**Cyfrowy Equalizer**

**Cel projektu**

Celem projektu jest zaprojektowanie i oprogramowanie odtwarzacza muzycznego za pomocą mikrokontrolera. Kluczową funkcjonalnością układu ma być equalizer pozwalający na filtrowanie i zmienianie barwy odtwarzanego dźwięku. Mikrokontroler będzie odtwarzał utwory zapisane na karcie SD i umożliwiał zdalną kontrolę poprzez moduł bluetooth.

**Obsługiwane funkcjonalności**

* Odczyt plików wave audio z karty microSD
* Odtwarzanie plików dźwiękowych
* Korektor (equalizer) pozwalający zmieniać tony dżwięku
* Odtwarzacz umożliwiający wybór utworów, przewijanie, pauzowanie i konfigurowanie equalizera
* Interfejs zdalnej obsługi bluetooth pozwalający na kontrolowanie odtwarzacza
* Aplikacja do zdalnej obsługi kontrolera na urządzenia z systemem Android

**Hardware**

* Mikrokontroler STM32F407G-DISC1

# Moduł czytnika kart microSD

# Moduł Bluetooth HC-06 ZS-040

# Karta pamięci Goodram microSD 16GB

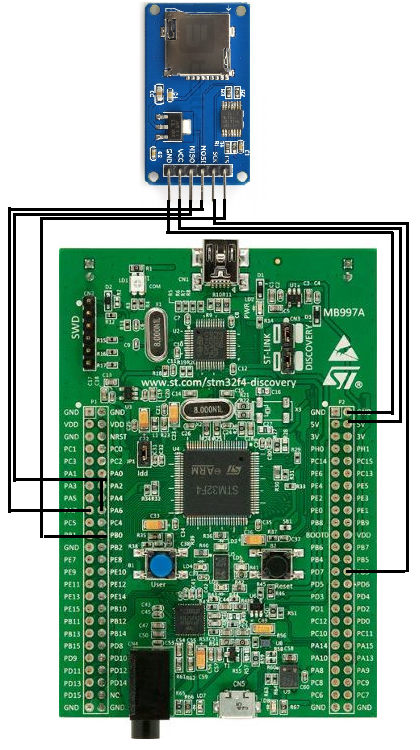
**Realizacja projektu**

1. **Zapoznanie się ze specyfikacją mikrokontrolera i modułów, oraz złożenie prototypu**

Do połączenia komponentów z kontrolerem wykorzystano kable stykowe żeńsko-żeńskie.

**Karta SD**

Moduł czytnika kart microSD podłączono do pinów PA5, PA6, PB3 oraz PB0, do napięcia 5V. Komunikacją z kartą SD odbywa się z wykorzystaniem protokołu SPI.



1. **Przygotowanie środowiska do pracy z mikrokontrolerem w systemie Ubuntu**

Projekt realizowany jest w języku C++ w standardzie C++17 z wykorzystaniem bibliotek HAL oraz FatFs. Za kompilację odpowiada CMake wykorzystujący kompilator gcc-arm-none-eabi. Do programowania pamięci Flash mikrokontrolera używany jest program OpenOCD.

Jako zintegrowane środowisko programistyczne używany jest program CLion.

1. **Implementacja inicjalizacji mikrokontrolera i jego poszczególnych modułów**

Do częściowej generacji kodu konfiguracji zegara oraz kodu inicjalizacji poszczególnych komponentów i bibliotek wykorzystywany jest program CubeMX.

Inicjalizacja komponentów użuwanych przez program kontrolera odbywa się w klasie **Application**. Odpowiada ona za inicjalizację:

* Biblioteki HAL
* Ustawień zegara
* Pinów do obsługi diod LED
* Komunikacji SPI z modułem kart microSD
* Systemu plików FAT dla karty microSD

Obsługuje ona również przerwania zegarowe.

1. **Implementacja prostego odtwarzania dźwięku wave audio z karty microSD**

Za odczyt metadanych audio oraz próbek dźwięku odpowiada klasa **WavAudioReader**. Umożliwia ona ciągły odczyt sekwencyjny danych, a także wykonuje walidację pliku, wykrywając błędy i informując o nieobsługiwanych formatach i parametrach.

Klasa **AudioPlayer** odtwarza pliki audio wykorzystując klasę **WavAudioReader**. Odtwarzanie porzebiega z wykorzystaniem sterowników BSP płytki discovery.

1. **Implementacja graficznego equalizera z wykorzystaniem szybkiej transformacji Fouriera**

Klasa **AudioPlayer** umożliwia filtrowanie dźwięku przed jego odtwarzaniem. Interfejs filtrowania dostarcza klasa **AudioFilter**, która wykonuje skalowanie próbek z wartości całkowitoliczbowych do wartości zmiennoprzecinkowych (-1, 1), rozdzielając ciąg próbek na poszczególne kanały, a następnie dla próbek każdego kanału wywołuje metodę **processNormalizedSamples**, która przetwarza znormalizowane próbki. Następnie przefiltrowane wartości rzeczywiste są konwertowane z powrotem do wartości całkowitoliczbowych.

Klasa **FFTAudioFilter** implementuję metodę **processNormalizedSamples**, wykonując Szybką Transformację Fouriera. Transformacja wykonywana jest bez dodatkowej pamięci, z wersją ze zwiniętą rekurencją w iterację. Algorytm wykorzystuje decymację częstotliwościową (Decimation-in-Frequency). Klasa wywołuję następnie metodę **processFFT** i wykonuje odwrotną transformację Fouriera.

Klasa **DigitalEqualizerFilter** implementuje metodę **processFFT**, w której wykonuje filtrowanie próbek transformaty. Dostępne parametry konfiguracji to podbicie wartości amplitudy dla wszystkich częstotliwości oraz podbicie amplitudy w bandach oktawy ISO (zestaw dziesięciu bandów).

1. **Przygotowanie funkcjonalnego interfejsu odtwarzacza.**

Klasa **Playlist** udostępnia listę plików audio do odtwarzania z danego katalogu karty SD do odtwarzania przez klasę **AudioPlayer**.

1. **Publikacja interfejsu odtwarzacza poprzez moduł bluetooth.**
2. **Przygotowanie aplikacji dla androida obsługującej mikrokontrolerowy odtwarzacz przez bluetooth.**
3. **Przetestowanie i ewentualne poprawki błędów całego systemu.**

**Źródła**

Strona producenta z dokumentacją mikrokontrolera:

<https://www.st.com/en/microcontrollers-microprocessors/stm32f407vg.html>

Szablonowy projekt CMake dla płytek STM32F4xx

<https://github.com/ahessling/STM32F4Template>

Specyfikacja kontenera WAVE

http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html

Przykładowy kod equalizera plików WAVE napisany w C

<https://github.com/kappalabs/fft_band_equalizer>

Przykładowy kod equalizera napisany w C++ wykorzystujący biblioteki do FFT I filtrowania dźwięku

https://sourceforge.net/projects/equalizerapo/

Artykuł o wykorzystaniu szybkiej transformacji Furiera w equalizerach.

[http://www.zytrax.com/tech/audio/equalization.html#overview](http://www.zytrax.com/tech/audio/equalization.html" \l "overview)

Decymacja częstotliwości

<https://www.brainkart.com/article/Decimation-In-Frequency-(DIFFFT)_13033/>