AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Systemy wbudowane

Odtwarzacz MP3

Julia Plewa
Michał Rudnik

1. Opis projektu

Celem projektu była implementacja odtwarzacza plików MP3 na płytce STM32 z wyjściem audio. Głównym problemem było dekodowanie danych z pliku i przekazywanie ich do odtwarzania za pomocą techniki DMA. Program jest oparty o udostępniony projekt startowy odtwarzający format Wave i został napisany w całości w języku C.

2. Użyty sprzęt / technologia

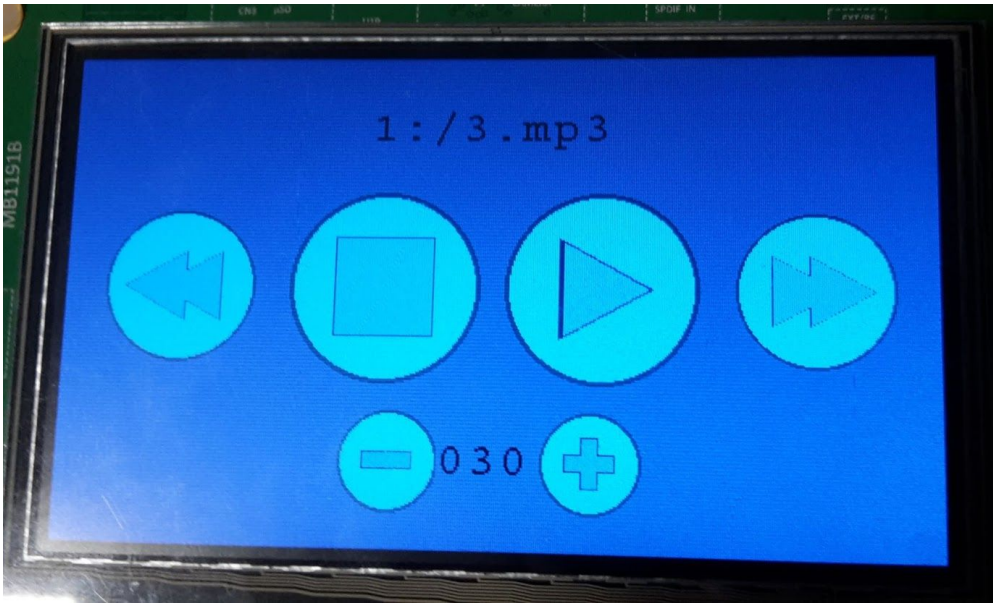
- Płytka rozwojowa: STM32F746G-DISCO
- Biblioteka Helix MP3 Decoder

3. Działanie programu

Program opiera się o dwa komunikujące się ze sobą zadania, jedno odpowiadające za dekodowanie i odtwarzanie plików, drugie zajmujące się obsługą ekranu dotykowego.

Odtwarzacz przy inicjalizacji przegląda katalog główny urządzenia wpiętego do wejścia microUSB FullSpeed w poszukiwaniu plików MP3, które miałby potencjalnie odtworzyć, i zapamiętuje ich ilość oraz nazwy.

4. Obsługa programu



Rys. 1. *Interfejs użytkownika*

Po podłączeniu nośnika z danymi i wstępnej inicjalizacji można korzystać z odtwarzacza za pomocą przycisków dostępnych na panelu dotykowym:

- play/pause: do odtwarzania/pauzowania obecnej piosenki
- stop: jeśli odtwarzana jest muzyka to ją zatrzymuje, naciśnięcie play po tej operacji rozpoczyna utwór od nowa
- minus/plus: służące do zmniejszania/zwiększania poziomu głośności.
- previous/next: do zmiany utworu na poprzedni/następny, cyklicznie.

5. Ciekawsze szczegóły implementacyjne

a. Przygotowanie dźwięku do odtwarzania:

Działanie odtwarzacza opiera się o trzy bufory danych.

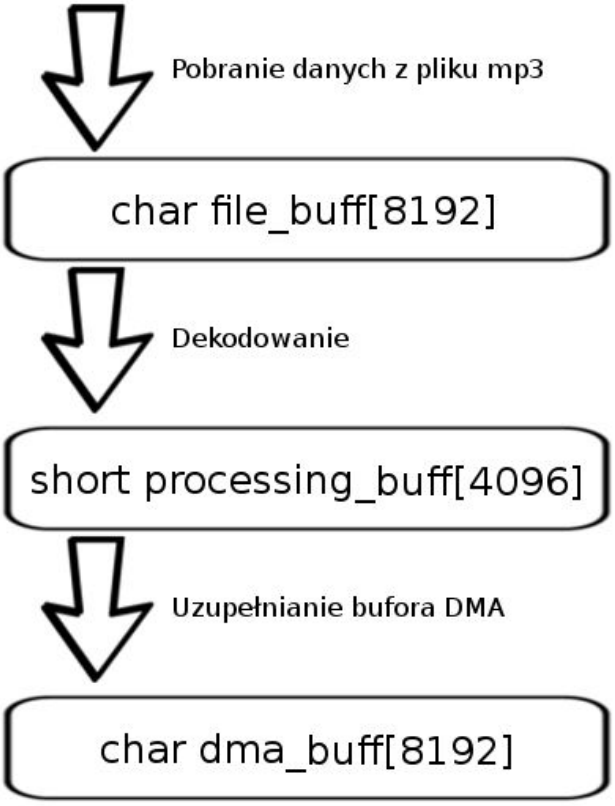
Pierwszy służy do przechowywania "surowych" danych z pliku MP3 i jest on cały czas zapełniany, dopóki nie zabraknie danych w pliku (czyli kiedy skończy się utwór).

W drugim buforze gromadzone są zdekodowane dane, aż do uzbierania się danych o rozmiarze połowy trzeciego bufora.

Z trzeciego bufora korzysta kontroler DMA, z którego odczytuje dane do odtworzenia. Dodatkowo kontroler zwraca w callbackach informacje o odczytaniu pierwszej i drugiej połowy bufora.
Działanie funkcji wykonującej się przy otrzymaniu informacji od DMA można opisać następująco:

1) Dekoduj dane z pierwszego bufora i umieszczaj je w drugim buforze.
2) Gdy w drugim buforze znajdzie się wystarczająco dużo danych, żeby zapełnić połowę trzeciego bufora, to przenieś tyle danych do tego bufora, zależnie od otrzymanego callbacka do pierwszej lub drugiej połowy.
3) Przesuń nieodczytane dane z pierwszego bufora na jego początek i uzupełnij resztę.
4) Ewentualne pozostałe dane w drugim buforze przenieś na początek.

Funkcja jest wywoływana, dopóki nie skończą się dane w odczytywanym pliku, wtedy odtwarzanie jest zatrzymywane.



Rys. 2. *Schemat przetwarzania danych*

Callback odpowiedzialny za dekodowanie danych:

```
int process_callback(int dma_offset)
{
   int bytes_read, offset;
   while (processing_buff_offs < DMA_BUFFER_SIZE / 4)
   {
       offset = MP3FindSyncWord((unsigned char *)file_buff_ptr, bytes_left);
       if (offset == -1)
       {
           bytes_left = 0;
           return 0;
       }
       bytes_left -= offset;
       file_buff_ptr += offset;
       if (MP3Decode(hMP3Decoder, (unsigned char **)&file_buff_ptr, (int *)&bytes_left,
processing_buff_ptr, 0))
       {
           xprintf("ERROR: Failed to decode the next frame\n");
           return -1;
       }
       MP3GetLastFrameInfo(hMP3Decoder, &mp3FrameInfo);
       processing_buff_offs += mp3FrameInfo.outputSamps;
       processing_buff_ptr = processing_buff + processing_buff_offs;
   }
   memcpy(dma_buff + dma_offset, processing_buff, DMA_BUFFER_SIZE / 2);
   memcpy(file_buff, file_buff_ptr, bytes_left);
   memcpy(processing_buff, &processing_buff[DMA_BUFFER_SIZE / 4], (processing_buff_offs
- DMA_BUFFER_SIZE / 4) * 2);

   file_buff_ptr = file_buff + bytes_left;

   if (f_read(&file, file_buff_ptr, (FILE_BUFFER_SIZE - bytes_left), (void *)
&bytes_read) != F_OK)
   {
       xprintf("ERROR: Failed to read from file\n");
       return -1;
   }
   file_buff_ptr = file_buff;
   bytes_left += bytes_read;
   processing_buff_offs -= DMA_BUFFER_SIZE / 4;
   processing_buff_ptr = processing_buff + processing_buff_offs;
   dma_buff_offs = BUFFER_OFFSET_NONE;
   return 0;
}
```

Rozpoczynanie odtwarzania z pliku:

```
int start_reading_file()
{
   if (f_open(&file, FILES[CURRENT_FILE], FA_READ) != FR_OK)
   {
       xprintf("ERROR: Failed to open file %s\n", FILES[CURRENT_FILE]);
       return -1;
   }
   file_buff_ptr = file_buff;

   if (f_read(&file, file_buff_ptr, FILE_BUFFER_SIZE, (void *)&bytes_left) != F_OK)
   {
       xprintf("ERROR: Failed to read from file %s\n", FILES[CURRENT_FILE]);
       return -1;
   }
   processing_buff_ptr = processing_buff;
   processing_buff_offs = 0;
   dma_buff_offs = BUFFER_OFFSET_NONE;

   if (BSP_AUDIO_OUT_Play((uint16_t *)&dma_buff[0], DMA_BUFFER_SIZE) != AUDIO_OK)
   {
       xprintf("ERROR: Failed to start the audio stream\n");
       return -1;
   }
   return 0;
}
```

Przykłady kontroli odtwarzania:

```
int mp3_stop()
{
   if (BSP_AUDIO_OUT_Stop(CODEC_PDWN_SW) != AUDIO_OK)
   {
       xprintf("ERROR: Failed to stop the audio stream\n");
       return -1;
   }

   memset(dma_buff, 0, DMA_BUFFER_SIZE);
   memset(file_buff, 0, FILE_BUFFER_SIZE);
   memset(processing_buff, 0, DMA_BUFFER_SIZE);

   if (f_close(&file) != F_OK)
   {
       xprintf("ERROR: Failed to close audio file\n");
       return -1;
   }
   player_state = STOPPED;
   return 0;
}
```

```
int mp3_pause()
{
   if (BSP_AUDIO_OUT_Pause() != AUDIO_OK)
   {
       xprintf("ERROR: Failed to pause the audio stream\n");
       return -1;
   }
   player_state = PAUSED;
   return 0;
}
```

b. komunikacja między zadaniami

Zadanie 1: obsługuje inicjalizację sterowników, odczyt plików, mechanizm dekodowania.

Zadanie 2: wyświetla i aktualizuje interfejs graficzny, reaguje na naciśnięcie przycisków na ekranie dotykowym.

Gdy drugie zadanie wykryje naciśnięcie któregoś z przycisków informuje pierwsze zadanie o tym fakcie po wykonaniu odpowiedniej dla przycisku czynności.

Komunikacja między zadaniami odbywa się poprzez zmienną przyjmującą jedną z wartości:

- stałych, ustawianych przez pierwsze zadanie (trzy)
- tymczasowych, ustawianych przez drugie zadanie (dwanaście)

Początkową wartością jest **STOPPED**, kolejne przejścia pokazane w tabeli poniżej.

Zadanie 2	Stan po zad. 2	Zadanie 1	Stan po zad. 1
Naciśnięto stop Rysuje przycisk play	STOP_PRESSED	Zatrzymaj odtwarzanie Przygotuj do odtwarzania od początku	STOPPED
Naciśnięto play przy stanie STOPPED Rysuje przycisk pause	PLAY_PRESSED	Rozpocznij odtwarzanie Zleć zadaniu 2 odświeżenie tytułu	PLAYING
Naciśnięto pause Rysuje przycisk play	PAUSE_PRESSED	Wstrzymaj odtwarzanie	PAUSED
Naciśnięto play przy stanie PAUSED Rysuje przycisk pause	RESUME_PRESSED	Wznów odtwarzanie	PLAYING
Naciśnięto previous przy stanie PLAYING Rysuje przycisk pause	PREV_PRESSED_PLAYING	Zatrzymaj aktualny plik Znajdź poprzedni plik Rozpocznij odtwarzanie pliku Zleć zadaniu 2 odświeżenie tytułu	PLAYING
Naciśnięto next przy stanie PLAYING Rysuje przycisk pause	NEXT_PRESSED_PLAYING	Zatrzymaj aktualny plik Znajdź kolejny plik Rozpocznij odtwarzanie pliku Zleć zadaniu 2 odświeżenie tytułu	PLAYING
Naciśnięto previous przy stanie STOPPED Rysuje przycisk play	PREV_PRESSED_STOPPED	Znajdź poprzedni plik Zleć zadaniu 2 odświeżenie tytułu	STOPPED

Naciśnięto next przy stanie STOPPED Rysuje przycisk play	NEXT_PRESSED_STOPPED	Znajdź kolejny plik Zleć zadaniu 2 odświeżenie tytułu	STOPPED
Naciśnięto previous przy stanie PAUSED Rysuje przycisk play	PREV_PRESSED_PAUSED	Znajdź poprzedni plik Przygotuj plik do odtworzenia Zleć zadaniu 2 odświeżenie tytułu	PAUSED
Naciśnięto next przy stanie PAUSED Rysuje przycisk play	NEXT_PRESSED_PAUSED	Znajdź kolejny plik Przygotuj plik do odtworzenia Zleć zadaniu 2 odświeżenie tytułu	PAUSED
Naciśnięto plus przy stanie PLAYING	VOL_UP_PRESSED	Zwiększa wartość głośności	PLAYING
Naciśnięto minus przy wartości PLAYING	VOL_DOWN_PRESSED	Zmniejsza wartość głośności	PLAYING

Tab. 1. *Przejścia między stanami i współpraca zadań*

Dodatkowo zadanie 1 posiada mechanizm zlecania zadaniu 2 odświeżenia tytułu odtwarzanego utworu. Służy do tego dodatkowa zmienna przyjmująca wartości 0 lub 1.

c. obsługa ekranu dotykowego

Do narysowania interfejsu oraz wykrywania dotyku wykorzystane zostały funkcje udostępniane przez sterownik BSP.

Po wykryciu dotyku w obrębie przycisków w zadaniu 2 odpowiednia informacja jest przekazywana do zadania 1. Zadanie 1 ma możliwość zlecenia polecenia odświeżenia tytułu utworu dla zadania 2, ale samo nie odbywa żadnej interakcji z ekranem.

Przypadki, w których naciśnięcie przycisku jest ignorowane:

- wciśnięcie przycisku stop, kiedy odtwarzacz jest w stanie PAUSED
- próba regulacji głośności w stanie innym niż PLAYING

6. Wyniesiona wiedza

- Budowanie dużego projektu w C
- Samodzielne szukanie źródeł oraz bibliotek
- Wiedza odnośnie formatu danych dźwiękowych (MP3 oraz Wave)
- Praktyczne doświadczenie w synchronizowaniu zadań
- Projektowanie interfejsu użytkownika

7. Dalszy rozwój projektu

- Obsługa obu wejść USB i czytnika SD (z opcją odświeżania)
- Radio internetowe

8. Kod źródłowy

Link do repozytorium z kodem: https://github.com/jplewa/SW-AGH