Filtracja + UI

Spis treści

Wstęp 1

Opis metod 1

Przeprowadzone testy 2

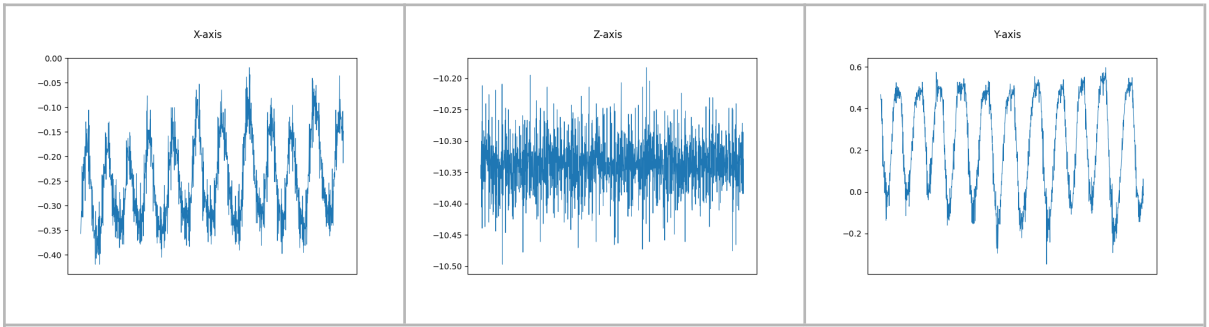
Funkcjonalność 3

Wnioski 5

Wstęp

Ze względu na to, że w pliku tekstowym zapisują się dane surowe odczytane wprost z akcelerometru, wymagana jest filtracja dla wygładzenia danych, usunięcia niepotrzebnych szumów i ułatwienia wyciągnięcia przydatnych w analizie oddechu parametrów.

Rysunek 1 przedstawia surowe sygnały wydobyty wprost z akcelerometru(różne osi), który zawiera dwie główne składowe - sygnał pochodzący z bicia serca i sygnał poruszania się klatki piersiowej podczas oddechu.



Rys.1

Oprócz tego, postanowiliśmy, że skoro nie wiemy na jakim sprzęcie zostaną wyświetlone dane w warunkach klinicznych, potrzebna jest aplikacja UI, pozwalająca na korzystaniu z oprogramowania nawet w konsoli.

Kiedy projektowaliśmy to urządzanie i kiedy robiliśmy oprogramowanie do niego, zakładaliśmy, że badania będą prowadzić się w warunkach kiedy pacjent się nie porusza(czyli leży, stoi milcząc, śpi). Jak i w ECG, poruszanie się pacjenta wprowadza zbyt duże szumy i mocno zakłóca główny sygnał.

Opracowana filtracja pozwala na wyciągnięcie takich parametrów, jak liczba oddechów na minutę, wyświetlenie dokładnej godziny śmierci badanego człowieka, wyświetlanie graficzne przebiegów oddechowych. Oprócz tego pozwala ona na wybranie przedziału czasu w którym chcemy przeprowadzić analizę.

Oprogramowanie zostało zaimplementowane w języki Python ze względu na kompatybilność z GUI wersją naszej aplikacji.

Opis metod

Proces filtracji zaczyna się już na poziomie oprogramowania arduino. Wprowadziliśmy tak zwany licznik, który zlicza w górę tylko w przypadku, jeśli nie “widzi” zmian wartości w chociażby jednej z osi akcelerometru. Buzzer zostaje włączony od wartości licznika równej

950. Jeśli licznik dochodzi do 1000 - stwierdzamy zgon. Dane są nadal zapisywane, ale licznik przestaje zliczać i pozostaje na stałej wartości 1000, która pozwala nam

W procesie analizy przebiegów każdego z wymiarów, doszliśmy do wniosku, że oś Z nie za bardzo nadaje się do analizy oddechowej. W przeciwieństwie do danych z osi Z, dane z osi Y i X nie były tak bardzo zakłócane biciem serca. W przeciwieństwie do osi X i Y, oś - Z mierzyła przyspieszenia spowodowane dźwiękami akustycznymi zastawek serca lepiej niż inne osie[Rys 1].

Jedna z celu naszego projektu było znalezienie średniej liczby oddechów na minutę w pewnym przedziale czasowym. Żeby to zrobić, wymyślona została metoda obliczenia liczby szczytów lokalnych otrzymanego przebiegu, a później skalowania otrzymanej liczby na liczbę oddechów na minutę za pomocą zwykłej proporcji.

Surowy sygnał, jak widziliśmy już wcześniej, jest bardzo zaszumiony. Utrudnia to znacznie prawidłowe wyznaczenie szczytów. Czyli potrzebne jest wygładzenie przebiegu. Oprócz tego, Python pozwala na znalezienie maksimów lokalnych odnośnie pewnego pozioma. Nie wiemy dokładnie jaki to jest poziom w każdym z przypadków, bo za każdym razem może on się różnić, więc otrzymane dane też powinny zostać znormalizowane.

W celu wygładzenia osi X i Y został użyty wbudowany w Python filtr Savitzky'ego-Golaya. W celi normalizacji została użyta metoda z-score. Warto zauważyć, że nawet po wygładzeniu przebiegu nadal są widoczne małe zakłócenia, ale już bardzo blisko poszukiwanego maksima lokalnego. Z tego względu wbudowana metoda find_peaks() wyznacza więcej szczytów niż tego potrzebujemy, ale później ręcznie pozbawiamy się od zbędnych we własnej metodzie find_peaks_custom(), jeśli te pliki znajdują się blisko siebie.

Zaimplementowana analiza wyświetla dokładną godzinę śmierci pacjenta, szukając wartości 1000 w liście z zapisanymi wartościami licznika. Później na podstawie indeksu pierwszej spotkanej wartości równej 1000, szuka poprawny czas w którym to zliczenie wystąpiło.

Przeprowadzone testy

Zaimplementowana analiza nie mogła powstać bez wcześniej przeprowadzonych testów. Warto podkreślenia to, że przeprowadzone testy nie są idealne i czasami zawierają w sobie odcinki spowodowane próbą odłączenia od sieci lub przedstawienie na powierzchnię, aby urządzenie się nie poruszało i arduino mogło wykryć zgon.

O to lista przeprowadzonych przez nas testów:

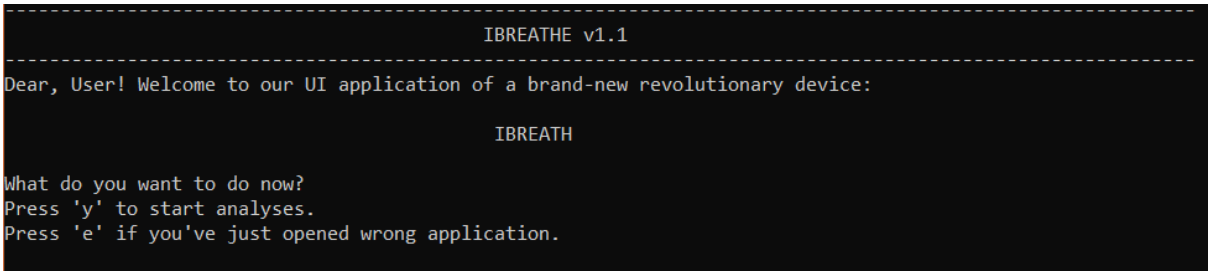
test1 - pacjent w ciągu 5 minut leży w łóżku i normalnie oddycha.
test2 - pacjent w ciągu 2 minut leży na boku w łóżku i normalnie oddycha.
test3 - pacjent w ciągu minuty stoi i rozmawia.
test4 - pacjent w ciągu 2 minut oddycha o później udaje, że umarł.

Jak i zakładaliśmy wcześniej, analiza nie nadaje się dla przypadku zbyt dużych zakłóceń, które powstały przy przeprowadzeniu testu trzeciego. Najbardziej wiarygodne wyniki uzyskujemy przy analizie testu pierwszego, bo warunki przeprowadzenia tego testu były najlepsze.

Funkcjonalność

Poniższe zrzuty ekranu i krótkie opisy tego, co się tam dzieje, przedstawiają funkcjonalność aplikacji UI.

Zaczynamy naszą przygodę od możliwości wybrania jedną z dwóch opcji:

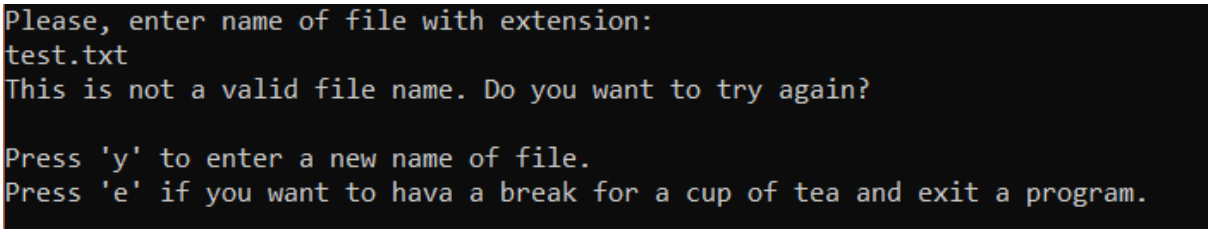


Wybierając opcję e, opuszczamy naszą aplikację

Wybierając opcję y zaczynamy dialog z nią:

Wyświetlany zostaje zawartość całego folderu z przeprowadzonymi wcześniej testami.

Od razu sprawdźmy czy wprowadzenie błędnej nazwy wyrzuci nas z programu:



Widzimy że proszą nas o powtórne wprowadzenie danych.

Teraz wprowadziliśmy dobrą nazwę pliku. Wyświetlony jest czasowy przedział, w którym przeprowadzony był test, a dalej my zostali poproszeni o wprowadzeniu czasu startowego od którego żądamy wyświetlenia analizy:

```
Please, enter name of file with extension:
test1.txt
Data was recorded in the time interval from 28.5.2022  at  18.26.5  to  28.5.2022  at  18.32.0

Enter the hour from which you want to start analysis:
18
Enter minute from which you want to start analysis:
26
Enter seconds from which you want to start analysis:
10

START TIME: 18:26:10
```

Dalej wprowadzamy godzinę zakończenia analizy:

```
Data was recorded in the time interval from 28.5.2022  at  18.26.5  to  28.5.2022  at  18.32.0

Enter stop hour:
```

Spróbujmy wprowadzić coś innego niż to o co nas proszą:

```
Enter stop hour:
dsf
Please input integer only...
```

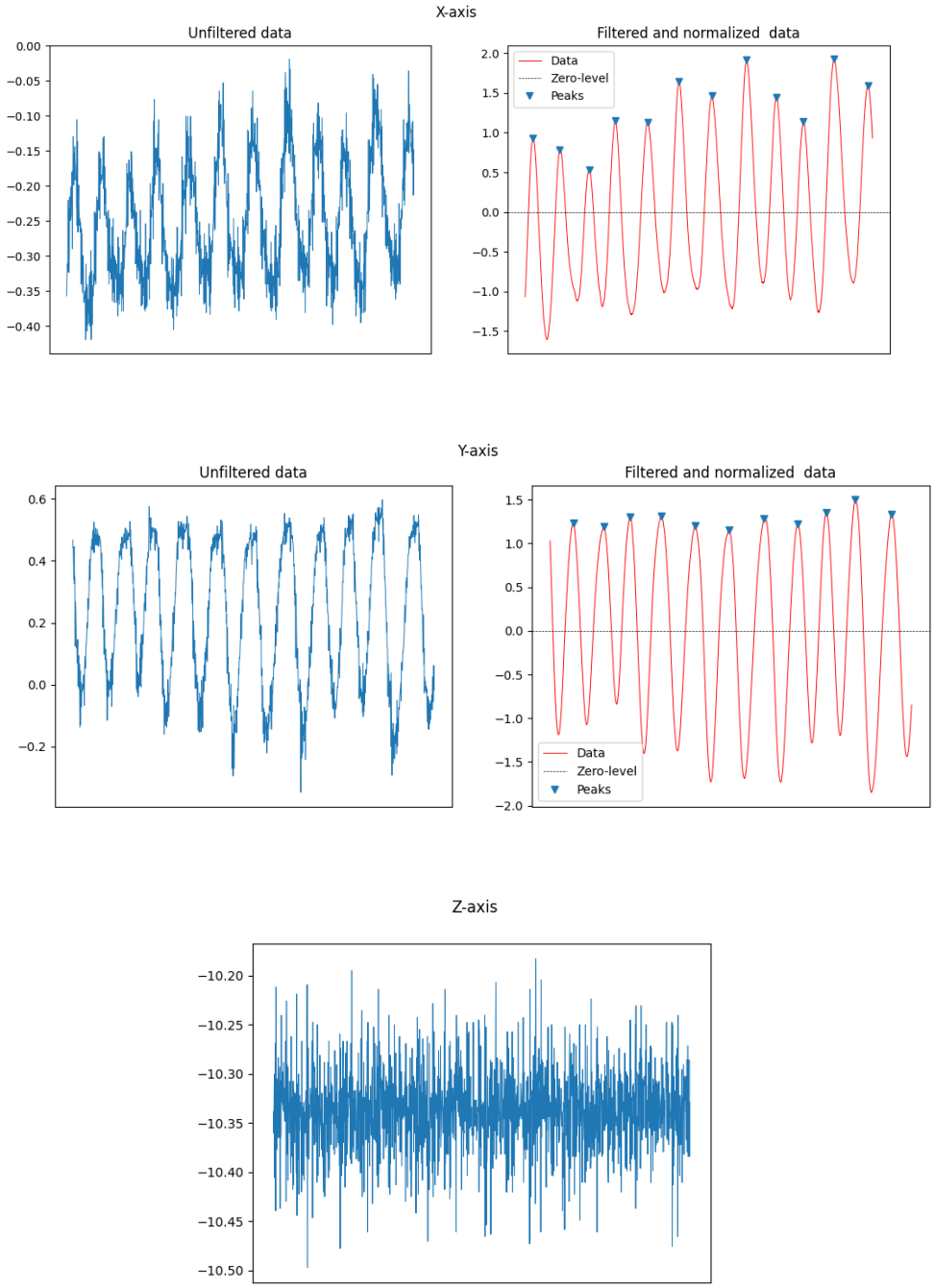
A teraz wprowadźmy czasowy przedział, który nie mieści się w czasowym przedziale przeprowadzonego badania:

```
Data you've provided is wrong. Check correct time interval. Please repeat again.

Data was recorded in the time interval from 28.5.2022  at  18.26.5  to  28.5.2022  at  18.32.0

Enter the hour from which you want to start analysis:
```

Po wprowadzeniu poprawnych danych, wyświetla się nam cała analiza:

```
ANALYSIS WAS STARTED...
START TIME: 18:26:10
STOP TIME: 18:27:20

INFORMATION:

The patient is still alive.

Czas, który zajmuje analiza:  70
Average breaths per minute from X axis: 11
Average breaths per minute from Y axis: 10
```

```
ANALYSIS WAS STARTED...
START TIME: 18:47:10
STOP TIME: 18:49:10

INFORMATION:

The patient died at 18.48.48

Czas, który zajmuje analiza:  120
Average breaths per minute from X axis: 8
Average breaths per minute from Y axis: 2
```



Wnioski

Wniosek jest taki: zaimplementowana analiza pokazuje najbardziej wiarygodne wyniki, kiedy badanie było zrobione w spokojnych warunkach a człowiek oddychał spokojnie.