

POLITECHNIKA ŚLĄSKA

WYDZIAŁ INŻYNIERII BIOMEDYCZNEJ



Katedra Biosensorów i Przetwarzania Sygnałów
Biomedycznych

Programowanie Python

Projekt

Narzędzie CLI do konwersji częstotliwości próbkowania

Suchanek Kamil

PiAIB semestr 2

Prowadzący projekt: Mgr inż. Konrad Duraj

Bytom, 28 grudnia 2019r.

Spis treści

1.	Cel i założenia	3
2.	Wstęp teoretyczny	3
2.1.	Interpolacja	3
2.2.	Ekspansja sygnału	4
3.	Realizacja projektu	4
4.	Specyfikacja wewnętrzna	4
4.1.	Wskazówki dla użytkownika	4
4.2.	Załadowanie potrzebnych modułów	5
4.3.	Funkcja resampleit(path, target_frequency)	6
5.	Specyfikacja zewnętrzna	7
6.	Podsumowanie	8

1. Cel i założenia

Celem projektu jest stworzenie narzędzia linii poleceń, służącego do konwersji częstotliwości próbkowania jednowymiarowych sygnałów bioelektrycznych.

Narzędzie umożliwia konwersję częstotliwości próbkowania z dowolnym czynnikiem poprzez wprowadzenie odpowiednich flag-argumentów oraz wskazania ścieżki do pliku z sygnałem.

2. Wstęp teoretyczny

Konwersja częstotliwości próbkowania sygnału realizowana jest poprzez ekspansję i podpróbkowanie sygnału.

2.1. Interpolacja

Wartości sygnału cyfrowego znane są w dyskretnych punktach czasu. W przypadku równomiernie próbkowanych sygnałów, odstęp między próbami jest stały i równy okresowi próbkowania $T = \frac{1}{fs}$, gdzie fs oznacza częstotliwość próbkowania, czyli liczbę próbek przypadających na jedną sekundę sygnału. Zdarza się, że zachodzi potrzeba uzyskania innej częstotliwości próbkowania niż pierwotna, albo potrzebna jest wartość w chwili nie zdeterminowanej dyskretną próbą sygnału [1].

Interpolacja sygnału cyfrowego polega na obliczeniu wartości sygnału w dowolnym punkcie między próbami. Należy zaznaczyć, że interpolacja nie tworzy nowych danych sygnału, powstał sygnał nie będzie identyczny z odpowiednikiem rejestrowanym z większą częstotliwością próbkowania. Wartości w dodatkowych punktach są estymowane na podstawie dostępnych [1].

2.2. Ekspansja sygnału

Ekspansja polega na “poszerzeniu” sygnału o dodatkowe próbki, zwiększając częstotliwość próbkowania. Operacja ta przeprowadzona w dziedzinie czasu powoduje zwężenie się widma częstotliwościowego, objawiające się poprzez wystąpienie niskich częstotliwości

3. Realizacja projektu

Zastosowano moduł docopt do sformułowania narzędzia CLI ze skryptu Pythona. W celu przetwarzania sygnałów zastosowano wrapper do narzędzi WFDB służących do tworzenia, odczytywania, przetwarzania i zaopatrywania w adnotacje sygnałów w serwisie PhysioNet.

4. Specyfikacja wewnętrzna

Skrypt narzędzia CLI składa się z czterech części:

- ❖ Wskazówki dla użytkownika,
- ❖ Załadowanie potrzebnych modułów,
- ❖ Funkcja spełniająca zadanie narzędzia,
- ❖ Fragment warunkujący uruchomienie narzędzia z argumentami i bez nich.

4.1. Wskazówki dla użytkownika

Na samej górze skryptu znajduje się notatka wskazująca na prawidłowe użycie narzędzia.

```
resampleit.py
"""
RESAMPLE IT CLI
Usage:
    resampleit.py
    resampleit.py <target_frequency> <path>
    resampleit.py -h|--help
    resampleit.py -v|--version
Options:
    <path> Full path to the record file with the extension .dat or .hea.
    <target_frequency> Target sampling rate.
    -h --help Show this screen.
    -v --version Show version.
"""


```

Rys. 1 - Wskazówka do wykorzystania narzędzia CLI

4.2. Załadowanie potrzebnych modułów

Narzędzie wykorzystuje następujące moduły:

- ❖ docopt - narzędzie do formułowania narzędzi CLI,
- ❖ numpy - pakiet narzędzi numerycznych i nie tylko,
- ❖ wfdb - narzędzia wspomniane w punkcie 3 - Realizacja projektu,
- ❖ zenipy - narzędzie do interaktywnego wyboru plików i folderów, wygodne i zachowujące natywny wygląd bieżącego interfejsu.

```
from docopt import docopt
import numpy
import wfdb
from wfdb import processing
from zenipy import file selection
```

Rys. 2 - Zależności narzędzia

4.3. Funkcja resampleit(path, target_frequency)

Funkcja wykonuje kolejno następujące czynności:

- ❖ usunięcie z pełnej ścieżki do pliku rozszerzenia .hea / .dat,
- ❖ wczytanie rekordu WFDB jako obiektu Python,
- ❖ dostosowanie formy tablicy do celów obliczeniowych,
- ❖ przepróbkowanie sygnału,
- ❖ stworzenie rekordu WFDB z przepróbkowanym sygnałem,
- ❖ informacja zwrotna.

```
def resampleit(path, target_frequency):

    # change file name
    path = path.replace('.dat','')
    path = path.replace('.hea','')

    # reading MIT file
    record = wfdb.rdsamp(path)

    # create signal vector
    signal = numpy.concatenate(record[0])

    # resampling signal
    signal_resample = processing.resample_sig(signal, record[1]['fs'], target_frequency)

    wfdb.wrsamp(path.split('/')[-1] + '_resampled_to' + str(target_frequency),
    fs = target_frequency, units = record[1]['units'],
    fmt=["16"],
    sig_name = record[1]['sig_name'],
    p_signal = numpy.asarray(signal_resample[0].reshape(len(signal_resample[0]), 1)) )

    print('Signal was resampled from %i [Hz] to %i [Hz]\n' %(record[1]['fs'], target_frequency))
    print('The resulting files are located in the current folder\n')
```

Rys. 3 - Funkcja resampleit.

5. Specyfikacja zewnętrzna

Istnieją dwie metody użycia narzędzia:

- ❖ Wraz z argumentami:
 - przykład zastosowania:
 - wywołanie interpretera "python3",
 - wskazanie pliku narzędzia "resampleit_py",
 - argument pierwszy - częstotliwość docelowa "200",
 - argument drugi - pełna ścieżka do rekordu wfdb, jak poniżej:

```
>> python3 resampleit.py 200 "/home/ciarambola/Pulpit/semestr  
2/PP/ResampleIt_CLI/data/ansiaami-ec13-test-waveforms-  
1.0.0/aami3a.dat"
```

- widok terminala:

```
ciarambola@CiarambolaNaUbuntu:~/Pulpit/semestr 2/PP/ResampleIt_CLI/program$  
python3 resampleit.py 200 "/home/ciarambola/Pulpit/semestr 2/PP/ResampleIt_C  
LI/data/ansiaami-ec13-test-waveforms-1.0.0/aami3a.dat"  
Signal was resampled from 720 [Hz] to 200 [Hz]
```

The resulting files are located in the current folder

Rys. 4 - Widok terminala przy zastosowaniu narzędzia z argumentami.

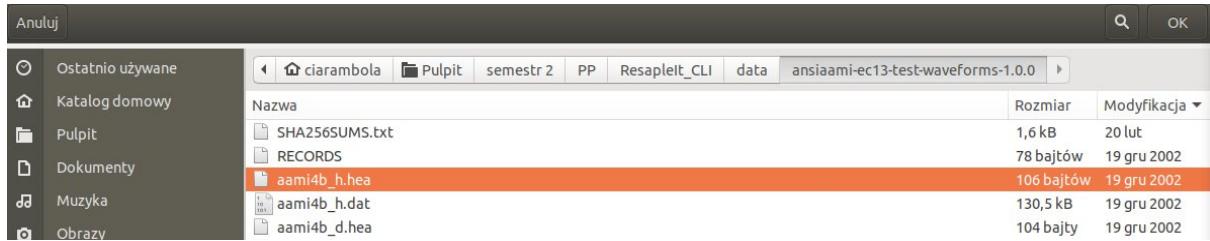
- ❖ Bez argumentów:
 - przykład zastosowania:
 - wywołanie interpretera python3 oraz użycie narzędzia resampleit.py bez argumentów:

```
ciarambola@CiarambolaNaUbuntu:~/Pulpit/semestr 2/PP/ResampleIt_CLI/program$  
python3 resampleit.py  
Gtk-Message: 15:21:50.808: GtkDialog mapped without a transient parent. Thi  
s is discouraged.
```

Rys. 5 - Widok terminala przy zastosowaniu narzędzia bez argumentów.

Ewentualne powiadomienia w terminalu należy zignorować o ile nie informują o błędzie, przerywając działanie narzędzia.

Od razu po wpisaniu polecenia do konsoli powinno pojawić się okno wyboru pliku.



Rys. 6 - Okno wyboru pliku.

- następnie należy wprowadzić w oknie terminala docelową częstotliwość próbkowania i zatwierdzić wybór.

```
Enter target frequency: 200
Signal was resampled from 720 [Hz] to 200 [Hz]
```

```
The resulting files are located in the current folder
```

Rys. 7 - Widok terminala po wprowadzeniu i zatwierdzeniu docelowej częstotliwości próbkowania.

Wynik działania narzędzia znajduje się w bieżącej pozycji terminala, w przykładach jest to folder "program".

6. Podsumowanie

Utworzono zgodne z założeniami narzędzie do konwersji częstotliwości próbkowania. Można go używać zarówno w formie interaktywnej oraz w innym skrypcie podając potrzebne argumenty.

Bibliografia

1. Opracowanie w notatniku Jupyter -
<https://sound.eti.pg.gda.pl/~greg/dsp/05-Interpolacja.html> [dostęp: 19.10.2019]
2. Wykłady, Zaawansowane metody analizy sygnałów biologicznych, Prof. dr hab. inż. Ewaryst Tkacz
3. Instrukcja do Laboratorium Cyfrowego przetwarzania sygnałów, Ćwiczenie 5 - Wielozestotliwościowe przetwarzanie sygnałów - interpolacja i decymacja -
<https://ioisp.el.pcz.pl/images/instrukcje/air/Cyfrowe%20Przetwarzanie%20Sygnalow/Laboratoria/Cw.5%20Wielozestotliwosciowe%20przetwarzanie%20sygnalow%20-%20interpolacja%20i%20decymacja.pdf> [dostęp: 19.10.2019]
4. Przetwarzanie sygnałów, Ćwiczenie 3 - Właściwości przekształcenia Fouriera - <http://www.w12.pwr.wroc.pl/ps/instrukcje/PS4.pdf> [dostęp: 19.10.2019]