Odtwarzacz FLAC na płytce STM32F746G-DISCOVERY

Łukasz Jezapkowicz, Maciej Sikora

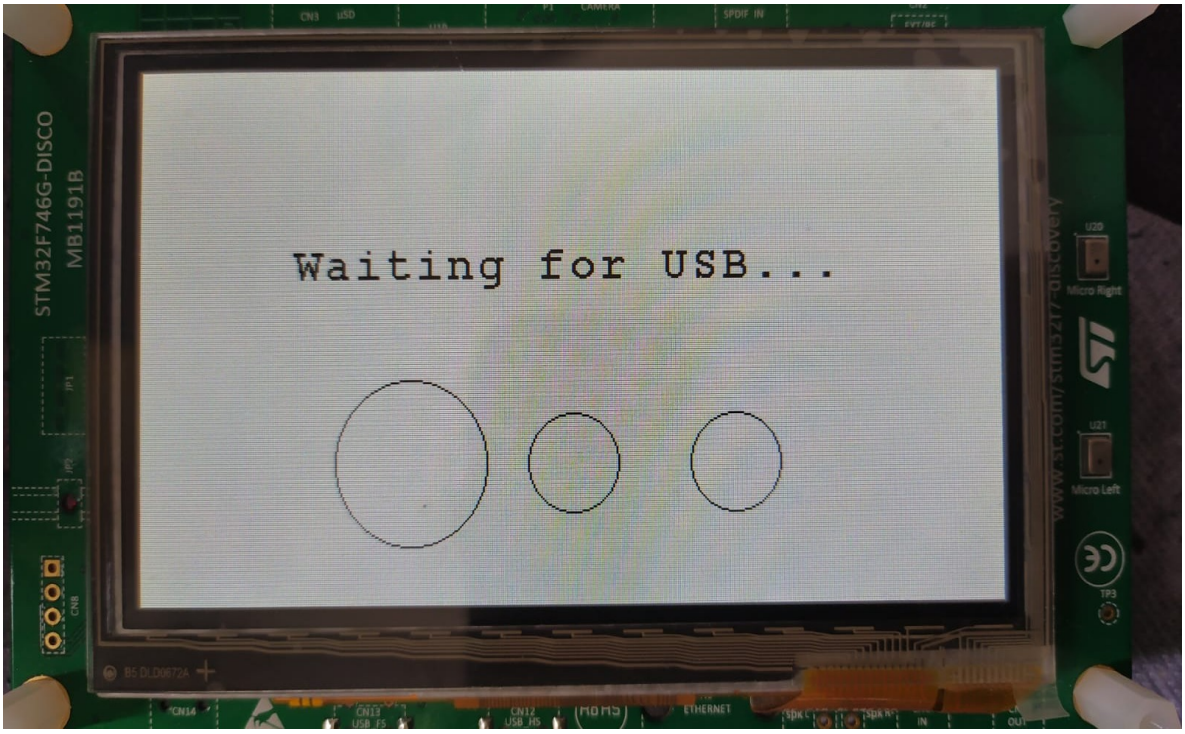
29 maja 2021

GitHub: https://github.com/LucasJezap/EmbeddedSystems

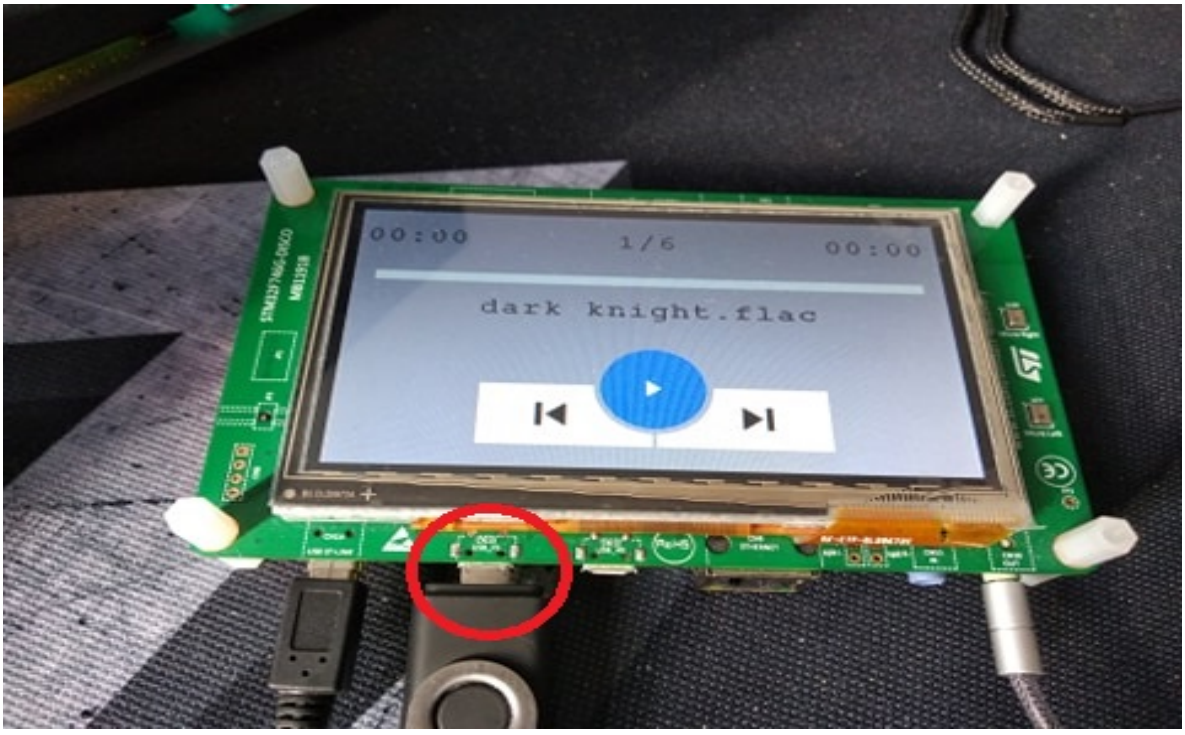
1 Dokumentacja użytkowa

- Przygotuj płytkę STM32F746G-DISCOVERY z wgranym oprogramowaniem odtwarzacza FLAC.
- Aby usłyszeć dźwięk potrzebne jest podłączenie głośników aktywnych lub słuchawek do zielonego wyjścia audio znajdującego się z boku płytki, wtyczką mini jack.

- Upewnij się, że na nośniku USB znajdują się pliki w formacie FLAC i znajdują się one w głównym katalogu.
- Poczekaj, aż zobaczysz ekran oczekiwania na podpięcie USB.



- Podłącz nośnik USB z plikami.
- Poczekaj, aż zobaczysz menu odtwarzacza muzycznego.



- Przyciśnięcie przycisku start/stop, gdy znajduje się na nim symbol trójkąta spowoduje rozpoczęcie odtwarzania pliku, którego nazwa jest wyświetlona pod paskiem postępu odtwarzania.

- Przyciśnięcie przycisków ze strzałkami w prawo/lewo spowoduje zmianę aktualnie odtwarzanego pliku. Odtwarzacz zapętla się.
- Na ekranie widzimy nazwę piosenki, postęp odtwarzania, numer piosenki, aktualny moment piosenki oraz całkowity czas trwania.

2 Dokumentacja techniczna

2.1 Ogólna idea

Dekodowanie pliku rozpoczyna się od odczytania metadanych znajdujących się na początku każdego pliku FLAC. Następnie czytane są ramki, które zawierają informacje o dźwięku. Dekodując plik ramka po ramce jesteśmy w stanie prosto przetworzyć cały plik poprawnie, ponieważ jest to zgodne z API udostępnionym przez bibliotekę libFLAC.

Aby móc płynnie odtwarzać dźwięk z mikroprocesora nie możemy odtwarzać całego pliku bit po bicie, ponieważ operacje które wykonują się w międzyczasie zajmują zbyt dużo czasu. Rozwiązaniem tego problemu jest równoległe wykorzystanie bufora podzielonego na dwie części. [1]

Jeden wątek programu ma dostęp do części **A**, nazwijmy go *writer*, na której zapisuje on kolejne bity zdekodowane z pliku FLAC, a wątek który zajmuje się odtwarzaniem dźwięku, nazwijmy go *player*, ma dostęp do części **B**, z której czyta on dane potrzebne do odtwarzania dźwięku.

W momencie, w którym *writer* kończy zapisywanie w swoim segmencie pozwala na wejście do niego *playerowi*, gdy tamten skończy swoją część i oczekuje na możliwość dostępu do sekcji, w której obecnie pracuje *player*. Proces ten powtarza się, aż do zakończenia pobierania danych z pliku. Należy pamiętać, że w ostatnim cyklu przetwarzania ramki mogą nie zapełnić w całości konkretnej sekcji bufora.

2.2 Struktura programu

- Podstawowa struktura projektu została wygenerowana za pomocą STM32CubeMX.
- Plik Makefile został wygenerowany z open source'owego programu [2] do generacji Makefile dla projektów wygenerowanych przy pomocy STM32CubeMX.
- Pliki nagłówkowe stworzone przez nas zawierają się w folderze Inc z prefiksem **stm32f476g-disco-LJMS-**.
- Pliki źródłowe zawierają się w folderze Src z identycznym prefiksem.

2.3 Najważniejsze funkcje

Poniższy opis zawiera jedynie wyjaśnienie najważniejszych funkcji w naszym projekcie. Uznaliśmy, że przeklejanie kodu źródłowego do dokumentacji jest zbędne i może zmniejszyć przejrzystość instrukcji. Kod źródłowy zawiera odpowiednie komentarze wyjaśniające działanie funkcji. Nazewnictwo plików oraz funkcji ma na celu jasne określenie do czego służy dany plik/funkcja.

- Plik źródłowy flac.c zapewnia interfejs do czytania metadanych oraz kolejnych ramek.
  - Flac *create_new_flac(InputStream *input);
    - Funkcja zajmująca się tworzeniem nowego obiektu *Flac* ze strumienia wejściowego *input*. Funkcja tworzy odpowiednie dekodery FLAC oraz ustawia odpowiednie callbacki (read_callback, write_callback, metadata_callback, error_callback).
  - bool read_flac_metadata(Flac *flac, FlacMetadata *info);
    - Funkcja zajmująca się odczytywaniem metadanych pliku w formacie FLAC do zmiennej $info$.
  - bool read_flac_frame(Flac *flac, FlacFrame **frame);
    - Funkcja zajmująca się odczytywaniem kolejnych ramek pliku w formacie FLAC do zmiennej $frame$.
- Plik źródłowy flac_frame_buffer.c zapewnia interfejs do tworzenia i usuwania buffora, oraz do czytania kolejnych ramek z pliku.
  - int read_flac_buffer(FlacBuffer *self, void *dest, int size);
    - Funkcja zajmująca się odczytywaniem *size* bajtów z bufera FLAC do bufera *dest*. Funkcja odczytuje kolejne ramki, aż do zapełnienia konkretnego segmentu bufora *dest*.

- Plik źródłowy flac_input_stream.c udostępnia interfejs do czytania z InputStream do buffora.
  - int read_input_stream(InputStream *self, void *buf, int len);
    - Funkcja zajmująca się odczytywaniem *len* bajtów ze strumienia wejściowego oraz zapisywaniem ich w buferze buf.
- Plik źródłowy main.c zawiera główną logikę działania, jest punktem startowym naszego programu. W nieskończonej pętli aktualizowany jest stan odtwarzacza FLAC.
- Plik źródłowy flac_player.c zawiera implementację odtwarzania muzyki z plików FLAC.
  - void init_flac_player(void);
    - Funkcja zajmująca się inicjalizacją odtwarzacza plików FLAC. Funkcja ustawia odpowiednią częstotliwość, głośność oraz audio frame slot.
  - void update_flac_player(void);
    - Funkcja zajmująca się aktualizowaniem odtwarzacza plików FLAC, działająca tylko jeżeli utwór jest aktualnie odtwarzany.
  - void flac_player_play(const char *filename);
    - Funkcja zajmująca się rozpoczynaniem odtwarzania utworu.
  - void flac_player_pause(void);
    - Funkcja zajmująca się pauzowaniem odtwarzania utworu.
  - void flac_player_resume(void);
    - Funkcja zajmująca się wznawianiem odtwarzania utworu.
  - void flac_player_stop(void);
    - Funkcja zajmująca się zatrzymywaniem odtwarzania utworu.
- Plik źródłowy flac_screen.c odpowiada za zarządzanie interfejsem graficznym. Główną funkcją jest *render_flac_player*.
  - void render_flac_player(int files_count, int current_file, const char *current_filename, unsigned sample_rate, unsigned current_timer, unsigned max_timer, bool is_playing);
    - Funkcja zajmująca się renderowaniem interfejsu graficznego odtwarzacza. Funkcja przyjmuje odpowiednie parametry wejściowe.
  - static void flip_layer(void);
    - Funkcja zajmująca się zamienianiem aktualnej warstwy wyświetlacza (projekt wykorzystuje podwójne buforowanie [1])
  - void init_screen(void);
    - Funkcja zajmująca się inicjalizacją wyświetlacza mikrokontrolera.
  - static void fill_lcd_polygon(Point position, const Point *points, uint16_t point_count);
    - Funkcja zajmująca się rysowaniem wielokąta z podanych punktów, w podanym miejscu na wyświetlaczu.
  - void display_info_on_screen(const char *info);
    - Funkcja zajmująca się pisaniem tekstu na wyświetlaczu.
  - void render_usb(int which_big);
    - Funkcja zajmująca się renderowaniem interfejsu graficznego oczekiwania na nośnik USB. W tej implementacji na wyświetlaczu widoczne są trzy przechodzące okręgi.
  - void handle_touch(void);
    - Funkcja zajmująca się rozpoznawaniem oraz reagowaniem na dotyk użytkownika na wyświetlaczu.
- Plik źródłowy flac_search_files.c zawiera funkcję, która umożliwia przeglądanie podłączonego USB w poszukiwaniu plików FLAC.
  - void search_for_flac_files(const char *path, Files *files);
    - Funkcja zajmująca się wyszukiwaniem plików w formacie FLAC w folderze $path$, zapisująca je w zmiennej $files$.

2.4 Wprowadzanie własnych zmian, rozwijanie oprogramowania

By wprowadzić własne zmiany, należy zmodyfikować kod źródłowy znajdujący się w plikach z prefiksem **stm32f476g-disco-LJMS-**. W celu zrozumienia działania poszczególnych funkcji stworzonego API zaleca się przeczytanie wcześniejszego podpunktu. Proponowane zmiany/rozwój oprogramowania:

- Zmiana interfejsu graficznego.
- Dodanie funkcjonalności zmiany poziomu głośności odtwarzacza.
- Próba polepszenia algorytmu dekodowania dźwięku, próba uzyskania poprawnie zdekodowanego dźwięku dla plików z bitratem większym niż 700kbps.
- Dodanie możliwości przewijania do konkretnych momentów utworów.
- Dodanie obsługi kart pamięci.

Nie są to oczywiście jedyne możliwe zmiany, jedynie proponowane. Dodatkowe informacje można znaleźć w dokumentacji płytki STM32F746G-DISCOVERY. [3]

3 Dokumentacja procesu realizacji projektu

3.1 Spotkanie 11.03

Spotkanie na MS Teams w sprawie wyboru tematu. Poruszanych było wiele tematów, między innymi odtwarzacz MP3 czy odtwarzacz stacji radiowych. Temat, który ostatecznie został wybrany to Odtwarzacz plików dźwiękowych FLAC na płytce STM32F746G-DISCOVERY.

3.2 Spotkanie 18.03

Spotkanie głównie służyło ustaleniu spraw technicznych takich jak termin odebrania płytki w D17 oraz kontemplacja na temat udźwignięcia przedsięwzięcia przez płytkę STM32F746G-DISCOVERY. Z naszej strony pojawił się również plan (kolejne kroki) pracy nad projektem.

3.3 Spotkanie 8.04

Spotkanie, na którym przedstawiono pierwsze postępy pracy. Na to spotkanie udało się skonfigurować środowisko programistyczne, stworzyć nowy projekt oraz sprawdzić działanie wyświetlacza na przykładzie projektu na tej płytce załączonego w kursie Technika Mikroprocesorowa.

3.4 Spotkanie 22.04

Spotkanie, na którym przedstawiono kolejne postępy pracy prowadzącemu. Projekt poprawnie wyświetla elementy UI oraz oczekuje na podpięcie USB. Dyskusja z prowadzącym na temat testowania odtwarzacza (słuchawki/głośniki aktywne, gdzie podpiąć).

3.5 Spotkanie 6.05

Spotkanie, na którym przedstawiono kolejne postępy pracy. Warstwa graficzna projektu jest już całkowicie gotowa. Mikrokontroler poprawnie działa po podpięciu USB (przed podpięciem - prosty interfejs graficzny oczekiwania na USB, po podpięciu - interfejs odtwarzacza). Prezentacja UI odtwarzacza oraz działających przycisków (play/pause, next, previous). Pliki są już przechowywane w pamięci, lecz nie są jeszcze odtwarzane. Na ekranie zaimplementowane mockowe wartości takie jak czas trwania, nazwa piosenki, ilość piosenek. Ustalenie, że na następne spotkanie projekt będzie gotowy (dodanie dekodowania).

3.6 Spotkanie 20.05

Spotkanie, na którym przedstawiono gotowy produkt. Mikrokontroler poprawnie dekoduje oraz odtwarza pliki w formacie FLAC. Prezentacja gotowego produktu oraz dokumentacji prowadzącemu. Dyskusja na temat ewentualnych poprawek oprogramowania/dokumentacji. Dyskusja na temat możliwości odtwarzania piosenek o wysokim bitracie (więcej niż 700kbps).

3.7 Spotkanie 27.05

Spotkanie, na którym przedstawiono gotowy produkt grupie laboratoryjnej. Podczas tego spotkania odbyła się również prezentacja projektów innych grup. W naszej grupie było bardzo wiele ciekawych projektów.

Bibliografia

[1] Karsten Schmidt. *Double Buffer*. URL: https://gist.github.com/postspectacular/61f17333c17b0206a73e4591c

[2] Juraj Ďuďák. *Makefile for CubeMX*. URL: https://github.com/duro80/Makefile4CubeMX.

[3] Multiple authors. *Documentation*. URL: https://documentation.help/STM32F429I-Discovery-BSP-Drivers/.