

# Music Player

## Introducere

- Proiectul consta intr-un player de muzica care reda melodii dupa un card SD.
- Acesta va beneficia de un ecran LCD pe care va fi proiectat numele melodiei care se reda si de led-uri RGB care isi vor schimba culoarea in functie de sunetul redat prin difuzor.
- De asemenea, melodiile vor putea fi schimbate inainte si inapoi cu ajutorul a doua butoane, iar sunetul va fi redat printr-un difuzor care va fi conectat la un amplificator pentru incrementarea sunetului.
- Am ales sa fac acest proiect din dorinta de a crea ceva practic si de care sa ma pot folosi in viitor.

## Descriere generală



## Hardware Design

### Lista de piese:

1. Arduino UNO
2. Led RGB
3. Rezistente 1k
4. Condesator
5. Fire tata-tata
6. Fire mama-tata
7. Modul de card SD
8. Card SD
9. Difuzor
10. Amplificator
11. Butoane
12. Ecran LCD
13. Breadboard
14. Difuzor SparkFun

## Schema cablaj



## Software Design

Bibliotecile folosite sunt:

1. SD.h
2. LiquidCrystal\_I2C.h
3. SPI.h

### Cod

```
#include <LiquidCrystal_I2C.h>
#include "SD.h"
#include "SPI.h"
#define Rpin1 10
#define Gpin1 9
#define Bpin1 8
#define Rpin2 14
#define Gpin2 15
#define Bpin2 16
#define Rpin3 17
#define Gpin3 18
#define Bpin3 19
```

```
uint8_t signal_values[] = {250, 240, 230, 220, 210, 200, 190, 180, 170, 160,
150, 140, 130, 120, 110, 100, 90, 80, 70, 60, 50, 40, 30, 20, 10};
// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);
File myFile;
// change this to make the song slower or faster
uint16_t tempo = 110;
uint8_t buzzer = 7;
short int melody[120];
uint8_t len = 0;
uint8_t notes;
volatile uint8_t song = 0, prev_song = 0;
unsigned long lastDebounceTime1 = 0, lastDebounceTime2 = 0, debounceDelay =
500;
```

```
void RGBColor(uint8_t red, uint8_t green, uint8_t blue)
```

```
analogWrite(Rpin1, red);
analogWrite(Gpin1, green);
analogWrite(Bpin1, blue);
```

```
analogWrite(Rpin2, red);
analogWrite(Gpin2, green);
analogWrite(Bpin2, blue);
```

```
analogWrite(Rpin3, red);
analogWrite(Gpin3, green);
analogWrite(Bpin3, blue);
```

```
void lights(uint8_t signal)
```

```
if (signal >= signal_values[0]) {
    // blue
    RGBColor(0, 0, 255);
} else if (signal >= signal_values[1]) {
    // Azure
    RGBColor(0, 255, 255);
} else if (signal >= signal_values[2]) {
    // Cyan
    RGBColor(0, 127, 255);
} else if (signal >= signal_values[3]) {
    // Aqua marine
    RGBColor(0, 255, 127);
} else if (signal >= signal_values[4]) {
    // Green
    RGBColor(0, 255, 0);
} else if (signal >= signal_values[5]) {
    // Yellow
    RGBColor(255, 255, 0);
} else if (signal >= signal_values[6]) {
    // Magenta
    RGBColor(255, 0, 255);
} else if (signal >= signal_values[7]) {
    // Rose
    RGBColor(255, 0, 127);
} else if (signal >= signal_values[8]) {
    // Orange
    RGBColor(255, 127, 0);
} else if (signal >= signal_values[9]) {
    // Red
    RGBColor(255, 0, 0);
} else if (signal >= signal_values[9]) {
    // Purple
    RGBColor(128, 0, 128);
} else if (signal >= signal_values[10]) {
    // Gold
    RGBColor(255, 215, 0);
```

```
} else if (signal >= signal_values[11]) {
    // Spring green
    RGBColor(0, 250, 154);
} else if (signal >= signal_values[12]) {
    // Turquoise
    RGBColor(64, 224, 208);
} else if (signal >= signal_values[13]) {
    // Indigo
    RGBColor(75, 0, 130);
} else if (signal >= signal_values[14]) {
    // Pink
    RGBColor(255, 192, 203);
} else if (signal >= signal_values[15]) {
    // Lavender
    RGBColor(230, 230, 250);
} else if (signal >= signal_values[16]) {
    // Chocolate
    RGBColor(210, 105, 30);
} else if (signal >= signal_values[17]) {
    // Sea green
    RGBColor(46, 139, 87);
} else if (signal >= signal_values[18]) {
    // Olive
    RGBColor(128, 128, 0);
} else if (signal >= signal_values[19]) {
    // Maroon
    RGBColor(128, 0, 0);
} else if (signal >= signal_values[20]) {
    // Midnight blue
    RGBColor(25, 25, 112);
} else if (signal >= signal_values[21]) {
    // Medium violet red
    RGBColor(199, 21, 133);
} else if (signal >= signal_values[22]) {
    // Orchid
    RGBColor(218, 112, 214);
} else if (signal >= signal_values[23]) {
    // Steel blue
    RGBColor(70, 130, 180);
} else if (signal >= signal_values[24]) {
    // Coral
    RGBColor(255, 127, 80);
} else {
    // White
    RGBColor(255,255,255);
}
```

```
void button1Pressed()
```

```
    unsigned long current = millis();
    if(millis() - lastDebounceTime1 > debounceDelay){
```

```
    lastDebounceTime1 = current;
    Serial.println(song);
    song++;
    song = song % 2;
}
```

void button2Pressed()

```
unsigned long current = millis();
if(current - lastDebounceTime2 > debounceDelay){
    lastDebounceTime2 = current;
    if(song != 5){
        prev_song = song;
        song = 5;
    } else if (song == 5){
        song = prev_song;
    }
}
```

void readFile(char \*file\_name)

```
Serial.print("Init SD card...");
if (!SD.begin(4)) {
    Serial.println("init failed!");
    while (1);
}
Serial.println("init done.");
// open the file for reading:
myFile = SD.open(file_name);
if (myFile) {
    Serial.println(file_name);
    uint8_t i = 0;
    // read from the file until there's nothing else in it:
    while (myFile.available()) {
        short int note = myFile.parseInt();
        melody[i] = note;
        if (i > 120)
            break;
        i++;
    }
    len = i;
    notes = len / 2;
    // close the file:
    myFile.close();
} else {
    // if the file didn't open, print an error:
    Serial.println("error opening file");
}
```

void sing(uint8\_t song\_id)

```
// this calculates the duration of a whole note in ms
int wholenote = (60000 * 4) / tempo;
int divider = 0, noteDuration = 0;
// iterate over the notes of the melody.
// the array is twice the number of notes (notes + durations)
for (uint8_t thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
  if (song_id != song) {
    return;
  }
  lights(abs(melody[thisNote]) % 255);
  // calculates the duration of each note
  divider = melody[thisNote + 1];
  if (divider > 0) {
    // regular note, just proceed
    noteDuration = (wholenote) / divider;
  } else if (divider < 0) {
    // dotted notes are represented with negative durations!!
    noteDuration = (wholenote) / abs(divider);
    noteDuration *= 1.5; // increases the duration in half for dotted notes
  }
  // we only play the note for 90% of the duration, leaving 10% as a pause
  tone(buzzer, melody[thisNote], noteDuration * 0.9);

  // Wait for the specified duration before playing the next note
  delay(noteDuration);
  // stop the waveform generation before the next note
  noTone(buzzer);
}
```

```
void play(char *song_name, uint8_t song_id)
```

```
  readFile(song_name);
  sing(song_id);
```

```
void setup()
```

```
// Start with the LEDs off.
pinMode(10, OUTPUT);
pinMode(9, OUTPUT);
pinMode(8, OUTPUT);
pinMode(14, OUTPUT);
pinMode(15, OUTPUT);
pinMode(16, OUTPUT);
pinMode(17, OUTPUT);
pinMode(18, OUTPUT);
pinMode(19, OUTPUT);

pinMode(2, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(2), button1Pressed, FALLING);
```

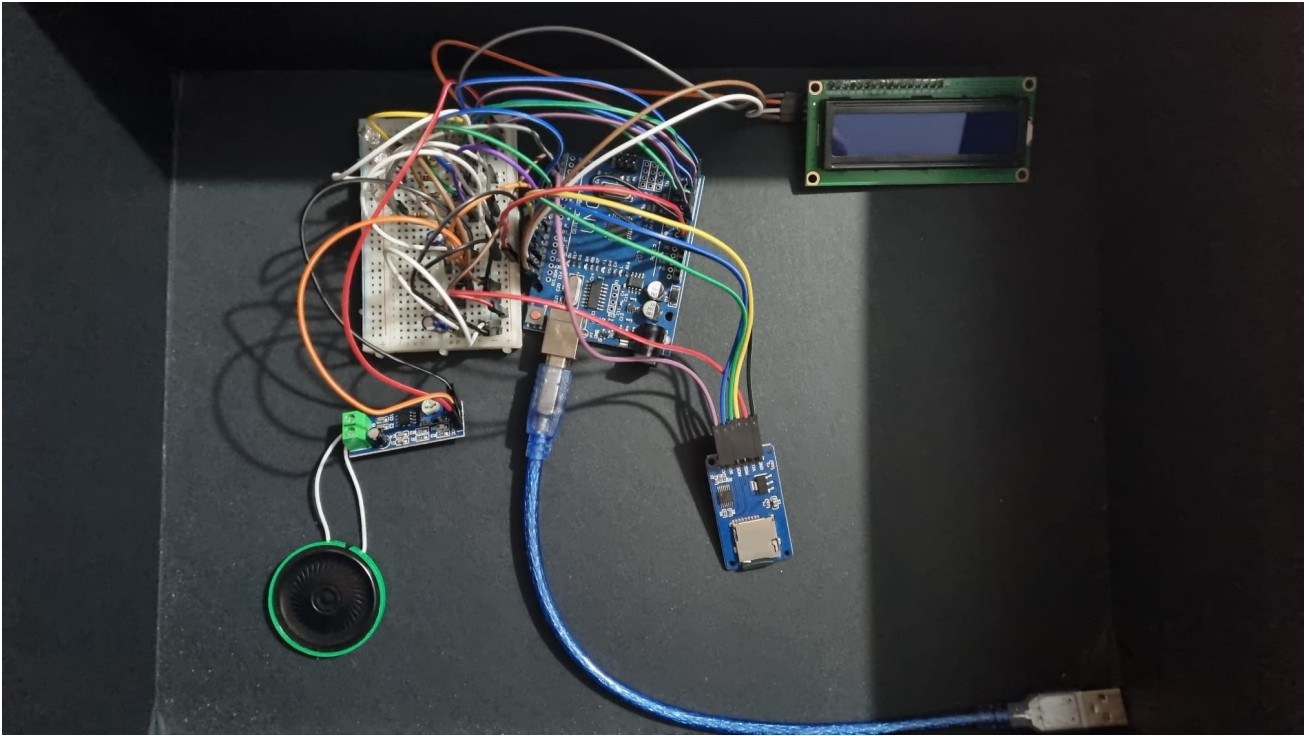
```
pinMode(3, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(3), button2Pressed, FALLING);
// Open serial communications and wait for port to open
Serial.begin(9600);
while (!Serial) {
    // wait for serial port to connect
}
// initialize the LCD
lcd.begin();
// Turn on the backlight and print a message.
lcd.backlight();
```

void loop()

```
lcd.clear();
switch(song) {
    case 0:
        lcd.clear();
        lcd.print("Game of thrones");
        play((char*)"got.txt", 0);
        break;
    case 1:
        lcd.clear();
        lcd.print("Fur Elise-");
        lcd.setCursor(3,1);
        lcd.print("Beethoven");
        play((char*)"furelise.txt", 1);
        break;
    default:
        break;
}
```

## Rezultate Obținute

Link catre demo: <https://www.youtube.com/shorts/H1VGBhjymu8>



## Concluzii

Am dorit inițial să implementez un media player care să redea fișiere audio .wav de pe un card SD și să adauge un efect luminos pentru a sincroniza culorile cu notele muzicale, însă din motive necunoscute, am întâmpinat dificultăți în redarea fișierelor .wav de pe cardul SD utilizând biblioteca TMRpcm.

Ca soluție alternativă, am decis să stochez mai multe fișiere .txt pe card, fiecare conținând notele corespunzătoare unei melodii. Am citit notele din fișierul selectat într-un vector și le-am redat folosind funcția "note" din Arduino IDE.

Am implementat un buton pentru schimbarea melodiilor și un alt buton pentru a pune pauza redării unei melodii. În timpul redării, cele 3 LED-uri își schimbă culorile în funcție de nota redată în prezent de difuzor.

Am încercat să îmbunătățesc procesul de eliminare a rezonanțelor provocate de contactele imperfecte ale butonului prin adăugarea unui condensator în serie cu fiecare buton. Cu toate acestea, butoanele tot nu funcționau corespunzător, așa că, am adăugat o metodă de debouncing pentru a detecta o singură apăsare de buton.

Am fost nevoit să folosesc o listă restrânsă de melodii, deoarece adăugarea ecranului LCD, și implicit a bibliotecii "LiquidCrystal\_I2C", supraîncărcă memoria plăcii Arduino și componentele nu mai funcționau corect.

După ce erau citite două melodii de pe cardul SD, memoria se încălzește și programul se bloca la citirea următoarei melodii.

Din punct de vedere hardware, versiunea finală a proiectului a suferit câteva modificări. Am economisit pini prin adăugarea a încă 2 LED-uri RGB și am optat pentru un LCD cu interfață I2C în

locul celui standard. De asemenea, am adăugat un amplificator LM386 pentru a amplifica sunetul unui difuzor SparkFun de 0.5W.

Dezvoltarea acestui proiect nu a fost una usoara. Pe parcursului acesteia am intampinat probleme in ceea ce priveste constructia hardware-ului cat si dezvoltarea corecta a software-ului, insa rezultatul final, desi este unul mai simplu decat mi-am propus initial, este unul complet functional.

## Download

[PDF Proiect](#)

## Jurnal

3 mai - documentatie initiala

14 mai - hardware

25 mai - software

## Bibliografie/Resurse

Diagrama bloc - <https://app.diagrams.net>

Schema cablaj - <https://fritzing.org>

LCD I2C - <https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>

SD library - <https://github.com/arduino-libraries/SD>

From:

<http://ocw.cs.pub.ro/courses/> - **CS Open CourseWare**

Permanent link:

[http://ocw.cs.pub.ro/courses/pm/prj2023/abirlica/music\\_player](http://ocw.cs.pub.ro/courses/pm/prj2023/abirlica/music_player)



Last update: **2023/05/25 21:12**