# Music Player

Nume: Eduard-Constantin Marin

Grupa: 331CAb

## Introducere

Proiectul consta intr-un Music Player cu doua moduri de functionare, care ii permite userului sa uploadeze melodii de pe laptop pe un card SD, si sa aleaga melodia pe care isi doreste sa o asculte, titlul acesteia fiind afisat pe un display LCD. Navigarea printre melodii se face prin intermediul butoanelor, iar volumul poate fi schimbat glisand cu mana de sus in jos sau invers, printr-un senzor de gesturi.

## Descriere generală

In cazul in care modul de upload este selectat, un fisier specificat de user va fi trimis byte cu byte de pe laptop, prin seriala, catre Arduino, care il va redirecta, folosind un buffer, catre cardul SD folosit ca storage. In timpul transferului, un array de leduri va afisa progresul.

Altfel, utilizatorul va folosi butoanele pentru a naviga printre fisierele stocate pe card, fiind afisat pe display-ul LCD numele fisierului la care a ajuns si durata melodiei din fisier.

In cazul in care decide sa asculte melodia din fisierul curent, un timer va incepe sa contorizeze timpul trecut de la inceputul melodiei, iar acesta va fi afisat pe display. Atunci cand opreste melodia, se aplica un efect de blinking asupra textului de pe display, pana cand fie utilizatorul continua ascultarea melodiei curente, fie trece la alta melodie. Pentru modificarea volumului speaker-ului, senzorul de gesturi va inregistra miscarile utilizatorului.

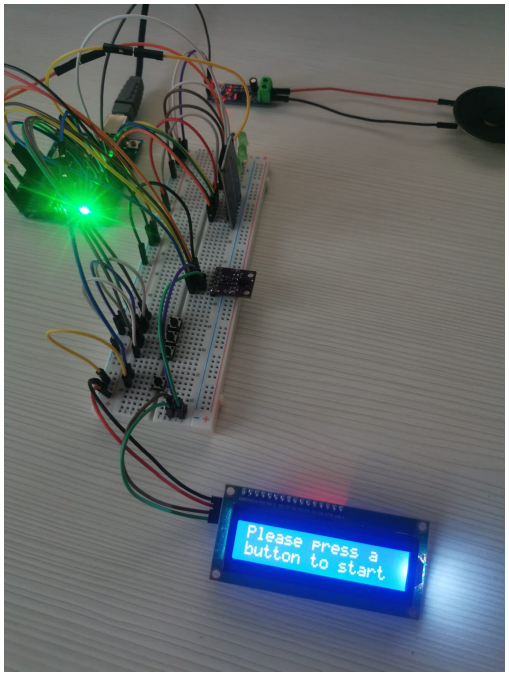
## Schema bloc

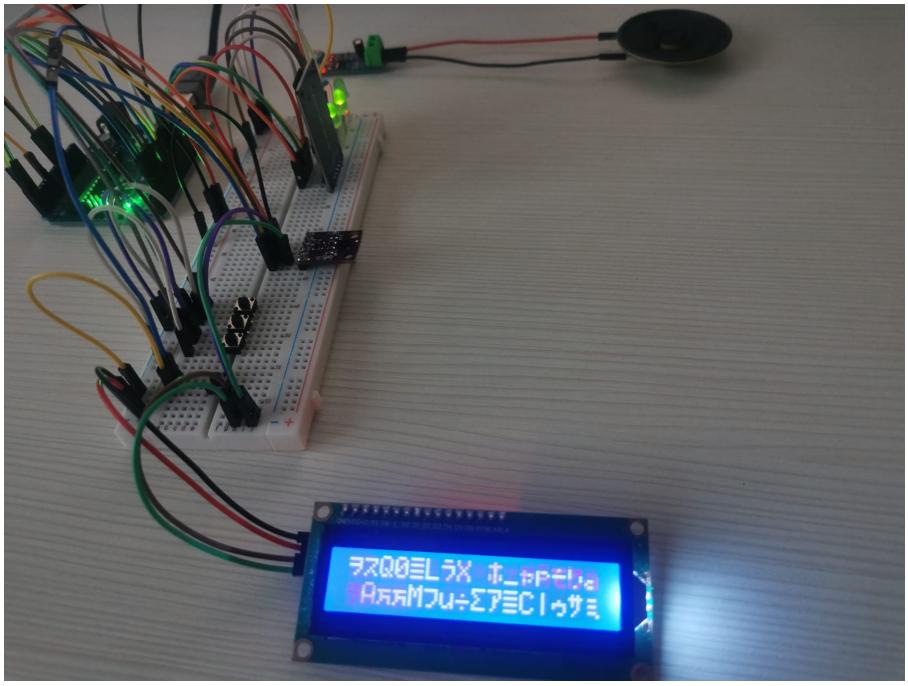
## Hardware Design

Listă piese:

- Arduino Uno
- Ecran LCD cu I2C
- Senzor gesturi
- SD card
- SD card reader
- Amplificator
- Difuzor 1W
- Breadboard
- 3 x Buton
- 5 x LED verde

Schema electrica:





## Software Design

Proiectul fost realizat in Arduino IDE.

Biblioteci folosite:

- SparkFun_APDS9960 - pentru a prelua datele de la senzorul de gesturi APDS9960
- LiquidCrystal_I2C - pentru a interactiona cu ecranul LCD-ul
- SdFat - citirea datelor stocate pe cardul MicroSD
- TMRpcm - pentru a putea reda, cu ajutorul speaker-ului, melodii in format WAV, cu:
  - 8 biti de rezolutie
  - 16 kHz rata de esantionare
  - mono audio

## Descriere implementare

## Redare melodii

Interactiunea cu melodiile pastrate pe cardul SD se face navigand prin sistemul de fisiere ca printr-o lista inlantuita. Fiecare fisier are numele format dintr-un index, de la 0 la numarul fisierelor de pe cardul SD, plus o extensie (.wav).

Am ales acest format din cauza limitarilor cu care venea biblioteca SD privitoare la numele acceptate pentru fisiere, si l-am pastrat pentru ca trebuia oricum sa folosesc un fisier aditional in care sa mapez fiecare melodiei cu durata ei, si nu avea sens sa consum memorie de 2 ori pentru a stoca intregul

nume al melodiilor, atat in fisierul de metadate, cat si ca nume in filesystem.

Astfel, pentru a putea afisa pe LCD titlul melodiei curente, pastrez continuu fisierul de metadate deschis, iar atunci cand se schimba melodia citesc linia care corespunde indicelui acesteia. Formatul unei linii foloseste ca si separator caracterul “|”:

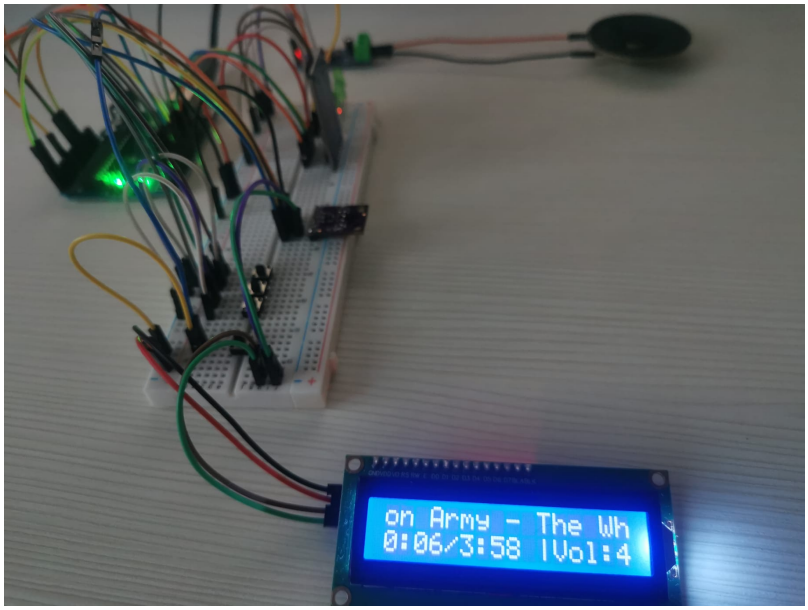
```
  Welcome to the Jungle - Guns'n Roses|4:40
  You Don't Get Me High Anymore|3:50
```

Ttilul melodiei si numele artistului sunt afisate intr-o maniera circulara, textul fiind iterat de la dreapta la stanga pe durata redarii. In cazul in care butonul de pauza este apasat, efectul de rotire se opreste.

```
if (musicOn && halfSecondPassed) {        // verifica trecerea unei jumatati
de secunda (marcata de ISR-ul timer-ului 2)
    halfSecondPassed = 0;
    halfSecondsCnt++;

    for (int i = 0; i < LCD_LEN; ++i) {   // shiftez spre stanga cu
halfSecondsCnt fiecare caracter
      lcd.setCursor(i, 0);
      lcd.print(songTitle[(halfSecondsCnt + i) % strlen(songTitle)]);
    }
  }
```

- **startSong(int i, char *title, char *len)** - citeste, de la inceputul fisierului de metadate, a i-a linie si extrage denumirea si durata folosind pointerii pasati ca parametri; in plus, afiseaza tot ce a citit + volumul curent
- **displayVolume(int volume)** - afiseaza pe LCD, in coltul din dreapta jos, volumul curent
- **getNumSongs()** - extrage numarul de intrari valide din fisierul de metadate, citindu-i prima linie
- **interpretGesture()** - in functie de gestul care a fost primit de la senzorul APDS9960, creste sau scade volumul aplicatiei
- **luckyStartup()** si **“luckyCleanup()“** sunt folosite pentru a activa/dezactiva **luckyMode-ul** - mod in care timp de cateva secunde sunt afisate caractere random pe LCD, iar apoi o este aleasa, tot random, o melodie pentru a fi ascultata (timpul cat se sta in luckyMode este contorizat cu ajutorul timer-ului 2).




## Lucky mode

User-ului i se alege o melodie random pentru a fi redata, iar timp de cateva secunde (timp care poate fi configurat din macro-ul LUCKY_TIME) sunt afisate caractere aleatorii pe display si se aude un sunet de zaruri pana cand se face alegerea melodiei urmatoare.

```
if (tenNsPassed && luckyModeLoop) {     // verific daca a timer 2 a generat
o intrerupere
    tenNsPassed = 0;
    luckyCnt++;

    int pos = luckyCnt % (LCD_WIDTH * LCD_LEN);     // se calculeaza pozitia
curenta unde
    lcd.setCursor(pos % LCD_LEN, pos / LCD_LEN);    // va fi inserat un
caracter random
    lcd.print((char)random(256));

    if (luckyCnt % 10 == 0) {           // in functie de luckyCnt, se
aprinde unul din cele 3 led-uri
      if (luckyCnt % 3 == 1) {
        PORTC |= (1 << (LED_PIN_1 - A0));
        PORTD &= ~(1 << LED_PIN_3);
      } else if (luckyCnt % 3 == 2) {
        PORTD |= (1 << LED_PIN_3);
        PORTB &= ~(1 << (LED_PIN_2 - PB0));
      } else {
        PORTB |= (1 << (LED_PIN_2 - PB0));
        PORTC &= ~(1 << (LED_PIN_1 - A0));
      }
    }

    if (luckyCnt == LUCKY_TIME) {       // se iese din lucky mode daca a
```

```
trecut timpul dedicat, LUCKY_TIME
      luckyCnt = 0;
      luckyModeEnded = 1;
    }
  }
```

# Transfer de fisiere

Initial am incercat sa transfer byte cu byte datele de pe laptop catre Arduino, pentru a implementa o politica de write-through de pe Arduino catre cardul SD, iar problema majora de care m-am lovit a fost buffer-ul foarte mic pe care il foloseste Arduino pentru transferurile seriale, mai exact de 64 de bytes. Astfel, in cazul in care nu asteptam sa fie consumate datele primite pe seriala, acestea erau suprascrise de noile date, pierzand foarte mult continut.

Pentru a rezolva aceastra problema, am implementat un protocol prin care laptop-ul trimite o fereastra de 32 de bytes, oprindu-se pana cand primeste o confirmare de la Arduino ca toate datele au fost procesate inainte sa trimita urmatorul batch.

Am marit baud rate-ul de la 9600 la 115200 pentru ca transferul de fisiere sa nu fie extrem de lent, si totusi sa fie stabil si sa nu pierd date.

Transferul se poate face daca si numai daca nu este redata in mod curent o melodie!

# Intreruperi

- **ISR(TIMER2_COMPA_vect)** - foloseste trei variabile pentru a marca trecerea unei secunde (pentru a actualiza pe LCD clock-ul), a unei jumatati de secunda (pentru a deplasa titlul spre stanga, astfel incat sa poata fi citit in totalitate pe LCD) si a unei sutimi de secunda, aproximativ:) - pentru a genera caractere random pe display.
- **ISR(INT0_vect), ISR(INT1_vect)** - detecteaza o intrerupere generata de senzorul exern de gesturi, si apasarea butonului de pauza
- **ISR(PCINT2_vect)** - restul butoanelor, care au input-urile multiplexate

# Probleme aparute

Initial foloseam biblioteca SD, in loc de SdFat, insa au aparut probleme in momentul in care iteram printre melodii cu ajutorul butoanelor, pentru ca trebuia sa fac citiri rapide din fisierul in care pastrez numele melodiilor asociate cu durata lor, iar atunci cand citeam string-uri de lungime mai mare de 10 caractere Arduino-ul se bloca si exista un delay foarte mare pana afisa pe LCD ceea ce citise din fisier. In plus, daca voiam sa redeschid un fisier inchis, tot proiectul devenea unresponsive, doar intreruperile de pe pinul INT1 putand fi folosite in continuare.

Odata cu folosirea bibliotecii SdFat, TMRpcm a incetat sa mai functioneze, si a trebuit sa o configurez

asstfel incat sa nu mai foloseasca biblioteca simpla SD.

## Rezultate obtinute

## Concluzii

Proiectul m-a adus mai aproape de lumea hardware, fiind prima ocazie, in afara lab-urilor de PM, cu care am interactionat cu un Arduino si am descoperit limitarile cu care vine, atat din punct de vedere al memoriei si al puterii de procesare, cat si din punct de vedere software, unele biblioteci fiind foarte ineficiente.

Ca si functionalitate, chiar daca este un proiect destul de voluminos, mi se pare foarte cool ca poate fi folosit practic. In plus, am invatat o tona de chestii lucrand la el, unele neplacute, cum ar fi ca nu poti sa te bazezi tot timpul pe hardware, in cazul meu timer-ul 2 avand probleme in modul CTC, si trebuind sa incerc diferite valori la configurare astfel incat sa il pot folosi.

## Download

em_musicplayer.zip

## Bibliografie

https://ocw.cs.pub.ro/courses/_media/pm/atmel-7810-automotive-microcontrollers-atmega328p_datasheet.pdf

https://learn.sparkfun.com/tutorials/apds-9960-rgb-and-gesture-sensor-hookup-guide/all

https://forum.arduino.cc/t/serial-buffer-limited-to-64-bytes/1025972/5

Export to PDF