Projekt 1 - Cechy sygnału audio w dziedzinie czasu

Kacper Trębacz, Jakub Knyspel

1 kwietnia 2023

1 Opis zadania

Celem projektu było stworzenie aplikacji analizującej plik .wav, obliczając cechy zapisanego w nim sygnału w dziedzinie czasu. Program miał za zadanie:

- Wyświetlać przebieg czasowy fali dźwiękowej;
- Wyświetlać wykresy chwilowych wartości parametrów dźwięku, obliczanych na poziomie ramki;
- Wykrywać fragmenty ciszy i oznaczać je na wykresie;
- Odróżniać głoski bezdźwięczne od dźwięcznych;
- Wykonywać parę innych zadań, których nie zdążyliśmy zrealizować.

2 Wykorzystane technologie

Program został napisany w języku Python, głównie ze względu na przystępność tego języka w zastosowaniach naukowych, jak i przez istnienie wielu bibliotek do najróżniejsztch naukowych zastosowań. Do utworzenia aplikacji okienkowej wykorzystana została biblioteka PyQt 5. Do wczytywania plików .wav użyta została biblioteka SciPy, do efektywnego przetwarzania danych sygnału w postaci wektorowej - NumPy, a do wyświetlania wyników w postaci wykresów - MatPlotLib. Aby przyspieszyć działanie programu, wykorzystana została biblioteka Numba.

3 Funkcjonalności

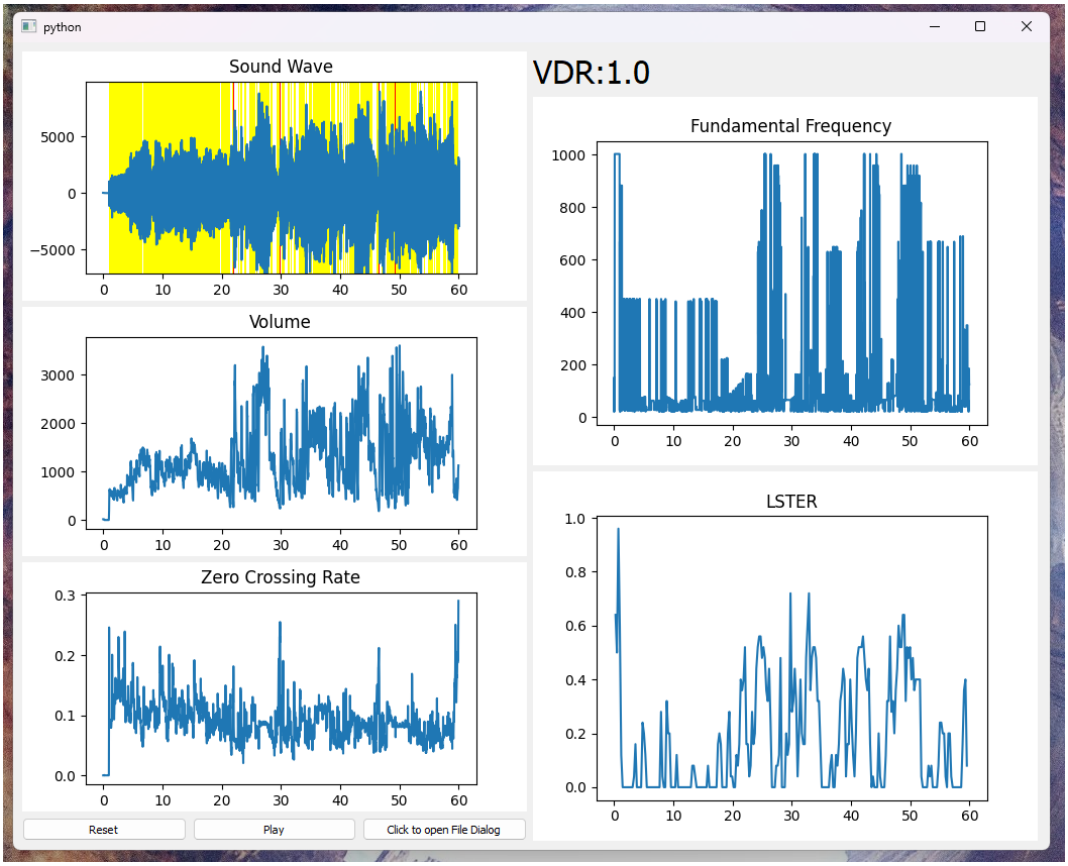
Interfejs programu został przedstawiony na zrzucie ekranu 1. Program pozwala na wczytanie plików .wav zawierających dźwięk, który ma być przeanalizowany. Aby to zrobić, należy kliknąć przycisk "Click to open File Dialog"u dołu okna. Po wczytaniu wykresy parametrów zostaną zaktualizowane. Użytkownik ma dodatkowo możliwość posłuchania wczytanego klipu - służy do tego przycisk "Play".

Przebieg czasowy fali dźwiękowej widać na wykresie w lewej kolumnie u góry (podpisanym "Sound Wave"). Pozioma oś wykresu reprezentuje czas trwania klipu (w sekundach), a pionowa - relatywne natężenie.

Na wykes nałożona zostałą informacja o fragmentach ciszy oraz głoskach bezdźwięcznych. Są one zamalowane odpowiednio na czerwono i żółto.

Zaznaczając zakres czasowy na wykresie przebiegu czasowego fali, możemy zmienić granice wyświetlanych wartości na tym, oraz wszystkich pozostałych, wykresach. Po zaznaczeniu wykresy będą prezentowały jedynie fragment w czasie, który był zaznaczony. Aby powrócić do widoku całego klipu, należy użyć przycisku "Reset".

Wykres widoczny po lewej stronie, pod wykresem przebiegu czasowego fali (podpisany "Volume"), prezentuje zależność głośności od czasu.



Rysunek 1: Interfejs programu

Kolejny wykres (po lewej stonie, na dole) przedstawia częstotliwość przecinania przez falę dźwiękową osi X (tzw. Zero Crossing Rate - ZCR). Parametr ten, razem z głośnością, jest wykorzystywany do określania, które fragmenty klipu mogą zostać uznane za ciszę.

Wykres górny po prawej stronie prezentuje wartość chwilowej częstotliwości podstawowej.

Ostatni z wykresów zawiera Low Short Time Energy Ratio - LSTER.

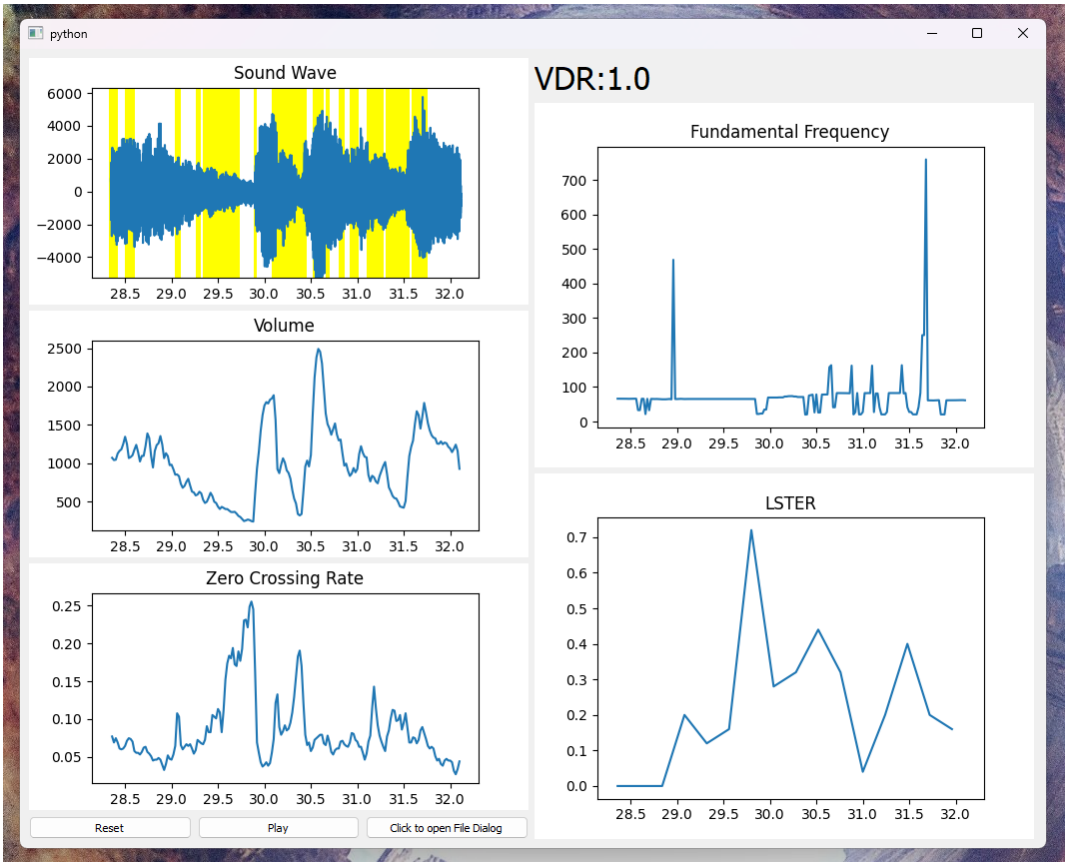
4 Algorytmy

Większość algorytmów zaimplementowanych w celu obliczania opisanych wyżej parametrów wynika wprost z ich definicji, więc dokładne ich opisywanie nie ma szczególnej wartości. Jedyny godny uwagi aspekt to konieczność użycia funkcji okna. Wykorzystana do tego została funkcja convolve z modułu NumPy, której drugim argumentem (tablicą, z którą była konwolowana tablica natężenia dźwięku w czasie) jest funkcja $\mathbb{1}_{[t_1,t_2]}$, gdzie t_1 i t_2 to granice rozważanego okna.

4.1 Częstotliwość podstawowa

```
def rn(sound, l):
    if len(sound) <= l:
        return 0
    N = len(sound)
    total = 0

    for i in range(N-l):
        total += (sound[i]*sound[i+l])
```



Rysunek 2: Przybliżenie fragmentu

```
    return total/(len(sound)-l)


def fundamental_frequency(sound, frame_rate, window_size):
    L = len(sound)
    distance = window_size // 2
    freqs = []
    min_freq = 20
    max_freq = 1000
    min_period = frame_rate // max_freq
    max_period = frame_rate // min_freq

    for i in range(window_size, L, distance):
        corelations = []
        step = 1

        for l in range(min_period, max_period, step):
            corelations.append(rn(sound[i-window_size:i+l], l))

        f0 = frame_rate / (np.argmax(np.array(corelations)) * step + min_period)
        freqs.append(f0)

    return np.array(freqs)
```

Algorytm obliczania chwilowej częstotliwości podstawowej nie jest zbyt nieoczywisty, jednak jest

bardziej skomplikowany niż pozostałe, więc warto go przytoczyć.

Funkcja `rn()` oblicza wartość funkcji autokorelacji $R_n(l)$, sumując iloczyny natężeń o stałym odstępie l (mierzonym w liczbie próbkowań).

Funkcja `fundamental_frequency()` dla każdego analizowanego okna (zewnętrzna pętla) i dla każdego odstępu z interesującego nas zakresu (wewnętrzna pętla), określanego za pomocą minimalnej i maksymalnej częstotliwości, oblicza wartość $R_n(l)$, a następnie znajduje argument maksymalizujący ją w tym przedziale. Wynik jest następnie konwertowany z powrotem do częstotliwości, i dodawany do listy wartości do wyświetlenia na wykresie.

4.2 Low Short Time Energy Ratio

```
def ste(sound, window_size):
    l = len(sound)
    distance = window_size//2
    ste = []
    sound_squared = np.power(sound, 2)

    for i in range(window_size, l, distance):
        ste.append(sound_squared[i-window_size:i].mean())

    return np.array(ste)


def lster(sound, window_size, big_window_size):
    calculated_ste = ste(sound, window_size)
    num_windows = big_window_size // window_size
    lster = []

    for i in range(num_windows, len(calculated_ste), num_windows // 2):
        chunk = calculated_ste[i-num_windows:i]
        avg = chunk.mean()
        v = np.sign(0.5 * avg - chunk) + 1
        lster.append(v.sum() * 0.5 / num_windows)

return np.array(lster)
```

Aby obliczyć Low Short Time Energy Ratio, konieczne jest obliczenie Short Time Energy. Dokonuje tego funkcja `ste()`. Jej działanie jest niezbyt skomplikowane, więc nie ma powodu aby ją dokładniej opisywać.

Jako że Short Time Energy to parametr, który możemy określić w pewnym oknie, funkcja `lster()` musi być funkcją operującą na "oknach okien". Wielkość tego dużego okna jest zdefiniowana przez parametr `big_window_size` (w ilości próbkowań). Według definicji parametru wielkość ta powinna odpowiadać jednej sekundzie, i tak rzeczywiście jest: wartość tego argumentu ustawiana jest na częstotliwość wczytanego klipu, a więc na ilość próbkowań w każdej sekundzie.

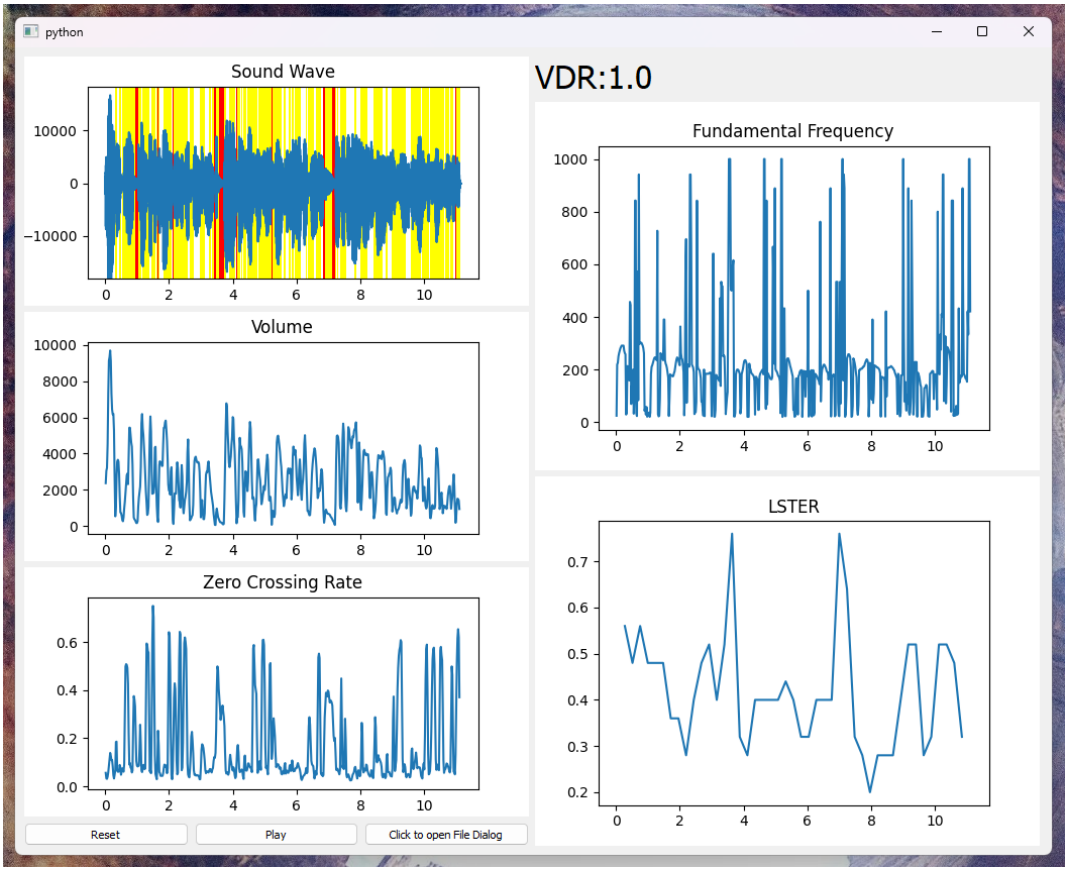
Dla każdego dużego okna (zewnętrzna pętla) obliczana jest średnia wartość STE. Następnie zliczana jest ilość okien, w których STE jest większe niż połowa średniej (poprzez `(np.sign(0.5 * avg - chunk) + 1).sum() * 0.5`). Wartość ta jest dzielona przez ilość analizowanych w jednym (dużym) oknie wartości STE. Otrzymana wartość jest dodawana do listy wartości do wyświetlenia na odpowiednim wykresie.

4.3 Detekcja ciszy i głosek dźwięcznych i bezdźwięcznych

Detekcje ciszy i głosek bezdźwięcznych działają na podobnej zasadzie.

Fragment jest klasyfikowany jako cisza, gdy $ZCR > 0.07$ oraz relatywna głośność $V < 300$; a jako głoska bezdźwięczna - kiedy $ZCR > 0.07$ i $V < 6000$. Działa to stosunkowo dobrze na nagraniach, które uzyskalismy podczas zajęć. Niestety w przypadku nagrań niezawierających głosu (jak widać na zrzucie ekranu 1) wiele fragmentów jest błędnie klasyfikowanych jako głoski bezdźwięczne.

5 Analiza wyników



Rysunek 3: Analiza klipu mowy

5.1 Porównanie muzyki i mowy

Zrzut ekranu 3 przedstawia analizę klipu mowy - dokładniej: fragmnent komunikatu radiowego w języku angielskim. Możemy te wyniki porównać z widocznymi na zrzucie ekranu 1, który przedstawia analizę klipu muzycznego - charakterystycznego motywu muzycznego z filmu "Różowa pantera".

Pierwsza różnica, która rzuca się w oczy, to obecność bardzo krótkich fragmentów ciszy w klipie mowy. Klip muzyczny nie zawiera ich.

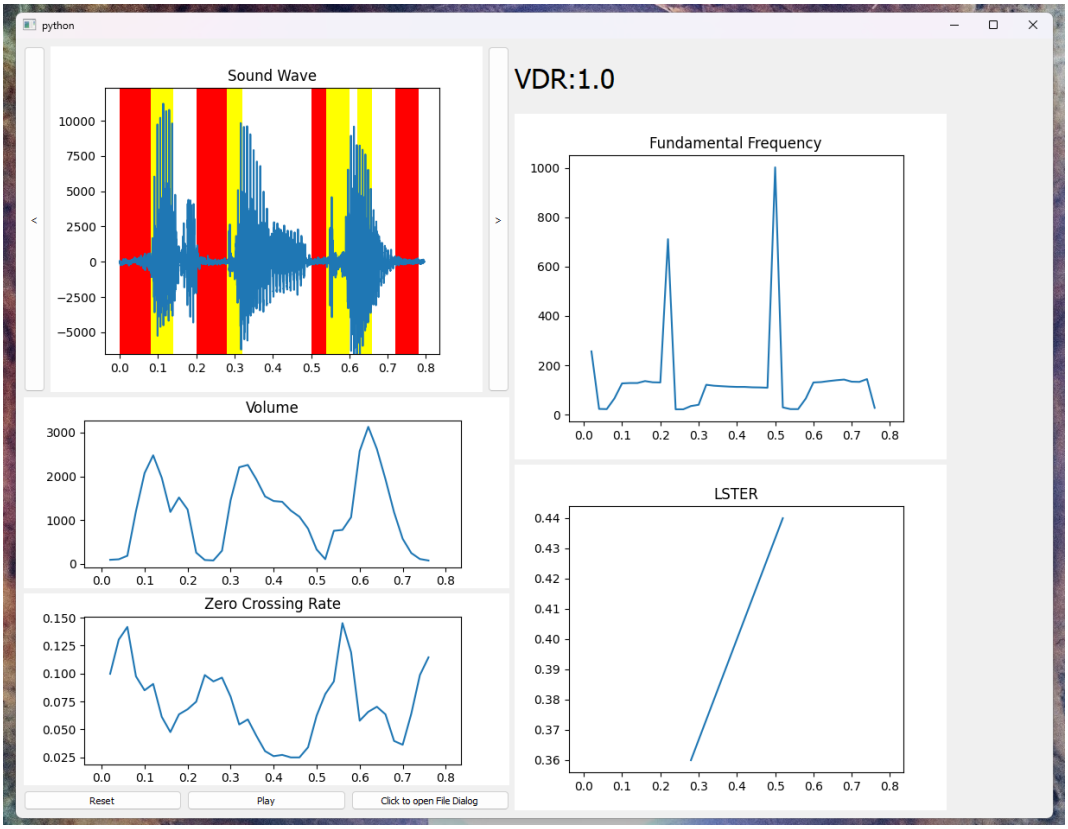
W przypadku wszystkich wykresów możemy zauważyć, że klip muzyczny cechuje się większą regularnością. Parametry zmieniają swoje wartości wolniej. W obrębie krótkiego okresu czasu piki wartości nie różnią się od siebie za bardzo.

Nieoczywiste rozróżnienie widzimy na wykresie STE: w przypadku mowy możemy wyodrębnić fragmenty, gdzie wartość tego parametru jest przez pewien czas bardzo mała. Natomiast w przypadku muzyki nie widzimy takich fragmentów.

Najbardziej rzuca się w oczy różnica na wykresie częstotliwości podstawowej. W przypadku muzyki piki tego wykresu mają niezwykle podobne do siebie wartości (w krótkim okresie czasu), tworząc poziome odcinki. W przypadku mowy też możemy wyróżnić kilka dominujących częstotliwości, jednak nie widzimy tak wyraźnych fragmentów, w których dominują. Są też mniej dokładnie określone - widzimy na przykład, że wartość na wykresie często oscyluje w okolicach $200Hz$, jednak przyjmowane wartości w tych okolicach całkiem znacznie się różnią.

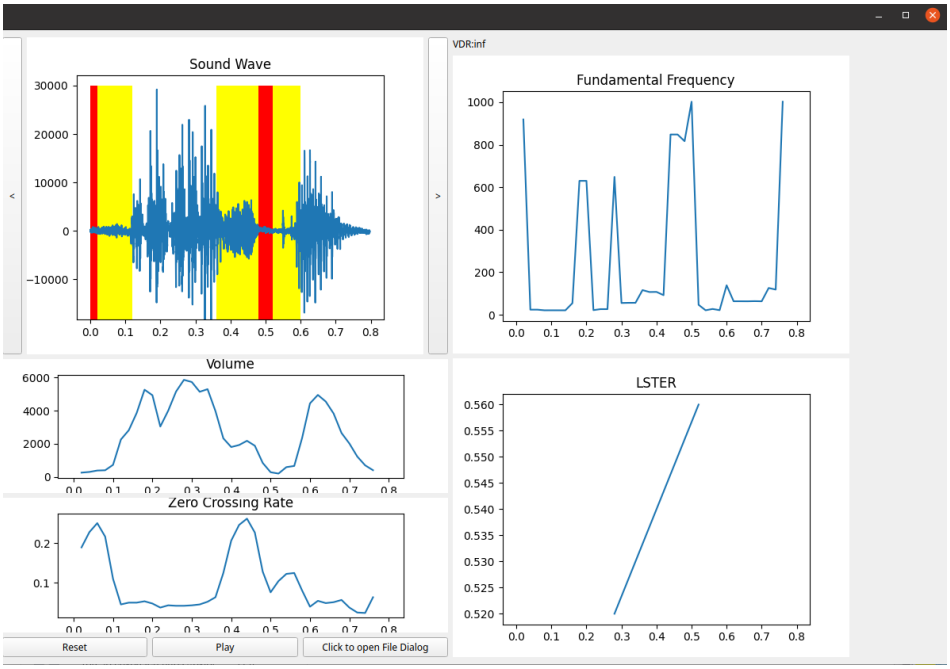
Mniej wyraźna jest różnica w parametrze $LSTER$. Tak jak w przypadku pozostałych wykresów, w przypadku mowy widzimy bardziej chaotyczne zachowanie, ale poza tym możemy zauważyć, że przyjmowane wartości są wyraźnie większe. Dla mowy średnia znajduje się w okolicach $0,5$, natomiast w przypadku muzyki jest to około $0,3$.

5.2 Wykrywanie głosek bezdźwięcznych



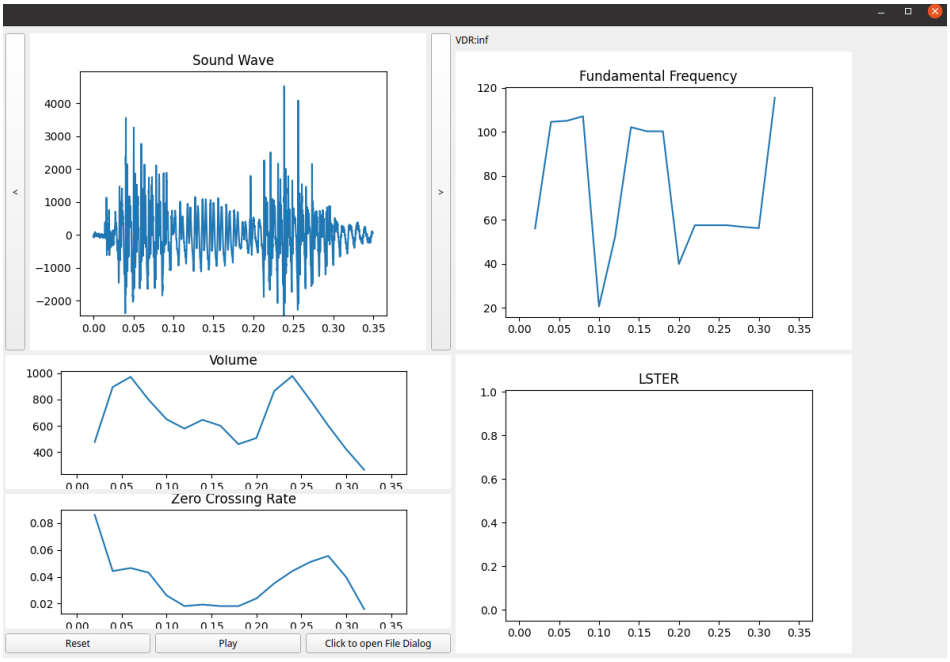
Rysunek 4: Analiza klipu zawierającego słowo 'artenko'

Jak widać na zrzutach ekranu 1 i 3, wykrywanie głosek bezdźwięcznych nie działa idealnie. Klip muzyczny jest w większości żółty, chociaż oczywiście nie zawiera żadnych głosek, a tym bardziej bezdźwięcznych. Jednak na krótkich klipach mowy funkcjonalność ta działa całkiem dobrze. Zrzut ekranu 4 przedstawia jedno z nagrań wykonanych podczas laboratorium. Zawiera ono słowo "artenko". Przyglądając się uważnie wzorowi zamalowanych obszarów możemy zauważyć, że rzeczywiście odpowiadają one głoskom bezdźwięcznym. Jedynym wyjątkiem jest tutaj głoska "a", która także została zakfalifikowana jako bezdźwięczna.



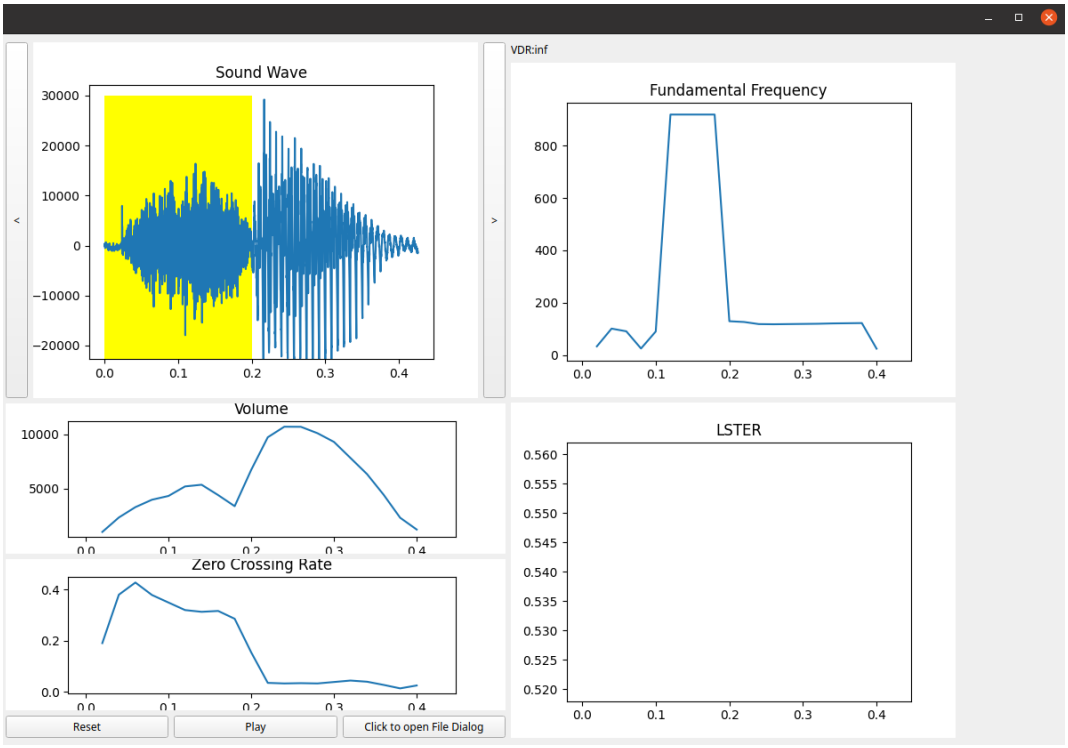
Rysunek 5: Analiza klipu zawierającego słowo 'faraszka'

Lepszym przykładem jest analiza słowa "faraszka", widoczna na zrzucie ekranu 5. Tutaj wszystkie głoski zostały przyporządkowane poprawnie: bezdźwieczne "f", następnie dźwięczne "a", (cisza) bezdźwięczne "r", znów "a", (cisza) bezdźwięczne "sz"i "k", oraz "a". Kilka ramek zostało jednak zakwalifikowanych jako cisza.



Rysunek 6: Analiza klipu zawierającego słowo 'aba'

Przyjrzyjmy się jeszcze kilku przykładom, które działają bardzo dobrze. Pierwszym z nich jest słowo "aba"przestawione na zrzucie ekranu 6. Widać tutaj, że wszystkie głoski zostały zakwalifikowane jako dźwięczne.



Rysunek 7: Analiza klipu zawierającego słowo ’trzy’

Ostatnim interesujacym przykładem jest analiza słowa "trzy", które jednak wymawia się jako "tszy", co nasz program wychwycił i zaznaczył "rz"słyszane jako "sz"jako głoskę bezdźwięczną.