Nostalgia Music Player - Raduta Lavinia-Maria 333CA

Introducere

Absolut nicio petrecere nu scapa de melodiile anilor 2000. De ce? Pentru ca au un ritm anume, au acel vibe de discoteca, iti aduc aminte de anii in care erai copil, nu ai aveai griji, facultate si teme. In plus, mai greu gasesti melodii pe care sa le stie toata lumea sa se ridice de la masa, sa cante (cu tot sufletul) si sa danseze.
M-am gandit astfel ca daca tot vreau sa fac un player de muzica, sa fie totusi unul cat de cat autentic. Azi doar deschizi Spotify si te conectezi la boxa, inainte aveai nevoie de Cd, Cd player etc. Asa ca proiectul meu incearca sa aduca nostalgia anilor 2000 combinand tehnologia cu amintirea de a veni dupa tine cu casetofonul.
Astfel, player-ul realizat poate reda melodi in format mono, de pe un card SD

Descriere generală

Functionalitate

Nostalgia player-ul realizat are urmatoarele functii:

- butoane pentru controlul melodiilor:
  - play/pause
  - next track
  - previous track
  - volume up
  - volume down
- buton de shuffle, care permite amestecarea ordinii melodiilor din playlist

Pentru ca atmosfera sa fie si mai bine intretinuta, am adaugat si 2 LED-uri care pulseaza in ritmul melodiei.

Pe un ecran LCD vor fi afisate informatii despre:

- melodia curenta - titlu, timpul scurs de la inceputul melodie
- informatii despre actiuni:
  - daca optiunea de shuffle este activata, pe LCD va aparea un simbol
  - la schimbarea melodiei curente, apare timp de putin timp un simbol de next/prev
  - cand se apasa pe butoanele de volume up/down se afiseaza prin intermediul unei bare de stare,

cat si printr-un numar nivelul volumului
- statusul player-ului - la inceput, inainte ca playerul sa fie pornit se va afisa un mesaj care anunta utilizatorul ca poate da play melodiei

Formatul melodiilor

Melodiile vor fi stocate pe un card SD si pentru a avea o calitate cat de cat decenta, vor avea urmatoarele specificatii:

- formatul va fi **.wav**
- Sampling Rate de **16000Hz**
- canalul audio de tip **Mono**
- Bit Resolution de **8 biti**

Schema Bloc

Hardware Design

Lista de componente

- Arduino Uno R3
- Ecran LCD 1602 + modul I2C
- Difuzor 1W
- Modul amplificator audio LM386
- Modul Card microSD + card microSD
- 2 x LED RGB
- 2 x rezistente 220 Ω
- 6 x Buton
- Modul microfon MAX4466, cu amplificare si castig reglabil

Schema electrica

Proiectul inainte de a realiza cutia exterioara

Software Design

Setup intreruperi

Apasarea celor 6 butoane declanseaza intreruperi, pentru ca microprocesorul sa nu citeasca la fiecare loop starea butoanelor, care de cele mai multe ori nu se schimba.
Butoanele folosesc intreruperile:

- External Interrupt Request 0 (INT0) - Next (pe pinul **PD2**)
- Pin Change Interrupt Request 0 (PCINT0) - Pause (pe pinul **PB0 - PCINT0**)
- Pin Change Interrupt Request 2 (PCINT2)
  - Shuffle - pe pinul **PD4 - PCINT20**
  - Prev - pe pinul **PD5 - PCINT21**
  - Volume Up - pe pinul **PD6 - PCINT22**
  - Volume Down - pe pinul **PD7 - PCINT23**

Astfel folosesc functia interrupts_setup() pentru setarea directiei pinilor si pentru setarea tuturor intreruperilor.

```
void interrupts_setup() {
  cli();

  // input pins
  DDRD &= ~(1 << PD5) & ~(1 << PD4) & ~(1 << PD6) & ~(1 << PD7) & ~(1 << PD2);
  DDRB &= ~(1 << PB0);

  // input pullup
  PORTD |= (1 << PD4) | (1 << PD5) | (1 << PD6) | (1 << PD7) | (1 << PD2);
  PORTB |= (1 << PB0);

  // external INTERRUPT
  EIMSK |=  (1 << INT0);

  // Interrupt constrol register
  PCICR |= (1 << PCIE2) | (1 << PCIE0);

  // interrupt PIN CHANGE
  PCMSK2 |= (1 << PCINT20) | (1 << PCINT21) | (1 << PCINT22) | (1 << PCINT23);
  PCMSK0 |= (1 << PCINT0);

  // falling edge of INT0
  EICRA |= (1 << ISC01);
```

```
  sei();
}
```

Iar rutinele de tratare a intreruperilor seteaza flag-uri de tip bool pentru fiecare actiune.

```
// button controls bools
volatile bool next_press = false;
volatile bool prev_press = false;
volatile bool up_vol_press = false;
volatile bool down_vol_press = false;
volatile bool pause_press = false;
volatile bool shuffle_press = false;

// next button - PD2 - INT0
ISR(INT0_vect) {
  next_press = true;
}

// PCINT2: shuffle - PD4, prev - PD5, down - PD6, up - PD7
ISR(PCINT2_vect) {
  if ((PIND & (1 << PD4)) == 0) {
    shuffle_press = true;
  } else if ((PIND & (1 << PD5)) == 0) {
    prev_press = true;
  } else if ((PIND & (1 << PD6)) == 0) {
    up_vol_press = true;
  } else if ((PIND & (1 << PD7)) == 0) {
    down_vol_press = true;
  }
}

// PCINT0: pause/play - PB0
ISR(PCINT0_vect) {
  if ((PINB & (1 << PB0)) == 0) {
    pause_press = true;
  }
}
```

Debounce butoane

Pentru a detecta corect apasarile butoanelor, pentru fiecare dintre butoane am aplicat o logica de debounce, care consta in numararea milisecundelor care au trecut de la ultima apasare. Daca numarul acestora este mai mic decat thresholdul ales (de 500ms) atunci detectarea apasarii a fost gresita si este ignorata (flagul corespunzator este setat pe false).

```
volatile unsigned long last_debounce = 0;
```

```
volatile unsigned long current_millis = millis();
const unsigned long debounce_delay = 500;

if (press_flag) {
    unsigned long current_millis = millis();
    if (current_millis - last_debounce_next < debounce_delay) {
      press_flag = false;
    } else {
      // button was really pressed => do work
    }
  }
```

Redarea Melodiilor

Pentru a reda melodiile de pe cardul SD, am folosit biblioteca TMRpcm.
Feature-uri ale bibliotecii:

- foloseste biblioteca standard SD.h pentru a accesa fisierele de pe cardul SD
- reda melodiile in mod asincron, astfel permite executia altui cod in loop in timpul redarii, fara ca melodia sa se opreasca
- accepta fisiere de tip .wav cu urmatoarele caracteristici:
  - rezolutie audio de 8 biti (fiecare esantion audio are 8 biti - ceea ce inseamna ca exista 256 de niveluri de amplitudine posibile)
  - frecventa de esantionare (sample rate) a sunetului intre 8 si 32kHz (frecventa de esantionare = cate esantioane audio sunt redate pe secunda)
  - Mono = un singur canal audio care va fi redat pe toti pinii de iesire
- permite accesarea metadatelor fiecarei melodii (nume, artist etc)
- foloseste un singur timer, TIMER1, restul fiind disponibile utilizatorului

Cum functioneaza?

- biblioteca utilizeaza un buffer cu dimensiunea implicita de 128 de bytes, in care stocheaza datele audio care vor fi redate luate de pe cardul SD
- folosind timerul 1, de fiecare data cand se genereaza o intrerupere, bufferul este din nou umplut, iar datele sunt trimise catre modulul de redare audio
- timerul este folosit pentru a genera intreruperi la un interval regulat de timp, pentru ca redarea melodiilor sa fie fluenta

Modul de folosire al bibliotecii:

```
TMRpcm audio;                   // instance of the TMRpcm class.
audio.play(filename);          // plays a file
audio.speakerPin = 9;           // the pin on which the speaker is
connected
audio.isPlaying();              // returns 1 if music playing, 0 if not
audio.pause();                  // pauses/unpauses playback
audio.volume(0);                // 1 (up) or 0 (down) to control volume
audio.setVolume(0);             // 0 to 7. Set initial volume level
audio.listInfo(filename, buf, n); // returns info from metadata in the
```

```
buffer. 0 - title, 1 - artist, 2- album
```

Pentru ca biblioteca SD limiteaza lungimea numelui fisierelor la 8 caractere + extensie, am decis ca cel mai ok mod de a le denumi sa fie 01.wav, 02.wav etc. Astfel, este super usor de a construi numele fisierului avand indexul melodiei pe care vreau sa o redau.

Microfonul si LED-urile

La fiecare interatie a loop-ului, daca o melodie este in momentul curent redata, se va citi valoarea de pe pinul A1, pe care e conectat microfonul, folosind analogRead. Valoarea citita astfel este intre 0 si 1023 si pentru a mapa aceasta valoare intre 0 si 255 (valoare necesara pentru analogWrite) folosesc functia map().
LED-urile sunt legate pe pinul PD3, care foloseste Tmer2 pentru PWM (ceea ce e ok, biblioteca de audio foloseste doar Timer1) si daca valoarea citita de la microfon e mai mare de un threshold (ales in functie de testele efectuate si de castigul ajustat al microfonului) cu analogWrite se transmite intensitatea cu valori intre 0-255. Altfel se transmite 0.

```
if (!paused) {
  int analogValue = analogRead(A1);  // read value from A1
  int intensity = map(analogValue, 0, 1023, 0, 255);  // map value between 0
and 255
  if (analogValue > 700) {
    analogWrite(3, intensity);
  } else {
    analogWrite(3, 0);  // turn off LED when analogValue under threshold
  }
}
```

Scrierea pe LCD

Pentru scrierea pe LCD-ul cu modul I2C am folosit biblioteca LiquidCrystal_I2C care ofera o interfata mult mai usor de folosit in comunicarea I2C.

- in etapa de initializare, aceasta stabileste parametrii comunicarii I2C
- atunci cand se apeleaza functii de control ale LCD-ului, biblioteca construieste pachete de date care contin informatii necesare si le trimite pe linia SDA catre LCD
- pentru sincronizarea transferului de date intre Arduino si LCD (cand datele sunt valide si pot fi citite) se foloseste linia SCL

Principalele functionalitati pe care le-am folosit din aceasta biblioteca:

```
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address and number of columns and
lines
lcd.init();                         // initializing I2C communication
lcd.backlight();                    // turn on backlight
```

```
lcd.clear();                          // clear LCD
lcd.createChar(SHUFFLE_C, shuffle_char); // create a custom character
lcd.write(SHUFFLE_C);                 // printing a custom character on
the LCD
lcd.setCursor(column, line);          // set cursor befor writing
lcd.print("starting...");              // printing a string on the LCD
```

Pentru a crea un caracter custom, am declarat un array de bytes, cu 0 pe pozitiile pe care vreau sa fie stinse punctele si 1 unde vreau sa fie aprinse. Fiecare caracter are asociat un numar (define-uri declarate in extras.h)

```
#define SHUFFLE_C 0
byte shuffle_char[] = {
  B00000,
  B01010,
  B01010,
  B01010,
  B01010,
  B00100,
  B01010,
  B01010
};
```

Rezultate Obţinute

Sunt multumita de cum a iesit proiectul, mereu fiind insa loc de imbunatatiri. Initial voiam sa folsesc un senzor de gesturi pentru a controla activitatea player-ului, insa nu am reusit sa integrez toate componentele. cel mai probabil microcontroller-ul ramanea fara memorie, asa ca am inlocuit aceasta functionalitatea cu butoane (sa fie mai nostalgic)
O alta chestie pe care nu am resit sa o fac, a fost sa citesc in functia de setup toate fisierele de pe cardul SD, sa verific daca sunt in format wav si sa le numar, asa ca am hardcodat numarul de melodii in cod. Acesta ar fi urmatorul update pe care l-as adauga proiectului.

Concluzii

Proiectul mi s-a parut foarte util pentru a intelege cum functioneaza intreruperile hardware, DAC-ul, PWM si I2C. A fost o experienta foarte placuta, ma bucur ca am ales un proiect pe care sa-l vad cum evolueaza in fiecare zi in care lucram la el.
Ce am mai invatat din acest proiect este sa verific cu MAI MARE atentie specificatiile componentelor si sa verific inainte de a da banii pe ele daca chiar o sa pot sa le conectez. Bibliotecile ascund multa informatie in spatele unui API dragut si cateodata e nevoie de ajustari pentru a le putea folosi impreuna.

Folositi modul de card MicroSD, nu de card SD mare, sau macar un modul cu level shifter integrat. Am invatat pe pielea mea.
Aveti grija ce biblioteci folositi, (probabil) ca o biblioteca pe care am incercat-o pentru prima oara mi-a facut placuta sa fumege.
Si componentele chinezesti nu sunt o investitie chiar buna

Download

Link GitHub

Jurnal

Bibliografie/Resurse

Software

documentatie biblioteca TMRpcm
biblioteca LiquidCrystal_I2C
despre modulul MicroSD
interfatare microfon cu Arduino
Pinout Arduino - SFANT CEA MAI UTILA CHESTIE

Export to PDF