Mini-DJ Board

rares.petruc@stud.acs.upb.ro
Rareș Petruc
Grupa 336CA

Introducere

Ce face

Proiectul reprezintă o plăcuță de DJ, cu funcționalități de ajustare a volumului, a vitezei de redare, un led RGB ce simulează un glob disco și un display de afișare a melodiei ce se aude în momentul curent.

Scop

Scopul proiectului este unul de relaxare/agrement pentru copii sau cei care sunt la început cu setup-ul complicat al unei plăci de DJ moderne. În același timp, adăugarea unei melodii noi pe cardul SD este un alt lucru ce se dorește a realiza foarte simplu prin intermediul acestui proiect.

Idee de bază

Îmi place destul de mult să pun muzică la diverse evenimente și am vrut o funcționalitate în plus pe lângă volum și eventual reclamele enervante, dacă nu ai un cont premium. În plus, am vrut să creez o atmosferă frumoasă cu ajutorul ledului RGB și a display-ului.

Utilitate

Proiectul poate fi folosit de oricine vrea să înțeleagă puțin mai low-level cum funcționează un echipament de DJ și care vrea să își asculte piesele favorite în mod repetat. Pentru mine e util că am înțeles mai bine cum să proiectez un astfel de echipament, dar o să îl folosesc și eu poate pentru melodiile favorite.

Descriere generală

Când prgramul rulează pe placuță, se începe cu prima melodie de pe SD Card, care va avea setat volumul definit de potențiometrul de volum și va avea viteză medie de redare. Totodată cu partea audio, odată cu frecvența melodiei, se aprinde și un led RGB, la aceeași frecvență, schimbându-și coloarea totodată cu schimbarea notei din melodie. Pe un ecran LCD este afișat numele melodiei ce se redă curent. Avem un buton, ce odată apăsat, schimbă melodia curentă cu următoarea din SD card, fiind vorba de o incrementare ciclică (când se ajunge la melodia cu indicele egal cu numărul maxim de melodii de pe SD card și este apăsat butonul, se trece la prima melodie). Potențiometrul pentru viteza de redare nu este suficient în schimbarea vitezei, întrucât am considerat că acesta poate fi lovit accidental, iar valoarea indicată de potențiometru este validată cu ajutorul celuilalt buton de pe breadboard, totodată cu apăsarea lui, schimbându-se viteza.

Hardware Design

Listă de componente

- Arduino Uno CH340
- Breadboard
- Ecran LCD 1602 IIC/I2C
- Modul adaptor SD
- LED RGB
- 2 x Buton
- Buzzer pasiv 5V
- 2 x Potențiometru liniar 10KΩ
- 3 x RESISTOR 1KΩ
- 1 x RESISTOR 100Ω
- Fire de legatură (mamă-tată și tată-tată)

Schemă electrică

Software Design

În realizarea proiectului, am utilizat următoarele biblioteci și funcții:

- <SPI.h> - pentru comunicarea între Arduino și cardul SD
- <SD.h> - pentru citirea de pe cardul SD
  - SD.begin(4): Inițializează cardul SD cu CS (CHIP SELECT) pe pinul 4
  - SD.open(melody_name, FILE_READ): deschide fișierul cu numele melody_name în modul de citire
- <LiquidCrystal_I2C.h> - pentru adaptorul I2C și display LCD
  - lcd.begin(16, 2): specifică dimensiunea display-ului, prima comandă înainte de utilizarea altor comenzi ce lucrează cu display-ul
  - lcd.clear(): curăță ecranul LCD de posibil junk (caractere lăsate de la afișările anterioare pe ecran)
  - lcd.setCursor(0, 0): setează poziția cursorului pe rândul 0, coloana 0
  - lcd.print("string_to_display"): afișează textul dintre ghilimele pe LCD

În **setup**, inițializez modul pinilor (spre exemplu butoanele ca *INPUT_PULLUP*, pinii spre ledul RGB ca *OUTPUT*). Totodată, configurez butonul corespunzător pinului PD2 pentru întrerupere (buton folosit pentru validarea vitezei de rotație stabilită de potențiometru) și pinul A0 pentru CAD. Se începe citirea primei piese de pe SD și se afișează pe ecranul LCD.

În **loop**, se verifică dacă butonul pentru schimbarea melodiei curente a fost apăsat, apoi dacă sfârșitul melodiei curente a avut loc (s-au difuzat toate notele, caz în care se ia de la capăt cu melodia tocmai terminată). În continuare, se difuzează nota curentă alături de incrementarea notei curente cântate, pentru a ști la fiecare pas unde ne aflăm în melodie.

Funcția **readCrtSong** deschide fișierul melodiei corespunzătoare numărului de apăsari ale butonului pentru schimbarea de melodie MOD numărul de melodii (implicit *3* în cazul default). Totodată, se citesc și primele informații importante despre fișierul în curs (numărul de note, tempo-ul minim și tempo-ul maxim al piesei).

Funcția **checkSwitchSong** verifică cu ajutorul unui debouncer dacă butonul pentru schimbarea melodiei s-a apăsat (vezi ieșire PD3 în schema electrică sau în pozele circuitului), iar în caz afirmativ, se incrementează numărul melodiei curente și se fac modificările necesare încheierii melodiei curente.

Restul funcțiilor utilizate, *RGB_color*, *playNote* sunt puțin ajustate față de referințele din bibliografie sau sunt triviale, ca de exemplu *displaySong*.

Pentru a adăuga o piesă nouă, se vor modifica macro-ul NO_SONGS din codul sursă alături de melodyNames unde se va adăuga stringul dorit pentru afișare corespunzător melodiei noi adăugate. Ulterior, pe cardul SD se va realiza un nou fișier cu numele melody{index_curent}.txt, unde {index_curent} este egal cu NO_SONGS înainte de actualizare (ex: dacă NO_SONGS era 3 și adăugăm o nouă melodie, aceasta se va numi melody3.txt).

Fișierele de pe cardul SD sunt următoarele:

Iată cum arată melody0.txt din poza de mai sus: Exemplu format melodie

Astfel, **fișierul** respectă următorul **format**:

<număr_note> <min_tempo> <max_tempo>

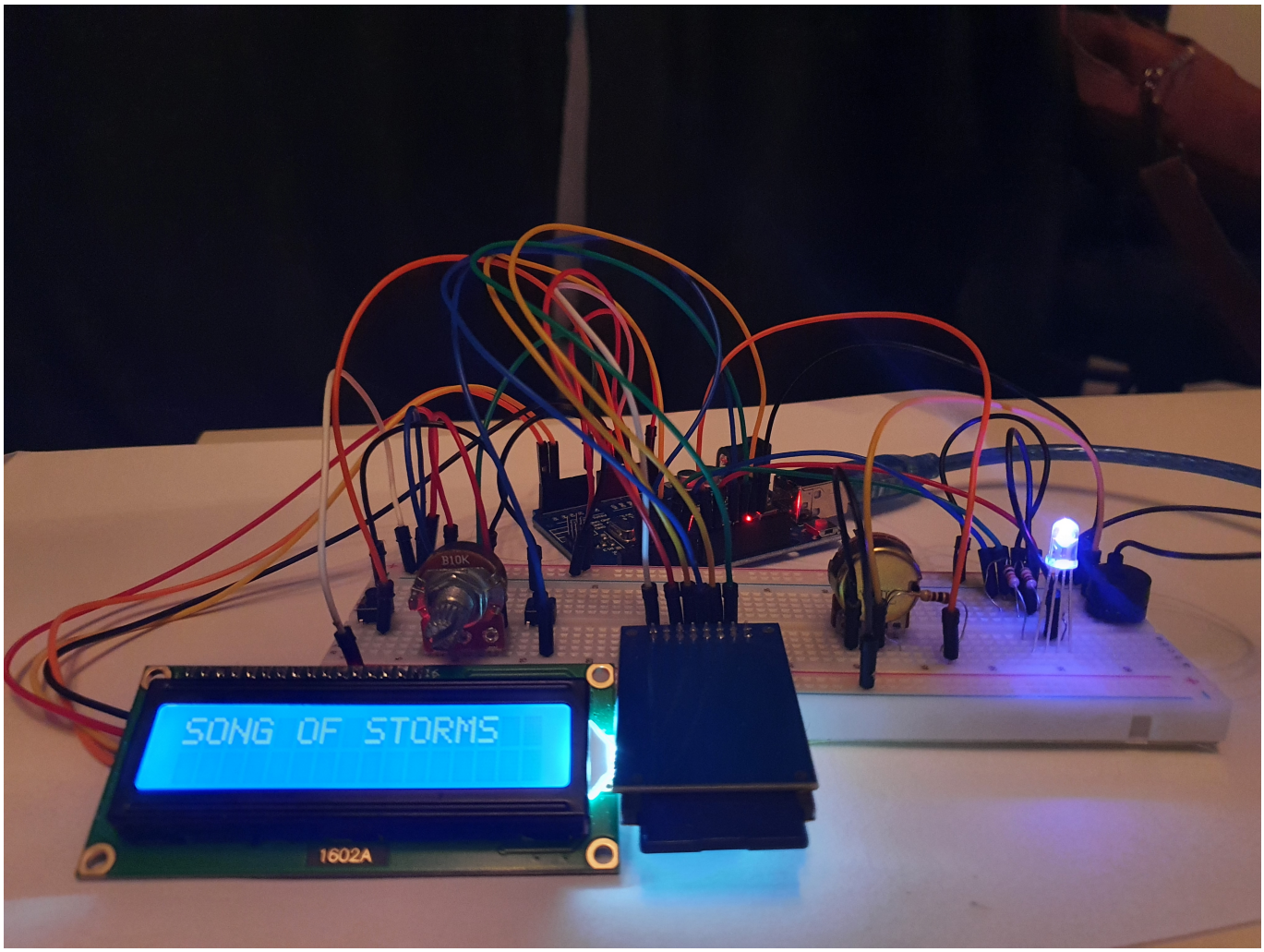
<notă>,<durată>,...

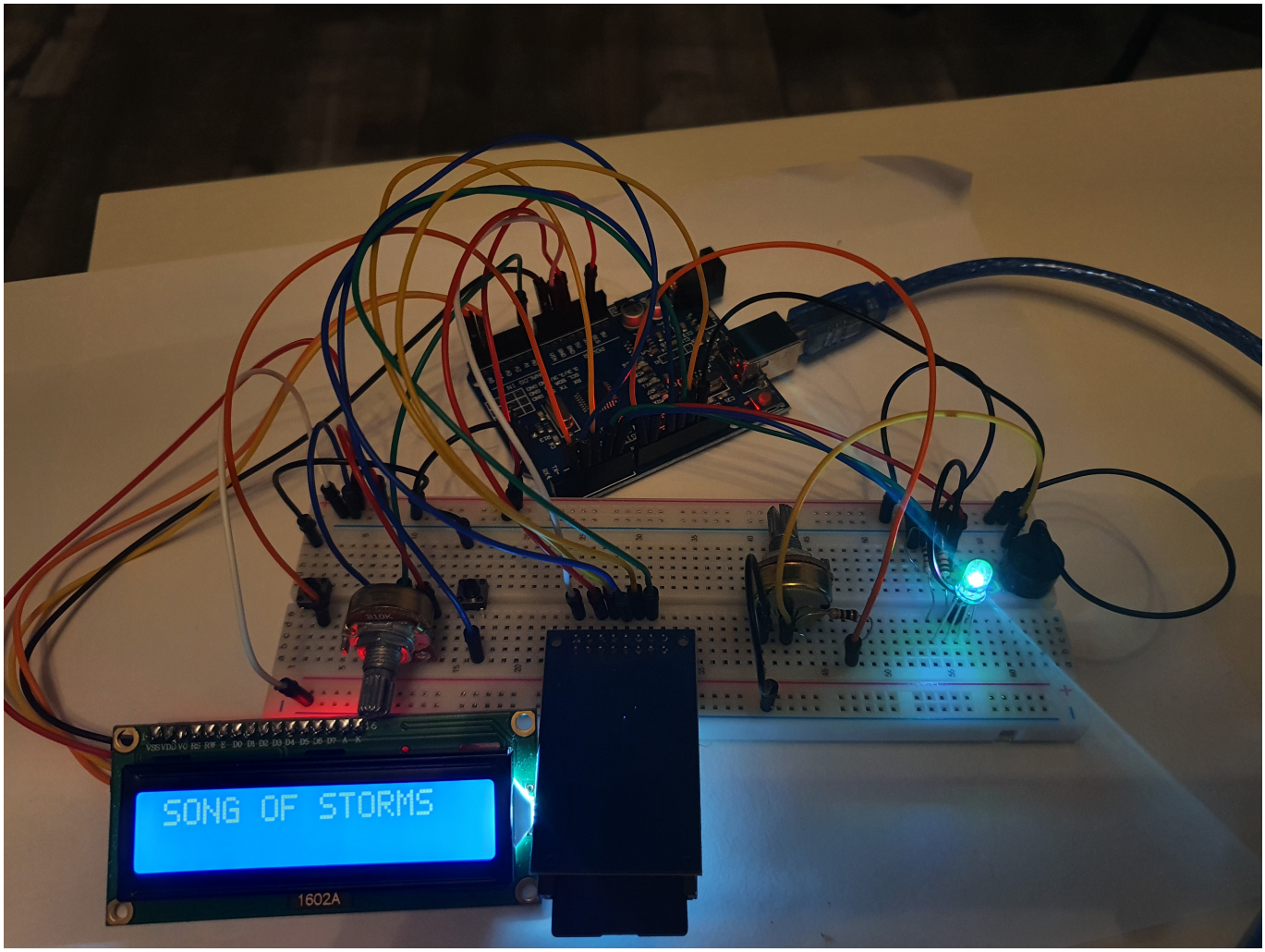
min_tempo și *max_tempo* le-am ales funcție de tempo-ul din repository-ul cu muzică pentru Arduino din Bibiliografie, scăzând 50, respectiv adunând 50 fată de valoarea tempo precizată pentru fiecare melodie aleasă de mine să ajungă pe cardul SD.

Nu este relevant dacă între diferitele variabile prezente în format este spațiu sau virgulă, se acceptă orice delimitatori, întrucat parsarea fișierelor cu melodii se realizează apelând metoda *parseInt* din clasa File, care ignoră caracterele dintre ele.

Rezultate Obţinute

Circuitul final





Demo

Link demo YouTube

Concluzii

A fost un proiect de-a lungul căruia am îmbinat o mulțime de cunoștințe din laboratoarele de PM, cum ar fi *ADC* pentru potențiometru de viteză, *PWM* pentru colorarea ledului RGB, *SPI* pentru comunicarea cu cardul SD, *I2C* pentru afișare pe display-ul LCD, întreruperi și multe altele.

Rezultatul este cel dorit. Am înțeles ceva mai bine, din practică de data asta, limitările de curent minim și maxim pentru unele componente (nu funcționau cum trebuie). Am reținut că ar fi trebuit să gândesc o așezare serioasă a pieselor alături de pinii de care sunt atașate încă de la început, pentru că am fost foarte aproape să rămân fără pinii de PWM de care aveam nevoie sau fără loc pe Breadboard, apelând la cabluri de tip mamă-tată.

Download

Arhivă cod sursă

Jurnal

- 13.04.2022: Confirmare temă proiect de către asistent
- 13.04.2022: Am comandat piesele
- 15.04.2022: Ridicare piese
- 15.04.2022: Realizare circuit + cod fără card SD și ecran LCD
- 19.04.2022: Fixare piese (pini îndoiți la adaptor SD, contrast prea mare pentru ecran LCD)
- 21.04.2022: Începere pagină wiki
- 04.05.2022: Adăugare melodii pe card SD și redarea acestora, alături de afișarea pe ecranul LCD (cod actualizat) + cable management refăcut
- 05.05.2022: Adăugare de comentarii în codul sursă, finalizare wiki

Bibliografie/Resurse

Export to PDF

Resurse Software:

https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce (Debouncer)

https://github.com/robsoncouto/arduino-songs (Melodii Arduino)

https://www.arduino.cc/en/reference/SD (Citire de pe cardul SD)

https://ocw.cs.pub.ro/courses/pm/lab/lab3-2022 (Colorare LED RGB)

Resurse Hardware:

https://www.youtube.com/watch?v=HmWf25-8uEY (SD Card and Adaptor configurare)

https://www.youtube.com/watch?v=xVC0X_PE_XE (LCD I2C configurare)

https://ocw.cs.pub.ro/courses/_detail/pm/lab/uno.jpg?id=pm%3Alab%3Alab1-2022 (PINOUT ARDUINO)