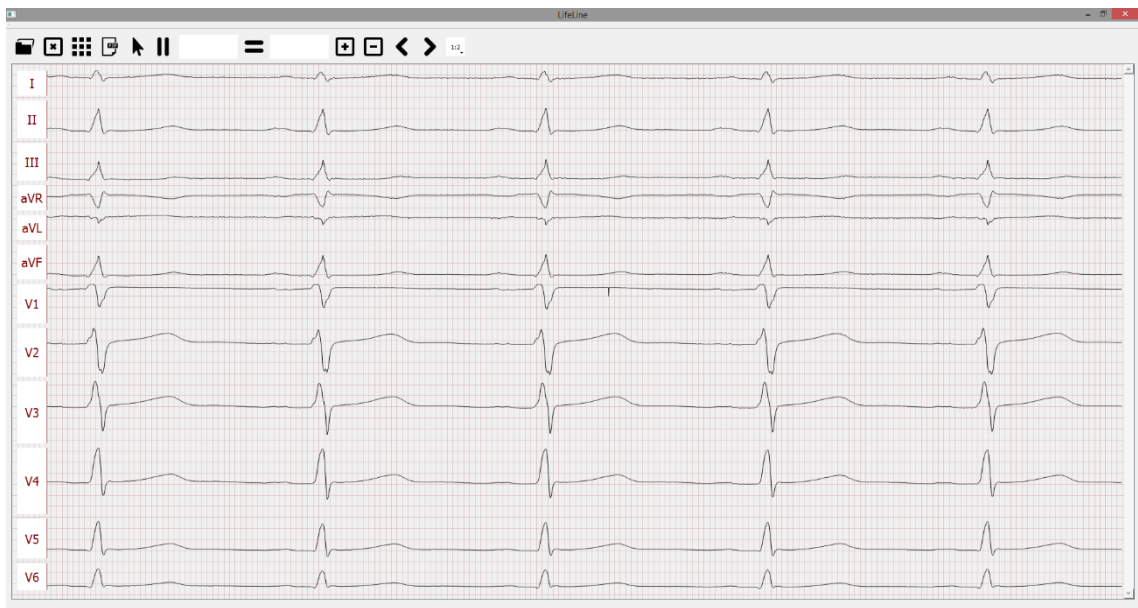
3.4.1.1 Elementy składowe głównego okna

Główne okno aplikacji jest widoczne na Rysunku 1.



Rysunek 1. Główne okno aplikacji.

Na Rysunku 1 widocznym powyżej możemy wyróżnić dwie przestrzenie, na które zostało podzielone okno : pasek narzędzi oraz arkusz papieru milimetrowego. Pasek narzędzi zawiera całą funkcjonalność udostępnioną przez tę aplikację, natomiast na arkuszu papieru milimetrowego wyświetlane są wykresy pomiarów dokonanych podczas badania elektrokardiograficznego, patrz Rysunek 2.



Rysunek 2. Główne okno aplikacji – przykładowe dane

3.4.1.2 Opis elementów graficznych

Wszystkie ikony graficzne zostały zapożyczone ze strony <http://www.flaticon.com/> [ref. 4], ponieważ nie były one częścią zadanego problemu i mają jedynie wpływ na odbiór wizualny aplikacji.

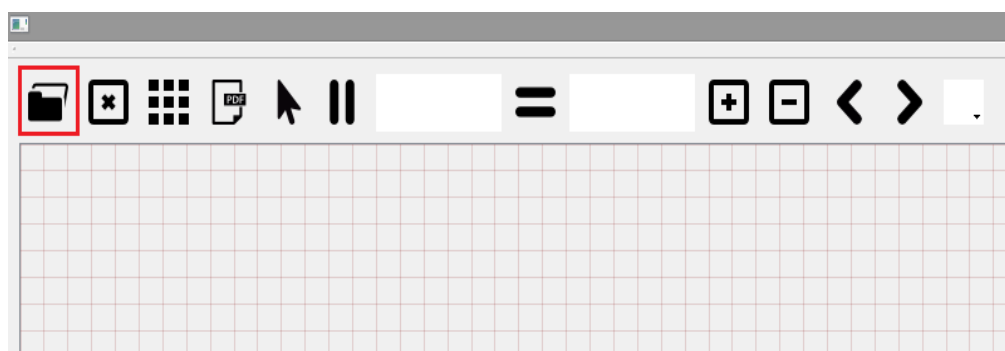
3.4.1.2.1 Przycisk **Wczytaj**

Przycisk Wczytaj służy do wczytywania danych z pliku. W programie został oznaczony ikoną widoczną na Rysunku 3. Jest to pierwszy widoczny element paska narzędzi głównego okna aplikacji.



Rysunek 3. Ikona przycisku Wczytaj.[ref. 4]

Rysunek 4 przedstawia umieszczenie ikony przycisku Wczytaj na pasku narzędzi. Został oznaczony czerwoną ramką.



Rysunek 4. Lokalizacja przycisku Wczytaj na pasku narzędzi.

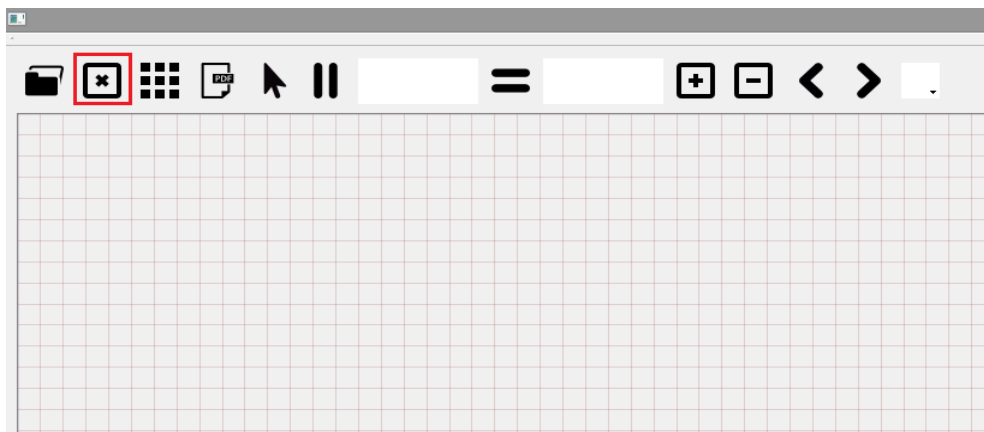
3.4.1.2.2 Przycisk **Zamknij plik**

Przycisk Zamknij plik służy do zamykania wczytanych danych. Czyści arkusz papieru milimetrowego ze wszystkich wykresów, pozwalając na wczytanie kolejnego pliku. Czynności wykonywane przez ten przycisk wykonywane są również podczas wczytywania pliku, więc nie ma niebezpieczeństwa utraty danych czy niepożądanych czynności. Przycisk został oznaczony ikoną widoczną na Rysunku 5.



Rysunek 5. Ikona przycisku Zamknij plik. [ref. 4]

Rysunek 6 przedstawia umieszczenie ikony przycisku Zamknij plik na pasku narzędzi. Został oznaczony czerwoną ramką.



Rysunek 6. Lokalizacja przycisku Zamknij plik na pasku narzędzi

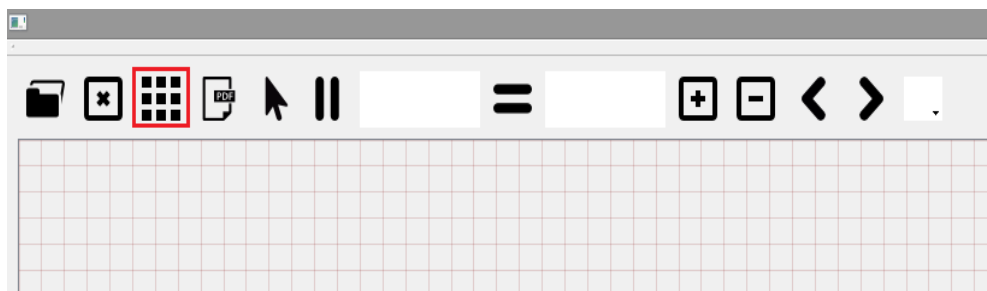
3.4.1.2.3 Przycisk **Wyłącz tło**

Przycisk Wyłącz tło służy do wyłączania/włączania wyświetlania papieru milimetrowego. Wyłączenie tła może być zalecane dla wolniejszych komputerów, w celu uzyskania lepszej wydajności uruchomionej aplikacji. Przycisk został oznaczony ikoną widoczną na Rysunku 7.



Rysunek 7. Ikona przycisk Wyłącz tło. [ref. 4]

Na Rysunku 8 widoczne jest umieszczenie przycisku na pasku narzędzi głównego okna aplikacji, oznaczone czerwoną ramką.



Rysunek 8. Lokalizacja przycisku Wyłącz tło na pasku narzędzi.

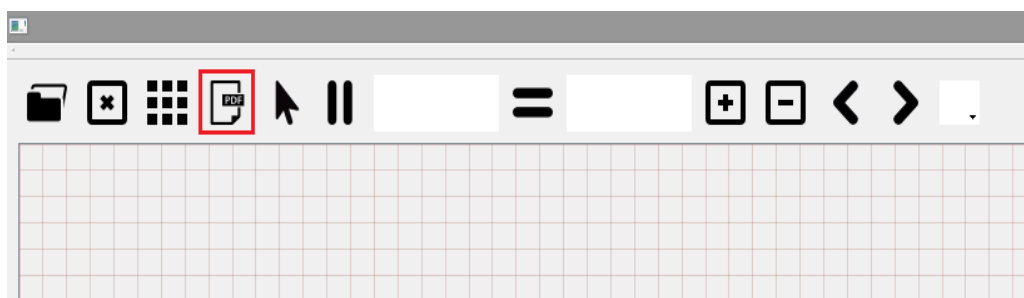
3.4.1.2.4 Przycisk **Utwórz plik *.pdf**

Naciśnięcie przycisku Utwórz plik *.pdf powoduje otwarcie nowego okna, w którym możliwe jest dostosowanie zawartości pliku. Okno tworzenia pliku zostało opisane w dalszej części rozdziału. Przycisk został oznaczony ikoną widoczną na Rysunku 9.



Rysunek 9. Ikona przycisku Utwórz plik *.pdf. [ref. 4]

Rysunek 10 przedstawia umieszczenie przycisku na pasku narzędzi. Przycisk został oznaczony czerwoną ramką.



Rysunek 10. Lokalizacja przycisku Utwórz plik *.pdf na pasku narzędzi.

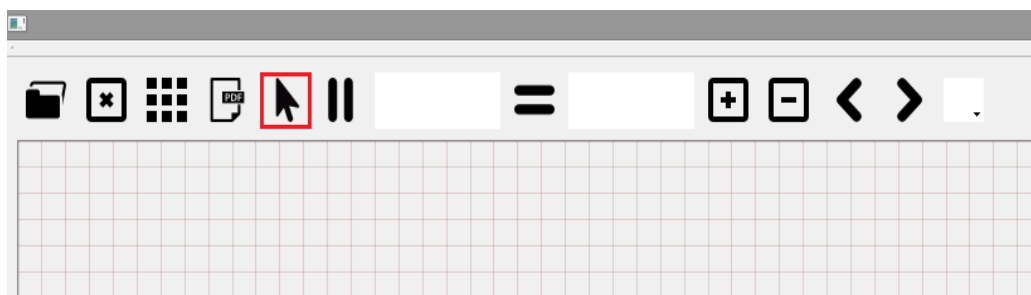
3.4.2.1.5 Przycisk **Kursor**

Przycisk Kursor został oznaczony ikoną widoczną na Rysunku 11. Przycisk służy do ustawiania klasycznego kursora i wyłączania możliwości pomiaru.



Rysunek 11. Ikona przycisku Kursor. [ref. 4]

Umieszczenie przycisku zostało zobrazowane na Rysunku 12.



Rysunek 12. Lokalizacja przycisku Kursor na pasku narzędzi.

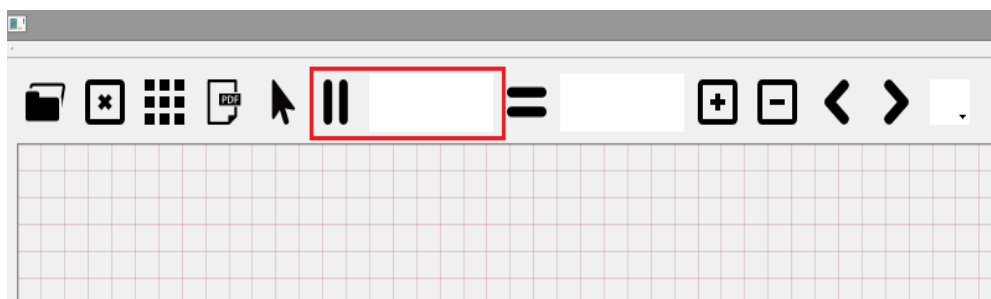
3.4.1.2.6 Przycisk oraz etykieta **Pomiar Wertykalny**

Przycisk Pomiar Wertykalny służy do uruchamiania narzędzia, dzięki któremu możliwe jest mierzenie odległości pomiędzy dwoma liniami pionowymi. Etykieta Pomiar Wertykalny wyświetla wyniki pomiaru. Ikona przycisku widoczna jest na Rysunku 13.



Rysunek 13. Ikona przycisku Pomiar Wertykalny. [ref. 4]

Etykieta Pomiar Wertykalny widoczna jest razem w przyciskiem na Rysunku 14.



Rysunek 14. Lokalizacja przycisku i etykiety Pomiar Wertykalny na pasku narzędzi.

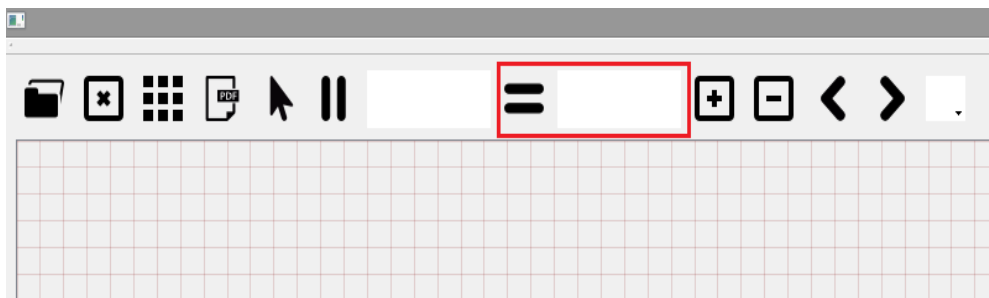
3.4.1.2.7 Przycisk oraz etykieta **Pomiar Horyzontalny**

Przycisk Pomiar Horyzontalny służy do uruchamiania narzędzia, które mierzy odległość pomiędzy dwoma poziomymi liniami. Etykieta Pomiar Horyzontalny wyświetla wyniki pomiaru. Ikona przycisku widoczna jest na Rysunku 15.



Rysunek 15. Ikona przycisku Pomiar Horyzontalny. [ref. 4]

Przycisk oraz etykieta Pomiar Horyzontalny zostały oznaczone czerwoną ramką na Rysunku 16.



Rysunek 16. Lokalizacja przycisku i etykiety Pomiar Horyzontalny na pasku narzędzi.

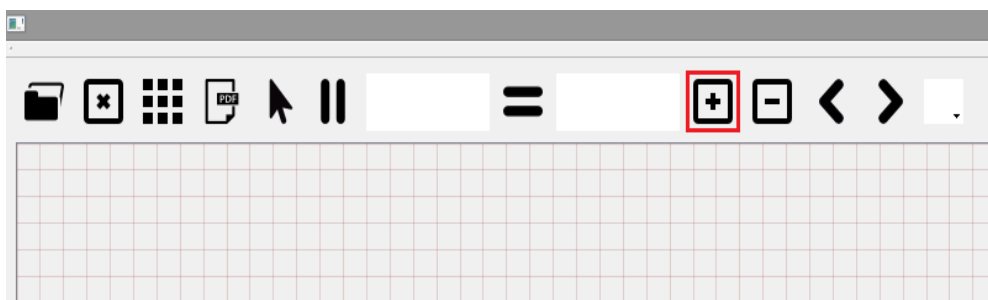
3.4.1.2.8 Przycisk **Powiększ**

Przycisk Powiększ służy do powiększania (przybliżania) wykresów widocznych w obszarze papieru milimetrowego. Ikona przycisku została pokazana na Rysunku 17.



Rysunek 17. Ikona przycisku Powiększ. [ref. 4]

Przycisk Powiększ jest widoczny na Rysunku 18. Został oznaczony czerwoną ramką.



Rysunek 18. Lokalizacja przycisku Powiększ na pasku narzędzi.

3.4.1.2.9 Przycisk **Pomniejsz**

Przycisk Pomniejsz służy do pomniejszania (oddalania) wykresów widocznych w obszarze papieru milimetrowego. Ikona przycisku została pokazana na Rysunku 19.



Rysunek 19. Ikona przycisku Pomniejsz. [ref. 4]

Lokalizacja przycisku na pasku narzędzi jest widoczna na Rysunku 20. Oznaczenie przycisku zostało wykonane przy pomocy czerwonej ramki.



Rysunek 20. Lokalizacja przycisku Pomniejsz na pasku narzędzi.

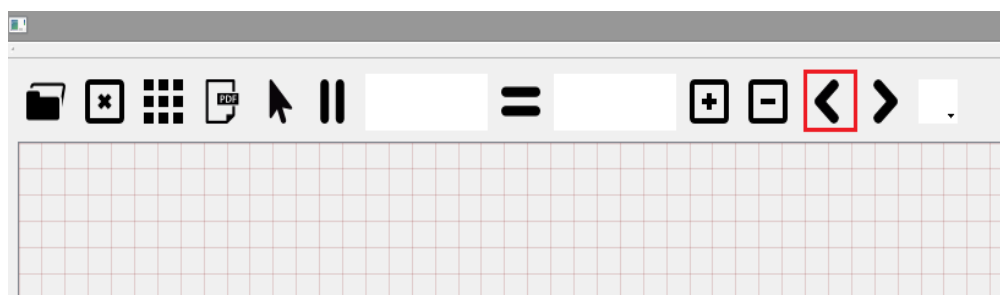
3.4.1.2.10 Przycisk **Lewo**

Przycisk Lewo służy do przesuwania wykresów w lewo i odrysowania części wykresu znajdującej się z lewej strony aktualnie rozpatrywanego obszaru. Rysunek 21 ukazuje ikonę przycisku Lewo.



Rysunek 21. Ikona przycisku Lewo. [ref. 4]

Lokalizacja przycisku Lewo na pasku narzędzi jest widoczna na Rysunku 22. Zaznaczony czerwoną ramką obszar obejmuje przycisk Lewo.



Rysunek 22. Lokalizacja przycisku Lewo na pasku narzędzi.

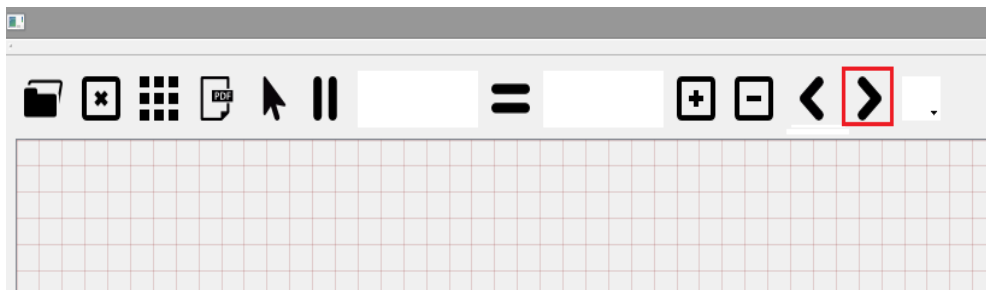
3.4.1.2.11 Przycisk **Prawo**

Przycisk Prawo służy do przesuwania wykresów w prawo i odrysowania części wykresu znajdującej się z prawej strony aktualnie rozpatrywanego obszaru. Rysunek 23 przedstawia ikonę przycisku Prawo.



Rysunek 23. Ikona przycisku Prawo. [ref. 4]

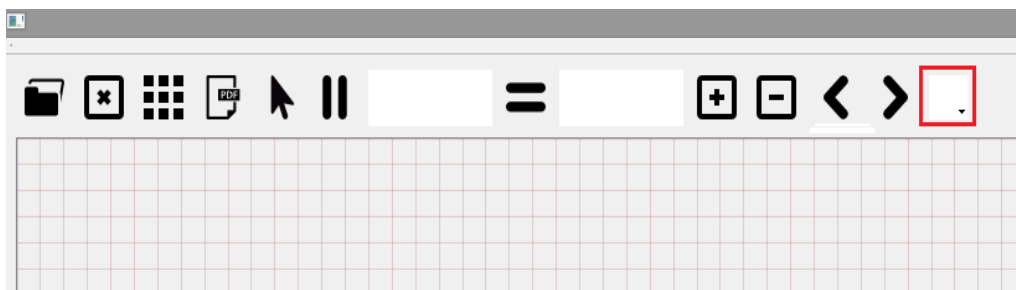
Lokalizacja przycisku Prawo została pokazana na Rysunku 24.



Rysunek 24. Lokalizacja przycisku Prawo na pasku narzędzi.

3.4.1.2.12 Rozwijane menu **Skala**

Menu Skala umożliwia wybranie stosunku wartości odległości pomiędzy punktami, innymi słowy, zmniejsza bądź zwiększa skalę szerokości wykresów i papieru milimetrowego. Rysunek 25 pokazuje umieszczenie rozwijanego menu Skala na pasku narzędzi, które zostało oznaczone czerwoną ramką.

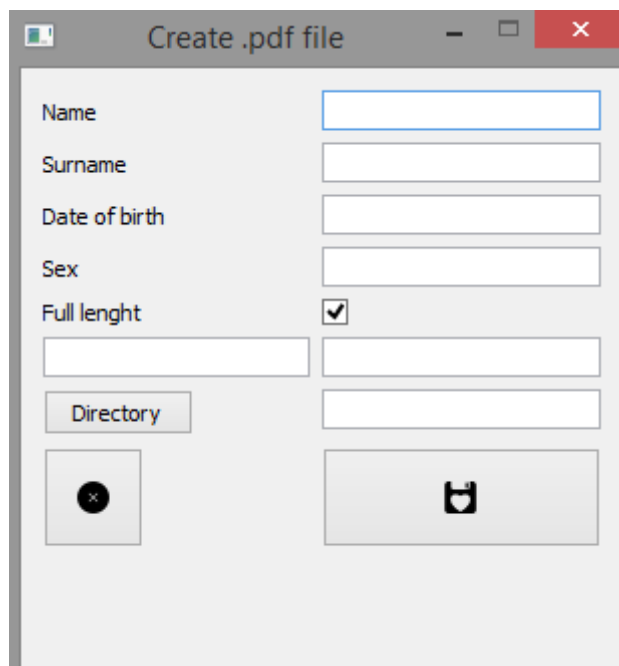


Rysunek 25. Lokalizacja menu Skala na pasku narzędzi.

3.4.2 Okno generowania pliku *.pdf

3.4.2.1 Wygląd okna

Rysunek 26 przedstawia wygląd okna do generowania pliku *.pdf. Ukazuje się ono po naciśnięciu przycisku Utwórz plik *.pdf, patrz 6.114.



The image shows a window titled "Create .pdf file". It contains the following elements:

- Input field for "Name"
- Input field for "Surname"
- Input field for "Date of birth"
- Input field for "Sex"
- Input field for "Full lenght" with a checked checkbox
- Two empty input fields below the "Full lenght" section
- A button labeled "Directory"
- A close button (X icon)
- A save button (floppy disk icon)

Rysunek 26. Okno generowania pliku *.pdf.

3.4.2.2 Opis elementów graficznych

3.4.2.2.1 Linia edycyjna **Imię**

Linia edycyjna Imię umożliwia osobie obsługującej program na podanie imienia pacjenta, dzięki czemu w pliku *.pdf będzie widoczne imię badanej osoby. Program korzysta również z danych wpisanych w to miejsce, tworząc nazwę pliku. Linia edycyjna Imię jest widoczna na Rysunku 27.

Create .pdf file

Name

Surname

Date of birth

Sex

Full lenght ☒

Directory

Rysunek 27. Linia edycyjna Imię.

3.4.2.2.2 Linia edycyjna **Nazwisko**

Linia edycyjna Nazwisko umożliwia osobie obsługującej program na podanie nazwiska pacjenta, dzięki czemu wygenerowany plik *.pdf będzie zawierał nazwisko badanej osoby. Program korzysta z danych wpisanych w to pole, tworząc nazwę pliku. Linia edycyjna Nazwisko jest widoczna na Rysunku 28.

Create .pdf file

Name

Surname

Date of birth

Sex

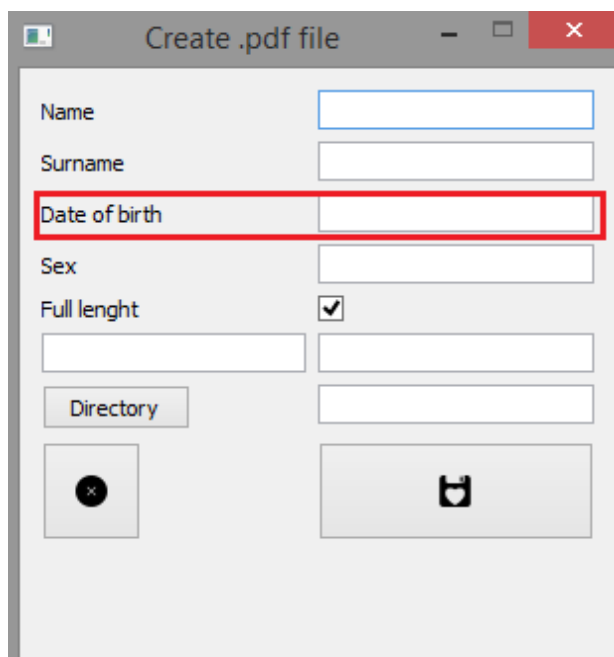
Full lenght ☒

Directory

Rysunek 28. Linia edycyjna Nazwisko.

3.4.2.2.3 Linia edycyjna **Data urodzenia**

Linia edycyjna data urodzenia umożliwia osobie posługującej się programem na podanie daty urodzenia pacjenta, dzięki czemu w wygenerowanym pliku *.pdf będzie widoczna jego data urodzenia. Program korzysta z danych wpisanych w to pole, tworząc nazwę pliku. Linia edycyjna Data urodzenia jest widoczna na Rysunku 29.

The image shows a software window titled "Create .pdf file". It contains several input fields: "Name", "Surname", "Date of birth", "Sex", and "Full lenght". The "Date of birth" field is highlighted with a red rectangular border. Below "Full lenght" is a checkbox that is checked. At the bottom, there is a "Directory" button, a close button (X), and a save button (floppy disk icon).

Rysunek 29. Linia edycyjna Data urodzenia.

3.4.2.2.4 Linia edycyjna **Płeć**

Linia edycyjna Płeć umożliwia użytkownikowi programu na określenie płci badanego pacjenta. Płeć pacjenta będzie widoczna w wygenerowanym pliku *.pdf. Linia edycyjna została oznaczona na Rysunku 30.

The screenshot shows a window titled "Create .pdf file". It contains several input fields: "Name", "Surname", "Date of birth", "Sex", and "Full lenght". The "Sex" field is highlighted with a red rectangle. Below the "Full lenght" label, there is a checked checkbox and two empty input fields. At the bottom, there is a "Directory" button, a close button (X), and a print button (P).

Rysunek 30. Linia edycyjna Płeć.

3.4.2.2.5 Pole wyboru **Długość** oraz linie edycyjne **Czas**

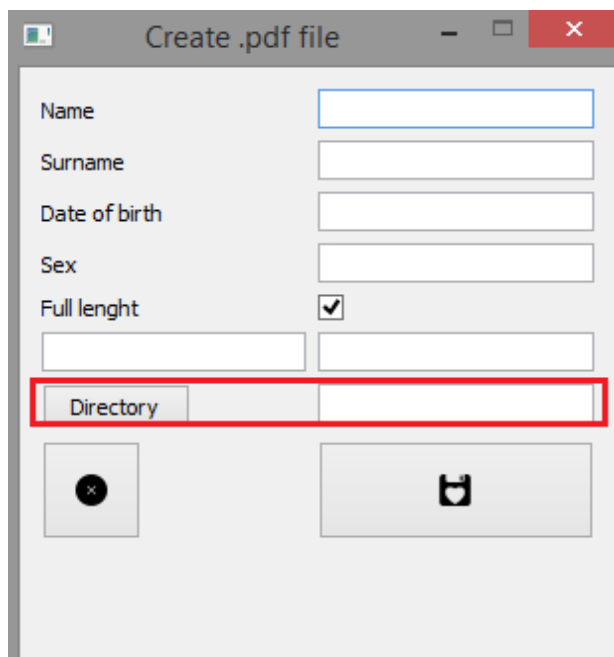
Pole wyboru **Długość** umożliwia zaznaczenie, czy wydruk do pliku ma zawierać pełne dane czy nie. Odznaczenie pola wyboru umożliwia wpisanie w dwie niżej położone linie edycyjne wartości graniczne w sekundach. Rysunek 31 przedstawia umieszczenie pola wyboru Długość oraz linii edycyjnych Czas w oknie.

The screenshot shows the same "Create .pdf file" window. In this view, the "Full lenght" checkbox is checked, and the two input fields below it are highlighted with a red rectangle. The other fields and buttons remain the same as in the previous screenshot.

Rysunek 31. Pole wyboru Długość oraz linie edycyjne Czas

3.4.2.2.6 Przycisk **Katalog**

Przycisk Katalog umożliwia wybranie katalogu, w którym będzie zapisany plik *.pdf. Po wybraniu ścieżki, w widocznej obok linii edycyjnej pojawi się wybrana ścieżka. Rysunek 32 przedstawia umieszczenie przycisku Katalog w oknie.



Rysunek 32. Przycisk Katalog.

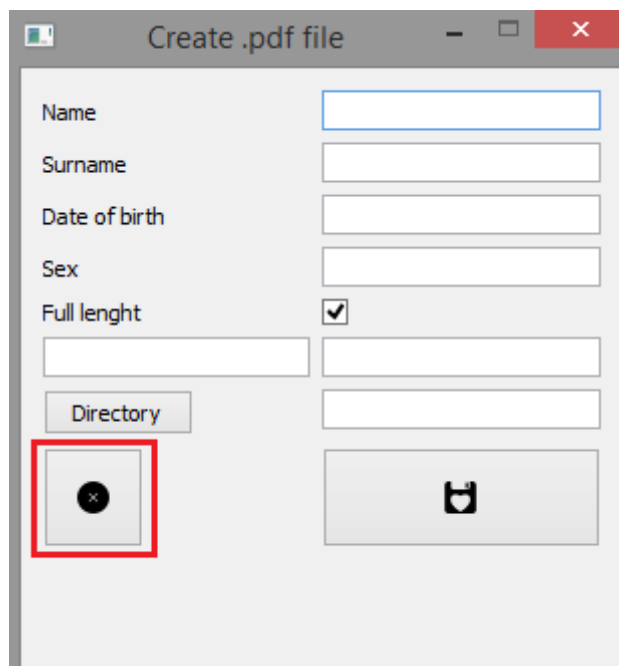
3.4.2.2.7 Przycisk **Zamknij**

Przycisk Zamknij umożliwia zamknięcie okna, jeżeli nie jesteśmy zainteresowani tworzeniem pliku *.pdf. Ikona użyta do przycisku jest widoczna na Rysunku 33.



Rysunek 33. Ikona przycisku Zamknij. [ref. 4]

Lokalizacja przycisku Zamknij jest widoczna na Rysunku 34.



Rysunek 34. Przycisk Katalog.

3.4.2.2.8 Przycisk **Zapisz plik**

Przycisk Zapisz plik umożliwia zapis pliku *.pdf. Po kliknięciu plik zapisywany jest w podanej ścieżce, tworząc plik o nazwie ImieNazwisko_DataUrodzenia.pdf. Ikona przycisku jest widoczna na Rysunku 34.



Rysunek 35. Ikona przycisku Zapisz. [ref. 4]

Lokalizacja przycisku Zapisz plik widoczna jest na Rysunku 36.

Create .pdf file

Name

Surname

Date of birth

Sex

Full lenght ☒

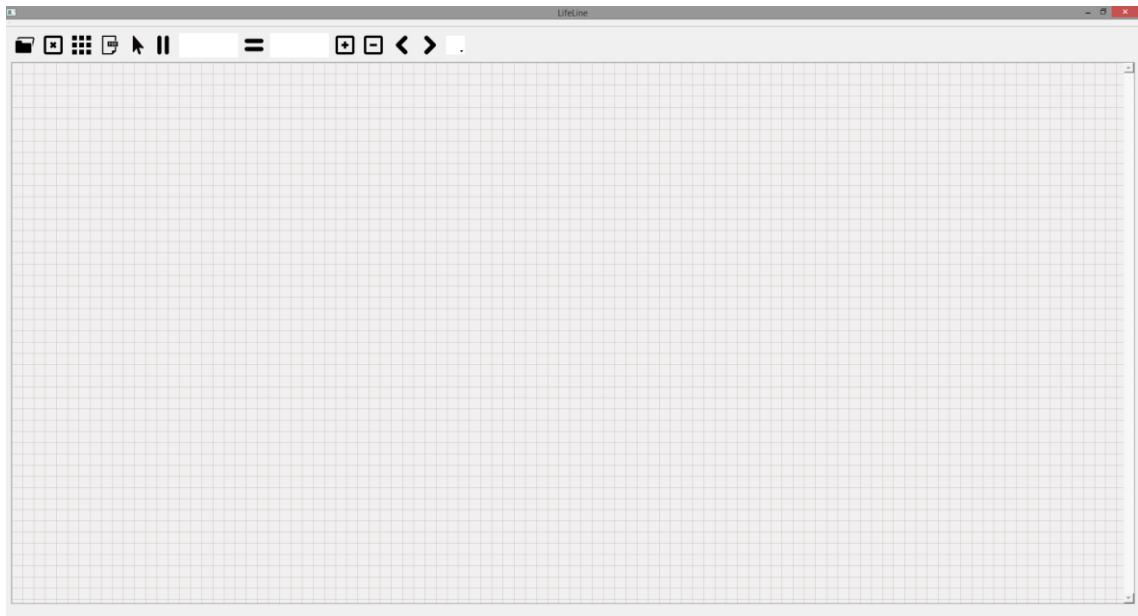
Directory

Rysunek 36. Przycisk Zapisz plik.

4. Opis korzystania z programu

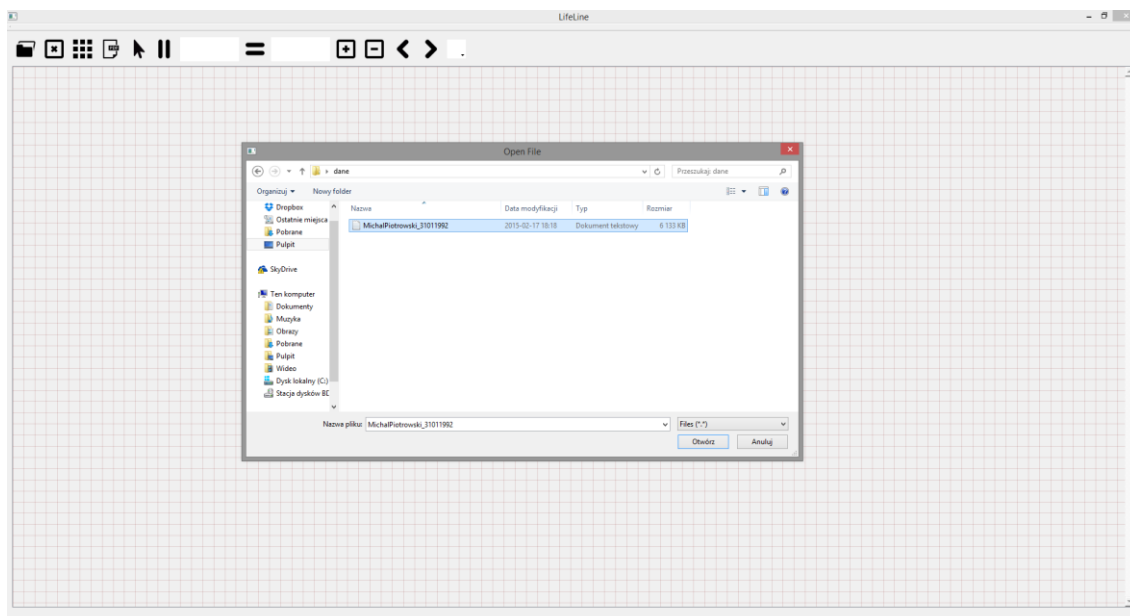
4.1 Wczytywanie pliku z danymi

Po uruchomieniu programu, widoczne jest główne okno aplikacji, patrz Rysunek 37.

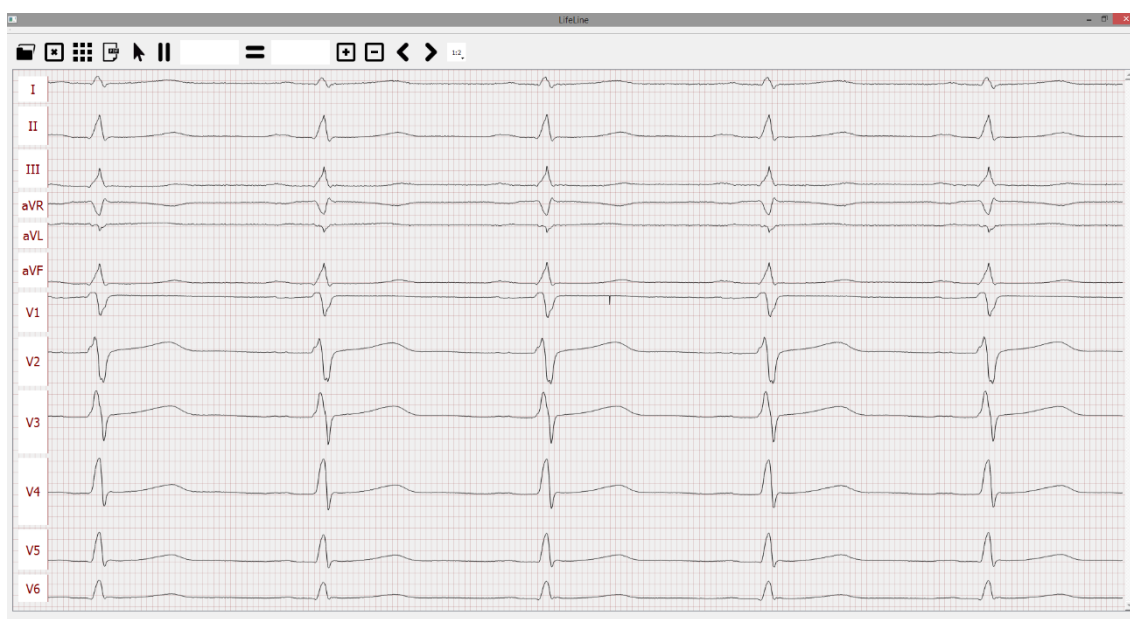


Rysunek 37. Główne okno aplikacji.

Aby rozpocząć pracę, pierwszym krokiem jest wczytanie pliku z danymi pacjenta. W tym celu należy kliknąć przycisk **Wczytaj**, patrz 6.111. Po naciśnięciu przycisku ukaze nam się okno systemowe, w którym należy wyszukać interesujący plik z danymi. Rysunek 38 przedstawia przykład takiego rozwiązania.



Rysunek 38. Wczytywanie pliku z danymi.
Po wczytaniu pliku, widoczne są przebiegi elektrokardiograficzne, patrz Rysunek 39.



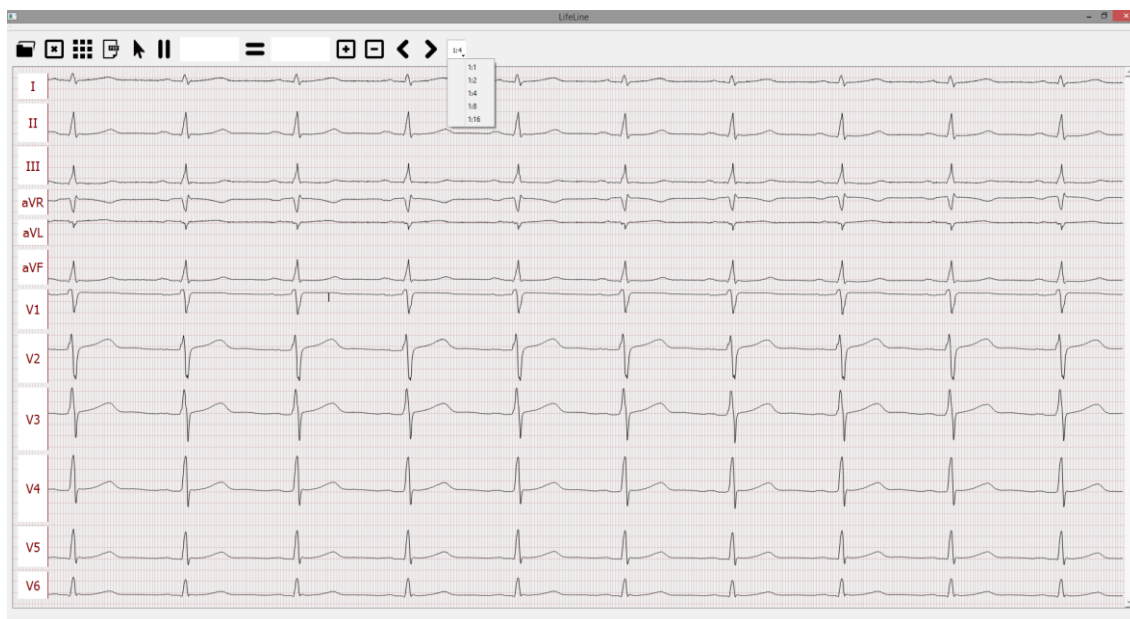
Rysunek 39. Główne okno aplikacji – przykładowe dane

4.2 Korzystanie z narzędzi

4.2.1 Zmiana skali

Wczytane dane mogą nie być przejrzyste. Wpływ na taki stan rzeczy mają różne czynniki, takie jak częstotliwość pomiaru, dokładność urządzenia pomiarowego, czas badania itp. Aby umożliwić jak najlepsze dopasowanie

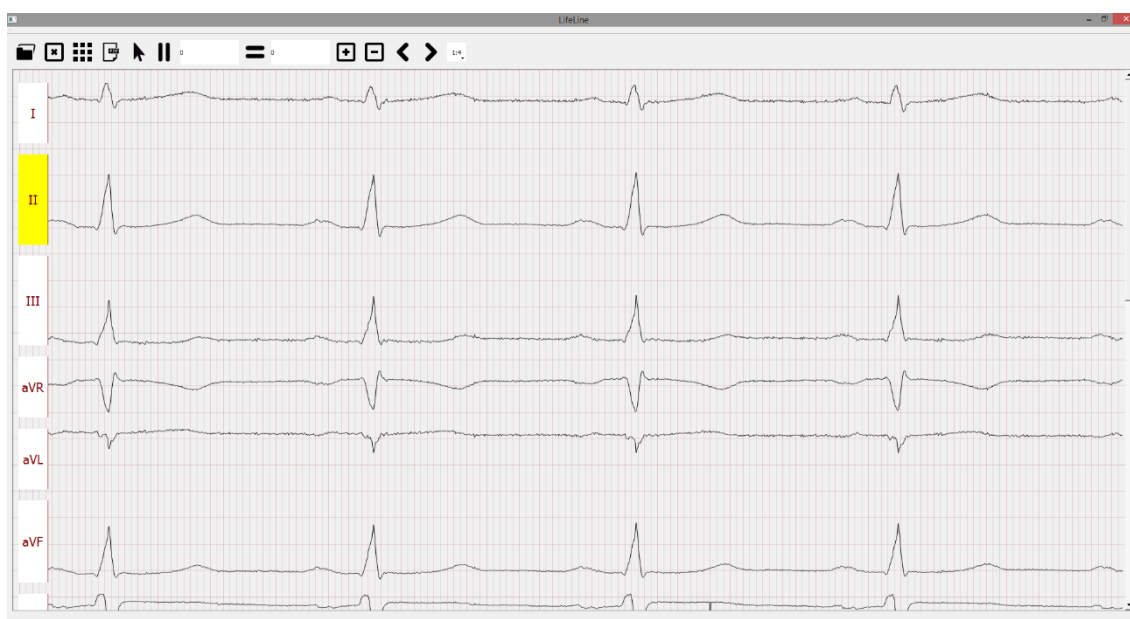
wykresu do potrzeb, należy kliknąć na przycisk menu Skala, patrz 6.122. Rysunek 40 przedstawia dane w skali 1:4.



Rysunek 40. Przykładowe dane w skali 1:4

4.2.2 Powiększanie

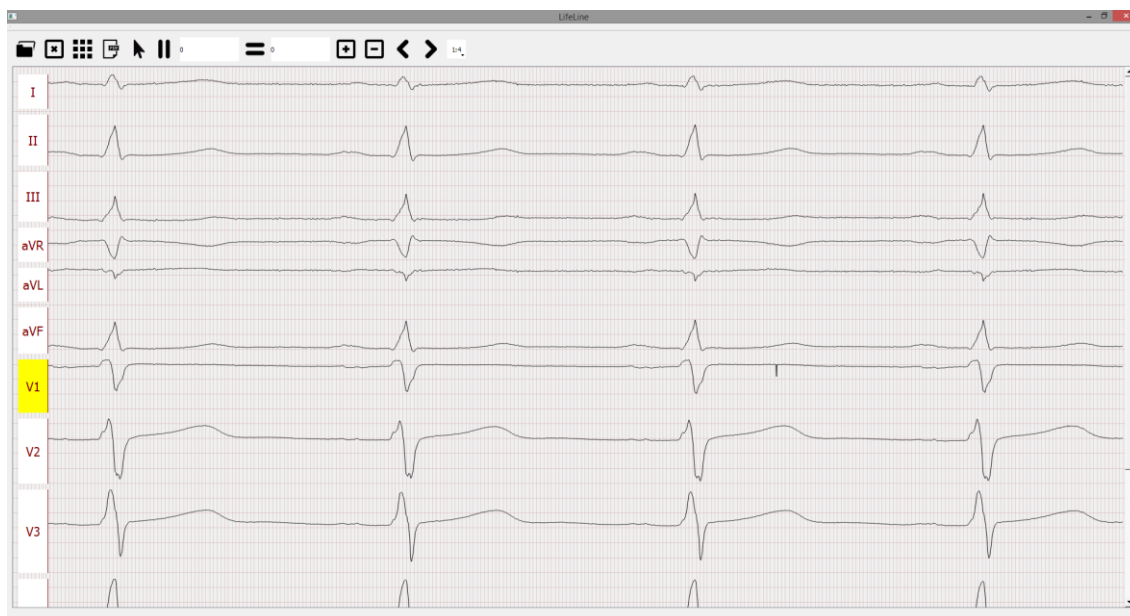
Aby powiększyć wykresy, należy użyć przycisku Powiększ, patrz 6.118. Rysunek 41 pokazuje pewne powiększenie wczytanych danych (w skali 1:4).



Rysunek 41. Powiększenie danych.

4.2.3 Pomniejszanie

Aby następnie pomniejszyć wykresy, należy użyć przycisku Pomniejsz, patrz 6.119. Rysunek 42 pokazuje pewne pomniejszenie wczytanych danych, w dalszym ciągu w stosunku 1:4.



Rysunek 42. Pomniejszenie danych

4.2.4 Przesuń w lewo

Aby przesunąć widoczne wykresy w lewo, należy użyć przycisku Lewo, patrz 6.120.

4.2.5 Przesuń w prawo

Aby przesunąć widoczne wykresy w prawo, należy użyć przycisku Prawo, patrz 6.121.

4.2.6 Pomiar horyzontalny wykresu

Dzięki temu narzędziu, użytkownik jest w stanie zmierzyć dowolną szerokość pomiędzy dwoma punktami. Aby użyć narzędzia pomiar horyzontalny, patrz 6.116. Rysunek 43 przedstawia użycie lewego przycisku pomiaru horyzontalnego.



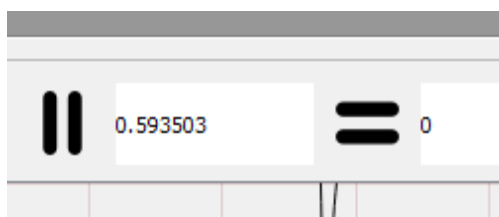
Rysunek 43. Użycie lewego przycisku myszy – pomiar horizontalny.

Do pomiaru długości potrzebny jest jeszcze druga linia, którą uruchamiamy kliknięciem prawym przyciskiem myszy na wybranym wykresie. Widoczne zaznaczenie widoczne jest na Rysunku 43.



Rysunek 44. Użycie prawego przycisku myszy – pomiar horizontalny.

Wartość zmierzonego odcinka widoczna jest na pasku narzędzi, patrz Rysunek 45.



Rysunek 45. Wartość zmierzonego odcinka

4.2.7 Pomiar wertykalny wykresu

Dzięki pomiarowi wertykalnemu, użytkownik ma możliwość pomiaru wysokości pomiędzy dwoma punktami przebiegu. Przykładowy pomiar przedstawiony został na Rysunku 46.



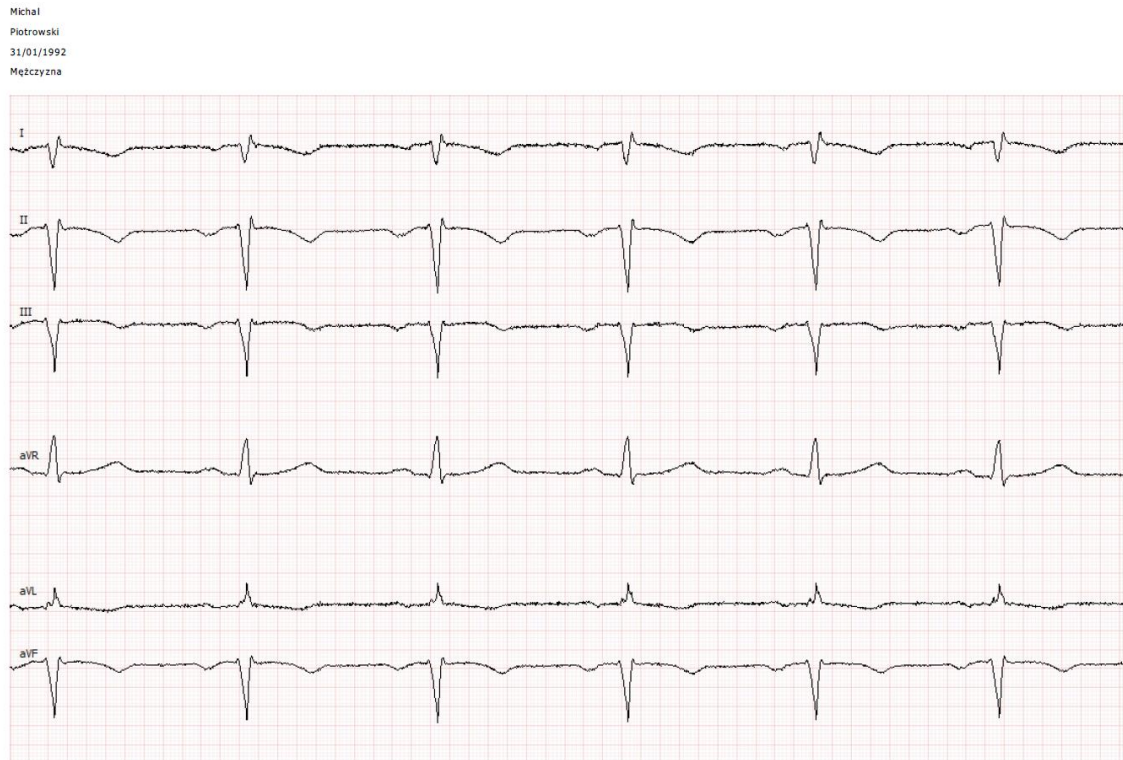
Rysunek 46. Przykładowy pomiar wertykalny.

4.3 Tworzenie pliku *.pdf

Aby utworzyć plik *.pdf należy użyć przycisku tworzenia pliku, patrz 6.114. Po naciśnięciu, ukazuje się okno służące do dostosowania pliku *.pdf. Rysunek 47 przedstawia przykładowo wypełnione dane okna tworzenia pliku *.pdf.

Rysunek 47. Przykładowe okno z danymi pacjenta

Wygenerowany plik zapisze się w ścieżce widocznej w polu Katalog. Na Rysunku 48 widoczny jest wygenerowany plik *.pdf.



Rysunek 48. Przykładowa strona z wygenerowanego pliku *.pdf

5. Podsumowanie

Celem niniejszej pracy dyplomowej było opracowanie graficznej aplikacji umożliwiającej wyświetlanie na ekranie krzywych kardiograficznych rejestrowanych podczas badania EKG oraz stworzenie narzędzi ułatwiających pomiar załamków. Motywacją do podjęcia tematu była współpraca Uniwersytetu Łódzkiego oraz Uniwersytetu Medycznego w Łodzi w celu opracowania tańszej i alternatywnej wersji badania elektrokardiograficznego.

W ramach niniejszej pracy dyplomowej zostały opracowane:

- aplikacja graficzna wyświetlająca wykresy elektrokardiograficzne
- narzędzia pomiarowe umożliwiające pomiar wyświetlanych przebiegów w pionie i w poziomie

Autor udostępnił również możliwość zapisu danych w pliku *.pdf, aby można było analizować dane badania w tradycyjnej, papierowej formie.

W aplikacji wykorzystano algorytm najbliższego sąsiedztwa, dzięki któremu autor mógł w łatwy i szybki sposób dokonywać skalowania obrazu.

Testy aplikacji pokazały, iż zarówno wymagania funkcjonalne jak i нефункционалне zostały spełnione. Aplikacja działa płynnie, bez widocznych dużych opóźnień w wyświetlaniu obrazu.

Aplikacja została utworzona z wykorzystaniem biblioteki Qt i napisana w języku C++, dzięki czemu jest aplikacją, którą można uruchomić na wielu systemach operacyjnych. Zapewniło to wieloplatformowość, która była jednym z celów niniejszej pracy dyplomowej.

Bibliografia

1. Tomasz Tomasik, Józef Kocemba, Anna Skalska, Adam Windak, Jolanta Kulczycka-Życzkowska: Elektrokardiografia dla lekarza praktyka.
2. Mark Summerfield: Biblioteki Qt. Zaawansowane programowanie przy użyciu Qt.
3. <http://fbt.cz/en/>
4. <http://www.flaticon.com/>