Sound mixer

Introducere

Where words fail, music speaks spunea Hans Christian Andersen. Muzica e cu noi oriunde, ne dă puterea de a ne exprima într-un mod unic, ne unește și ne face să ne simțim mai bine. Dar, mai presus de toate, muzica e despre creativitate. Iar de multe ori creativitatea înseamnă să vii cu propria reinterpretare asupra muncii altor oameni.

Astfel, m-am gândit să construiesc un dispozitiv care să ofere oricui și oriunde posibilitatea de a remixa melodii. Fără echipament costisitor și dificil de transportat ca cel de DJ, oricând te lovește inspirația poți începe să creezi fără întârziere!

Sound mixer-ul îți oferă posibilitatea selectării și redării melodiilor de pe un card MicroSD, pentru care aplică live funcții precum reglare tempo, low, mid, high, treble, bass și volum. Informații utile despre melodii și parametrii de mixare sunt afișate pe un ecran OLED, iar ție nu îți rămâne decât să conectezi o pereche de căști sau cel mai mare difuzor disponibil la port-ul audio jack și să îți asculți creația!

Descriere generală

Schemă bloc

Descriere

1. Modul Card MicroSD:

- Asigură legătura MCU cu sistemul de fișiere unde sunt stocate fișierele audio, cel de pe cardul microSD. La comanda microcontroller-ului, acesta începe transferul unei anumite melodii.

2. Potențiometru liniar:

- Se ocupă de reglarea tempo-ului, prin deplasarea slider-ului. Atunci când este situat la jumătatea cursei, tempo-ul este cel original al melodiei (viteză de citire de pe card implicită). La deplasarea din această poziție, viteza de redare (cea de citire de fapt) a melodiei este modificată.

3. Buton selectare melodie:

- Atunci când sistemul de fișiere este încărcat, lista de melodii de pe card sunt afișate pe display. Un buton este folosit pentru deplasarea prin această listă (NEXT).

4. Butoane pentru gestionarea melodiei:

- După ce am ajuns în listă la melodia preferată, folosim butoanele de play și stop pentru redare, respectiv revenirea la lista de melodii.

5. Arduino Uno:

- Controlează toate celelalte componente. Comunică modulului microSD fișierul ce va fi transferat pentru redare; preia datele de la acesta și le direcționează către modulul de audio tune; citește periodic valoarea indicată de potențiometru și ajustează viteza de citire a melodiei dacă este necesar (comunică modulului microSD o schimbare a vitezei de transfer); trimite date despre statusul întregului proces la display pentru a fi afișate.

6. Display OLED:

- Afișează date precum lista de melodii, minutul și secunda la care ne aflăm în redarea melodiei curente, tempo-ul curent atunci când ajustăm poziția potențiometrului etc.

7. Modul Amplificare Audio:

- Amplifică semnalul audio analogic slab provenit de la output-ul Arduino pentru a face posibilă redarea melodiei la un difuzor mai puternic decât căști clasice de telefon, dar și pentru a putea distinge modificările efectuate asupra semnalului de către modulul de audio tone tunning.

8. Modul Audio Tone Tunning:

- Prelucrează semnalul audio analogic pentru a ajusta diverse atribute ale acestuia și a-l altera astfel.

9. Modul Jack Audio Stereo 3.5 mm:

- Preia semnalul audio analogic prelucrat și asigură legătura cu un difuzor.

Hardware Design

Listă de piese

- Arduino Uno R3 AtMega328P
- Modul card microSD
- Modul Jack Audio Stereo 3.5 mm
- Display OLED 1.3 inch
- Modul Audio Tone Tunning
- Modul Amplificator Audio
- Potențiometru liniar
- Butoane

Schemă electrică

Conectarea componentelor:

1. Modul Card MicroSD: Un pin conectat la GND, unul la VCC, iar restul pinilor SCK, MISO, MOSI, CS conectați la pinii specifici de pe Arduino UNO pentru aceste semnale (PB5, PB4, PB3, PB2).

2. Potențiometru liniar Un pin conectat la GND, unul la VCC, iar al treilea pin conectat la ADC0 (PC0).

3. Butoane Conectate la GND și pini de pe placă, cu rezistența de pull-up activată, astfel:

- stop - PD4
- play - PD3
- next - PD2

4. Display OLED Un pin conectat la GND, unul la VCC, iar pinii SCL și SDA concectați la pinii specifici de pe Arduino UNO pentru aceste semnale (PC5 și PC4).

5. Modul Jack Audio Stereo 3.5 mm Pinii conectați astfel:

- TIP - PB1 (output PWM Timer 1)
- RING1 - PB1 (replicăm output mono în stereo)
- RING2 - GND
- SLEEVE - GND

6. Modul Audio Tone Tunning Un cablu Jack 3.5 mm - RCA face legătura între modulul Jack 3.5 mm descris mai sus ce primește output-ul PWM și intrarea acestui modul.

7. Modul Amplificator Audio Un cablu 2.54-3p face legătura între ieșirea modulului de Tone Tunning și intrarea acestui modul.

8. Difuzor Conectat prin două fire de cupru la ieșirea modului de amplificare.

9. Sursă de alimentare PC 220V Conectată la priză pentru alimentare, iar de la ieșire am folosit pinii de -12V, 12V, GND pentru alimentarea modulului de Audio Tone Tunning, respectiv cei de 5V, GND pentru alimentarea modulului Amplificator.

Software Design

GitHub: https://github.com/lucianpandelica/Sound-mixer

Mediu de dezvoltare: PlatformIO, Visual Studio Code

Biblioteci utilizate:

- Petit FatFs - FAT file system library: pentru citirea melodiilor de pe cardul microSD
- C Library for SSD1306 0.96” OLED display: pentru afișarea informațiilor pe display

Detalii implementare:

- Citirea butoanelor se face folosind întreruperi de tip pin-change interrupt și rezistențele de pull-up interne.
- Pentru citirea potențiometrului liniar am folosit funcția implementată în laboratorul de ADC (myAnalogRead).
- Pentru citirea de pe cardul microSD am folosit scheletul și implementarea din laboratorul de SPI, pe care l-am modificat pentru a adăuga funcționalități noi / a oferi suport pentru cazuri netratate.
- Pentru afișarea pe display prin I2C am folosit funcțiile din biblioteca menționată mai sus.
- Redarea melodiei, afișarea timpului parcurs în redarea acesteia și debounce-ul butoanelor se realizează folosind timere.
- Redarea melodiei se realizează prin output PWM al Timer1.

Programul configurează pentru început toate elementele folosite în implementare (butoane, întreruperi, timere, display, ADC). Apoi, se montează sistemul de fișiere de pe card folosind funcțiile din biblioteca Petit FatFs. Într-o buclă for se verifică în mod constant dacă valoarea citită de ADC (shiftată pentru a intra în intervalul 0-7) și-a modificat valoarea. Tot aici se verifică dacă unul dintre cele trei butoane a fost apăsat (s-a setat un flag de apăsare pentru butonul respectiv în întrerupere), iar în caz afirmativ se deviază execuția spre funcțiile aferente.

Pentru schimbarea melodiei se apelează funcția next_file(), iar pentru redarea ei funcția play(). Butonul de stop este verificat (și are efect) în cadrul funcției continue_play() utilizată în funcția play() - toate aceste funcții fiind implementate în laboratorul de SPI.

Timer-ul 2, configurat să numere milisecunde, este utilizat pentru efectuarea debounce-ului pentru butoane (reținem timpul global la care a fost apăsat un buton și verificăm dacă diferența dintre timpul global actual și cel reținut la apăsare este mai mare decât un prag setat prin experimentare). Timer-ul 1 este folosit pentru FastPWM pe 8 biți, pentru redarea melodiei. Timer-ul 0 este folosit pentru citirea melodiei de pe card, mai exact pentru avansarea bufferului în care citim octeții.

Viteza de redare a melodiei este modificată prin schimbarea sample rate-ului, înainte de începerea citirii unui fișier. Astfel, este luată în calcul valoarea citită de ADC și introdusă într-o formulă care să crească viteza proporțional cu valoarea indicată de potențiometru.

Durata de redare a melodiei curente este și ea manipulată în funcție de tempo, pentru a ilustra realitatea (doar o aproximare, rezultatul nu este exact).

De menționat că am încercat folosirea output-ului pe 16 biți pus la dispoziție de Timer 1 pentru a obține o redare la calitate mai bună, dar am întâmpinat dificultăți în adaptarea codului existent și am abandonat ideea pentru a mă concentra pe alte aspecte din implementare.

Am folosit exclusiv lucru cu registrii în implementarea proiectului, cu excepția folosirii bibliotecii pentru display.

Rezultate Obţinute

Ansamblul obținut nu este tocmai ușor portabil, așa cum îl imaginam inițial (din cauza necesității sursei de alimentare), dar cu excepția acesteia are dimensiuni reduse.

Concluzii

Prin acest proiect am învățat extrem de multe lucruri noi, mai ales când vine vorba de hardware (de la lipit pinii pentru o componentă la studiat dacă o componentă se poate adapta de la curent alternativ la continuu), iar, mai ales, am învățat să găsesc soluții folosind ceea ce am deja la dispoziție (de exemplu, alternativa la folosirea sursei de alimentare de la un PC vechi era achiziționarea unui transformator cu tole care, pe lângă faptul că avea timpi de livrare de ordinul săptămânilor, costa și destul de mult:)

Download

mihai_pandelica_proiect_pm.zip

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

- https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
- https://www.circuito.io/blog/arduino-uno-pinout/
- https://ocw.cs.pub.ro/courses/pm/lab/lab0-2023
- https://ocw.cs.pub.ro/courses/pm/lab/lab2-2023
- https://ocw.cs.pub.ro/courses/pm/lab/lab3-2023-2024
- https://ocw.cs.pub.ro/courses/pm/lab/lab4-2023-2024
- https://ocw.cs.pub.ro/courses/pm/lab/lab5-2023-2024
- https://github.com/Matiasus/SSD1306
- https://github.com/greiman/PetitFS
- https://en.wikipedia.org/wiki/Power_supply_unit_(computer)

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF