Student: Kacper Połatajko
Nr indeksu: 241603
Kierunek: AiR ARK IV rok VII semestr
Termin: Poniedziałek $8^{15} - 11^{00}$ TN
Data wysłania 1. etapu: 03.12.2020 r.

PROCESORY SYGNAŁOWE – PROJEKT

ETAP 4 – ODDANIE PROJEKTU

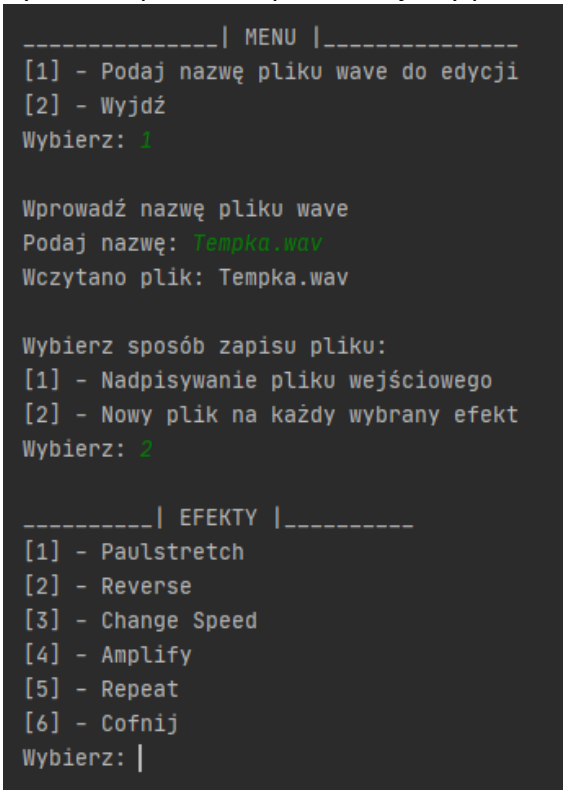
1. Opis wykonanych prac.

Projekt w poprzednim etapie zawierał podstawowe menu w terminalu oraz jeden efekt do wybrania – Paulstretch. Od tamtej fazy pojawiły się nowe efekty:

- Reverse,
- Change speed,
- Amplify,
- Repeat (w miejsce wcześniej planowanego Bass and Treble).

Dodatkowo oprócz stworzonego wcześniej menu tekstowego, wykonano GUI w PyQt5 zapewniające taką samą funkcjonalność jak menu kontekstowe oraz zapewniające dodatkowe zabezpieczenia i będące o wiele bardziej intuicyjne.

Menu tekstowe wygląda tak samo jak w poprzednim etapie. Jedyną zmianą jest sposób wyboru zapisu pliku wave z zastosowanymi efektami (każdy efekt będzie osobnym plikiem dźwiękowym albo każdy kolejny efekt zostanie nałożony na bazowy plik dźwiękowy). Po wybraniu sposobu zapisu zostajemy przeniesieni do menu efektów.



Efekt Reverse – odwraca wybrany dźwięk w czasie, tak że koniec będzie na początku a początek na końcu. Próbki dla lewego i prawego kanału zostają wpisane do osobnych tablic a następnie zostaje odwrócona ich kolejność i łączone są ponownie w jeden dwukanałowy dźwięk.

```
__________| EFEKTY |__________
[1] - Paulstretch
[2] - Reverse
[3] - Change Speed
[4] - Amplify
[5] - Repeat
[6] - Cofnij
Wybierz: 2

Wybrano efekt: Reverse
Reverse - Gotowe!
```

Efekt Change Speed – zmienia prędkość odtwarzanego dźwięku (przyspiesza dla mnożnika > 1, spowalnia dla mnożnika < 1). W programie istnieją dwa sposoby implementacji efektu. Pierwszy zmienia ilość próbek w dźwięku, częstotliwość próbkowania pozostaje taka sama, ale dźwięk jest znacznie pogorszony. Drugi nie zmienia ilości próbek, dźwięk jest wciąż dobrej jakości, ale zmieniamy częstotliwość próbkowania. Efekt jako jedyny potrzebuje wbudowanej, dodatkowej biblioteki „random". Reszta efektów posiada jedynie biblioteki „numpy" i „wave".

```
__________| EFEKTY |__________
[1] - Paulstretch
[2] - Reverse
[3] - Change Speed
[4] - Amplify
[5] - Repeat
[6] - Cofnij
Wybierz: 3

Wybrano efekt: Change Speed
Wprowadź wartość mnożnika prędkości: 1.2

Który parametr uchronić przed edycją?
[1] - Częstotliwość próbkowania
[2] - Ilość próbek
Wybierz: 2
Change speed (metoda II) - Gotowe!
```

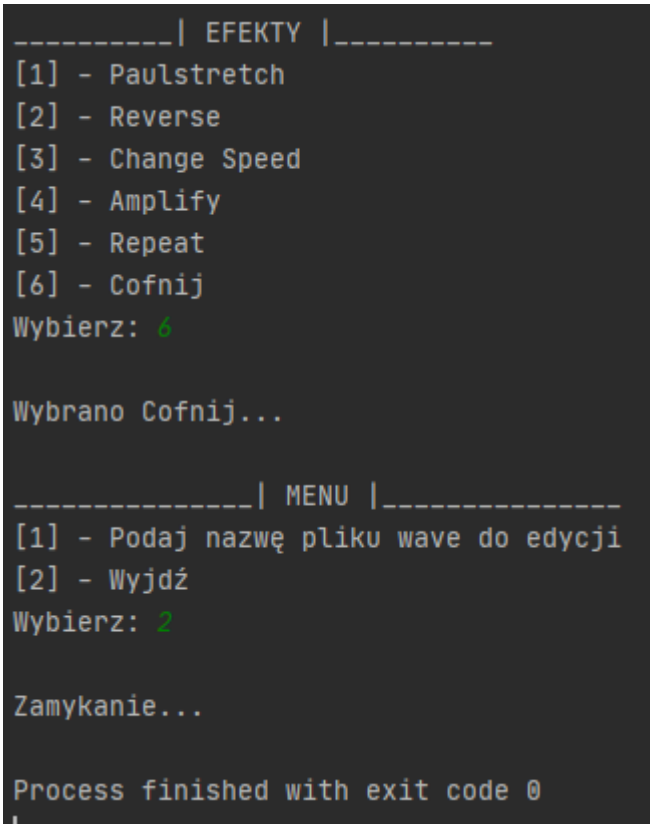
Efekt Amplify – wzmacnia dźwięk o wybraną wartość decybeli. Jeśli próbki przekraczają wartość 1 lub -1 zostaną one obcięte do maksymalnych wartości przedziałów (przedział [-1, 1]). Próbki zostają przemnożone według wzoru: $wzmocnienie = \left(\frac{mnożnik}{20}\right)^{10}$.

```
__________| EFEKTY |__________
[1] - Paulstretch
[2] - Reverse
[3] - Change Speed
[4] - Amplify
[5] - Repeat
[6] - Cofnij
Wybierz: 4

Wybrano efekt: Amplify
Wprowadź wartość wzmocnienia (dB): 3.12
Amplify - Gotowe!
```

Efekt Repeat – powtarza wybrany przez nas fragment dźwięku o daną ilość razy. Na podstawie wybranego przedziału czasu wybierane są graniczne próbki z tablicy i następnie wycinek tablicy, pomiędzy tymi próbkami, jest powtarzany o wybraną ilość razy.

```
__________| EFEKTY |__________
[1] - Paulstretch
[2] - Reverse
[3] - Change Speed
[4] - Amplify
[5] - Repeat
[6] - Cofnij
Wybierz: 5

Wybrano efekt: Repeat
Podaj czas początku i końca fragmentu do powtórzenia (sekundy): 8.123,9.487
Ile razy powtórzyć wybrany fragment?: 3
Repeat - Gotowe!
```

Wyjście z programu wygląda identycznie do poprzedniego etapu.

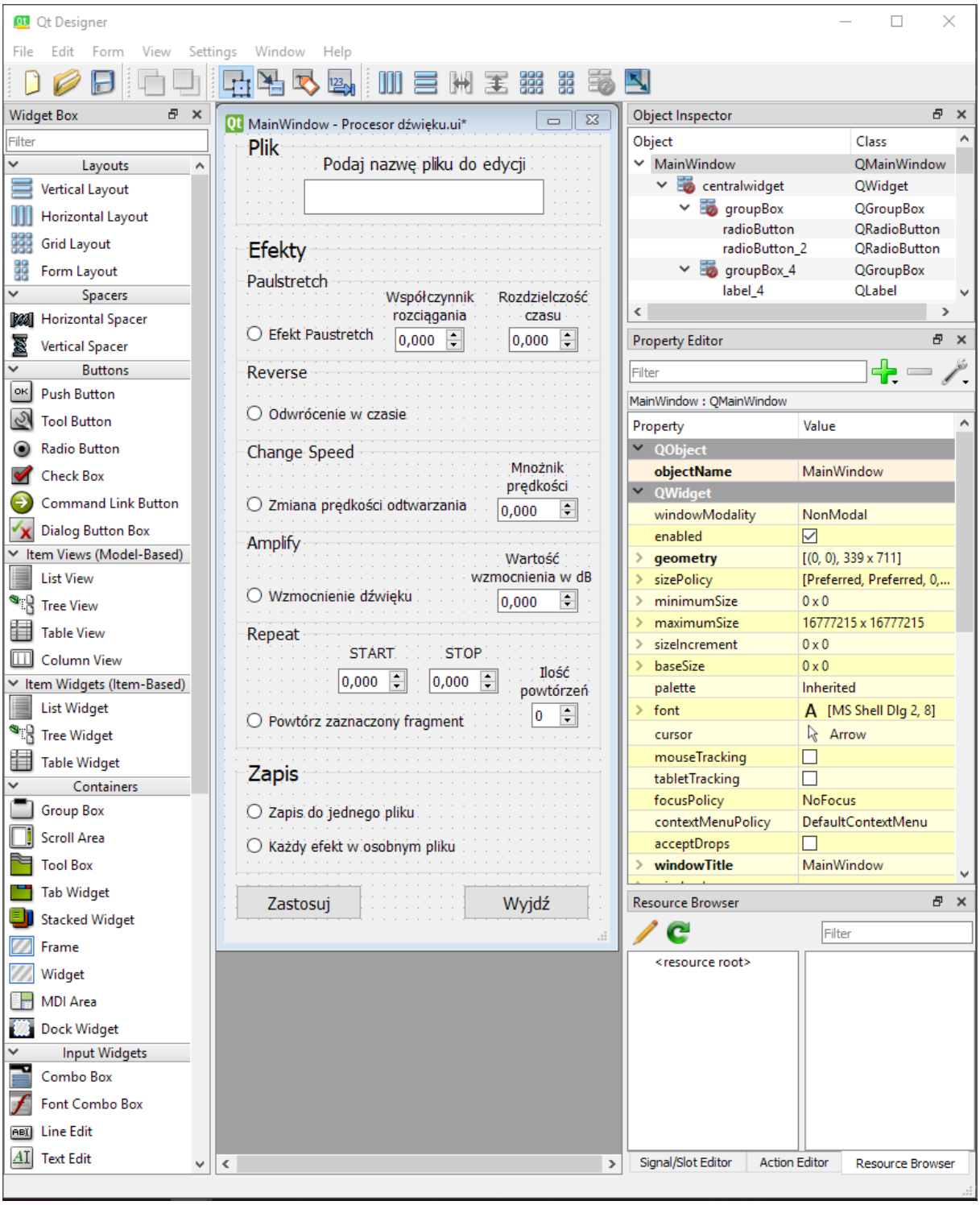


Program z GUI zapewnia identyczną funkcjonalność co menu tekstowe z dodatkowymi zabezpieczeniami dla braku wybrania pliku, braku wybrania formy zapisu oraz braku wybrania efektu. GUI zostało stworzone przy pomocy biblioteki PyQt5 posiłkując się dodatkowym programem QtDesigner przy pomocy którego można w łatwy sposób zaprojektować wygląd programu i poszerzać okno o dodatkowe funkcjonalności.

Okno programu podzielone jest na trzy sekcje:

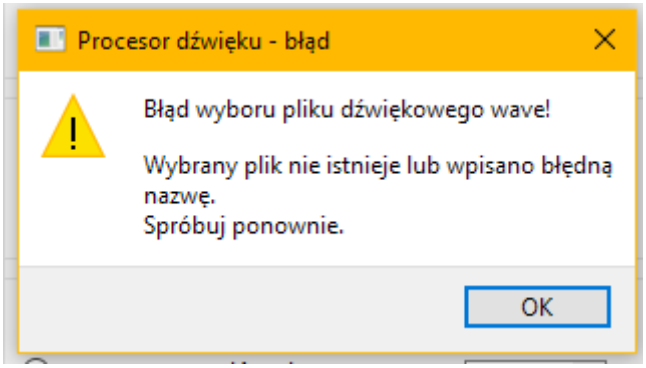
- Plik – w którym wpisujemy nazwę pliku wave do edycji (razem z rozszerzeniem),
- Efekty – w którym wybieramy interesujący nas efekt/efekty wraz z parametrami (jeśli takie posiadają),
- Zapis – w którym wybieramy opcję zapisu przetworzonych plików (jako osobne pliki bądź nadpisanie bazowego pliku).

W każdej sekcji programu musi zostać wybrana/wpisana minimalnie jedna opcja.

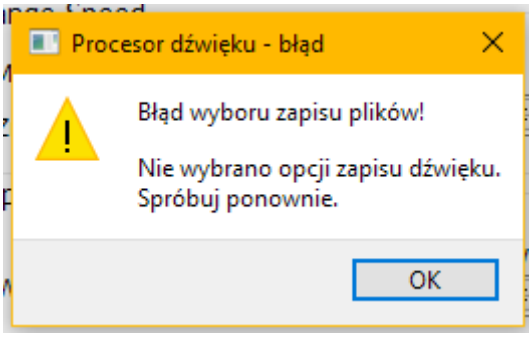


Program posiada funkcję zabezpieczeń oraz powiadomień w postaci wyświetlanych informacyjnych okien.

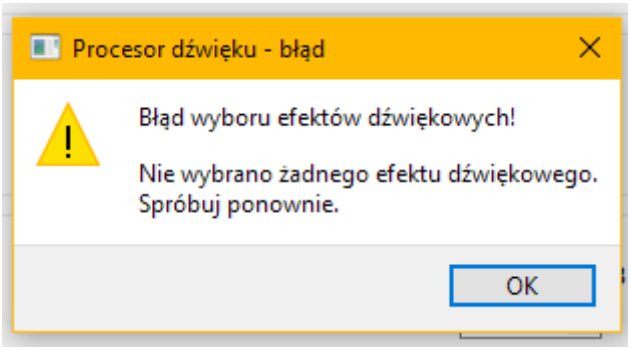
W przypadku niewpisania żadnej nazwy, podaniu nazwy pliku, który nie istnieje lub niedodania rozszerzenia w polu „Podaj nazwę pliku do edycji” wyświetli się komunikat poniżej



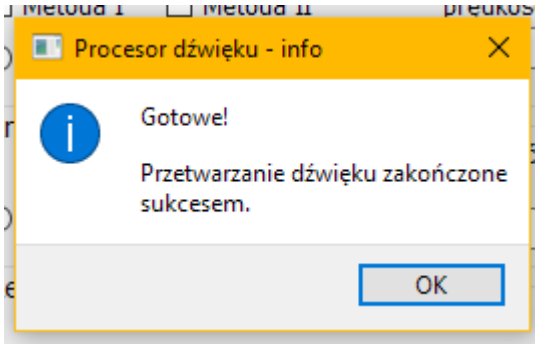
W przypadku niewybrania metody zapisu plików wyświetli się ostrzeżenie poniżej.



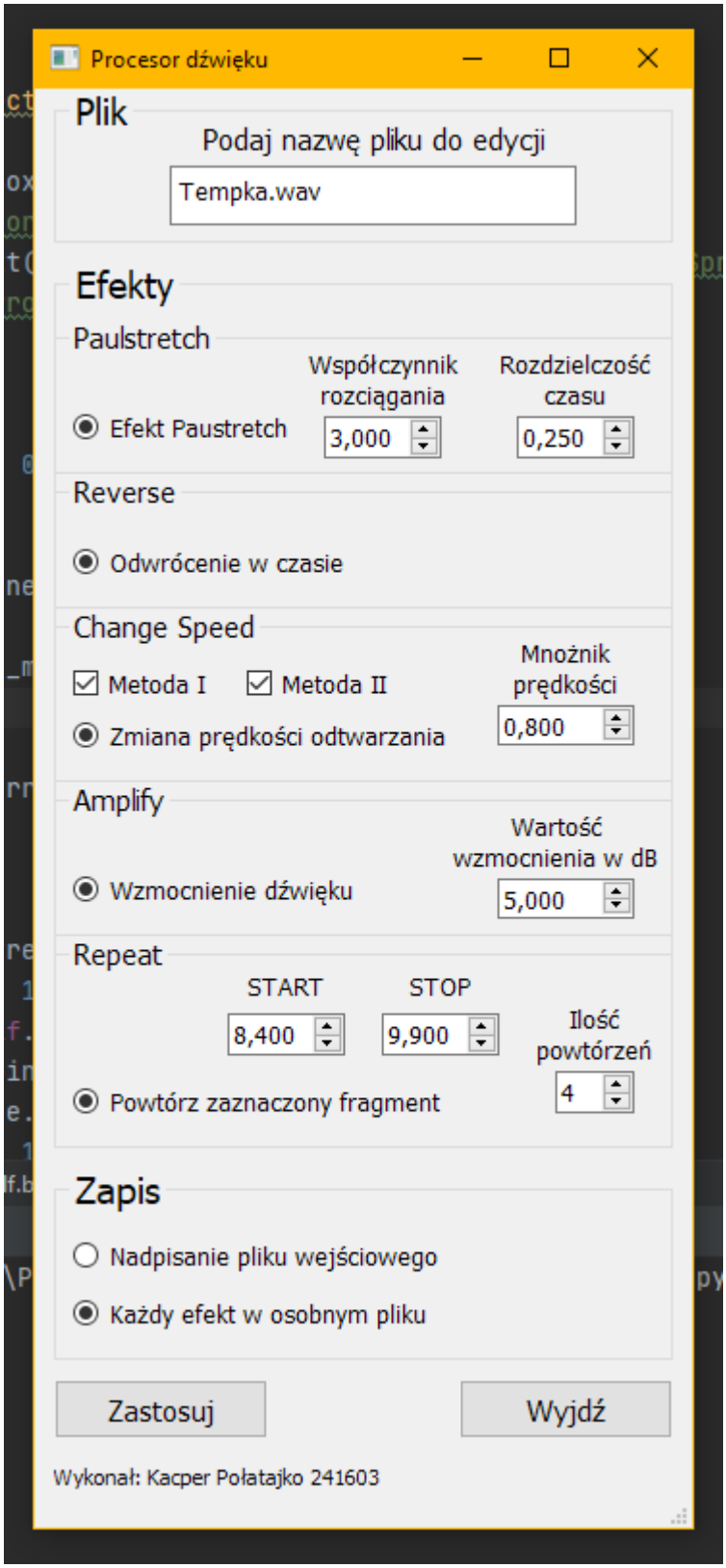
W przypadku niewybrania żadnego efektu wyświetli się komunikat poniżej.



Jeśli poprawnie wybierzemy wszystkie opcje i parametry wyświetli się komunikat o udanej operacji nałożenia efektu/efektów na dźwięk.

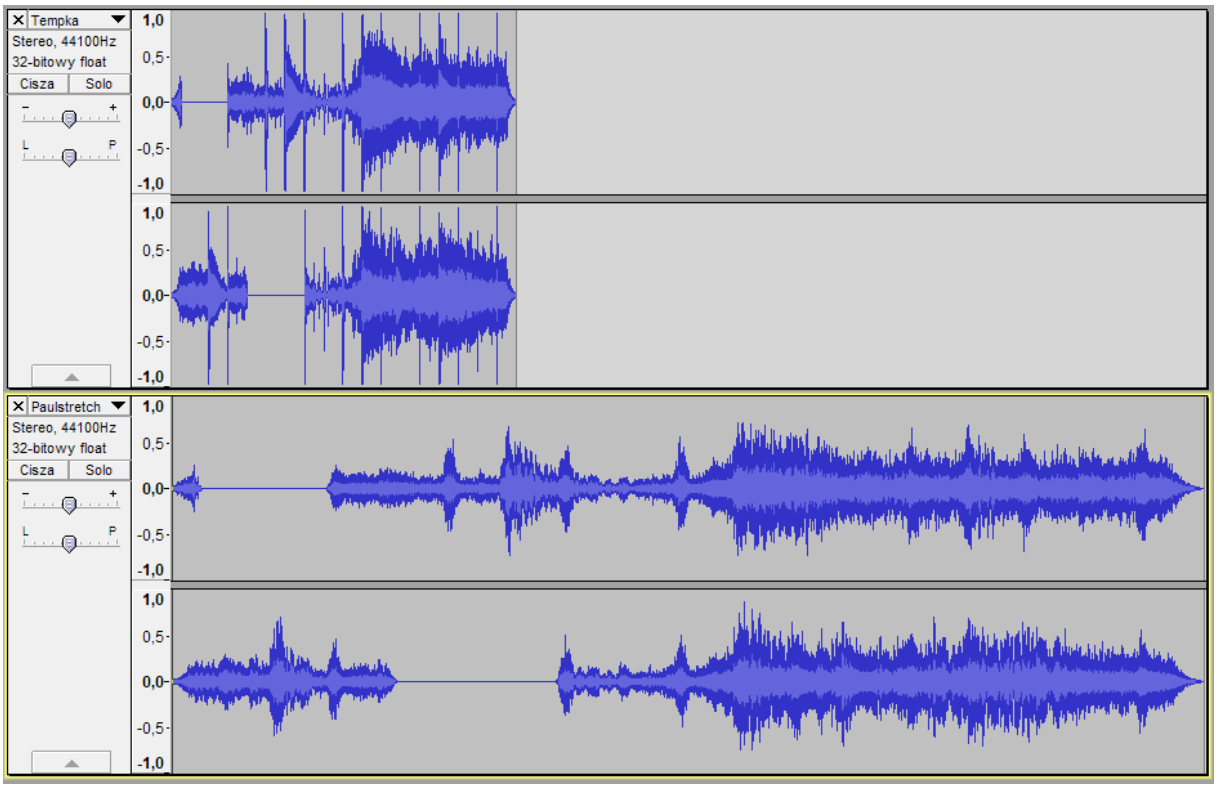


Gotowe okno programu wygląda tak jak zaprojektowane w programie QtDesigner z dodatkowym podpisem autora na samym dole.

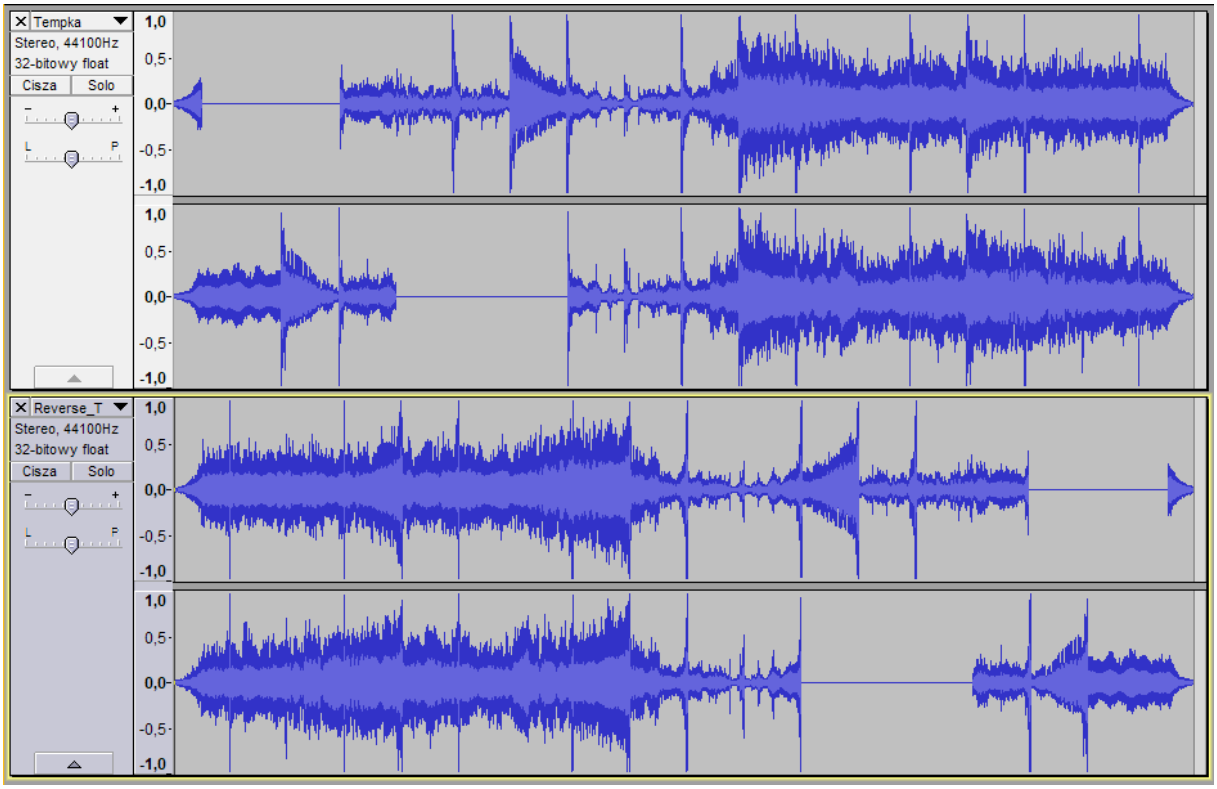


Wynik zastosowanej powyżej konfiguracji przedstawiono poniżej. W oryginalnym pliku Tempka.wav wyciszono w różnych momentach lewy i prawy kanał, aby sprawdzić czy łączenie kanałów po edycji jest poprawne oraz jak cisza wpływa na algorytmy.

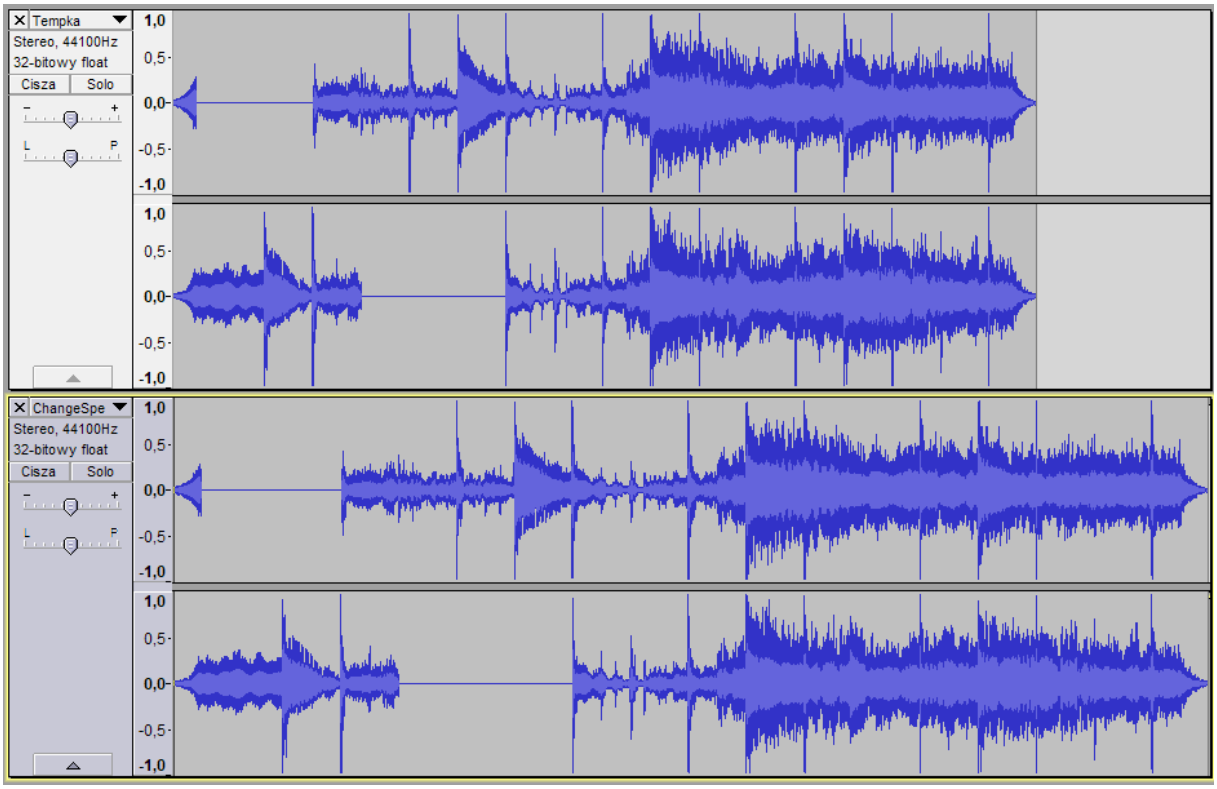
Efekt Paulstretch:



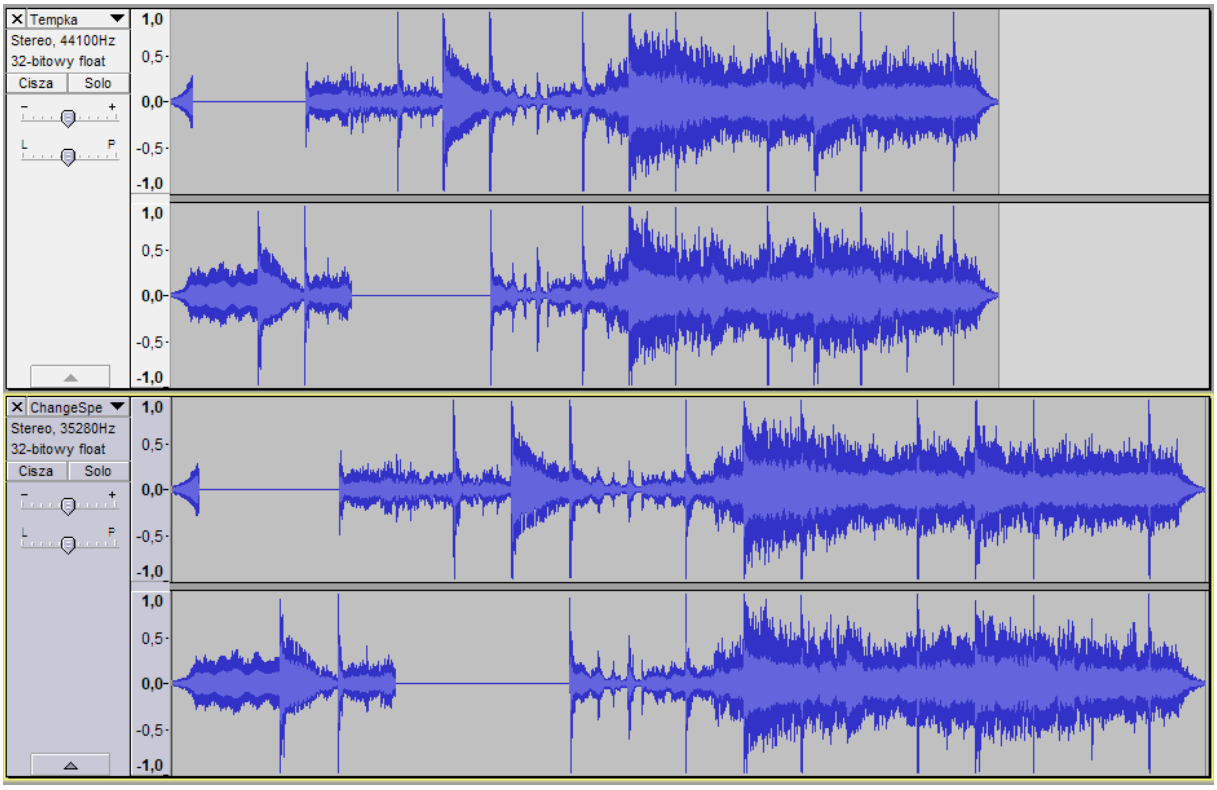
Efekt Reverse:



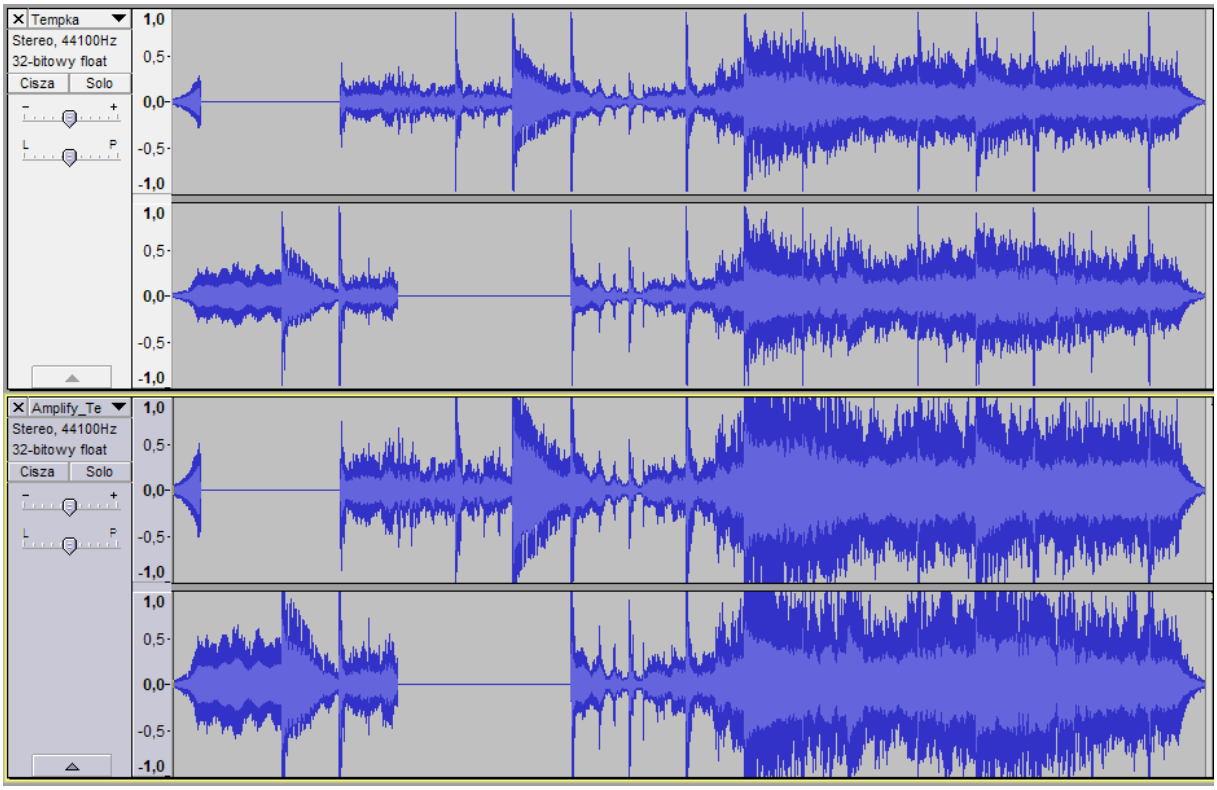
Efekt Change Speed 1 (zachowanie częstotliwości próbkowania):



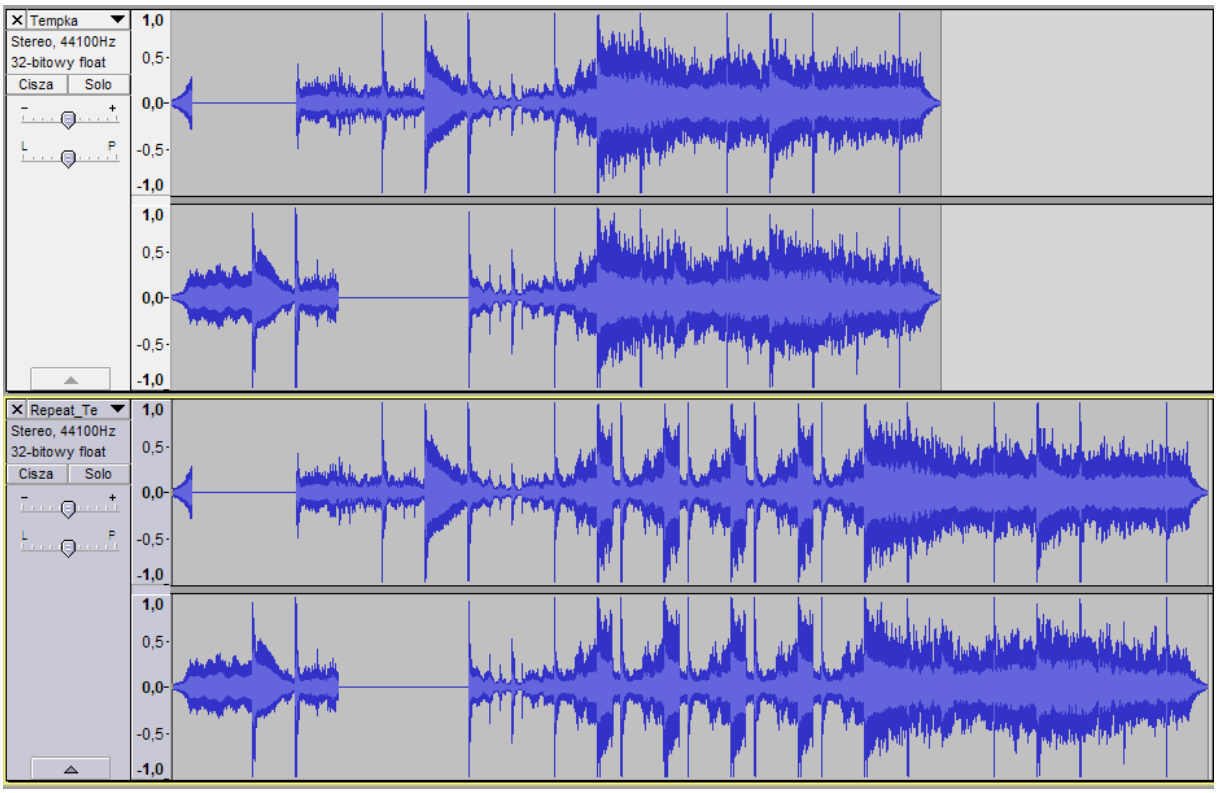
Efekt Change Speed 2 (zachowanie ilości próbek):



Efekt Amplify:

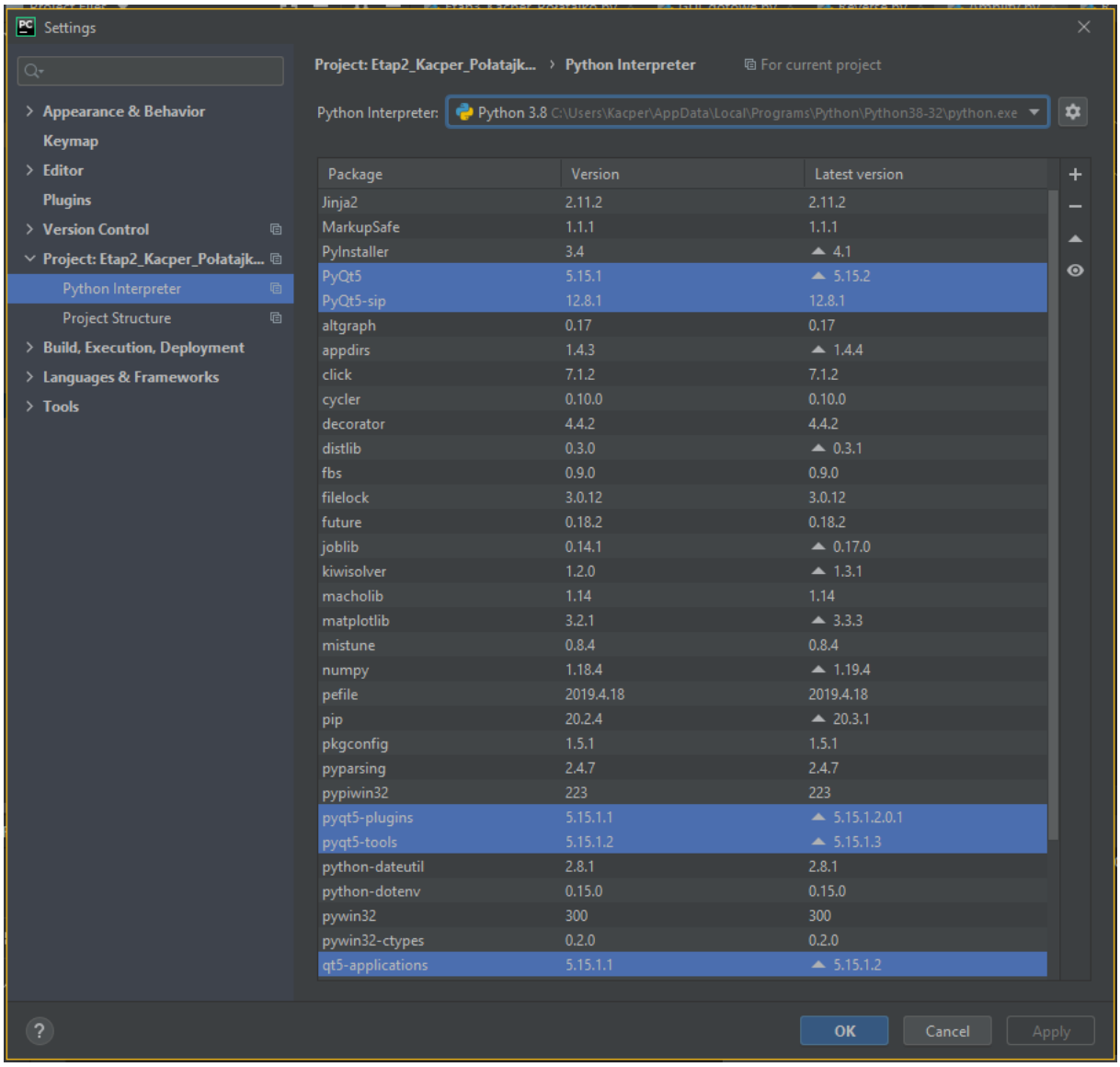


Efekt Repeat:

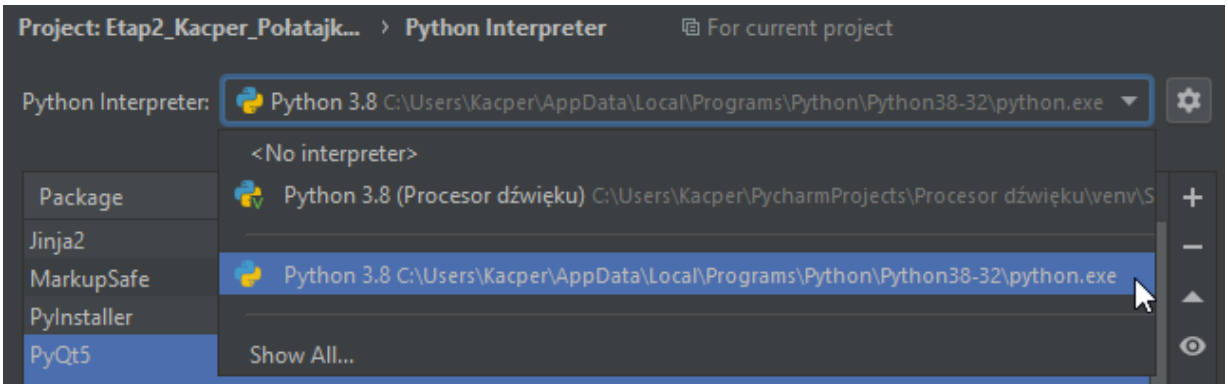


2. Instrukcja kompilacji.

Po wykonaniu procesu kompilacji opisanego w poprzednim etapie, wystarczy do wcześniejszych bibliotek dodać „PyQt5”, „PyQt5-sip”, „pyqt5-tools”, „pyqt5-plugins” oraz „qt5-applications”.



Istnieje możliwość, że po dodaniu tych bibliotek program wciąż nie będzie chciał się uruchomić. Wtedy należy zmienić interpreter Pythona na ten wbudowany bazowo w program PyCharm.



3. Załączniki.

W skompresowanej paczce ZIP (o nazwie Połatajko_Kacper_Etap4.zip) znajduje się folder o tej samej nazwie a w nim wszystkie potrzebne do działania skryptu pliki:

- plik muzyczny Tempka.wav – plik wave przed zastosowaniem jakiegokolwiek efektu,
- plik muzyczny Tempka_wszystkie_efekty.wav – plik wave po zastosowaniu wszystkich efektów na jednym dźwięku,
- plik muzyczny Paulstrech_Tempka.wav – plik wave po zastosowaniu efektu Paulstrech,
- plik muzyczny Reverse_Tempka.wav – plik wave po zastosowaniu efektu Reverse,
- plik muzyczny ChangeSpeed1_Tempka.wav – plik wave po zastosowaniu efektu Change Speed metoda I,
- plik muzyczny ChangeSpeed2_Tempka.wav – plik wave po zastosowaniu efektu Change Speed metoda II,
- plik muzyczny Amplify_Tempka.wav – plik wave po zastosowaniu efektu Amplify,
- plik muzyczny Repeat_Tempka.wav – plik wave po zastosowaniu efektu Repeat,
- skrypt Etap4_Kacper_Połatajko_Terminal.py – program napisany w języku Python3 z menu wykonanym w terminalu,
- skrypt Etap4_Kacper_Połatajko_GUI.py – program napisany w języku Python3 z GUI,
- osobne skrypty dla każdego efektu,
- plik programu QtDesigner Procesor_dźwięku.ui – wygenerowany plik z QtDesigner zawierający czyste GUI bez żadnych funkcjonalności programowych,
- dokument Połatajko_Kacper_Etap4.pdf – sprawozdanie z 4. Etapu projektu.

Film z działania programu z terminalem i GUI (napisy należy włączyć funkcją w serwisie Youtube [skrót klawiszowy „c”]): https://youtu.be/fcB1iRVjJYQ

Kody źródłowe efektów:

Paulstretch:

```
from numpy import *
import wave
```

```
# ------------------------------------------------------------------------
------------------------
# Funkcja implementująca efekt dźwiękowy paulstretch.
# ===================================================
def paulstretch(samplerate, samples, stretch, window_size_seconds,
outfilename):
    # Przygotowanie dźwięku (próbek) do edycji
    # ========================================
    nchannels = samples.shape[0]
    outfile = wave.open(outfilename, 'wb')
    outfile.setsampwidth(2)
    outfile.setframerate(samplerate)
    outfile.setnchannels(nchannels)

    # Sprawdzenie czy rozmiar okna (ilość próbek) jest parzysty i większy
od 16
    window_size = int(window_size_seconds * samplerate)
    if window_size < 16:
        window_size = 16
    window_size = opt_window_size(window_size)
    window_size = int(window_size / 2) * 2
    half_window_size = int(window_size / 2)

    # Poprawienie końca zbioru próbek
    nsamples = samples.shape[1]
    end_size = int(samplerate * 0.05)
    if end_size < 16:
        end_size = 16
    samples[:, nsamples - end_size:nsamples] *= linspace(1, 0, end_size)

    # Obliczenie przesunięcia wewnątrz pliku wejściowego
    start_pos = 0.0
    displace_pos = (window_size * 0.5) / stretch

    # Stworzenie okna okna czasu
    window = pow(1.0 - pow(linspace(-1.0, 1.0, window_size), 2.0), 1.25)
    old_windowed_buffer = zeros((2, window_size))

    # Właściwe zastosowanie efektu
    # ============================
    while True:
        # Znalezienie bufora okna
        istart_pos = int(floor(start_pos))
        buffer = samples[:, istart_pos:istart_pos + window_size]
        if buffer.shape[1] < window_size:
            buffer = append(buffer, zeros((2, window_size -
buffer.shape[1])), 1)
        buffer = buffer * window

        # Uzyskanie amplitudy składowych częstotliwości i pominięcie fazy
        frequencies = abs(fft.rfft(buffer))

        # Dobranie losowo faz poprzez pomnożenie przez losową liczbę
zespoloną o module = 1
        phase = random.uniform(0, 2 * pi, (nchannels,
frequencies.shape[1])) * 1j
        frequencies = frequencies * exp(phase)

        # Odwrotna szybka transformata Fouriera
        buffer = fft.irfft(frequencies)
```

```
        # Dodanie do okna wyjściowego bufferora
        buffer *= window

        # Nałożenie wyjścia
        output = buffer[:, 0:half_window_size] + old_windowed_buffer[:,
half_window_size:window_size]
        old_windowed_buffer = buffer

        # Ograniczenie wartości w przedziale <-1.0, 1.0>
        output[output > 1.0] = 1.0
        output[output < -1.0] = -1.0

        # Zapisz wyjście do pliku wave (.wav)
        outfile.writeframes(int16(output.reshape(-1, order='F') *
32767.0).tobytes())

        # Sprawdzenie czy wszystkie próbki zostały przetworzone (jeśli tak
- wyświetl "Gotowe!")
        start_pos += displace_pos
        if start_pos >= nsamples:
            print('Paulstretch - Gotowe!')
            break

    outfile.close()  # zamknięcie edycji pliku
# ----------------------------------------------------------------------------
-------------------------



# ----------------------------------------------------------------------------
-------------------------
# Funkcja optymalizująca rozdzielczość okna czasu.
# (a dokładniej liczbę próbek mieszczących się w danym oknie) efektu
paulstretch.
#
================================================================================
====
def opt_window_size(n):
    temp_n = n
    while True:
        n = temp_n
        while (n % 2) == 0:
            n /= 2
        while (n % 3) == 0:
            n /= 3
        while (n % 5) == 0:
            n /= 5
        if n < 2:
            break
        temp_n += 1
    return temp_n
# ----------------------------------------------------------------------------
-------------------------
```

Reverse:

```
from numpy import *
import wave

#----------------------------------------------------------------------------
-------------------------
```

```
# Funkcja odwracająca utwór w czasie.
#====================================
def reverse(samplerate, samples, outfilename):
   nchannels = samples.shape[0]
   outfile = wave.open(outfilename, 'wb')
   outfile.setsampwidth(2)
   outfile.setframerate(samplerate)
   outfile.setnchannels(nchannels)

   newL = samples[0][::-1]
   newR = samples[1][::-1]
   new = vstack((newL, newR))

   # new = new.transpose()
   # scipy.io.wavfile.write('odwr.wav', samplerate, new)

   outfile.writeframes(int16(new.reshape(-1, order='F') *
32767.0).tobytes())
   outfile.close()
   print('Reverse - Gotowe!')
#-------------------------------------------------------------------------
------------------------
```

Change Speed:

```
from numpy import *
import wave
import random

#-------------------------------------------------------------------------
------------------------
# Funkcja zmieniająca prędkość utworu.
# W zależności od zmiany prędkości usuwamy losowo lub dodajemy zdublowane
losowo próbki.
#=========================================================================
=============
def change_speed_1(samplerate, samples, multiplier, outfilename):
   nchannels = samples.shape[0]
   outfile = wave.open(outfilename, 'wb')
   outfile.setsampwidth(2)
   outfile.setframerate(samplerate)
   outfile.setnchannels(nchannels)

   newL = samples[0]
   newR = samples[1]
   # print('Max value: ', max(newL))
   # print('Min value: ', min(newR))

   # procent = multiplier * 100.00 - 100.00

   to_del = int(newL.size * (multiplier - 1.00))
   # new_smp = newL.size - to_del

   # Jeśli multiplier > 1.00 to zmniejsz liczbę próbek
   # Próbki losowane są z tablicy próbek i usuwane
   if multiplier > 1.00:
      randomList = random.sample(range(0, newL.size), to_del)
      newL = delete(newL, randomList)
      newR = delete(newR, randomList)
   # Jeśli multiplier < 1.00 to zwiększ liczbę próbek
```

```
    # Próbki losowane są z tablicy próbek i dublowane
    elif multiplier < 1.00:
        randomList = random.sample(range(0, newL.size), -to_del)
        itemListL = newL[randomList]
        itemListR = newR[randomList]
        newL = insert(newL, randomList, itemListL)
        newR = insert(newR, randomList, itemListR)

    new = vstack((newL, newR))

    outfile.writeframes(int16(new.reshape(-1, order='F') *
32767.0).tobytes())
    outfile.close()
    print('Change speed (metoda I) - Gotowe!')
#-------------------------------------------------------------------------
------------------------



#-------------------------------------------------------------------------
------------------------
# Funkcja zmieniająca prędkość utworu.
# Jedyny parametr, który zmieniamy to częstotliwość próbkowania.
#=========================================================
def change_speed_2(samplerate, samples, multiplier, outfilename):
    nchannels = samples.shape[0]
    outfile = wave.open(outfilename, 'wb')
    outfile.setsampwidth(2)
    outfile.setframerate(samplerate * multiplier)
    outfile.setnchannels(nchannels)

    newL = samples[0]
    newR = samples[1]
    new = vstack((newL, newR))

    outfile.writeframes(int16(new.reshape(-1, order='F') *
32767.0).tobytes())
    outfile.close()
    print('Change speed (metoda II) - Gotowe!')
#-------------------------------------------------------------------------
------------------------
```

Amplify:

```
from numpy import *
import wave

#-------------------------------------------------------------------------
------------------------
# Funkcja wzmacniająca cały utwór o wybraną wartość wzmocnienia (w dB.).
#=========================================================================
def amplify(samplerate, samples, dB, outfilename):
    nchannels = samples.shape[0]
    outfile = wave.open(outfilename, 'wb')
    outfile.setsampwidth(2)
    outfile.setframerate(samplerate)
    outfile.setnchannels(nchannels)

    newL = samples[0]
    newR = samples[1]
```

```
    amp = pow(10, dB / 20)

    newL = newL * amp
    newR = newR * amp

    newL[newL > 1.0] = 1.0
    newL[newL < -1.0] = -1.0
    newR[newR > 1.0] = 1.0
    newR[newR < -1.0] = -1.0

    # print(max(newL))
    # print(max(newR))
    # print(min(newL))
    # print(min(newR))
    new = vstack((newL, newR))

    outfile.writeframes(int16(new.reshape(-1, order='F') *
32767.0).tobytes())
    outfile.close()
    print('Amplify - Gotowe!')
#------------------------------------------------------------------------
------------------------
```

Repeat:

```
from numpy import *
import wave

#------------------------------------------------------------------------
------------------------
# Funkcja powtarzająca wybrany fragment dźwięku (w sek.) o wybraną ilość
razy.
#========================================================================
===
def repeat_sample(samplerate, samples, start_time, stop_time, amount,
outfilename):
    nchannels = samples.shape[0]
    outfile = wave.open(outfilename, 'wb')
    outfile.setsampwidth(2)
    outfile.setframerate(samplerate)
    outfile.setnchannels(nchannels)

    # print("Długość: ", samples.shape[1] / samplerate)

    newL = samples[0]
    newR = samples[1]

    start_id = int(start_time * samplerate)
    stop_id = int(stop_time * samplerate)

    # print(start_id)
    # print(stop_id)

    sample_pieces = arange(start_id, stop_id + 1, 1)
    # print(sample_pieces)

    newL_piece = newL[sample_pieces]
    newR_piece = newR[sample_pieces]

    for i in range(amount):
```

```
      newL = insert(newL, start_id, newL_piece)
      newR = insert(newR, start_id, newR_piece)

   new = vstack((newL, newR))

   outfile.writeframes(int16(new.reshape(-1, order='F') *
32767.0).tobytes())
   outfile.close()
   print('Repeat - Gotowe!')
# ------------------------------------------------------------------------
-------------------------
```

Kod źródłowy programu bez GUI:

```
# PROCESORY SYGNAŁOWE - PROJEKT
# ETAP 4 - ODDANIE PROJEKTU
# KACPER POŁATAJKO 241603

# Importowanie wszystkich potrzebnych bibliotek
import scipy.io.wavfile
# from numpy import *

from Paulstretch import *
from Reverse import *
from Change_speed import *
from Repeat import *
from Amplify import *

#------------------------------------------------------------------------
-------------------------
# Funkcja wczytująca plik wave i pobierająca z niego dane (częstotliwość
próbkowania i próbki).
#========================================================================
====================
def load_wav_file(filename):
   try:
      wavedata = scipy.io.wavfile.read(filename)
      samplerate = int(wavedata[0])
      samples = wavedata[1] * (1.0 / 32768.0) # 1/(2^15)
      samples = samples.transpose()
      if len(samples.shape) == 1:  # jeśli plik mono to przekonwertuj do
stereo
         samples = tile(samples, (2, 1))
      return (samplerate, samples)
   except:
      print("Błąd wczytywania pliku wave: " + filename)
      return None
#------------------------------------------------------------------------
-------------------------



#------------------------------------------------------------------------
-------------------------
# Funkcja wyświetlająca menu główne.
#====================================
def print_menu_1():
   print('\n_____________| MENU |_____________')
   print('[1] - Podaj nazwę pliku wave do edycji')
   print('[2] - Wyjdź')
#------------------------------------------------------------------------
```

```
------------------------


#-------------------------------------------------------------------------
------------------------
# Funkcja wyświetlająca menu efektów (po dodaniu pliku wave do edycji).
#=========================================================================
def print_menu_2():
   print('\n__________| EFEKTY |__________')
   print('[1] - Paulstretch')
   print('[2] - Reverse')
   print('[3] - Change Speed')
   print('[4] - Amplify')
   print('[5] - Repeat')
   print('[6] - Cofnij')
#-------------------------------------------------------------------------
------------------------


#-------------------------------------------------------------------------
------------------------
# Główna pętla programu.
#=======================
while True:
   print_menu_1()
   choice_1 = input('Wybierz: ')
   choice_1 = int(choice_1)

   if choice_1 == 1:
      print('\nWprowadź nazwę pliku wave')
      name_wav = input('Podaj nazwę: ')

      if load_wav_file(name_wav) == None:
         # print('Nie udało się wczytać pliku wave')
         choice_2 = False
      else:
         print('Wczytano plik: '+ name_wav)
         choice_2 = True

      print('\nWybierz sposób zapisu pliku:')
      print('[1] - Nadpisywanie pliku wejściowego')
      print('[2] - Nowy plik na każdy wybrany efekt')
      saving_choice = input('Wybierz: ')
      saving_choice = int(saving_choice)

      # Pętla efektów
      while choice_2:
         print_menu_2()
         choice_3 = input('Wybierz: ')
         choice_3 = int(choice_3)

         if choice_3 == 1:
            print('\nWybrano efekt: Paulstretch')
            (samplerate, samples) = load_wav_file(name_wav)

            if saving_choice == 1:
               new_name_wav = name_wav
            else:
               new_name_wav = 'Paulstretch_' + name_wav

            stretch, window_size = input('Wprowadź współczynnik rozciągania
```

```
i '
                                        'rozdzielczość czasu (sekundy):
').split(',')
            print('\nWspółczynnik rozciągania: ', stretch)
            print('Rozdzielczość czasu: ' + window_size + '\n' )
            stretch = float(stretch)
            window_size = float(window_size)

            paulstretch(samplerate, samples, stretch, window_size,
new_name_wav)

         elif choice_3 == 2:
            print('\nWybrano efekt: Reverse')
            (samplerate, samples) = load_wav_file(name_wav)

            if saving_choice == 1:
               new_name_wav = name_wav
            else:
               new_name_wav = 'Reverse_' + name_wav

            reverse(samplerate, samples, new_name_wav)

         elif choice_3 == 3:
            print('\nWybrano efekt: Change Speed')
            (samplerate, samples) = load_wav_file(name_wav)

            if saving_choice == 1:
               new_name_wav_1 = name_wav
               new_name_wav_2 = name_wav
            else:
               new_name_wav_1 = 'ChangeSpeed1_' + name_wav
               new_name_wav_2 = 'ChangeSpeed2_' + name_wav

            multiplier = input('Wprowadź wartość mnożnika prędkości: ')
            multiplier = float(multiplier)

            print('\nKtóry parametr uchronić przed edycją?')
            print('[1] - Częstotliwość próbkowania')
            print('[2] - Ilość próbek')
            chsp_choice = input('Wybierz: ')
            chsp_choice = int(chsp_choice)

            if chsp_choice == 1:
               change_speed_1(samplerate, samples, multiplier,
new_name_wav_1)
            elif chsp_choice == 2:
               change_speed_2(samplerate, samples, multiplier,
new_name_wav_2)
            else:
               print('Podano zły numer. Spróbuj jeszcze raz.')

         elif choice_3 == 4:
            print('\nWybrano efekt: Amplify')
            (samplerate, samples) = load_wav_file(name_wav)

            if saving_choice == 1:
               new_name_wav = name_wav
            else:
               new_name_wav = 'Amplify_' + name_wav

            dB = input('Wprowadź wartość wzmocnienia (dB): ')
```

```
            dB = float(dB)

            amplify(samplerate, samples, dB, new_name_wav)

         elif choice_3 == 5:
            print('\nWybrano efekt: Repeat')
            (samplerate, samples) = load_wav_file(name_wav)

            if saving_choice == 1:
               new_name_wav = name_wav
            else:
               new_name_wav = 'Repeat_' + name_wav

            start_time, stop_time = input('Podaj czas początku i końca
fragmentu do powtórzenia (sekundy): ').split(',')
            start_time = float(start_time)
            stop_time = float(stop_time)

            amount = input('Ile razy powtórzyć wybrany fragment?: ')
            amount = int(amount)

            repeat_sample(samplerate, samples, start_time, stop_time,
amount, new_name_wav)

         elif choice_3 == 6:
            print('\nWybrano Cofnij...')
            choice_2 = False

   elif choice_1 == 2:
      print('\nZamykanie...')
      break

   else:
      print('Podano zły numer. Spróbuj jeszcze raz.\n')
# ------------------------------------------------------------------------
-------------------------
```

Kod źródłowy programu z GUI:

```
# PROCESORY SYGNAŁOWE - PROJEKT
# ETAP 4 - ODDANIE PROJEKTU
# KACPER POŁATAJKO 241603

from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QMessageBox
import scipy.io.wavfile
import sys

from Paulstretch import *
from Reverse import *
from Change_speed import *
from Repeat import *
from Amplify import *

#-------------------------------------------------------------------------
------------------------
# Funkcja wczytująca plik wave i pobierająca z niego dane (częstotliwość
próbkowania i próbki).
#=========================================================================
====================
```

```
def load_wav_file(filename):
   try:
      wavedata = scipy.io.wavfile.read(filename)
      samplerate = int(wavedata[0])
      samples = wavedata[1] * (1.0 / 32768.0) # 1/(2^15)
      samples = samples.transpose()
      if len(samples.shape) == 1:  # jeśli plik mono to przekonwertuj do
stereo
         samples = tile(samples, (2, 1))
      return (samplerate, samples)
   except:
      # print("Błąd wczytywania pliku wave: " + filename)
      return None
#-------------------------------------------------------------------------

#-------------------------------------------------------------------------
# Funkcja dla przycisku "Wyjdź" - wyjście z programu
#=================================================
def exit_prog():
    sys.exit(0)
#-------------------------------------------------------------------------

#-------------------------------------------------------------------------
# Klasa opisująca wszytskie potrzebne elementy GUI oraz
# inicjalizująca wszystkie potrzebne do działania funkcje.
#=========================================================
class Ui_MainWindow(object):
    # Inicjalizacja wszystkich elementów GUI (tak jak zostały
zaprojektowane w QtDesignerze)
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("Procesor dźwięku")
        MainWindow.resize(341, 747)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")

        self.groupBox_6 = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_6.setGeometry(QtCore.QRect(10, 90, 321, 461))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.groupBox_6.setFont(font)
        self.groupBox_6.setObjectName("groupBox_6")

        self.groupBox_9 = QtWidgets.QGroupBox(self.groupBox_6)
        self.groupBox_9.setGeometry(QtCore.QRect(0, 110, 321, 71))
        font = QtGui.QFont()
        font.setPointSize(11)
        self.groupBox_9.setFont(font)
        self.groupBox_9.setObjectName("groupBox_9")

        self.button_reverse = QtWidgets.QRadioButton(self.groupBox_9)
        self.button_reverse.setGeometry(QtCore.QRect(10, 30, 151, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.button_reverse.setFont(font)
        self.button_reverse.setObjectName("button_reverse")

        self.groupBox_7 = QtWidgets.QGroupBox(self.groupBox_6)
        self.groupBox_7.setGeometry(QtCore.QRect(0, 30, 321, 81))
        font = QtGui.QFont()
        font.setPointSize(11)
        self.groupBox_7.setFont(font)
```

```
        self.groupBox_7.setObjectName("groupBox_7")

        self.button_paulstretch = QtWidgets.QRadioButton(self.groupBox_7)
        self.button_paulstretch.setGeometry(QtCore.QRect(10, 40, 121, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.button_paulstretch.setFont(font)
        self.button_paulstretch.setObjectName("button_paulstretch")

        self.label_5 = QtWidgets.QLabel(self.groupBox_7)
        self.label_5.setGeometry(QtCore.QRect(230, 10, 81, 41))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.label_5.setFont(font)
        self.label_5.setAlignment(QtCore.Qt.AlignCenter)
        self.label_5.setWordWrap(True)
        self.label_5.setObjectName("label_5")

        self.spinbox_paulstretch_wsp =
QtWidgets.QDoubleSpinBox(self.groupBox_7)
        self.spinbox_paulstretch_wsp.setGeometry(QtCore.QRect(140, 50, 61,
22))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.spinbox_paulstretch_wsp.setFont(font)
        self.spinbox_paulstretch_wsp.setDecimals(3)
        self.spinbox_paulstretch_wsp.setSingleStep(0.5)

self.spinbox_paulstretch_wsp.setObjectName("spinbox_paulstretch_wsp")

        self.label_6 = QtWidgets.QLabel(self.groupBox_7)
        self.label_6.setGeometry(QtCore.QRect(130, 10, 81, 41))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.label_6.setFont(font)
        self.label_6.setAlignment(QtCore.Qt.AlignCenter)
        self.label_6.setWordWrap(True)
        self.label_6.setObjectName("label_6")

        self.spinbox_paulstretch_roz =
QtWidgets.QDoubleSpinBox(self.groupBox_7)
        self.spinbox_paulstretch_roz.setGeometry(QtCore.QRect(240, 50, 61,
22))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.spinbox_paulstretch_roz.setFont(font)
        self.spinbox_paulstretch_roz.setDecimals(3)
        self.spinbox_paulstretch_roz.setSingleStep(0.05)

self.spinbox_paulstretch_roz.setObjectName("spinbox_paulstretch_roz")

        self.groupBox_10 = QtWidgets.QGroupBox(self.groupBox_6)
        self.groupBox_10.setGeometry(QtCore.QRect(0, 270, 321, 81))
        font = QtGui.QFont()
        font.setPointSize(11)
        self.groupBox_10.setFont(font)
        self.groupBox_10.setObjectName("groupBox_10")

        self.spinbox_amplify_wzm =
QtWidgets.QDoubleSpinBox(self.groupBox_10)
        self.spinbox_amplify_wzm.setGeometry(QtCore.QRect(230, 50, 71, 21))
```

```
        font = QtGui.QFont()
        font.setPointSize(10)
        self.spinbox_amplify_wzm.setFont(font)
        self.spinbox_amplify_wzm.setDecimals(3)
        self.spinbox_amplify_wzm.setMinimum(-50.0)
        self.spinbox_amplify_wzm.setMaximum(50.0)
        self.spinbox_amplify_wzm.setSingleStep(0.5)
        self.spinbox_amplify_wzm.setObjectName("spinbox_amplify_wzm")
        self.button_amplify = QtWidgets.QRadioButton(self.groupBox_10)
        self.button_amplify.setGeometry(QtCore.QRect(10, 40, 161, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.button_amplify.setFont(font)
        self.button_amplify.setObjectName("button_amplify")

        self.label_8 = QtWidgets.QLabel(self.groupBox_10)
        self.label_8.setGeometry(QtCore.QRect(200, 10, 121, 41))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.label_8.setFont(font)
        self.label_8.setAlignment(QtCore.Qt.AlignCenter)
        self.label_8.setWordWrap(True)
        self.label_8.setObjectName("label_8")

        self.groupBox_8 = QtWidgets.QGroupBox(self.groupBox_6)
        self.groupBox_8.setGeometry(QtCore.QRect(0, 180, 321, 91))
        font = QtGui.QFont()
        font.setPointSize(11)
        self.groupBox_8.setFont(font)
        self.groupBox_8.setObjectName("groupBox_8")

        self.spinbox_speed_mnoz = QtWidgets.QDoubleSpinBox(self.groupBox_8)
        self.spinbox_speed_mnoz.setGeometry(QtCore.QRect(230, 50, 71, 21))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.spinbox_speed_mnoz.setFont(font)
        self.spinbox_speed_mnoz.setDecimals(3)
        self.spinbox_speed_mnoz.setMinimum(0.0)
        self.spinbox_speed_mnoz.setMaximum(5.0)
        self.spinbox_speed_mnoz.setSingleStep(0.5)
        self.spinbox_speed_mnoz.setObjectName("spinbox_speed_mnoz")

        self.button_speed = QtWidgets.QRadioButton(self.groupBox_8)
        self.button_speed.setGeometry(QtCore.QRect(10, 50, 201, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.button_speed.setFont(font)
        self.button_speed.setObjectName("button_speed")

        self.label_7 = QtWidgets.QLabel(self.groupBox_8)
        self.label_7.setGeometry(QtCore.QRect(230, 10, 71, 41))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.label_7.setFont(font)
        self.label_7.setAlignment(QtCore.Qt.AlignCenter)
        self.label_7.setWordWrap(True)
        self.label_7.setObjectName("label_7")

        self.checkBox_speed_1 = QtWidgets.QCheckBox(self.groupBox_8)
        self.checkBox_speed_1.setGeometry(QtCore.QRect(10, 30, 81, 17))
        font = QtGui.QFont()
```

```
        font.setPointSize(10)
        self.checkBox_speed_1.setFont(font)
        self.checkBox_speed_1.setObjectName("checkBox_speed_1")

        self.checkBox_speed_2 = QtWidgets.QCheckBox(self.groupBox_8)
        self.checkBox_speed_2.setGeometry(QtCore.QRect(100, 30, 81, 17))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.checkBox_speed_2.setFont(font)
        self.checkBox_speed_2.setObjectName("checkBox_speed_2")

        self.groupBox_11 = QtWidgets.QGroupBox(self.groupBox_6)
        self.groupBox_11.setGeometry(QtCore.QRect(0, 350, 321, 111))
        font = QtGui.QFont()
        font.setPointSize(11)
        self.groupBox_11.setFont(font)
        self.groupBox_11.setObjectName("groupBox_11")

        self.label = QtWidgets.QLabel(self.groupBox_11)
        self.label.setGeometry(QtCore.QRect(90, 10, 61, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.label.setFont(font)
        self.label.setAlignment(QtCore.Qt.AlignCenter)
        self.label.setObjectName("label")

        self.label_2 = QtWidgets.QLabel(self.groupBox_11)
        self.label_2.setGeometry(QtCore.QRect(170, 10, 61, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.label_2.setFont(font)
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setObjectName("label_2")

        self.spinbox_repeat_start =
QtWidgets.QDoubleSpinBox(self.groupBox_11)
        self.spinbox_repeat_start.setGeometry(QtCore.QRect(90, 40, 61, 22))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.spinbox_repeat_start.setFont(font)
        self.spinbox_repeat_start.setDecimals(3)
        self.spinbox_repeat_start.setMaximum(99999.99)
        self.spinbox_repeat_start.setSingleStep(0.1)
        self.spinbox_repeat_start.setObjectName("spinbox_repeat_start")

        self.spinbox_repeat_stop =
QtWidgets.QDoubleSpinBox(self.groupBox_11)
        self.spinbox_repeat_stop.setGeometry(QtCore.QRect(170, 40, 61, 22))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.spinbox_repeat_stop.setFont(font)
        self.spinbox_repeat_stop.setDecimals(3)
        self.spinbox_repeat_stop.setMinimum(0.0)
        self.spinbox_repeat_stop.setMaximum(99999.99)
        self.spinbox_repeat_stop.setSingleStep(0.1)
        self.spinbox_repeat_stop.setObjectName("spinbox_repeat_stop")

        self.spinboxone_repeat_powt = QtWidgets.QSpinBox(self.groupBox_11)
        self.spinboxone_repeat_powt.setGeometry(QtCore.QRect(260, 70, 41,
22))
        font = QtGui.QFont()
```

```
        font.setPointSize(10)
        self.spinboxone_repeat_powt.setFont(font)
        self.spinboxone_repeat_powt.setObjectName("spinboxone_repeat_powt")

        self.label_3 = QtWidgets.QLabel(self.groupBox_11)
        self.label_3.setGeometry(QtCore.QRect(250, 30, 61, 41))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.label_3.setFont(font)
        self.label_3.setAcceptDrops(False)
        self.label_3.setAlignment(QtCore.Qt.AlignCenter)
        self.label_3.setWordWrap(True)
        self.label_3.setObjectName("label_3")

        self.button_repeat = QtWidgets.QRadioButton(self.groupBox_11)
        self.button_repeat.setGeometry(QtCore.QRect(10, 70, 201, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.button_repeat.setFont(font)
        self.button_repeat.setObjectName("button_repeat")

        self.groupBox_4 = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_4.setGeometry(QtCore.QRect(10, 0, 321, 81))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.groupBox_4.setFont(font)
        self.groupBox_4.setObjectName("groupBox_4")

        self.label_4 = QtWidgets.QLabel(self.groupBox_4)
        self.label_4.setGeometry(QtCore.QRect(60, 10, 211, 31))
        font = QtGui.QFont()
        font.setPointSize(11)
        self.label_4.setFont(font)
        self.label_4.setAlignment(QtCore.Qt.AlignCenter)
        self.label_4.setObjectName("label_4")

        self.filename_to_edit = QtWidgets.QPlainTextEdit(self.groupBox_4)
        self.filename_to_edit.setGeometry(QtCore.QRect(60, 40, 211, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.filename_to_edit.setFont(font)
        self.filename_to_edit.setObjectName("filename_to_edit")

        self.groupBox = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox.setGeometry(QtCore.QRect(10, 560, 321, 101))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.groupBox.setFont(font)
        self.groupBox.setObjectName("groupBox")

        self.button_save_many = QtWidgets.QRadioButton(self.groupBox)
        self.button_save_many.setGeometry(QtCore.QRect(10, 60, 291, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.button_save_many.setFont(font)
        self.button_save_many.setObjectName("button_save_many")
        # self.button_save_many.toggled.connect(self.btnstate)

        self.button_save_one = QtWidgets.QRadioButton(self.groupBox)
        self.button_save_one.setGeometry(QtCore.QRect(10, 30, 291, 31))
        font = QtGui.QFont()
```

```
        font.setPointSize(10)
        self.button_save_one.setFont(font)
        self.button_save_one.setObjectName("button_save_one")
        # self.button_save_one.toggled.connect(self.btnstate)

        self.button_apply = QtWidgets.QPushButton(self.centralwidget)
        self.button_apply.setGeometry(QtCore.QRect(10, 670, 111, 31))
        font = QtGui.QFont()
        font.setPointSize(11)
        self.button_apply.setFont(font)
        self.button_apply.setObjectName("button_apply")
        self.button_apply.clicked.connect(self.exec_prog)

        self.button_exit = QtWidgets.QPushButton(self.centralwidget)
        self.button_exit.setGeometry(QtCore.QRect(220, 670, 111, 31))
        font = QtGui.QFont()
        font.setPointSize(11)
        self.button_exit.setFont(font)
        self.button_exit.setDefault(False)
        self.button_exit.setObjectName("button_exit")
        self.button_exit.clicked.connect(lambda: exit_prog())

        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.label_9 = QtWidgets.QLabel(self.centralwidget)
        self.label_9.setGeometry(QtCore.QRect(10, 710, 321, 21))
        font = QtGui.QFont()
        font.setBold(False)
        font.setWeight(50)
        self.label_9.setFont(font)
        self.label_9.setObjectName("label_9")
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    # Funkcja wyświetlająca okno powiadomienia o poprawnym wykonaniu edycji
dźwięku
    def show_popup_ok(self):
        msg = QMessageBox()
        msg.setIcon(QMessageBox.Information)
        msg.setText("Gotowe!")
        msg.setInformativeText("Przetwarzanie dźwięku zakończone
sukcesem.")
        msg.setWindowTitle("Procesor dźwięku - info")
        msg.exec_()

    # Funkcja wyświetlająca okno błędnego wyboru sposobu zapisu plików.
    def show_popup_error_save(self):
        msg = QMessageBox()
        msg.setIcon(QMessageBox.Warning)
        msg.setText("Błąd wyboru zapisu plików!")
        msg.setInformativeText("Nie wybrano opcji zapisu dźwięku.\nSpróbuj
ponownie.")
        msg.setWindowTitle("Procesor dźwięku - błąd")
```

```
        msg.exec_()

    # Funkcja wyświetlająca okno błędnego wyboru pliku dźwiękowego wave.
    def show_popup_error_file(self):
        msg = QMessageBox()
        msg.setIcon(QMessageBox.Warning)
        msg.setText("Błąd wyboru pliku dźwiękowego wave!")
        msg.setInformativeText("Wybrany plik nie istnieje lub wpisano
błędną nazwę.\nSpróbuj ponownie.")
        msg.setWindowTitle("Procesor dźwięku - błąd")
        msg.exec_()

    # Funkcja wyświetlająca okno błędnego wyboru efektu dźwiękowego.
    def show_popup_error_effects(self):
        msg = QMessageBox()
        msg.setIcon(QMessageBox.Warning)
        msg.setText("Błąd wyboru efektów dźwiękowych!")
        msg.setInformativeText("Nie wybrano żadnego efektu
dźwiękowego.\nSpróbuj ponownie.")
        msg.setWindowTitle("Procesor dźwięku - błąd")
        msg.exec_()

    # Funkcja inicjalizująca zapewnienie całej funkcjonalności programu
przy pomocy GUI
    def exec_prog(self):
        effect_param = [0, 0, 0, 0, 0]

        # Wybór opcji zapisu
        if self.button_save_one.isChecked():
            save_param = 1
        elif self.button_save_many.isChecked():
            save_param = 2
        else:
            self.show_popup_error_save()
            return

        # Wybór efektów
        if self.button_paulstretch.isChecked():
            effect_param[0] = 1
            window_size = self.spinbox_paulstretch_roz.value()
            stretch = self.spinbox_paulstretch_wsp.value()
        if self.button_reverse.isChecked():
            effect_param[1] = 1
        if self.button_speed.isChecked():
            effect_param[2] = 1
            multiplier = self.spinbox_speed_mnoz.value()
            chsp_choice_1 = self.checkBox_speed_1.isChecked()
            chsp_choice_2 = self.checkBox_speed_2.isChecked()
        if self.button_amplify.isChecked():
            effect_param[3] = 1
            dB = self.spinbox_amplify_wzm.value()
        if self.button_repeat.isChecked():
            effect_param[4] = 1
            start_time = self.spinbox_repeat_start.value()
            stop_time = self.spinbox_repeat_stop.value()
            amount = self.spinboxone_repeat_powt.value()
        elif (self.button_paulstretch.isChecked() or
self.button_reverse.isChecked() or self.button_speed.isChecked()
              or self.button_amplify.isChecked() or
self.button_repeat.isChecked()) == False:
            self.show_popup_error_effects()
```

```
            return

        # Wybór pliku do edycji
        name_wav = self.filename_to_edit.toPlainText()
        if load_wav_file(name_wav) == None:
            self.show_popup_error_file()
            return

        if effect_param[0] == 1:
            (samplerate, samples) = load_wav_file(name_wav)
            if save_param == 1:
               new_name_wav = name_wav
            elif save_param == 2:
               new_name_wav = 'Paulstretch_' + name_wav
            paulstretch(samplerate, samples, stretch, window_size,
new_name_wav)

        if effect_param[1] == 1:
            (samplerate, samples) = load_wav_file(name_wav)
            if save_param == 1:
               new_name_wav = name_wav
            elif save_param == 2:
               new_name_wav = 'Reverse_' + name_wav
            reverse(samplerate, samples, new_name_wav)

        if effect_param[2] == 1:
            (samplerate, samples) = load_wav_file(name_wav)
            if save_param == 1:
               new_name_wav_1 = name_wav
               new_name_wav_2 = name_wav
            elif save_param == 2:
               new_name_wav_1 = 'ChangeSpeed1_' + name_wav
               new_name_wav_2 = 'ChangeSpeed2_' + name_wav
            if chsp_choice_1 == True:
               change_speed_1(samplerate, samples, multiplier,
new_name_wav_1)
            if chsp_choice_2 == True:
               change_speed_2(samplerate, samples, multiplier,
new_name_wav_2)

        if effect_param[3] == 1:
            (samplerate, samples) = load_wav_file(name_wav)
            if save_param == 1:
               new_name_wav = name_wav
            elif save_param == 2:
               new_name_wav = 'Amplify_' + name_wav
            amplify(samplerate, samples, dB, new_name_wav)

        if effect_param[4] == 1:
            (samplerate, samples) = load_wav_file(name_wav)
            if save_param == 1:
               new_name_wav = name_wav
            elif save_param == 2:
               new_name_wav = 'Repeat_' + name_wav
            repeat_sample(samplerate, samples, start_time, stop_time,
amount, new_name_wav)
        self.show_popup_ok()

    # Funkcja nadająca każdemu elementowi GUI nazwę w oknie
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
```

```
        MainWindow.setWindowTitle(_translate("MainWindow", "Procesor
dźwięku"))
        self.groupBox_6.setTitle(_translate("MainWindow", "Efekty"))
        self.groupBox_9.setTitle(_translate("MainWindow", "Reverse"))
        self.button_reverse.setText(_translate("MainWindow", "Odwrócenie w
czasie"))
        self.groupBox_7.setTitle(_translate("MainWindow", "Paulstretch"))
        self.button_paulstretch.setText(_translate("MainWindow", "Efekt
Paustretch"))
        self.label_5.setText(_translate("MainWindow", "Rozdzielczość
czasu"))
        self.label_6.setText(_translate("MainWindow", "Współczynnik
rozciągania"))
        self.groupBox_10.setTitle(_translate("MainWindow", "Amplify"))
        self.button_amplify.setText(_translate("MainWindow", "Wzmocnienie
dźwięku"))
        self.label_8.setText(_translate("MainWindow", "Wartość wzmocnienia
w dB"))
        self.groupBox_8.setTitle(_translate("MainWindow", "Change Speed"))
        self.button_speed.setText(_translate("MainWindow", "Zmiana
prędkości odtwarzania"))
        self.label_7.setText(_translate("MainWindow", "Mnożnik prędkości"))
        self.checkBox_speed_1.setText(_translate("MainWindow", "Metoda I"))
        self.checkBox_speed_2.setText(_translate("MainWindow", "Metoda
II"))
        self.groupBox_11.setTitle(_translate("MainWindow", "Repeat"))
        self.label.setText(_translate("MainWindow", "START"))
        self.label_2.setText(_translate("MainWindow", "STOP"))
        self.label_3.setText(_translate("MainWindow", "Ilość powtórzeń"))
        self.button_repeat.setText(_translate("MainWindow", "Powtórz
zaznaczony fragment"))
        self.groupBox_4.setTitle(_translate("MainWindow", "Plik"))
        self.label_4.setText(_translate("MainWindow", "Podaj nazwę pliku do
edycji"))
        self.groupBox.setTitle(_translate("MainWindow", "Zapis"))
        self.button_save_many.setText(_translate("MainWindow", "Każdy efekt
w osobnym pliku"))
        self.button_save_one.setText(_translate("MainWindow", "Nadpisanie
pliku wejściowego"))
        self.button_apply.setText(_translate("MainWindow", "Zastosuj"))
        self.button_exit.setText(_translate("MainWindow", "Wyjdź"))
        self.label_9.setText(_translate("MainWindow", "Wykonał: Kacper
Połatajko 241603"))
#-------------------------------------------------------------------------
------------------------

#-------------------------------------------------------------------------
------------------------
# Funkcja main programu.
# Inicjalizuje wszystkie zmienne potrzebne do uruchomienia programu z GUI.
#=========================================================================
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
#-------------------------------------------------------------------------
------------------------
```