Streamowanie dźwięku na Rapsberry Pi – sprawozdanie.

Opis problemu:

Naszym zadaniem było zaimplementowanie prostego programu do streamowania dźwięku, który byłby w bardzo prosty sposób modyfikowalny. Miało to umożliwiać łatwe rozszerzanie go o nowe funkcjonalności.
W implementacji posłużyliśmy się językiem Python, zważywszy na to, że jest preinstalowany na Raspbianie, a jego kod jest łatwy do zrozumienia.

Pierwsza implementacja:

Implementacja prostego streamowania dźwięku w real-time, bez gui i zbędnych modyfikacji.
Do przechwytywania dźwięku z mikrofonu posłużyliśmy się biblioteką pyaudio, która umożliwia obsługę urządzeń nagrywających dźwięk. Pyaudio pozwala dodatkowo na modyfikacje danych zebranych z urządzenia.

Problemy:

Po zaimplementowaniu zwykłego streamowania dźwięku chcieliśmy dodać możliwość zmiany częstotliwości odtwarzanego dźwięku w real-time, do czego okazało się potrzebne użycie algorytmu phase vocode'owania. Niestety po wielu próbach nie udało nam się tego podołać zaimplementowaniu tego algorytmu, zawsze kończyliśmy z szumami, syczeniem bądź buczeniem w otrzymanym outpucie.
Okazało się to dla nas zbyt skomplikowane, nie mając dostatecznych wiadomości o teorii dźwięku pomimo starań nie byliśmy w stanie zlokalizować problemu w naszej implementacji.

Opis algorytmu:

Sam algorytm phase vocodowania polega na skalowaniu częstotliwości dźwięku wraz z przedziałem czasu używając informacji o fazie dźwięku. Główną częścią algorytmu jest STFT, czyli szybka transformata Fouriera. STFT konwertuję przedział czasowy, jego reprezentację na reprezentację czas-częstotliwość. Pozwalając na modyfikacje konkretnych amplitud częstotliwości. Zmiana częstotliwości inputu odbywa się poprzez zmianę pozycji czasowych ramek STFT przed operacją resyntezacji pozwalając na modyfikację time-scale oryginalnego inputu.
Aby zmodyfikować w łatwy sposób dane dostarczone przez mikrofon użyliśmy biblioteki numpy.

Drugie podejście implementacji:

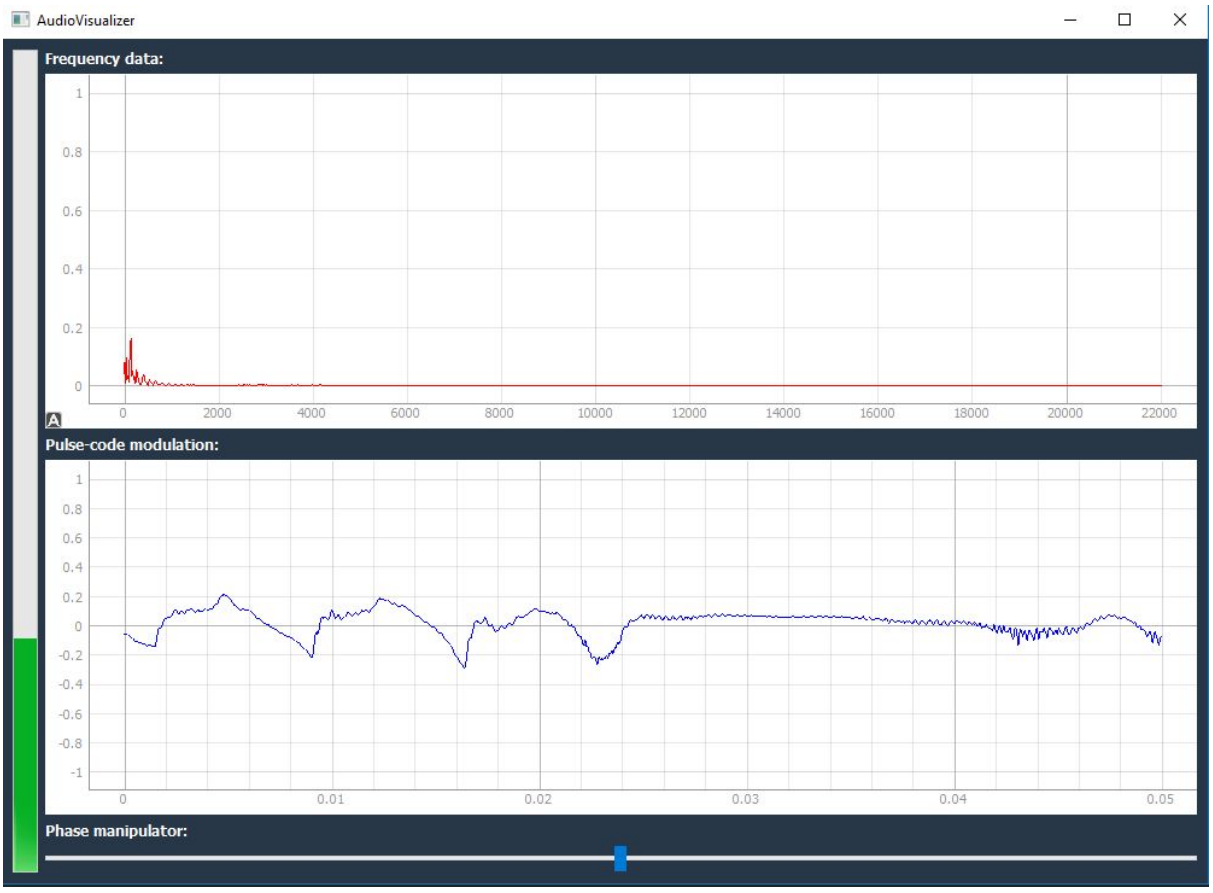
Napotkane problemy wymusiły na nas wykorzystanie gotowej implementacji phase vocodera z biblioteki audiolazy, która udostępnia algorytm phase vocodera. Nie musimy przejmować się poprawnością ramek zaciągniętych z inputu i aby zgrywały się odpowiednio ze sobą.

Gui i możliwość animowania wykresów dała nam biblioteka PyQt4.
Do zaimplementowania Gui posłużyliśmy się pogramem Qt Creator, gdzie mogliśmy w łatwy sposób poprzez gui tworzyć nowe gui. Z wygenerowanego pliku XML za pomocą API PyQt4 mogliśmy wygenerować kod w Pythonie stworzonego GUI i działać na nim w programie.

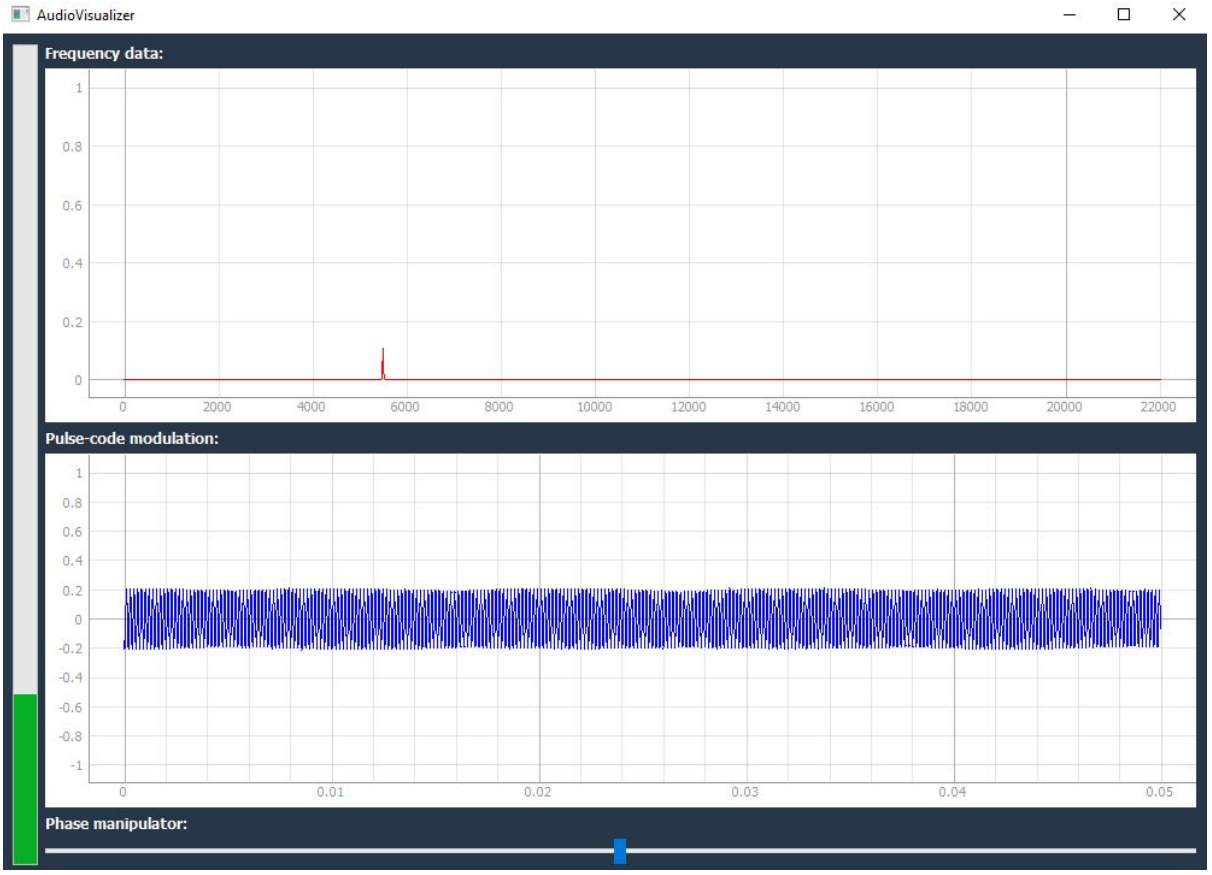
Opis użycia, instrukcja do instalacji bibliotek orazi krótka notka o programie znajdują się na repozytorium github:

https://github.com/mwoss/sound_stream_raspberry_pi

Normalny glos i niski głos:



Stałe częstotliwości:


AudioVisualizer
Frequency data:
1
0.8
0.6
0.4
0.2
0
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
22000
Pulse-code modulation:
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.01
0.02
0.03
0.04
0.05
Phase manipulator:

Muzyka:


AudioVisualizer
Frequency data:
1
0.8
0.6
0.4
0.2
0
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
22000
Pulse-code modulation:
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.01
0.02
0.03
0.04
0.05
Phase manipulator:

Autorzy:
Michał Matusiak, Mateusz Woś

Źródła:

https://en.wikipedia.org/wiki/Phase_vocoder

http://edu.pjwstk.edu.pl/wyklady/mul/scb/main33.html

http://zulko.github.io/blog/2014/03/29/soundstretching-and-pitch-shifting-in-python