<table>
<tr><td>  </td><td colspan="3">Instytut Informatyki Politechniki Śląskiej

Zespół Mikroinformatyki i Teorii Automatów Cyfrowych</td><td>  </td></tr>
<tr><td>Rok akademicki</td><td>Rodzaj studiów*: SSI/NSI/NSM</td><td>Przedmiot (Języki Asemblerowe/SMiW):</td><td>Grupa</td><td>Sekcja</td></tr>
<tr><td>2016/2017</td><td>SSI</td><td>SMiW</td><td>3</td><td>10</td></tr>
<tr><td>Imię:</td><td>Radosław</td><td rowspan="2">Prowadzący:</td><td colspan="2" rowspan="2">HM</td></tr>
<tr><td>Nazwisko:</td><td>Wojaczek</td></tr>
<tr><td colspan="5">Raport końcowy</td></tr>
<tr><td colspan="5">Temat ćwiczenia:

Szafa grająca</td></tr>
<tr><td>Data oddania:
dd/mm/rrrr</td><td colspan="4">10/01/2017</td></tr>
</table>

1. Temat projektu i opis założeń, w tym opis funkcji urządzenia.

Tematem mojego projektu jest „Szafa grająca” (ang. Jukebox). Powszechnie znana definicja Szafy grającej jako symbolu lat 50. i 60. brzmi następująco: *„automatyczne urządzenie do odtwarzania płyt gramofonowych zamknięte w dużej obudowie. Składa się z odtwarzacza i urządzenia wybierającego jedną płytę z zestawu.”* (źródło.: wikipedia.pl)

Dla celów projektowych zmieniłem specyfikę tego urządzenia na środowisko mikrokontrolerowe.

Funkcje jakie ma spełniać ten projekt:

- Odtwarzanie melodii zadanej przez użytkownika
- Wyświetlanie tytułu aktualnie wybranej melodii po wybraniu przyciskami nawigacyjnymi

Założenia:

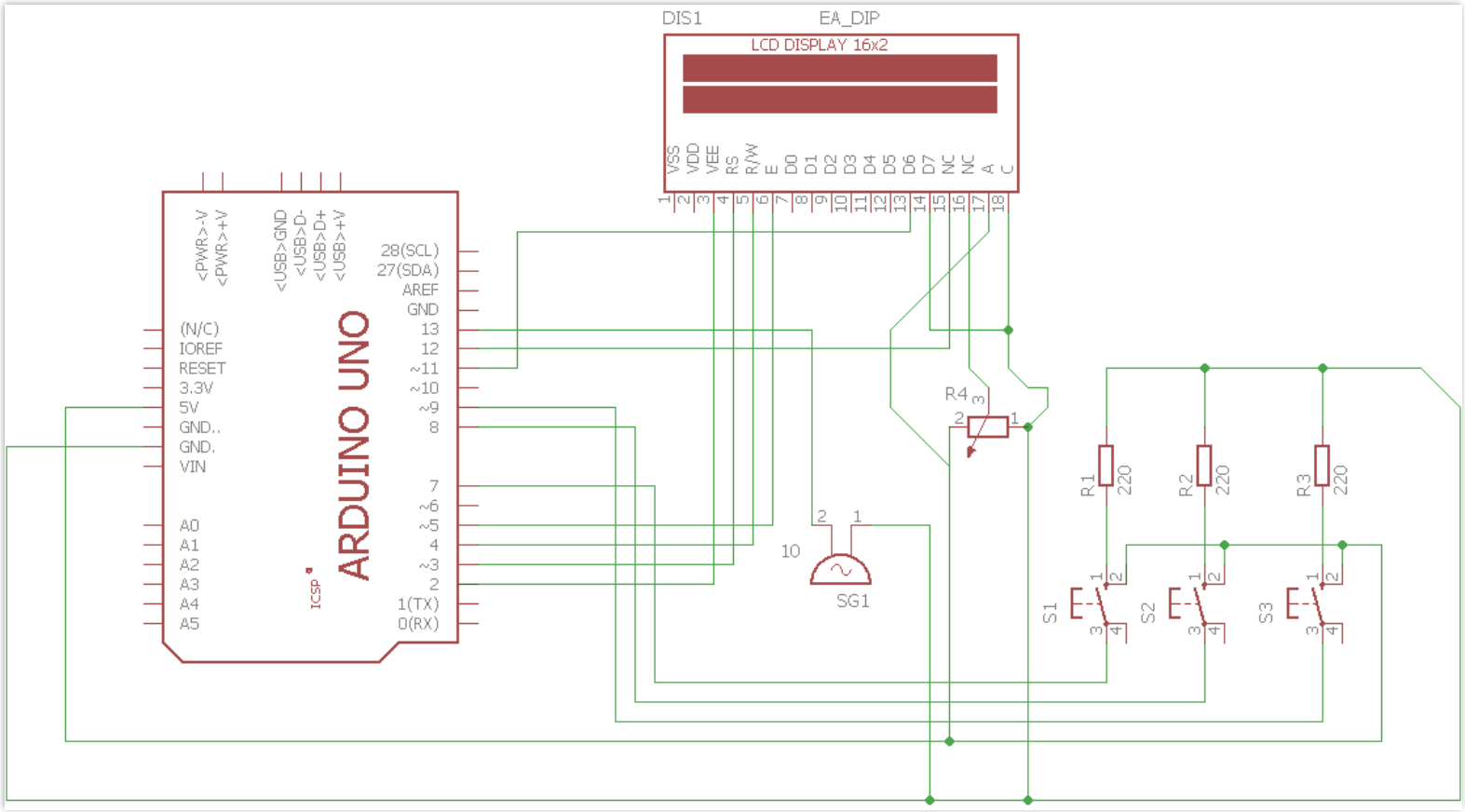
- Zmieszczenie 16 melodii w pamięci Arduino
- Nawigacja po tytułach za pomocą dwóch zewnętrznych przycisków (poprzedni i następny). Po natrafieniu na szesnastą melodię licznik cofa się do pierwszej melodii

2. Analiza zadania w tym uzasadnienie wyboru elementów elektronicznych i narzędzi użytych do realizacji projektu.

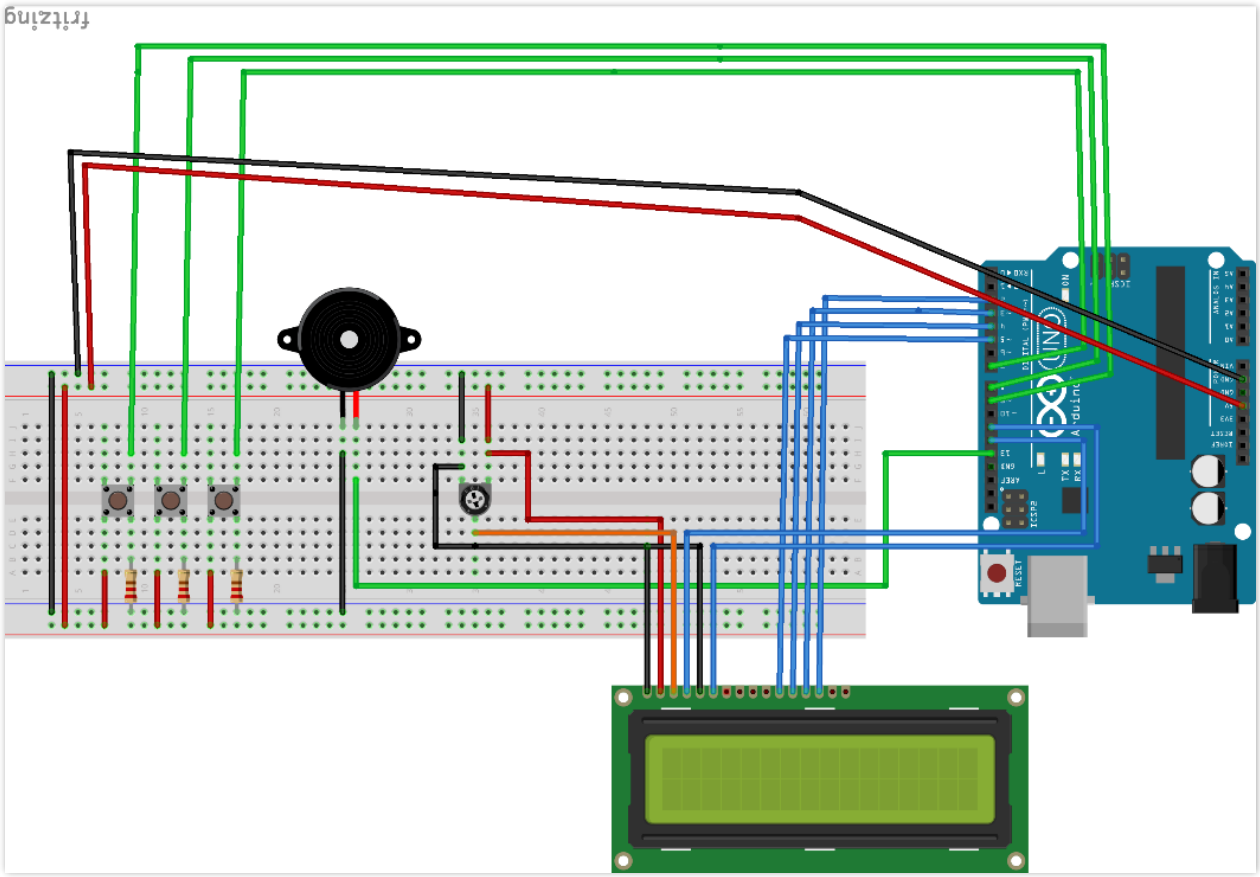
Lista części użytych w projekcie, wraz z uzasadnieniem:

- Arduino Uno – zdecydowałem się na tą platformę ze względu na jej popularność, oraz łatwy dostęp najróżniejszych materiałów – wykonanie dowolnego projektu nie ogranicza kreatywności
- Zworki
- Buzzer (pol. brzęczyk) z generatorem 5V 12 mm– część, bez której projekt nie mógłby się obejść, potrzebna jest do odgrywania dźwięków
- Wyświetlacz LCD (w typie HD44780) – potrzebny dla ułatwienia nawigacji po melodiach, wyświetla także tytuł aktualnie odgrywanej melodii, dostępna także opcja podświetlenia tylnego wyświetlacza, dzięki czemu możliwym staje się korzystanie z Szafy grającej w ciemnościach
- 3 przyciski – pierwszy (od lewej) służy do przesuwania się o jedną pozycję w tył po liście melodii, drugi do odtworzenia melodii, trzeci do przesuwania się o jedną pozycję naprzód po liście melodii
- 3 rezystory 220 Ohm – podłączone do przycisków w trybie pull-down
- Potencjometr liniowy 10kOhm – służy do zmieniania mocy kontrastu na wyświetlaczu LCD
- Płytka stykowa – pozwala ona na wykonanie prototypu urządzenia, bez konieczności wykonywania i trawienia płytek drukowanych, co zwiększa możliwości szybkiej optymalizacji projektu

3. Specyfikacja wewnętrzna:
   a. Schemat blokowy urządzenia (wykonany w programie Eagle).



   b. Schemat ideowy urządzenia (wykonany w programie Fritzing).



c. Kod programu:

```
#include <LiquidCrystal.h>
#include "pitches.h" //biblioteka z zapisanymi częstotliwościami tonów
w Hz

#define OCTAVE_OFFSET 0
#define MAX_SONG_NUMBER 16

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //inicjalizacja LCD
const char* tytuly[] = {"Game of Thrones", "Wsrod nocnej ciszy", "Ona
tanczy dla mnie","Never gonna give you up","Classical","He's
Pirate","Star Wars", "Underworld", "Hallelujah","Wlazl kotek na
plotek","Melody","Koziolek Matolek","Sto lat", "Melodia", "Ona tanczy
dla mnie","Melodyja"}; //tablica melodii

const int playPin = 8; //inicjalizacja przycisku odpowiedzialnego za
odtworzenie melodii
const int prevPin = 9; //inicjalizacja przycisku odpowiedzialnego za
przełączenie na poprzednią melodię
const int nextPin = 7; //inicjalizacja przycisku odpowiedzialnego za
przełączenie na następną melodię

//////////////////////POCZĄTEK LISTY MELODII//////////////////////

////////////////////////////Classical/////////////////////////////////////

int classic_melody[] = {      //tablica z tonami( wysokości dźwięków)

    NOTE_AS5  ,    NOTE_D5   ,    NOTE_E5   ,    NOTE_F5   ,
    NOTE_A5   ,    NOTE_A5   ,    NOTE_C6   ,    NOTE_A5   ,
    NOTE_G5   ,    NOTE_G5   ,    NOTE_F5   ,    NOTE_E5   ,
    NOTE_F5   ,    NOTE_E5   ,    NOTE_D5   ,    NOTE_D4

};

  int classic_duration[] = {   //tablica z długościami tonów

          5,        2,          2,          2,
          3,        3,          3,          2,
          7,        8,          8,          3,
          3,        3,          8,          10
  };

////////////////////////////UNDERWORLD/////////////////////////////////////

int underworld_melody[] = {
  NOTE_C4, NOTE_C5, NOTE_A3, NOTE_A4,

};

int underworld_duration[] = {
  2, 2, 2, 2,
};

//////////////////////GAME OF THRONES/////////////////////////

int thrones_melody[] = {
NOTE_G3 , NOTE_C3, NOTE_DS3, NOTE_F3, NOTE_G3 ,NOTE_C3, NOTE_DS3,
NOTE_F3, NOTE_D3,
NOTE_F3 , NOTE_B2, NOTE_D3, NOTE_DS3, NOTE_F3, NOTE_B2, NOTE_DS3,
NOTE_D3,NOTE_C3 }; //18

int thrones_duration[] = {
3, 3, 1, 1, 2, 2, 1, 1, 9,
3, 3, 1, 1, 2, 2, 1, 1, 9};
```

```
//////////////////WŚRÓD NOCNEJ CISZY//////////////////

 int wsrod_melody[] = {
C, D, H, G, E, E, F, D, E,
C, D, H, G, E, E, F, D, E,
C, E, C, E, F, D, H, G,
C, E, C, E, F, D, H, G,
C, C, D, D, C }; //39

int wsrod_duration[] =
{4,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2};


//////////////////////////MELODY/////////////////////////////////////

int melody[] = {6553}; //1

int melody_duration[3] = {4};

//////////////////////////MELODIA/////////////////////////////////////

int melodia[] = {208}; //1

int melodia_duration[6] = {4};

//////////////////////////MELODYJA/////////////////////////////////////

int melodyja[] = {543}; //1

int melodyja_duration[1] = {4};

//////////////////////////Rick Astley/////////////////////////////////

int rick_melody[] = {
C,D,F,D,A,
A,G,
C,D,F,D,G,
G,F}; //14

int rick_duration[] = {
1,1,1,1,1,
4,4,
1,1,1,1,1,
4,4};

//////////////////ONA TAŃCZY DLA MNIE//////////////////////////

int onatanczy_melody[] =
{C,
C,D,C,C,
E,F,E,C,E,
D,H,C,D,C,
C,C,D,H,C,
E,F,E,H,
D,H,C,D,C};

int onatanczy_duration[] = {
 3,
 1,1,1,3,
 1,1,1,1,3,
 1,1,1,1,3,
 1,1,1,1,3,
 1,1,1,3,
 1,1,1,1,3,}; //32



//////////////////////STO LAT/////////////////////////////////////
```

```
int stolat_melody[] =  {
G, E, G, E, //sto lat sto lat
G, A, G, F, //niech zyje zyje nam
E, F, F, D, //sto lat sto lat
F, D, F, G, //niech zyje zyje nam
F, E, D, E, //jeszcze raz
G, G, E, G, G, E, G, C, H, A, G, A,
H, H, H, CE}; //36

int stolat_duration[] = {
  3,2,3,2,
  4,2,2,2,
  3,2,3,2,
  4,2,2,2,
  2,2,2,2,
  2,2,2,2,2,2,2,2,5,2,2,2,
  6,3,3,6

 };

//////////////////////WLAZŁ KOTEK NA PŁOTEK///////////////

int kotek_melody[] =  {
G, E, E,
F, D, D,
C, E, G,
G, E, E,
F, D, D,
C, E, C}; //18

int kotek_duration[] = {
  2,2,2,
  2,2,2,
  1,1,5,
  2,2,2,
  2,2,2,
  1,1,6
 };

////////////////////////////HALLELUJAH/////////////////////////////////

 int halelujah_melody[] =  {
E, G, A, A,
A, G, E, E,
E, G, A, CE,
A, G, E, F, E, D, C}; //19

int halelujah_duration[] = {
  2,2,2,6,
  2,2,2,6,
  2,2,2,4,
  2,2,2,3,3,3,5
 };


//////////////////////STAR WARS//////////////////////

int starwars_melody[] =  {
C, G, F, E, D, C, G,
F, E, D, C, G,
F, E, F, D,
C, G, F, E, D, C, G,
F, E, D, C, G,
F, E, F, D}; //32

int starwars_duration[] = {2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
 };
```

```
//////////////////////CZERWONE KORALE//////////////////////////

int korale_melody[] =  {
F, E, G, F, E, D,
D, D, C, D, E,
F, E,
E, D, E,
F, E, G, F, E, D,
D, C, D, E, C, D, A, D }; //30 tonów

int korale_duration[] = {2,2,2,2,2, 2,2,2,2,2, 2,2,2,2,2, 2,2,2,2,2,
2,2,2,2,2, 2,2,2,2,2
 };

//////////////////////WIOSNA//////////////////////

int wiosna_melody[] =  {
C, E, E, E, D, C, G,
G, F, E, E, E, D, C, G,
G, F, E, F, G, F, E, D}; //30 tonów

int wiosna_duration[] = {
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2 };


//////////////////KOZIOŁEK MATOŁEK/////////////////

int koziolek_melody[] =  {
                NOTE_C5, NOTE_D5, NOTE_E5, NOTE_F5, NOTE_E5,
                     NOTE_D5, NOTE_C5, NOTE_C6, NOTE_G5, NOTE_C5,
                     NOTE_C6, NOTE_G5, NOTE_C5, NOTE_C6, NOTE_G5
};

int koziolek_duration[] = {
  1,2,1,1,2,2,2,2,2,2,2,2,2,2,4
 };

//////////////////////////STAR WARS////////////////////

int pirate_melody[] =  {

   NOTE_D4 ,   NOTE_D4 ,   NOTE_D4 ,   NOTE_D4 ,
   NOTE_D4 ,   NOTE_D4 ,   NOTE_D4 ,   NOTE_D4 ,
   NOTE_D4 ,   NOTE_D4 ,   NOTE_D4 ,   NOTE_D4 ,
   NOTE_D4 ,   NOTE_A3 ,   NOTE_C4 ,   NOTE_D4 ,
   NOTE_D4 ,   NOTE_D4 ,   NOTE_E4 ,   NOTE_F4 ,
   NOTE_F4 ,   NOTE_F4 ,   NOTE_G4 ,   NOTE_E4 ,
   NOTE_E4 ,   NOTE_D4 ,   NOTE_C4 ,   NOTE_C4 ,
   NOTE_D4 ,   NOTE_A3 ,   NOTE_C4 ,   NOTE_D4 ,
   NOTE_D4 ,   NOTE_D4 ,   NOTE_E4 ,   NOTE_F4 ,
   NOTE_F4 ,   NOTE_F4 ,   NOTE_G4 ,   NOTE_E4 ,
   NOTE_E4 ,   NOTE_D4 ,   NOTE_C4 ,   NOTE_D4 ,
   NOTE_A3 ,   NOTE_C4 ,   NOTE_D4 ,   NOTE_D4 ,
   NOTE_D4 ,   NOTE_F4 ,   NOTE_G4 ,   NOTE_G4 ,
   NOTE_G4 ,   NOTE_A4 ,   NOTE_AS4    NOTE_AS4 ,
   NOTE_A4 ,   NOTE_G4 ,   NOTE_A4 ,   NOTE_D4 ,
   NOTE_D4 ,   NOTE_E4 ,   NOTE_F4 ,   NOTE_F4 ,
   NOTE_G4 ,   NOTE_A4 ,   NOTE_D4 ,   NOTE_D4 ,
   NOTE_F4 ,   NOTE_E4 ,   NOTE_E4 ,   NOTE_F4 ,
   NOTE_D4 ,   NOTE_E4 ,   NOTE_A4 ,   NOTE_E5 ,
   NOTE_E4 ,   NOTE_E4 ,   NOTE_F4 ,   NOTE_D4 ,
   NOTE_E4 ,   NOTE_A4 ,   NOTE_C5 ,   NOTE_D5 ,
   NOTE_D5 ,   NOTE_D5 ,   NOTE_E5 ,   NOTE_F5 ,
   NOTE_F5 ,   NOTE_F5 ,   NOTE_G5 ,   NOTE_E5 ,
   NOTE_E5 ,   NOTE_D5 ,   NOTE_C5 ,   NOTE_C5 ,
   NOTE_D5 ,   NOTE_A4 ,   NOTE_C5 ,   NOTE_D5,
   NOTE_D5,    NOTE_D5  ,   NOTE_E5 ,   NOTE_F5  ,
   NOTE_F5  ,  NOTE_F5  ,   NOTE_G5  ,  NOTE_E5  ,
```

```
   NOTE_E5 ,   NOTE_D5  ,   NOTE_C5  ,  NOTE_D5  ,
   NOTE_A4  ,  NOTE_C5  ,   NOTE_D5 ,   NOTE_D5  ,
   NOTE_D5  ,  NOTE_F5  ,   NOTE_G5  ,  NOTE_G5  ,
   NOTE_G5  ,  NOTE_A5  ,   NOTE_AS5,    NOTE_AS5  ,
   NOTE_A5   ,  NOTE_G5   ,   NOTE_A5   ,  NOTE_D5   ,
   NOTE_D5   ,  NOTE_E5   ,   NOTE_F5   ,  NOTE_F5   ,
   NOTE_G5   ,  NOTE_A5   ,  NOTE_D5   ,  NOTE_D5   ,
   NOTE_F5   ,  NOTE_E5   ,  NOTE_E5   ,  NOTE_D5   ,
   NOTE_C5   ,  NOTE_D5   ,  NOTE_D5   ,  NOTE_E5
};

short pirate_duration[] = {

          1,         2,          1,         1,
          1,          1,          2,          1,
          2,          1,          2,          1,
          1,          1,          1,          2,
          2,          1,          1,
          2,          2,          1,          1,
          2,          2,          1,          1,
          1,          2,          1,          1,
          2,          2,          1,          1,
          2,          2,          1,          1,
          2,          2,          1,          1,
          3,          1,          1,          2,          2,
          1,          1,          2,          2,
          1,          1 ,          2,          2,
          1,          1,          1,          2,
          1,          1,          2,          2,
          2,          1,          2,          1,
          1,          2,          2,          1,
          1,          3,          1,          1 ,
          2,          2,          1,          1,
          2,          2,          1,          1 ,
          2,          2,          1,          1,
          1,          2,          1,          1 ,
          2,          2,          1,          1,
          2,          2,          1,          1 ,
          2,          2,          1,          1,
          3,          1,          1,          2,
          2,          1,          1,          2,
          2,          1,          1 ,          2,
          2,          1,          1,          1,
          2,          1,          1 ,          2,
          2,          2,          1,          2,
          1,          1 ,          2,          2,
          1,          1,          2,          2,
          2,          2,          1,          1,
          2,          3,          1,          1
 };

//////////////////KONIEC LISTY MELODII//////////////////////////

int song = 0;

void setup()
{
  pinMode(playPin, INPUT);
  pinMode(nextPin, INPUT);
  pinMode(prevPin, INPUT);
  Serial.begin(9600);
  lcd.print("Starting up!"); //wyświetlenie komunikatu początkowego
  lcd.autoscroll();
}

void loop()
{
  int playState = digitalRead(playPin);
```

```
  int prevState = digitalRead(prevPin);
  int nextState = digitalRead(nextPin);
 // lcd.setBacklight(HIGH);

  if (playState)  //jeśli wciśnięty przycisk play
 {
   lcd.begin(16, 2);

       switch(song) //pętla typu switch
       {
         case 0:
         {
           lcd.begin(16, 2);
           lcd.print("Wlasnie gram:");
           lcd.setCursor(0, 1);
           lcd.print(tytuly[song]); //przy odtwarzaniu melodii zostaje
wyświetlany tytuł tejże
            for (int thisNote = 0; thisNote < 18; thisNote++) //iteracja
po tablicy melodii - w części warunku została wpisana liczba
odpowiadająca ilości tonów w tablicy tonów dla każdej melodii
           {
             int thrones_durationn = 120 * thrones_duration[thisNote];
             tone(10, thrones_melody[thisNote],thrones_durationn);
             delay(thrones_durationn +30);
            }
           break;
         }
         case 1:
         {
           lcd.begin(16, 2);
           lcd.print("Wlasnie gram:");
           lcd.setCursor(0, 1);
           lcd.print(tytuly[song]);
            for (int thisNote = 0; thisNote < 39; thisNote++)
           {
             int wsrod_durationn = 120 * wsrod_duration[thisNote];
             tone(10, wsrod_melody[thisNote],wsrod_durationn);
             delay(wsrod_durationn +30);
           }
           break;
         }
         case 2:
         {
         lcd.begin(16, 2);
           lcd.print("Wlasnie gram:");
           lcd.setCursor(0, 1);
           lcd.print(tytuly[song]);
            for (int thisNote = 0; thisNote < 29; thisNote++)
           {
             int onatanczy_durationn = 120 *
onatanczy_duration[thisNote];
             tone(10, onatanczy_melody[thisNote],onatanczy_durationn);
             delay(onatanczy_durationn +30);
           }
           break;
         }
         case 3:
          {
         lcd.begin(16, 2);
           lcd.print("Wlasnie gram:");
           lcd.setCursor(0, 1);
           lcd.print(tytuly[song]);
            for (int thisNote = 0; thisNote < 14; thisNote++)
           {
             int rick_durationn = 120 * rick_duration[thisNote];
             tone(10, rick_melody[thisNote],rick_durationn);
             delay(rick_durationn +30);
           }
```

```
              break;
              }
            case 4:
            {
            lcd.begin(16, 2);
              lcd.print("Wlasnie gram:");
              lcd.setCursor(0, 1);
              lcd.print(tytuly[song]);
               for (int thisNote = 0; thisNote < 39; thisNote++)
              {
                int classic_durationn = 120 * classic_duration[thisNote];
                tone(10, classic_melody[thisNote],classic_durationn);
                delay(classic_durationn +30);
            }
              break;
            }
            case 5:
            {
            lcd.begin(16, 2);
              lcd.print("Wlasnie gram:");
              lcd.setCursor(0, 1);
              lcd.print(tytuly[song]);
               for (int thisNote = 0; thisNote < 272; thisNote++)
              {
                int pirate_durationn = 120 * pirate_duration[thisNote];
                tone(10, pirate_melody[thisNote],pirate_durationn);
                delay(pirate_durationn +30);
            }
              break;
            }
            case 6:
            {
              lcd.begin(16, 2);
              lcd.print("Wlasnie gram:");
              lcd.setCursor(0, 1);
              lcd.print(tytuly[song]);
               for (int thisNote = 0; thisNote < 18; thisNote++)
              {
                int starwars_durationn = 120 * starwars_duration[thisNote];
                tone(10, starwars_melody[thisNote],starwars_durationn);
                delay(starwars_durationn +30);
               }
               break;
            }
             case 7:
            {
            lcd.begin(16, 2);
              lcd.print("Wlasnie gram:");
              lcd.setCursor(0, 1);
              lcd.print(tytuly[song]);
               for (int thisNote = 0; thisNote < 18; thisNote++)
              {
                int underworld_durationn = 120 *
underworld_duration[thisNote];
                tone(10, underworld_melody[thisNote],underworld_durationn);
                delay(underworld_durationn +30);
               }
              break;
            }
            case 8:
            {
            lcd.begin(16, 2);
              lcd.print("Wlasnie gram:");
              lcd.setCursor(0, 1);
              lcd.print(tytuly[song]);
               for (int thisNote = 0; thisNote < 19; thisNote++)
              {
```

```
            int halelujah_durationn = 120 *
halelujah_duration[thisNote];
            tone(10, halelujah_melody[thisNote],halelujah_durationn);
            delay(halelujah_durationn +30);
           }
          break;
        }
        case 9:
        {
        lcd.begin(16, 2);
          lcd.print("Wlasnie gram:");
          lcd.setCursor(0, 1);
          lcd.print(tytuly[song]);
           for (int thisNote = 0; thisNote < 18; thisNote++)
          {
            int kotek_durationn = 200 * kotek_duration[thisNote];
            tone(10, kotek_melody[thisNote],kotek_durationn);
            delay(kotek_durationn +30);
           }
          break;
        }
        case 10:
        {
        lcd.begin(16, 2);
          lcd.print("Wlasnie gram:");
          lcd.setCursor(0, 1);
          lcd.print(tytuly[song]);
           for (int thisNote = 0; thisNote < 1; thisNote++)
          {
            int melody_durationn = 120 * melody_duration[thisNote];
            tone(10, melody[thisNote],melody_durationn);
            delay(melody_durationn +30);
           }
          break;
        }
        case 11:
        {
        lcd.begin(16, 2);
          lcd.print("Wlasnie gram:");
          lcd.setCursor(0, 1);
          lcd.print(tytuly[song]);
           for (int thisNote = 0; thisNote < 15; thisNote++)
          {
            int koziolek_duration_obj = 120 *
koziolek_duration[thisNote];
            tone(10, koziolek_melody[thisNote],koziolek_duration_obj);
            delay(koziolek_duration_obj +30);
           }
          break;
        }
        case 12:
        {
        lcd.begin(16, 2);
          lcd.print("Wlasnie gram:");
          lcd.setCursor(0, 1);
          lcd.print(tytuly[song]);
           for (int thisNote = 0; thisNote < 36; thisNote++)
          {
            int stolat_durationn = 120 * stolat_duration[thisNote];
            tone(10, stolat_melody[thisNote],stolat_durationn);
            delay(stolat_durationn +30);
           }
          break;
        }
        case 13:
        {
        lcd.begin(16, 2);
          lcd.print("Wlasnie gram:");
```

```
          lcd.setCursor(0, 1);
          lcd.print(tytuly[song]);
           for (int thisNote = 0; thisNote < 1; thisNote++)
          {
            int melodyja_durationn = 120 * melodyja_duration[thisNote];
            tone(10, melodyja[thisNote],melodyja_durationn);
            delay(melodyja_durationn +30);
           }
          break;
        }
        case 14:
        {
        lcd.begin(16, 2);
          lcd.print("Wlasnie gram:");
          lcd.setCursor(0, 1);
          lcd.print(tytuly[song]);
           for (int thisNote = 0; thisNote < 32; thisNote++)
          {
            int onatanczy_durationn = 120 *
onatanczy_duration[thisNote];
            tone(10, onatanczy_melody[thisNote],onatanczy_durationn);
            delay(onatanczy_durationn +30);
           }
          break;
        }
        case 15:
        {
        lcd.begin(16, 2);
          lcd.print("Wlasnie gram:");
          lcd.setCursor(0, 1);
          lcd.print(tytuly[song]);
           for (int thisNote = 0; thisNote < 1; thisNote++)
          {
            int melodia_durationn = 120 * melodia_duration[thisNote];
            tone(10, melody[thisNote],melodia_durationn);
            delay(melodia_durationn +30);
           }
          break;
        }
       }
     }
```

```
 if (prevState)  //jeśli wciśnięty przycisk "poprzedni"
 {
   lcd.begin(16, 2);
   lcd.print("Numer melodii:");
   song = --song%MAX_SONG_NUMBER;
   if (song<0)
   {
     song=MAX_SONG_NUMBER + song; //jeśli lista dotrwa do pozycji
poprzedzającej pierwszą piosenkę z tablicy piosenek, wtedy zostaje
wyświetlona pozycja ostatniej
   }
   lcd.print(song);
   lcd.setCursor(0, 1);
   lcd.print(tytuly[song]);
      delay(400);
}

 if (nextState)  //jeśli wciśnięty przycisk "następny"
 {
   lcd.begin(16, 2);
   lcd.print("Numer melodii:");
   song = ++song%MAX_SONG_NUMBER; //jeśli lista dotrwa do pozycji
poprzedzającej zero, wtedy zostaje wyświetlona pozycja ostatniej
piosenki z tablicy piosenek
```

```
    lcd.print(song); //wyświetlany numer piosenki
    lcd.setCursor(0, 1);
    lcd.print(tytuly[song]);
       delay(400);
}
}
```

d. Opis kluczowych zmiennych:

- thisNote – zmienna typu int sterująca pętlą służącą do przesuwania się po tablicy tonów konkretnej melodii
- song - zmienna typu int, zakres liczbowy jaki przyjmuje w programie to 0-15, każda liczba odpowiada pozycji melodii z tablicy piosenek

Kluczową funkcją wbudowaną, która jest używana to `Tone()`.

Opis funkcji Tone():

Funkcja `Tone()` umożliwia wygenerowanie przebiegu prostokątnego o wypełnieniu 50%. Na zadanym pinie. Długość trwania może zostać zadana, w przeciwnym razie funkcja działa do momentu wywołania `noTone()`. Pin może zostać podłączony do buzzera lub innego głośnika. Zakres możliwych do wygenerowania częstotliwości to 31 do 65535Hz.

Składnia funkcji:

`Tone(pin, częstotliwość)`

4. Specyfikacja zewnętrzna:

Dla uruchomienia (zapewnienia zasilania) układu Arduino musiał być podłączony do komputera za pośrednictwem kabla USB (po prezentacji projektu dokupiłem zasilacz wtyczkowy 12V/1,2A 15W DC, dzięki któremu wygoda użytkowania niewątpliwie wzrasta). Po podłączeniu zasilania na wyświetlaczu LCD pojawia się komunikat „Starting up!” – szafa grająca jest gotowa do użycia. Po naciśnięciu dowolnego przycisku zostaje wyświetlona pierwsza piosenka z listy, od teraz dostępne są 3 opcje:

1. Po naciśnięciu na lewy przycisk, zostaje wyświetlony numer oraz tytuł poprzedniej piosenki
2. Po naciśnięciu na środkowy przycisk, zostaje odegrana aktualnie wybrana melodia
3. Po naciśnięciu na prawy przycisk, zostaje wyświetlony numer oraz tytuł następnej piosenki

5. Wnioski z uruchamiania i testowania.

W pierwszej kolejności do przetestowania elementem był wyświetlacz LCD, w następnej kolejności buzzer, zaś na samym końcu obsługa przycisków. Każda melodia wymagała kilkukrotnego jej odtworzenia w celu wykrycia ewentualnych przekłamań w tonacji zapisanych dla każdej z osobna, lub nieprawidłowego przesuwania się po tablicy tonów. Nieodzownym elementem przygotowań było także odpowiednie zaplanowanie rozmieszczenia części na płytce stykowej dla zachowania dobrego poziomu estetyki oraz intuicyjności obsługi.

Ilość melodii początkowo miała być uzależniona od ilości przycisków, jednak bardziej optymalnym jakościowo rozwiązaniem okazało się użycie dwóch przycisków jako nawigacyjnych, a tylko jednego jako inicjalizującego odegranie wybranej melodii.

Układ działa zgodnie z początkowymi założeniami.

Podczas wykonywania projektu musiałem się zmierzyć m.in. z takimi problemami jak:

- Przechowywanie piosenek w formie dostępnej do ich odtworzenia
- Podłączenie całego układu, dopasowanie rezystorów, rozmieszczenie logistyczne podzespołów na płytce stykowej

Także nauczyłem się z kolei obsługi Arduino oraz środowiska programistycznego dla niego przeznaczonego, a także wyświetlacza LCD i przypomniałem sobie o podstawach elektroniki.