# MODradio

Grzegorz Koperwas

14 stycznia 2022

## 1. Temat projektu:

Celem powstałego programu jest strumieniowanie piosenek z `trackerów` poprzez *Http Live Streaming*[1] w celu łatwego odsłuchu na urządzeniach mobilnych.

Zatem musi on dynamicznie łączyć kolejne pliki w jeden strumień, bez wcześniejszego wczytania ich wszystkich (Ilość tych plików wynosi 29 tysięcy, rozmiar około 50 gigabajtów w formie skompresowanej).

### Opis problemu:

Archiwa strony `modarchive.org` są udostępniane w następującej formie:

1. Piosenki znajdują się w drzewie folderów rozróżniającym je ze względu na format, artystę czy rok dodania.

   Jakiekolwiek informacje zawarte w strukturze folderów mają być ignorowane, strumieniujemy piosenki w losowy sposób.

   Odtwarzacz VLC może uzyskiwać dostęp po protokole SMB do serwera z plikami, lecz nie odtwarza ich losowo w prosty sposób.

2. Każda piosenka jest skompresowana jako archiwum `ZIP`.

3. Piosenki są przechowywane jako *moduły trackerów*, gdzie zamiast danych `PCM` przechowywane są w formie sampli i informacji jak je odtwarzać. Wynika to z architektury komputerów *Amiga*, gdzie ten rodzaj muzyki powstał.

Zatem program musi:

1. Przyjmować pliki audio w formie pozwalającej na utrzymywanie *bufora* następnych plików audio.

2. Przyjmować pliki, lub ścieżki do nich ze źródła łatwo dostępnego dla jakiegoś skryptu. Na przykład przez `stdin`.

3. Spełniać wymagania do łatwego zahostowania na moim klastrze *Docker Swarm*, brak GUI itp.

[1] Dalej będę korzystał ze skrótu **HLS**

# 2. Opis pobieranych danych przez program:

Program pobiera przez standardowe wejście ścieżki do kolejnych plików audio. Pliki te są **usuwane** po zakończeniu ich strumieniowania.

Program po napotkaniu pliku, którego nie da się otworzyć lub zdekodować pomija dany plik.

Program stara się otwierać dwa pliki naraz. Plik, który jest właśnie strumieniowany, oraz plik, który jest następny w kolejce.

Program wspiera wiele formatów wejściowych, ich dokładna lista zależy od flag użytych do kompilacji biblioteki ffmpeg. Jeżeli plik wejściowy zawiera więcej niż jeden strumień[2], to program wybiera pierwszy strumień audio.

# 3. Opis otrzymanych rezultatów

## Wydruk z programu

Program wypisuje do konsoli logi diagnostyczne, za przechowywanie ich w plikach odpowiada **systemd** lub **docker**.

Przykładowe logi znajdują się na załączniku 1. Logi w załączniku składają się z paru części:

- Logi ze znakami < oraz >, - Logi informujące jakie obiekty są tworzone przez program. Przykładowo:
  - `<Encoder for aac>` - Stworzono obiekt kodera formatu `AAC`, zawsze wyświetla się na początku programu.
  - `<Reader for /path/to/file>` - Stworzono obiekt demuxera, który czyta zawartość pliku. Jeżeli dany plik zawiera parę strumieni wideo lub audio, program wybiera pierwszy strumień audio. Powinien wspierać większość standardów (pewnie nawet zasoby sieciowe, zależy od wersji biblioteki `libav`).
  - `<Decoder for $codec>` - Stworzono obiekt dekodera kompresji, program *powinien* wspierać wiele różnych kodeków, jednak standard mp3 generuje niepoprawny dźwięk.
  - `<Resampler from $foo to $bar>` - Stworzono obiekt resamplera, który normalizuje częstotliwość próbkowania oraz zapis bitowy.
- Logi z znakami [ oraz ][3], - Logi generowane przez bibliotekę `libav` - są zwykle w formie:

[ $źródło @ adres źródła ] $wiadomość

[2]Na przykład plik wideo będzie zawierał zwykle jedną ścieżkę wideo, jedną lub więcej audio oraz nawet kilkanaście ścieżek napisów, czasami nawet miniaturkę jako jednoklatkowy strumień MJPEG.

[3]te kolorowe

Zwykle są to logi o statusie muxera `HLS`, lecz w przypadku złego pliku wejściowego zawierają one dodatkowe informacje o błędzie. Program `ffmpeg`, który jest frontendem do biblioteki `libav` generuje te same logi, więc jego dokumentacja pomoże w diagnozowaniu problemu.

- Reszta:

  Część logów w przykładzie pochodzi od skryptu realizującego przykładowe wykorzystanie programu. Jest on dołączony do kodu jako `./feeder.sh`.

```
[mpegts @ 0x7f0600355480] frame size not set
<Encoder for aac>
extracted /tmp/modfiles/lazertrack_heaven_2.mod
[aac @ 0x7f05f4005240] Estimating duration from bitrate, this may be inaccurate
<Reader for /tmp/modfiles/lazertrack_heaven_2.mod.aac>
<Decoder for aac>
<Resampler from 44100hz to 44100hz>
[hls @ 0x7f060034de00] Opening 'stream0.ts' for writing
[hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
extracted /tmp/modfiles/the_hardliner_-_whoronzon_gohonzon.xm
[aac @ 0x7f05ec001100] Estimating duration from bitrate, this may be inaccurate
<Reader for /tmp/modfiles/the_hardliner_-_whoronzon_gohonzon.xm.aac>
extracted /tmp/modfiles/the_savannus_never_never.669
[hls @ 0x7f060034de00] Opening 'stream1.ts' for writing
[hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
[hls @ 0x7f060034de00] Opening 'stream2.ts' for writing
[hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
extracted /tmp/modfiles/gustavo6046_-_trulix.it
extracted /tmp/modfiles/jabdah-cover.xm
extracted /tmp/modfiles/jason_ee-futurefuckballs2010_cover.it
extracted /tmp/modfiles/owcfullfrontal.it
[hls @ 0x7f060034de00] Opening 'stream3.ts' for writing
[hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
extracted /tmp/modfiles/skyline_-_boners.it
extracted /tmp/modfiles/ko0x_-_galaxy_guppy.it
[hls @ 0x7f060034de00] Opening 'stream4.ts' for writing
[hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
extracted /tmp/modfiles/pasyada_alex_-_decil.xm
extracted /tmp/modfiles/badboyremixhypnosis.mod
[hls @ 0x7f060034de00] Opening 'stream5.ts' for writing
[hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
Waiting for modradio to pickup data
[hls @ 0x7f060034de00] Opening 'stream6.ts' for writing
[hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
[hls @ 0x7f060034de00] Opening 'stream7.ts' for writing
[hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
...
```

Załącznik 1: Przykładowe logi z programu.

## Pliki tworzone przez program:

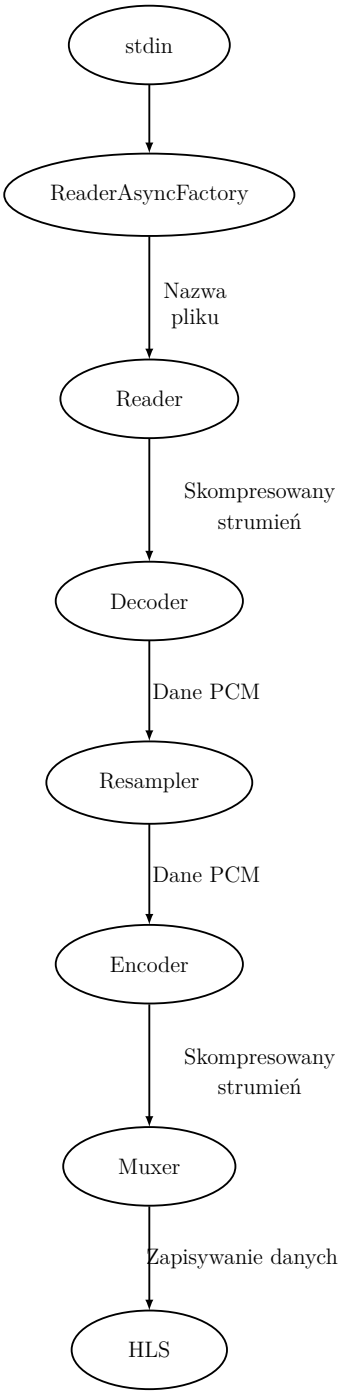
Program tworzy w aktualnym katalogu strumień w formacie `HLS` jako pliki `stream$x.ts` oraz plik „spis” `stream.m3u8`. Te pliki powinny być hostowane przez serwer HTTP, jako pliki statyczne. Programy takie jak *VLC Media Player*, *ffplay*, *safari* czy przeglądarki internetowe na systemie android odtworzą je bez problemu, nawet z dysku. Dla przeglądarek na komputerach PC trzeba dostarczyć demuxer w *javascript'cie*, co nie jest przedmiotem tego projektu.

# 4. Zastosowane algorytmy:

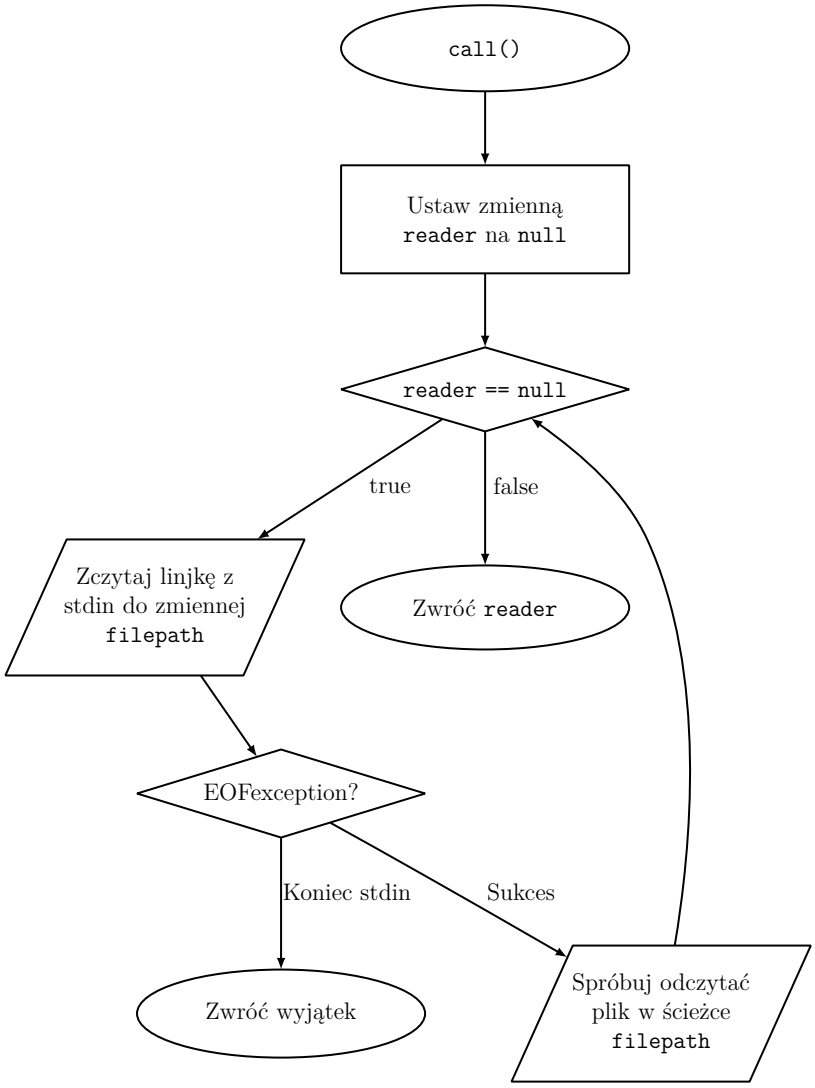
Obieg danych jest przedstawiony na załączniku 2.

W programie został zastosowany mechanizm asynchroniczności poprzez klasę standardową `Future` oraz `ReaderAsyncFactory`, proces ten jest pokazany w załączniku 3.

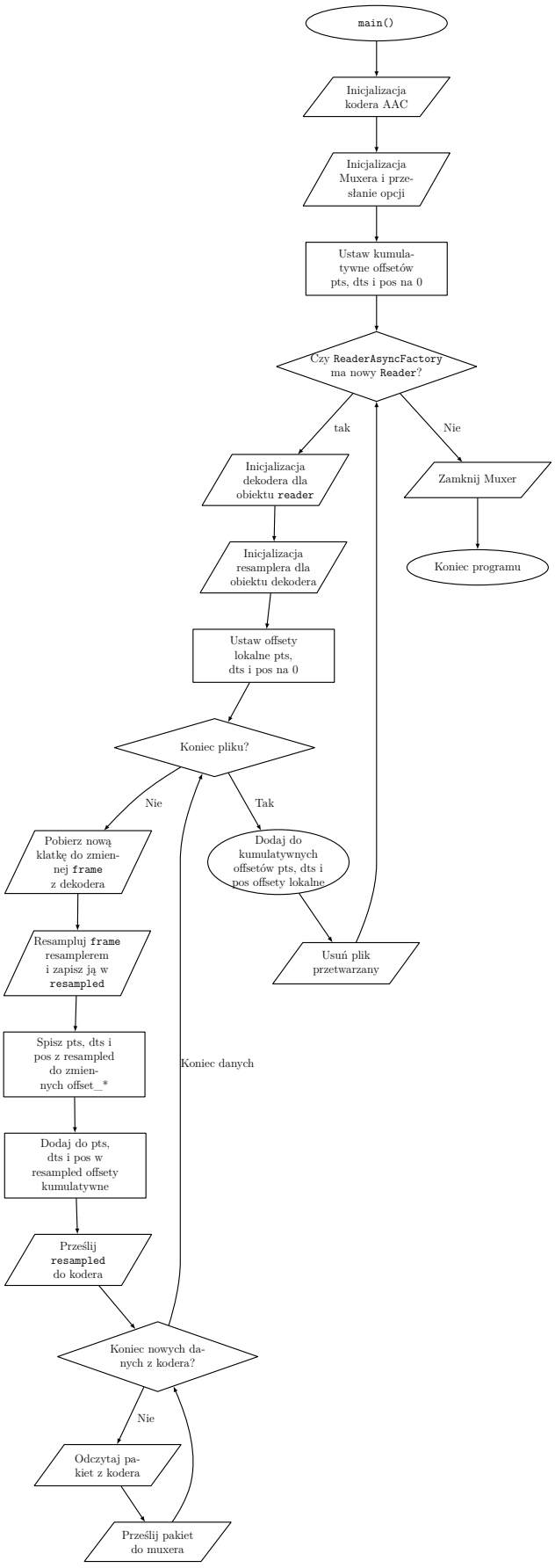
Ogólny schemat blokowy jest pokazany w załączniku 4.




Załącznik 2: Pipeline danych




Załącznik 3: Schemat blokowy działania klasy `ReaderAsyncFactory`




Załącznik 4: Ogólny schemat działania programu.

# 5. Testy na poprawność działania programu:

Poprawność działania była sprawdzana poprzez odsłuch strumieni generowanych przez program oraz hostowanie ich serwerem `http` z biblioteki standardowej języka *Python*, poprzez komendę `python -m http.server`.

# 6. Wnioski:

## 6.1. Znane błędy

Program generuje strumień w tempie leciutko zawyżonym. Podczas dłuższego odsłuch może wystąpić konieczność przewinięcia do przodu strumienia, bo starsze części mogą już nie być dostępne. Wynika to z niejasności dokumentacji `libav` na temat tego, w jakim formacie są dane synchronizujące zawarte w klatkach strumienia. Przykładowo pliki w formacie `mp3` oraz `aac` posiadają kompletnie różne wartości.

Jednym z możliwych rozwiązań tego problemu było by manualne obliczanie długości poszczególnych klatek w resamplerze, w oparciu o ilość próbek (sampli), częstotliwość próbkowania oraz ilość kanałów.

Innym błędem, raczej związanym z poprzednim[4], jest błąd przy odczycie plików w formacie `mp3`, program otwiera je poprawnie, resampluje z 48000hz do 41000hz, koduje do mp3, ale efekt wyjściowy nie przypomina zbytnio pliku wejściowego.

## 6.2. Inne:

Po stworzeniu tego programu lepiej rozumiem architekturę biblioteki `libav`, gdyż wcześniej wykorzystywałem ją tylko do dekodowania audio w poprzednim projekcie z programowania II. w języku C++ oraz w projekcie komercyjnym dla Gliwickiej firmy *APA Group*[5] w języku python.

[4] lub flag kompilacyjnych związanych z `libmp3lame` w bibliotece `libav`, kwestie patentowe komplikują sprawę wsparcia tego kodowania.

[5] Plakat rekrutacyjny „Pracuj w Czarnym Domu” wisi na czwartym piętrze wydziału Matematyki Stosowanej obok schodów.