Dokumentacja Programu : Wykrywacz dźwięków

Autor: Juliusz Stańczyk 107408

1.Temat

a. Program wczytuje dźwięk (.wav, mono) za pomocą okna wyboru plików windows

b. Program pokazuje wykres częstotliwości do amplitudy widma

c. Program pokazuje jaki dźwięk został wybrany (z przedziału 381 – 806 [Hz])

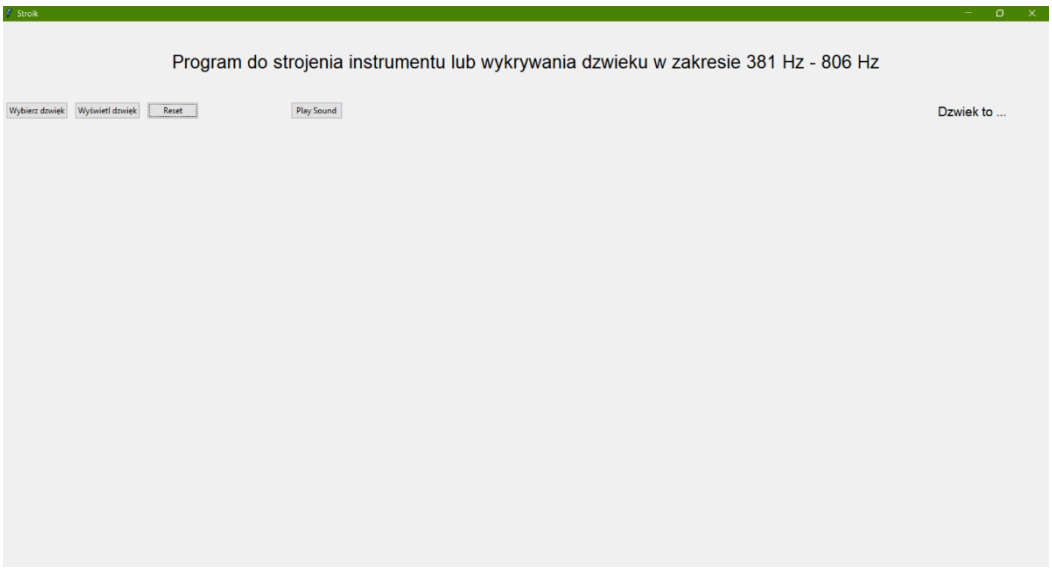
d. Program odtwarza dźwięk za pomocą programu windows

2.Specyfikacja użytkownika

Program tworzony był języku programowania Python 3.7 i tylko na tej wersji był testowany. Środowiskiem programistycznym był PyCharm w wersji 2019.2.3.

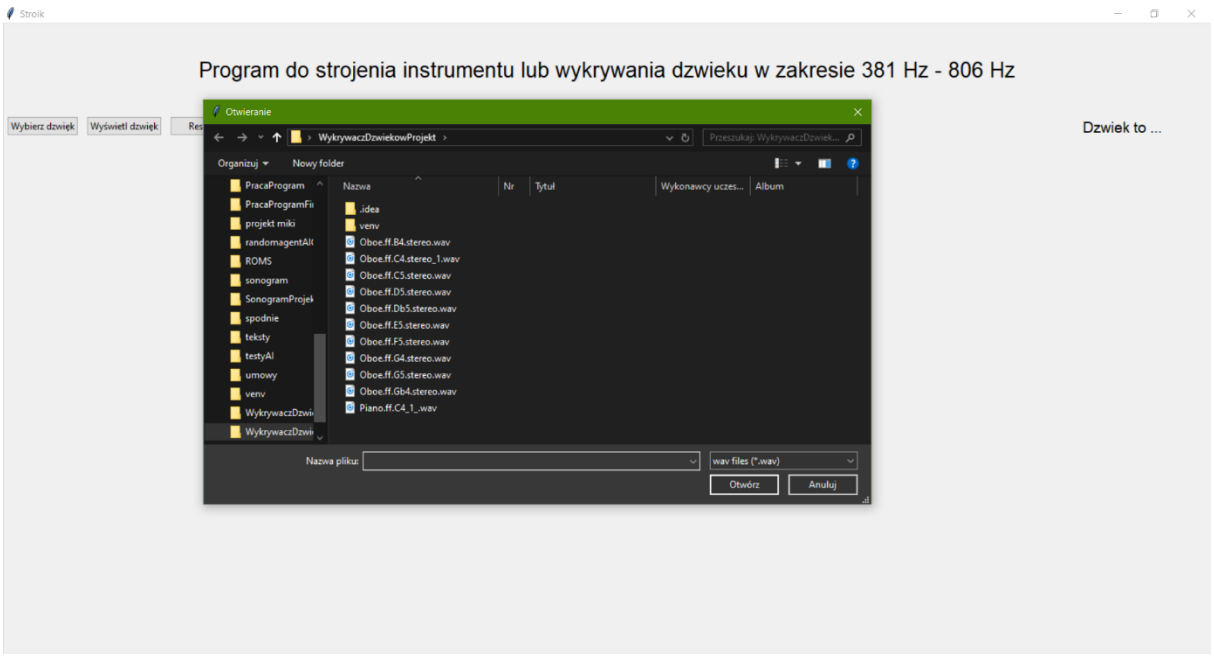
Po wypakowaniu programu należy uruchomić środowisko PyCharm. Następnie File -> Open... -> (wyszukać folder programu o nazwie WykrywaczDzwiekowProjekt) -> OK. Program jest gotowy do skompilowania i uruchomienia. Plikiem głównym programu jest „MainWykrywacz.py”. Plik zawiera również przykładowe dźwięki do użycia.

Po uruchomieniu pojawia się menu:

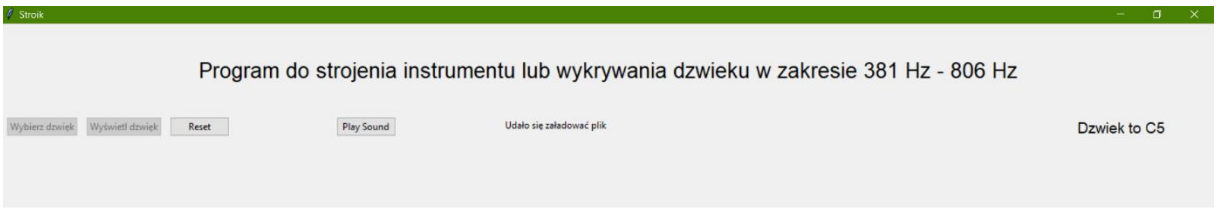


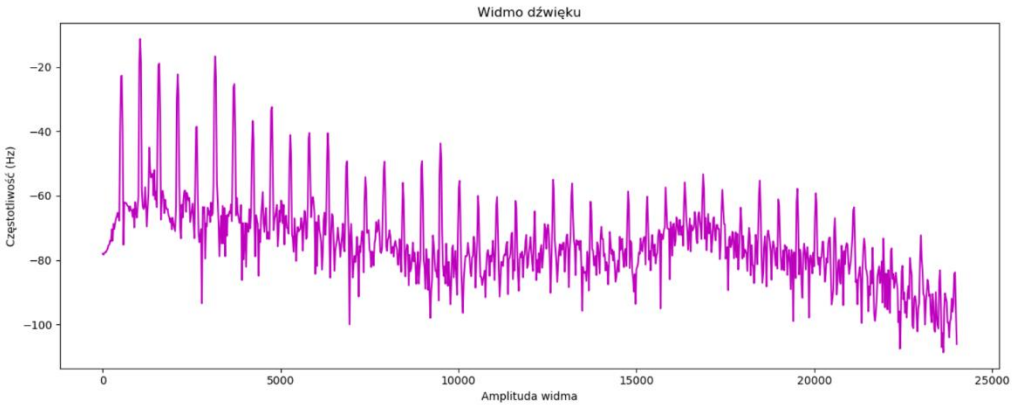
Mamy do wyboru

1. Wybierz dźwięk – otwiera okno wyboru dźwięku (.wav):

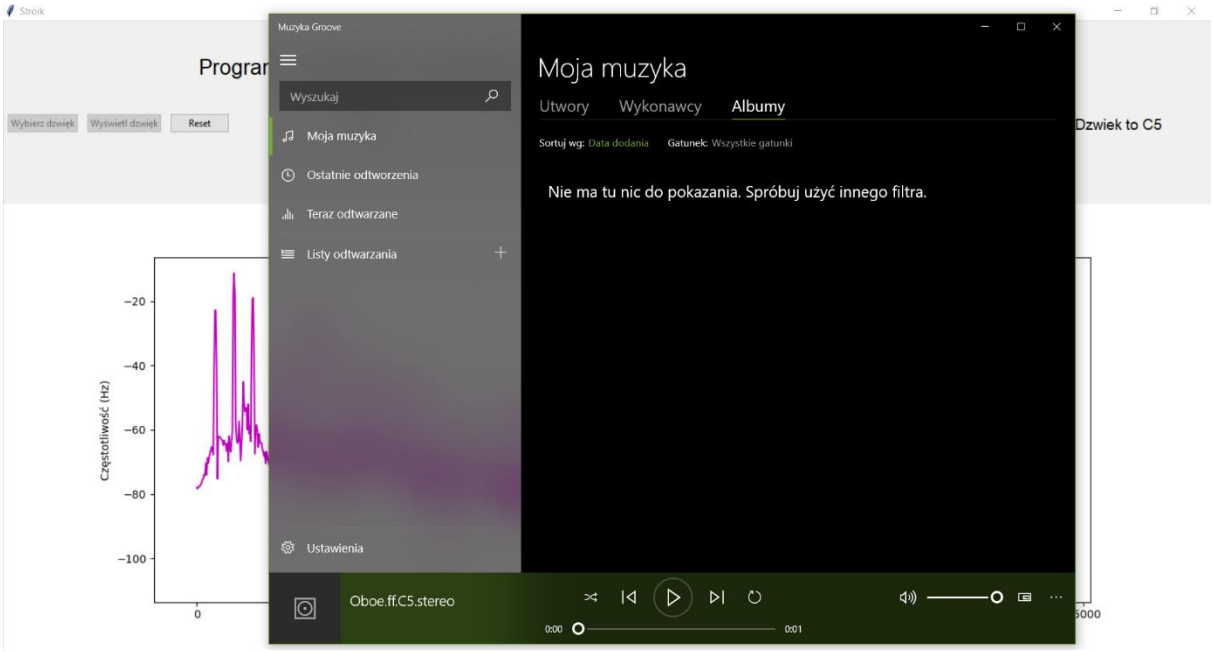


2. Wyświetl dźwięk – wyświetla wykres oraz nazwę dźwięku (po wcześniejszym wybraniu dźwięku):





3. Play sound – wyświetla okno windows do odtwarzania dźwięku (po wcześniejszym wybraniu dźwięku):



3. Kod programu

a. Używane biblioteki

```
import matplotlib

import tkinter as tk
from tkinter import ttk
from tkinter import filedialog as fd

matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

import numpy as np
import matplotlib.pyplot as plot
from scipy.io import wavfile

import os

import sys
import numpy
```

b. Klasa class StartPage

i. funkcja getsnd() – pobiera dźwięk .wav

```
def getsnd(self):
    global filename
    filename = None
```

```
    filename = fd.askopenfilename(filetypes=[("wav files", ".wav")])
    self.button6.config(state="disabled")
    #self.button1.config(state="disabled")
    if(filename != None and filename!= ""):
            self.label8 = tk.Label(self, text="Udało się załadować plik")
            self.label8.pack(anchor="nw",side='left', padx=5)
    else:
            self.label7 = tk.Label(self, text="Nie udało się załadować pliku")
            self.label7.pack(anchor="nw", side='left', padx=5)
```

ii. funkcja MsgDestr() – usuwa napisy i wykres po kliknięciu przycisku Reset

```
def MsgDestr(self):
    self.button6.config(state="enabled")
    #self.button1.config(state="enabled")
    self.button2.config(state="enabled")

    self.label5.destroy()
    self.label7.destroy()
    self.label8.destroy()
    self.button8.destroy()
    self.label4.destroy()
    self.canvas3.get_tk_widget().pack_forget()

    self.label6.config(text="Dzwiek to ...")
```

iii. funkcja Show() – wyświetla wykres i wyszukuje dźwięk. Algorytm wczytuje dźwięk i wyszukuje wartości powyżej progu -45 dB (prążki). Zapisuje je i liczy różnice bieżącej i poprzedniej wartości. Lista przyjmuje wartości 1 na początku prążka i -1 na końcu. Reszta wartości jest 0. Następnie wybieramy pierwszy prążek, bierzemy wartość poprzednią i następną, co tworzy parabole. Znajdujemy jej maksimum z którego liczymy wzorem wartość pomiędzy prążkami w celu dokładniejszego pomiaru. Dodajemy tą wartość do wartości w wcześniejszym indeksie i zmieniamy na częstotliwość w Hz. Następnie funkcja wyświetla odpowiedni tekst w zależności od wyniku. Funkcja tworzy również wykres częstotliwości do amplitudy widma.

```
def Show(self):
    self.button2.config(state="disabled")

    rate, k = wavfile.read(filename)

    k = k[30000:32048]
    k = k / np.max(np.abs(k))
    widmo = 20 * np.log10(np.abs(np.fft.rfft(k * np.hamming(2048))) / 1024)
    f = np.fft.rfftfreq(2048, 1 / 48000)

    self.f2, self.a2 = plot.subplots(1)
    self.a2.clear()
    self.a2.set_title('Widmo dźwięku')
    plot.xlabel("Amplituda widma")
    plot.ylabel("Częstotliwość (Hz)")
    self.a2.plot(f, widmo, color="m")

    self.canvas3 = FigureCanvasTkAgg(self.f2)
```

```
    self.canvas3._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

    nad = (widmo >= -45).astype(np.int)
    pom = np.diff(nad)
    start = np.where(pom == 1)[0] + 1
    end = np.where(pom == -1)[0] + 1

    p = np.argmax(widmo[start[0]:end[0]]) + start[0]
    a, b, c = widmo[p - 1:p + 2]
    k = 0.5 * (a - c) / (a - 2 * b + c)
    maks = (p + k) * rate / 2048

    if maks <= 380:
        self.label6.config(text="Nie można określić dźwięku. Hz poniżej 381")
    if 381 <= maks <= 403:
        self.label6.config(text="Dzwiek to G4")
    if 404 <= maks <= 427:
        self.label6.config(text="Dzwiek to G#4")
    if 428 <= maks <= 452:
        self.label6.config(text="Dzwiek to A4")
    if 453 <= maks <= 479:
        self.label6.config(text="Dzwiek to A#4")
    if 480 <= maks <= 508:
        self.label6.config(text="Dzwiek to B4")
    if 509 <= maks <= 538:
        self.label6.config(text="Dzwiek to C5")
    if 539 <= maks <= 570:
        self.label6.config(text="Dzwiek to C#5")
    if 571 <= maks <= 604:
        self.label6.config(text="Dzwiek to D5")
    if 605 <= maks <= 640:
        self.label6.config(text="Dzwiek to D#5")
    if 641 <= maks <= 678:
        self.label6.config(text="Dzwiek to E5")
    if 679 <= maks <= 718:
        self.label6.config(text="Dzwiek to F5")
    if 719 <= maks <= 761:
        self.label6.config(text="Dzwiek to F#5")
    if 762 <= maks <= 806:
        self.label6.config(text="Dzwiek to G5")
    if maks >= 805:
        self.label6.config(text="Nie można określić dźwięku. Hz powyżej 381")
```

iv. Funkcja Play() – otwiera okno odtwarzacza windows

```
def Play(self):
    os.startfile(filename)
```

v. Funkcja __init__() – tworzy napisy i przyciski

```
def __init__(self, parent, controller):
    tk.Frame.__init__(self, parent)
    label = tk.Label(self, text="Program do strojenia instrumentu lub wykrywania
dzwieku w zakresie 381 Hz - 806 Hz", font=LARGE_FONT)
    label.pack(pady=40, padx=10)
    self.canvas3 = None
    self.label5 = tk.Label()
```

```
    self.label7 = tk.Label()
    self.label8 = tk.Label()
    self.button8 = ttk.Button()
    self.label4 = tk.Label()

    self.button6 = ttk.Button(self, text="Wybierz dzwięk", command=self.getsnd)
    self.button6.pack(anchor="nw", side='left', padx=5)

    #self.button1 = ttk.Button(self, text="Nagraj dziwięk", command=lambda:
self.Record())
    #self.button1.pack(anchor="nw", side='left', padx=5)

    self.button2 = ttk.Button(self, text="Wyświetl dzwięk", command=lambda:
[self.Show()])
    self.button2.pack(anchor="nw", side='left', padx=5)

    self.button3 = ttk.Button(self, text="Reset", command=lambda: self.MsgDestr())
    self.button3.pack(anchor="nw", side='left', padx=5)

    self.label6 = tk.Label(self, text="Dzwiek to ...", font=("Helvetica", 14))
    self.label6.pack(anchor="ne", side='right', padx=60)

    self.button7 = ttk.Button(self, text="Play Sound", command=lambda:
self.Play())
    self.button7.pack(anchor="nw", side='left', padx=130)
```

c. Klasa class PageTwo – nic nie robi, bez niej program nie działał. Wiem gdzie jest błąd, ale chciałem go przebudowywać programu żeby go naprawić.

```
class PageTwo(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
```

d. Klasa class MMain – klasa tworząca okno i stronę.

```
class MMain(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        tk.Tk.wm_title(self, "Stroik")
        tk.Tk.wm_state(self,'zoomed')


        cont = tk.Frame(self)
        cont.pack(side="top", fill="both", expand=True)
        cont.grid_columnconfigure(0, weight=1)
        cont.grid_rowconfigure(0, weight=1)

        self.frames = {}

        for FR in (StartPage, PageTwo):
            frame = FR(cont, self)

            self.frames[FR] = frame

            frame.grid(row=0, column=0, sticky="nsew")
```

```
        self.show_frame(StartPage)

    def show_frame(self, contt):
        frame = self.frames[contt]
        frame.tkraise()
```