Politechnika Rzeszowska
Elektronika dla Informatyków

Music Player 8bit

Prosty syntezator muzyczny na bazie Arduino

Dokumentacja projektu

Autorzy:
Daniel Pawelec — nr albumu 183922
Wojciech Piątek — nr albumu 174752

Rzeszów, 2026

Spis treści

1. Wstęp

Niniejszy dokument stanowi pełną dokumentację techniczną projektu „Music Player 8bit” — prostego syntezatora muzycznego zrealizowanego na platformie Arduino UNO w ramach przedmiotu „Elektronika dla Informatyków”. Projekt obejmuje zarówno część sprzętową, jak i programową, a także dodatkowe narzędzie wspomagające w języku Python.

Celem urządzenia jest odtwarzanie zapisanych w pamięci melodii za pomocą buzzera, przy jednoczesnym prezentowaniu informacji o stanie systemu na wyświetlaczu LCD. Użytkownik steruje urządzeniem za pomocą czterech przycisków, poruszając się po menu wyboru utworów. Dokumentacja powstała na podstawie założeń projektowych, dwóch raportów cząstkowych oraz kodu źródłowego oprogramowania.

2. Opis i cel projektu

Projekt obejmuje zaprojektowanie i implementację prostego systemu odtwarzania muzyki na platformie Arduino UNO, wykorzystującego buzzer jako generator dźwięku oraz wyświetlacz LCD jako interfejs użytkownika.

System realizuje następujące funkcje:

- wyświetlanie ekranu powitalnego (ekranu tytułowego z nazwą projektu i autorami),
- prezentację menu wyboru utworów wraz z możliwością nawigacji,
- odtwarzanie wybranej melodii zapisanej w postaci funkcji w języku C/C++ (Arduino),
- możliwość przerwania odtwarzania w dowolnym momencie i powrotu do menu.

Dodatkowo opracowano narzędzie w języku Python służące do konwersji plików MIDI na funkcje kompatybilne z Arduino, co znacznie ułatwia dodawanie nowych utworów do systemu.

3. Założenia projektu

Na etapie analizy określono główne wymagania stawiane urządzeniu:

- możliwość odtwarzania melodii na buzzerze,
- prezentacja informacji na wyświetlaczu LCD,
- prosty i intuicyjny interfejs użytkownika oparty na przyciskach,
- łatwe dodawanie nowych utworów do biblioteki.

Zdecydowano o wykorzystaniu Arduino UNO jako platformy sprzętowej oraz języka Python do stworzenia narzędzia konwersji plików MIDI. Architekturę programu oparto na maszynie stanów (automacie skończonym).

4. Zastosowane technologie

4.1. Sprzęt

Element	Opis / zastosowanie
Arduino UNO	Główna platforma sprzętowa, mikrokontroler ATmega328P
Wyświetlacz LCD 16x2 (I2C)	Interfejs użytkownika z konwerterem I2C ograniczającym liczbę pinów
Buzzer	Generator dźwięku (sygnał audio sterowany funkcją tone())

Element	Opis / zastosowanie
4 przyciski (tact switch)	Sterowanie menu: poprzedni / następny / odtwórz / powrót
Rezystory pull-up (10 kΩ)	Stabilizacja stanów logicznych wejść przycisków

4.2. Oprogramowanie

Narzędzie	Zastosowanie
Arduino IDE	Środowisko programistyczne, język C/C++
Biblioteka LiquidCrystal_I2C	Obsługa wyświetlacza LCD przez magistralę I2C
Biblioteka Wire	Komunikacja po magistrali I2C
Python 3	Narzędzie konwersji plików MIDI na kod Arduino
Biblioteka mido	Wczytywanie i analiza plików MIDI w Pythonie
Tinkercad	Wizualizacja i symulacja układu, schemat ideowy

5. Architektura sprzętowa

5.1. Wyświetlacz LCD i magistrala I2C

Kluczowym elementem warstwy wizualnej jest alfanumeryczny wyświetlacz LCD 16x2 znaków wyposażony w moduł konwertera I2C. Wybór standardu I2C podyktowany był koniecznością oszczędności pinów mikrokontrolera Arduino UNO — komunikacja wymaga jedynie dwóch linii sygnałowych:

- SDA (Serial Data) — podłączona do pinu analogowego A4 Arduino UNO,
- SCL (Serial Clock) — podłączona do pinu analogowego A5 Arduino UNO.

Moduł zasilany jest napięciem 5 V bezpośrednio z szyny zasilającej mikrokontrolera, ze wspólną masą (GND). W kodzie wyświetlacz inicjalizowany jest pod adresem magistrali 0x27 (typowo 0x27 lub 0x3F).

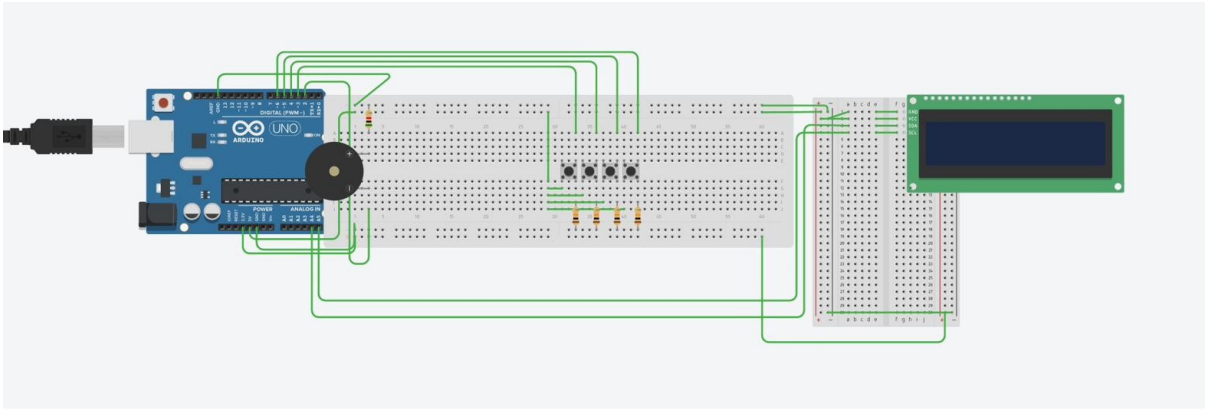
5.2. Buzzer i przyciski

Buzzer podłączono do wyjścia cyfrowego mikrokontrolera, a cztery przyciski sterujące do kolejnych pinów cyfrowych, ze stabilizacją rezystorami pull-up. Poniższa tabela przedstawia przypisanie pinów zgodne z kodem źródłowym:

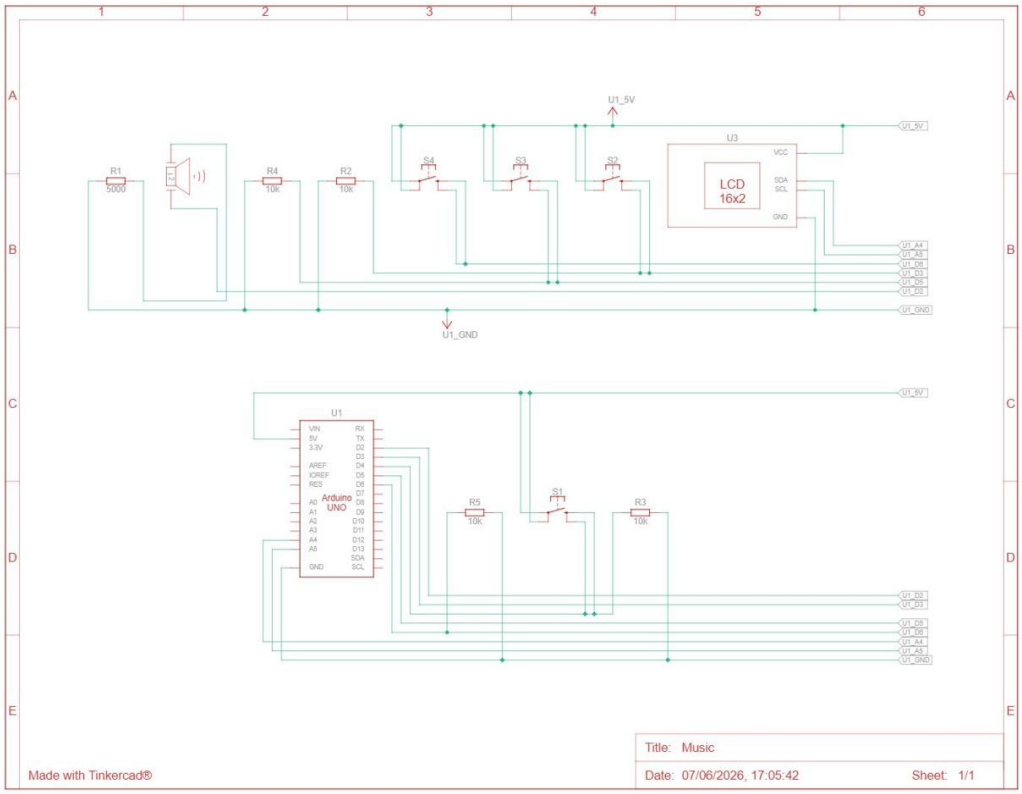
Pin Arduino	Element	Funkcja w programie
D2	Buzzer	Wyjście sygnału dźwiękowego (BUZZER_PIN)
D3	Przycisk PREV	Poprzedni utwór w menu
D4	Przycisk STOP / Powrót	Przerwanie odtwarzania i powrót do menu
D5	Przycisk PLAY	Odtworzenie wybranego utworu
D6	Przycisk NEXT	Następny utwór w menu
A4	LCD — SDA	Linia danych I2C
A5	LCD — SCL	Linia zegara I2C

5.3. Wizualizacja i schemat układu

Układ został zaprojektowany i zweryfikowany w środowisku Tinkercad. Poniżej przedstawiono wizualizację montażu na płytce stykowej oraz schemat ideowy połączeń.



Rysunek 1. Wizualizacja układu w programie Tinkercad.



Rysunek 2. Schemat ideowy połączeń.

6. Architektura oprogramowania

Prace programistyczne zrealizowano w środowisku Arduino IDE w języku C/C++. Logika interfejsu została ściśle zintegrowana z maszyną stanów sterującą całym urządzeniem.

6.1. System stanów (automat)

Działanie urządzenia opiera się na automacie skończonym, który zarządza przepływem informacji między interfejsem a modułem dźwiękowym. System operuje w trzech stanach:

Stan	Opis działania
START	Inicjalizacja komponentów i wyświetlenie animowanego ekranu tytułowego
MENU	Obsługa pętli wyboru utworu; oczekiwanie na sygnały z przycisków i aktualizacja pozycji na LCD
PLAYING	Przekazanie sterowania do funkcji odtwarzającej dźwięk wraz z wyświetleniem nazwy utworu

Ze stanu START przejście do MENU następuje po wciśnięciu przycisku. W stanie MENU przyciski PREV/NEXT zmieniają wybrany utwór, a przycisk PLAY uruchamia odtwarzanie (przejście do PLAYING). W trakcie odtwarzania wciśnięcie przycisku powrotu (pin D4) przerywa melodię i przywraca stan MENU.

6.2. Moduł odtwarzania dźwięku

Sercem warstwy dźwiękowej są dwie funkcje: play() generująca pojedynczy dźwięk o zadanej częstotliwości i czasie trwania oraz play_Pause() realizująca pauzę. Obie funkcje w każdym wywołaniu sprawdzają stan przycisków (detectBtn()), dzięki czemu odtwarzanie może zostać przerwane w dowolnej chwili i nastąpi natychmiastowy powrót do menu. Jest to kluczowe rozwiązanie zapewniające responsywność interfejsu mimo blokujących wywołań delay().

```
bool play(unsigned int frequency, unsigned long duration) {
  if (detectBtn() == 4) {
    state = MENU;
    return false;
  }
  tone(BUZZER_PIN, frequency);
  delay(duration);
  noTone(BUZZER_PIN);
  return true;
}

int play_Pause(int czas) {
  if (detectBtn() == 4) {
    state = MENU;
    return false;
  }
  noTone(BUZZER_PIN);
  delay(czas);
  return true;
}
```

Listing 1. Funkcje odtwarzania dźwięku oraz pauzy.

Każda nuta w utworze zapisana jest jako wywołanie play() w konstrukcji warunkowej (if (! play(...)) return;). Dzięki temu zwrócenie wartości false (czyli wykrycie przycisku powrotu) natychmiast przerywa wykonywanie całej funkcji melodii.

6.3. Definicje nut i wartości rytmiczne

W kodzie zdefiniowano częstotliwości wszystkich nut w zakresie od oktawy 0 do 8 (od C0 do C8) za pomocą dyrektyw #define. Wartości rytmiczne wyliczane są względem długości szesnastki (NOTE_SIXTEENTH), która ustalana jest indywidualnie dla każdego utworu na podstawie tempa (BPM):

Wartość	Definicja	Wartość	Definicja
Trzydziestodwójka	NOTE_SIXTEENTH / 2	Półnuta	8 × NOTE_SIXTEENTH
Ósemka	2 × NOTE_SIXTEENTH	Półnuta z kropką	12 × NOTE_SIXTEENTH
Ósemka z kropką	3 × NOTE_SIXTEENTH	Cała nuta	16 × NOTE_SIXTEENTH
Ćwierćnuta	4 × NOTE_SIXTEENTH	Cała z kropką	24 × NOTE_SIXTEENTH
Ćwierćnuta z kropką	6 × NOTE_SIXTEENTH	—	—

6.4. Obsługa wyświetlacza LCD

Do obsługi modułu wykorzystano bibliotekę LiquidCrystal_I2C. Implementacja obejmowała inicjalizację komunikacji pod adresem magistrali, definiowanie znaków oraz opracowanie funkcji odświeżających ekran w sposób nieblokujący pracy procesora — co jest kluczowe podczas jednoczesnego generowania dźwięku przez buzzer. Ekran powitalny realizuje efekt przewijania tekstu „Music Player v1.0” w oparciu o nieblokujący pomiar czasu millis().

6.5. Interfejs użytkownika

Interfejs został podzielony na trzy główne moduły wizualne:

- Ekran powitalny (Splash Screen) — wyświetlany po uruchomieniu; prezentuje tytuł projektu oraz autorów (Daniel & Wojtek).
- Menu wyboru utworów — dynamiczna lista melodii pobierana z tablicy zdefiniowanej w pamięci programu; nawigacja przyciskami.
- Ekran odtwarzania — wyświetla nazwę aktualnie wykonywanego utworu.

Wykrywanie naciśnięć przycisków zaimplementowano w funkcji detectBtn(), która porównuje bieżący stan wejścia z poprzednim, eliminując wielokrotne wyzwalanie przy jednym wciśnięciu. Funkcja zwraca kod 1–4 odpowiadający kolejno przyciskom PREV, NEXT, PLAY i STOP/Powrót.

7. Narzędzie konwersji MIDI → Arduino

W celu ułatwienia dodawania nowych utworów opracowano w języku Python narzędzie konwertujące pliki MIDI na gotowe funkcje w języku Arduino. Narzędzie wykorzystuje bibliotekę mido i realizuje następujące zadania:

1. wczytanie pliku MIDI,
2. analiza zdarzeń note_on oraz note_off,
3. obliczenie czasu trwania poszczególnych nut,
4. konwersja nut do formatu Arduino (nazwa nuty + wartość rytmiczna),
5. wygenerowanie gotowego kodu źródłowego funkcji.

Ze względu na to, że buzzer jest urządzeniem monofonicznym (brak obsługi polifonii), narzędzie automatycznie wybiera jedną nutę spośród równocześnie brzmiących dźwięków. Poniżej fragment kodu generującego funkcję Arduino:

```
func_name = "generatedSong"
print(f"void {func_name}() {{")
print(f"  NOTE_SIXTEENTH = {int(NOTE_SIXTEENTH)}; // tempo w ms z BPM")
```

```
last_time = 0
for start_ms, note, duration_ms in filtered_events:
    pause = start_ms - last_time
    if pause > 1:
        print(f"  if (!play_Pause({int(pause)})) return;")
    note_name = midi_to_note(note)
    note_define = duration_to_define(duration_ms, NOTE_SIXTEENTH)
    print(f"  if (!play({note_name}, {note_define})) return;")
    last_time = start_ms + duration_ms

print("  state = MENU; // koniec melodii")
print("}")
```

Listing 2. Fragment generatora kodu Arduino z pliku MIDI.

8. Baza utworów

Utwory przechowywane są jako oddzielne funkcje, a ich nazwy oraz wskaźniki do funkcji zebrano w tablicach indeksowanych pozycją w menu. Aktualnie biblioteka zawiera następujące utwory:

Lp.	Nazwa utworu	Funkcja w kodzie
1	Polka	polkaDziadek()
2	Cicha noc	silentNight()
3	Gravity Falls Theme	gravityFalls()
4	Bink's Sake (One Piece Theme)	binksSake()
5	Maiden Voyage (Sea of Thieves)	maidenVoyage()

Dodanie nowego utworu sprowadza się do napisania (lub wygenerowania konwerterem MIDI) funkcji melodii oraz dopisania jej nazwy i wskaźnika do odpowiednich tablic, dzięki czemu architektura jest łatwo rozszerzalna.

9. Napotkane problemy i ich rozwiązania

Problem	Rozwiązanie
Brak obsługi polifonii przez buzzer	Uproszczenie melodii i automatyczny wybór jednej nuty w skrypcie Python
Blokujące funkcje delay() uniemożliwiały reakcję menu	Sprawdzanie stanu przycisków (detectBtn()) w każdym wywołaniu funkcji odtwarzającej
Migotanie ekranu przy częstym odświeżaniu	Selektywne odświeżanie — aktualizacja tylko przy zmianie stanu lub pozycji kursora
Różnorodność i różnice długości nut w plikach MIDI	Normalizacja wartości i dostosowanie skryptu konwertującego
Ograniczenia pamięci RAM	Wykorzystanie makra F() oraz dyrektywy PROGMEM dla statycznych ciągów znaków

10. Testowanie i integracja

Integracja polegała na połączeniu modułu wizualnego z logiką odtwarzania melodii. Przeprowadzono następujące testy:

- weryfikację poprawności działania interfejsu i nawigacji w menu,
- sprawdzenie poprawności odtwarzania melodii oraz zgodności z danymi MIDI,
- weryfikację wyświetlania znaków diakrytycznych i specjalnych,
- sprawdzenie stabilności magistrali I2C przy długotrwałym obciążeniu procesora generowaniem dźwięku,
- testy responsywności menu przy różnych tempach utworów (BPM).

System przeszedł testy pomyślnie, osiągając pełną funkcjonalność oraz stabilność działania.

11. Podział pracy

11.1. Daniel Pawelec

- implementacja logiki odtwarzania dźwięku na buzzerze,
- opracowanie struktury funkcji muzycznych,
- stworzenie narzędzia w Pythonie do konwersji plików MIDI na kod Arduino,
- przetwarzanie danych muzycznych.

11.2. Wojciech Piątek

- implementacja obsługi wyświetlacza LCD 16x2,
- zaprojektowanie interfejsu użytkownika oraz logiki menu,
- integracja warstwy wizualnej z systemem sterowania odtwarzaczem.

12. Wnioski

Realizacja projektu „Music Player 8bit” pozwoliła na stworzenie kompletnego i przyjaznego dla użytkownika urządzenia. Zastosowanie wyświetlacza LCD z komunikacją I2C okazało się rozwiązaniem efektywnym, pozwalającym zachować wolne zasoby sprzętowe dla dalszej rozbudowy systemu.

Prace nad integracją uświadomiły wagę poprawnego projektowania systemów czasu rzeczywistego, w których warstwa wizualna nie może negatywnie wpływać na krytyczne czasowo funkcje generowania dźwięku. Projekt pozwolił również zdobyć doświadczenie w pracy z mikrokontrolerami, integracji sprzętu i oprogramowania oraz pracy zespołowej.

13. Podsumowanie

Projekt został ukończony i jest w pełni funkcjonalny oraz gotowy do prezentacji. System umożliwia odtwarzanie melodii, wygodne zarządzanie biblioteką utworów oraz monitorowanie stanu urządzenia w czasie rzeczywistym. Dzięki modularnej architekturze i narzędziu konwersji MIDI system można w prosty sposób rozbudowywać o kolejne utwory i funkcje.