Arduino iPod

Introducere

Ce este Arduino iPod?

Arduino iPod este un dispozitiv care reda fisiere audio in format wavelength. Fisierele audio sunt stocate pe un card SD, iar dispozitivul ofera functionalitati de playback similare cu acelea ale unui Music Player obisnuit:

- Display LCD pentru a oferi detalii despre melodia curenta
- Butoane pentru a trece la fisierul urmator/precedent si pentru a porni/opri playback-ul
- Butoane pentru a schimba modul de playback (repeat, shuffle)

Use Case

Dispozitivul ofera posibilitatea de a asculta muzica, podcast-uri si orice alt continut audio, oferind o experienta vintage (in opozitie cu playback de pe telefon/PC) fara sacrificiile aduse de dispozitive de playback mai clasice, cum ar fi spatiu mic de stocare pentru melodii.

Descriere generală

Schema Bloc

Descriere Module

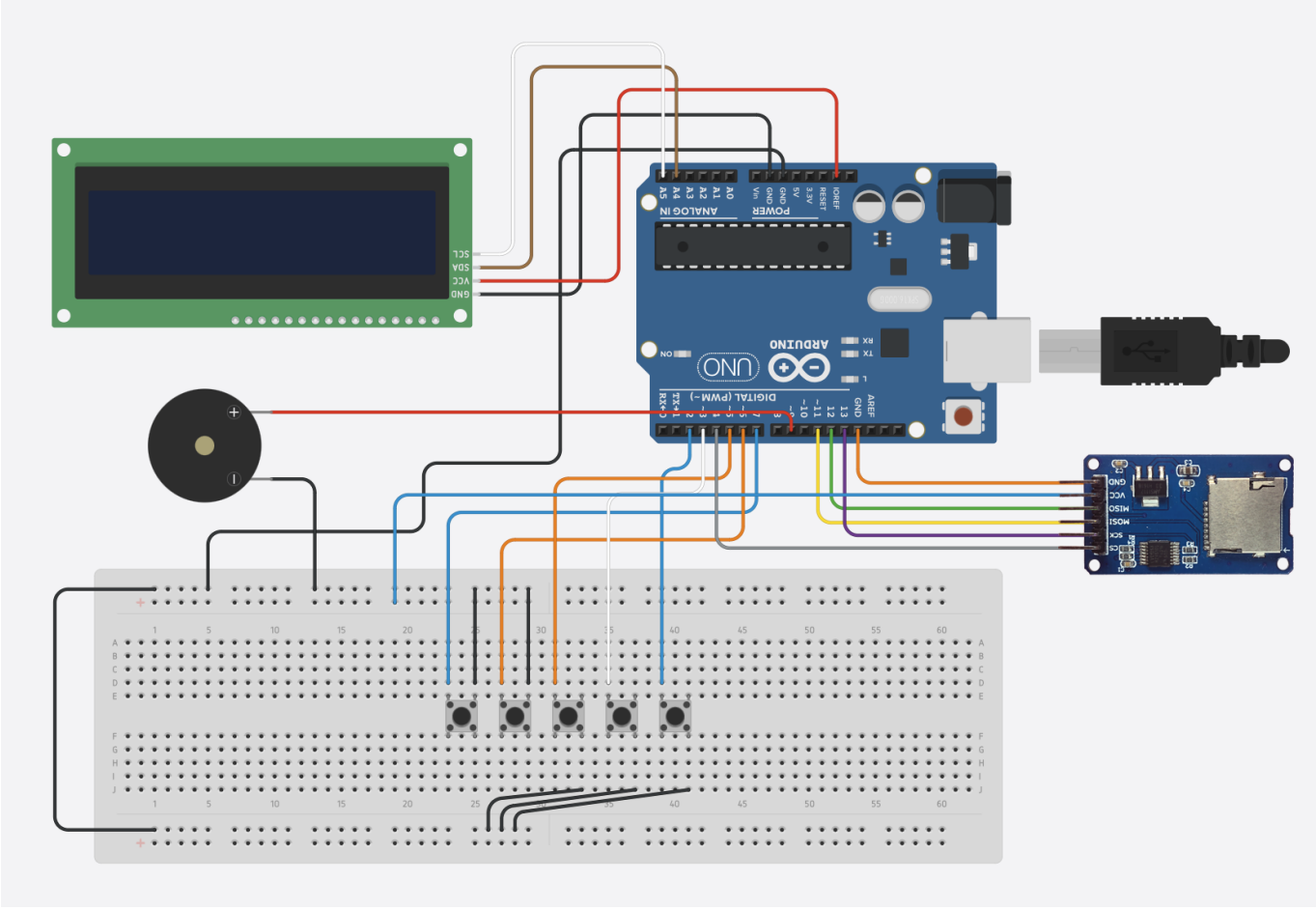
- Dispozitivul este implementat pe un Arduino UNO
- Fisierele audio se citesc folosing un modul de citit card-uri SD
- In absenta cardului SD se va afisa o eroare corespunzatoare
- Un ecran LCD I2C afiseaza informatii despre melodia curenta
- Redarea audio se face printr-un speaker
- Sistemul de playback poate fi controlat folosing 5 butoane (Prev, Next, Start/Stop, Repeat, Shuffle)
- Starea initiala a dispozitivului este Paused, asteptand apasarea butonului de Start inainte de redare

Hardware Design

Lista Piese

- Arduino
- LCD I2C
- Card SD
- Cititor Card SD
- Difuzor
- Butoane
- Breadboard
- Fire

Schema Hardware



Software Design

Componente de Interacțiune cu Sistemul

Funcționalitate Buton Play/Pause

Acest buton pornește/oprește playback-ul melodiei curente. De asemenea, el trebuie apăsat prima oară când pornește dispozitivul pentru a începe redarea primei melodii.

Funcționalitate Buton Next

Acest buton trece la următoarea melodie (determinată în funcție de modurile de Repeat și Shuffle) și pornește sau oprește redarea acesteia (în funcție de modul de Repeat).

Funcționalitate Buton Prev

Acest buton trece la melodia precedentă (determinată în funcție de modurile de Repeat și Shuffle) și pornește sau oprește redarea acesteia (în funcție de modul de Repeat).

Funcționalitate Buton Repeat

Apăsarea acestui buton duce la ciclarea circulară între 3 moduri posibile de Repeat, în ordinea următoare (și revenind la prima când se ajunge la ultima):

1. **No Repeat** (default) - În toate cazurile, trecerea la o alta melodie va avea loc, ducând însă la pauzarea playback-ului dacă:
   - Se apasă Prev, iar melodia curentă este prima melodie
   - Se apasă Next, iar melodia curentă este ultima melodie
   - Se termină ultima melodie
2. **Repeat Playlist** - În toate cazurile (Prev, Next, se termină melodia curentă) se va trece la o altă melodie, al cărei playback va incepe imediat.
3. **Repeat Song** - În toate cazurile (Prev, Next, se termină melodia curentă) va reîncepe melodia curentă.

Funcționalitate Buton Shuffle

Apăsarea acestui buton duce la ciclarea circulară între 2 moduri posibile de Shuffle, în ordinea următoare (și revenind la prima când se ajunge la ultima):

1. **Serial** (default) - Melodiile sunt redate în ordinea lexicografică.

2. **Shuffle** - Melodiile sunt redate într-o ordine aleatorie, fiecare fiind urmată de una aleasă aleator din lista disponibilă. Apăsarea butoanelor Prev și Next duc la același rezultat, la fel ca terminarea melodiei curente. Funcționalitatea acestui mod nu este afectata de modul de Repeat.

Algoritmul Next Song

```
// Computes the next song index, depending on Prev/Next, Repeat and Shuffle
// Returns whether or not the end has been reached (playback should stop if
the
// Repeat Mode is NO_REPEAT).
bool get_next_song_index(int &song_idx, bool increment = true) {
    bool end_reached = false;
    if (repeat_mode != REPEAT_SONG) {
        if (shuffle_mode == SERIAL_PLAYBACK) {
            if (increment) {
                if(song_idx < song_max_number) {
                    song_idx = song_idx + 1;
                } else {
                    song_idx = 1;
                    end_reached = true;
                }
            } else {
                if(song_idx > 1) {
                    song_idx = song_idx - 1;
                } else {
                    song_idx = song_max_number;
                    end_reached = true;
                }
            }
        } else {
            song_idx = random(1, song_max_number + 1);
        }
    }

    return end_reached;
}
```

Flow de Funcționare

1. **Start** - Dispozitivul pornește, se inițializează afișând un mesaj de welcome și asteaptă apăsarea butonului Play/Pause.
2. **Playing** - Melodiile sunt redate începând cu prima, iar interacțiunea cu dispozitivul are efectele descrise în secțiunea anterioară. LCD-ul afișează modurile de Repeat și Shuffle, urmat de melodia curent redată.
3. **Paused** - Similar cu starea de start, dizpozitivul nu are playback și asteaptă apăsarea butonului Play/Pause. LCD-ul afișează melodia care era în playback inainte de pauzare.
4. **Resumed** - Se continuă playback-ul de unde a rămas, în rest comportamentul este identic cu cel

de la **Playing**. LCD-ul afișează melodia curentă.

Format Fisiere

Arduino iPod acceptă fișiere care au:

- Formatul WAV
- Bit Resolution = 8 Bit
- Audio Frequency = 16000 Hz
- Audio Channels = Mono

Pentru a seta acest format audio în mod automat melodiilor, am creat un script de Python care primește folder-ul de melodii în orice format (mp3, flac etc.) și le setează ca mai sus.

Script Python WAV

```
from pydub import AudioSegment
import os

path = input("Path: ")
if not os.path.exists("wav_out"):
    os.mkdir("wav_out")

def convert_audio(file):
    audio = AudioSegment.from_file(os.path.join(path, file))

    # Set channels to mono
    audio = audio.set_channels(1)

    # Set frame_rate (frequency) to 16000
    audio = audio.set_frame_rate(16000)

    # Set sample_width (bit resolution) to 8
    audio = audio.set_sample_width(1)  # 1 byte = 8 bit

    # Save the result
    audio.export(os.path.join("wav_out", os.path.splitext(file)[0] + ".wav"),
format="wav")

if __name__ == '__main__':
    files = [filename for filename in os.listdir(path) if
                os.path.isfile(os.path.join(path, filename))]
    for file in files:
        convert_audio(file)
```

Pentru a pune fișierele pe SD, ele trebuiesc formatate astfel încât să poată fi interpretate de codul arduino. Pentru a facilita acest lucru, am creat un script de Python care primește path-ul către un

director, extrage toate fișierele WAV și creează un nou folder în acel director care conține versiunea formatată a fișierelor, pregătită pentru copiere pe SD.

Script Python Formatare

```
import re
import os
import shutil

if __name__ == '__main__':
    path = input("Song Dir Path: ")

    os.chdir(path)
    files = [filename for filename in os.listdir() if
             os.path.isfile(os.path.join(path, filename)) and filename[-4:]
== ".wav"]
    files_copy = files.copy()

    for i in range(len(files)):
        while not re.match(".* - .*", files[i]):
            print(f"The file \"{files[i]}\" does not respect the format
expected.")
            print("Please manually enter the song's details in the following
format:")
            print("artist_name - song_name")
            files[i] = input("Enter Here: ") + ".wav"

    files = [filename[:-4] for filename in files]
    files = [filename.split(" - ") for filename in files]
    files = [(file[0], file[1]) for file in files]
    files = [(artist[:16], song[:16]) for (artist, song) in files]
    files = [(artist + (16 - len(artist)) * " ", song + (16 - len(song)) * "
") for (artist, song)
             in files]

    if not os.path.exists('formatted'):
        os.mkdir("formatted")
    for i in range(len(files)):
        shutil.copyfile(files_copy[i], "formatted/song" + str(i + 1) + ".wav")
        file = open("formatted/song_info" + str(i + 1) + ".txt", "w")
        file.write(files[i][0] + "\n")
        file.write(files[i][1] + "\n")
        file.close()
```

Rezultate Obținute

Demo: https://drive.google.com/file/d/1WAYAqx93Ky4MkJhvtuAF9bdixta-ynb7/view?usp=sharing

Concluzii

Arduino iPod a fost un proiect foarte plăcut de implementat, oferind atât experiența relaxantă de a scrie cod voluminos și simplu, cât și cea engaging de a purta o luptă îndelungată pentru o funcție de câteva linii de care depinde tot restul programului. Partea de design a fost diferită de ce am lucrat până acum și, mai mult ca orice, resursele pe care le aveam puneau challenge-uri cu totul noi pe care niciodată nu le-am avut.

Dificultăți întâlnite:

- Cum era de așteptat pentru un breadboard, la un moment dat am fost surprins să văd că 3 butoane diferite nu fac absolut nimic. A fost o luptă îndelungată până când am acordat atenție faptului ca butoanele erau consecutiv așezate pe breadboard și am realizat că lamela de GND nu mai era funcțională de la un anumit punct până în capăt.
- Memoria unui Arduino este extrem de limitată față de un PC sau chiar și un Android. Am fost foarte suprins să văd un vector de string-uri ocupând întreaga memorie, și a fost o experiență nouă și educativă să vin cu un sistem care să nu aibă codul de Arduino ca bottleneck pentru numărul maxim de melodii acceptate.
- Logica Butoanelor a fost mai complicată decât mă așteptam. Eu le estimam dificultatea de implementare ca și componente separate însă nu mă gândisem la combinația lor, care s-a dovedit a fi un challenge mult mai mare.

Experiențele preferate:

- A fost ceva foarte util și nou să învăț să lucrez cu librării în afara celei standard de Arduino, în mod particular TMRpcm.
- Dezvoltarea sistemului de next_index și a scriptului de Python a fost un challenge foarte satisfăcător, atât ca design, cât și ca logică algoritmică.
- Testarea finală în care am văzut toata munca de circuit și coding funcționând exact așa cum ar trebui mi-a oferit foarte multă satisfacție. Încă folosesc proiectul pe post de music player în loc de Spotify doar pentru că știu că este music player-ul “meu”, făcut de mine.

Download

petolea_cosmin_335cb.zip

Jurnal

07.05.2023 - Adăugare documentație
19.05.2023 - Completat hardware
25.05.2023 - Completat Software

27.05.2023 - Finalizat Documentație

Bibliografie/Resurse

- https://www.instructables.com/Audio-Player-Using-Arduino-With-Micro-SD-Card/
- https://ocw.cs.pub.ro/courses/pm
- https://reference.arduino.cc/reference/en/libraries/sd/
- https://reference.arduino.cc/reference/en/libraries/tmrpcm/
- https://reference.arduino.cc/reference/en/libraries/liquidcrystal-i2c/

Export to PDF