Spotify Remote Controller

Introducere

- **Autor**: Cosmina Mihoreanu
- **Grupă**: 334CB

În proiectul de față îmi propun implementarea unui controller remote pentru player-ul de muzică de pe Spotify. Dispozitivul se va conecta la internet și va comunica cu Spotify pentru a controla diverse aspecte ale player-ului curent. Controlul se va face printr-o suită de butoane, din care se va putea pune pauză piesei curente, se va putea da play în continuare, skip forward sau backward în playlistul curent, seta starea de shuffle și controla volumul. Se va putea alege și un mod de funcționare în care volumul este ajustat automat în funcție de zgomotul din jur.

Scopul principal al proiectului este de a ușura interacțiunea cu Spotify, reducând-o la câteva butoane cu cele mai importante funcții. Astfel, nu mai este nevoie să accesezi dispozitivul pe care asculți muzică de fiecare dată când vrei să controlezi starea player-ului curent. Ideea provine tocmai din dorința de a avea o astfel de soluție de control centralizată pentru uzul personal, pentru a elimina pe cât posibil nevoia de a tot naviga continuu între ceea ce lucrez și player-ul de muzică. Din aceleași motive consider că proiectul ar fi util pentru persoanele care sunt obișnuite să asculte des muzică în timp ce se ocupă și de alte activități.

Descriere generală

Proiectul va cuprinde modulele reprezentate, la modul general, în schema bloc de mai jos.

Piesa centrală a sistemului este placa de dezvoltare ESP32. Prin pinii GPIO placa va primi input de la o suită de butoane care sunt acționate de utilizator pentru controlul dispozitivului. Acțiunile disponibile vor fi cele de bază pentru controlul unui player de muzică: on/off, pauză, play, skip înainte și înapoi, modificare status shuffle, ajustare volum și activare a modului de ajustare automată a volumului pe baza inputului analog primit de la un senzor de sunet cu microfon, conectat la ESP32.

Placa va comunica prin protocolul SPI cu un ecran LCD spre care va trimite date despre starea curentă a player-ului: vor fi afișate titlul și numele artistului, precum și timpul de redare la care s-a ajuns în orice moment.

Pentru toate informațiile și funcționalitățile implementate pe placă, aceasta se va conecta la o rețea Wi-Fi disponibilă și se va autentifica și comunica prin request-uri HTTP cu API-ul Spotify. Muzica disponibilă este, desigur, cea asociată user-ului pentru care se face autentificarea.

Hardware Design

Piese utilizate:

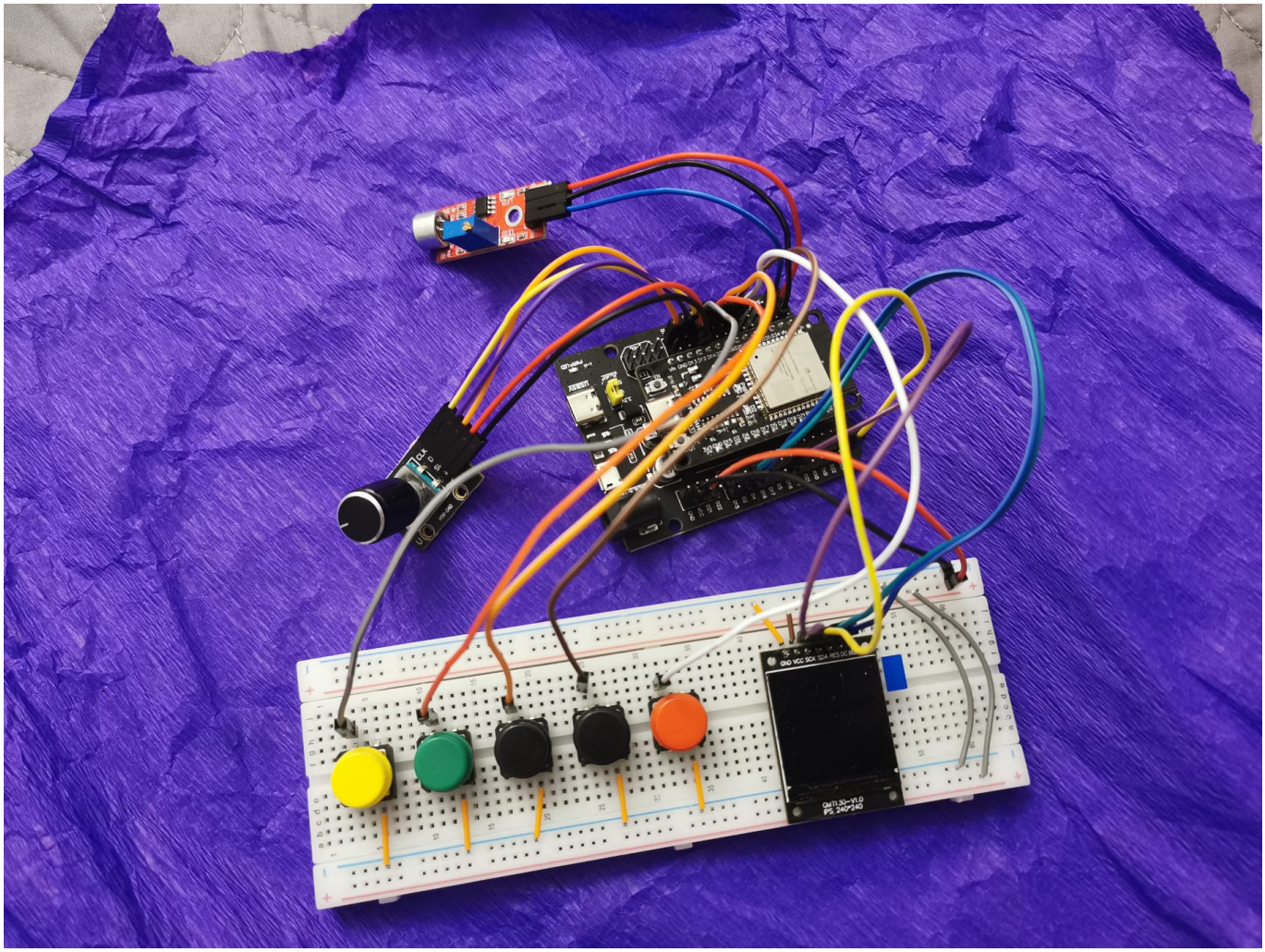
- placă ESP32-WROOM-32D
- display LCD TFT ST7789
- senzor sunet KY-037
- rotary encoder KY-040
- 5 push buttons
- 1 buton rotary encoder

Schemă electrică:

Constrângeri cablaj

- Pinul de input de la senzorul de sunet este tras la pinul GPIO36 al plăcii, deoarece este nevoie de un pin de tip ADC_1. Pinii de ADC_2 pun probleme de funcționare în cazul activării funcției de WiFI pe ESP32, esențială pentru acest proiect.
- Butoanele sunt conectate la placă la pinii GPIO33, 32, 25, 26, 27, aceștia fiind pini pe care există rezistențe interne de pull-up. Pinii 36, 39, 34, 35 nu au legate astfel de rezistențe.
- Display-ul este conectat la pinii corespunzători protocolului SPI, conform datasheet-ului plăcii (VSPI_MOSI și VSPI_CLK).

Cablaj final



Software Design

- **Mediu de dezvoltare:** Visual Studio Code + PlatformIO
- **Biblioteci externe:**
  - Adafruit ST7735 and ST7789 Library
  - Adafruit GFX Library
  - Adafruit BusIO
  - Adafruit Unified Sensor
  - ArduinoJson
  - SPI
  - WiFi
  - HTTPClient
- **Logică generală:** la inițializare ESP32-ul se va conecta la internet folosind biblioteca WiFi și apoi va comunica prin cereri HTTP cu API-ul Spotify, prin funcțiile expuse de bibliotecii HTTPClient; apăsarea butoanelor și acționarea rotary encoder-ului vor genera întreruperi care vor seta și modifica variabile și flag-uri globale, ale căror valori vor fi verificate continuu în bucla „loop” din main.cpp; aceste modificări vor declanșa generarea de noi cereri HTTP corespunzătoare, iar răspunsurile la acestea vor updata informația afișată pe ecran.
- **Flow și componente principale:**

WiFI

- configurarea WiFi-ului este una manuală: este pornit dispozitivul in mod de Access Point, și se pornește pe el un server cu o rută care întoarce o pagină simplă cu un form de login, unde pot fi introduse datele rețelei dorite. Datele introduse sunt extrase de server și salvate în variabilele globale **ssid** și **pasword**;
- se trece dispozitivul în modul Station și se conectează la rețeaua dată, după care se oprește serverul; toate funcțiile de inițializare și configurare se găsesc in WiFi.h;
- adresele IP și instrucțiunile de accesare sunt afișate pe ecran.

Spotify API

- după conectarea la WiFi se face automat o cerere de acces către Spotify API (**accounts.spotify.com/authorize**), de la care se primește ca răspuns un link de autentificare;
- se deschide un nou server pe un port diferit (același port poate provoca bind errors în unele cazuri), cu o rută care redirecționează către link-ul de autentificare de la Spotify;
- după verificarea identității, Spotify Web API răspunde prin redirecționarea către un URL dat de program prin **redirectURI**, și care trebuie să fie înregistrat în Dashboard-ul personal de pe Spotify for Developers; redirecționarea include un fragment denotat de **#** care conține access token-ul, care va fi extras de un script din pagina întoarsă de ruta de pe server setată pentru această redirectare;
- adresele IP și instrucțiunile de accesare sunt afișate pe ecran;
- următoarele interacțiuni cu API-ul se fac prin cereri la **/me/player** pentru informații legate de piesa curentă, **/me/player/previous** și **/me/player/next** pentru schimbarea piesei, **/me/player/shuffle** pentru toggle-ul stării de shuffle, **/me/player/pause** și **/me/player/play** pentru start-stop, **/me/player/volume** to set the volume.

Periferice

- **display:** conectat prin SPI și acționat prin bibliotecile Adafruit pentru grafică; se configurează inițial culorile și orientarea, după care la fiecare piesa este re-randat întreg ecranul;
- **butoane:** butoanele sunt setate pe modul INPUT_PULLUP, și li se atașează întreruperi care modifică în true flag-urile corespunzătoare, care vor fi verificate în main loop; este implementat și un debounce software, prin setarea unui timp minim de apăsare între apăsări succesive valide;
- **rotary encoder:** pinii sunt configurați pentru INPUT, respectiv INPUT_PULLUP în cazul pinului de switch care expune funcționalitatea de push-button; se atașează o întrerupere pe pinul CLK, care se declanșează atunci cand encoderul este rotit, iar citirea digitală a pinului DT indică direcția de rotire, după care se modifică variabila globală **volume**;
- **senzor sunet:** atunci câand este activat modul de auto-volume (prin buton), este citită cu o rezoluție de 12 biți valoarea pinul Analog Output a senzorului, si transformată într-o valoare a volumului; astfel, plaja de valori 960 - 1045 este mapată la intervalul 100 - 0 pentru volum; doar atunci cand se detectează variații de volum de minim 10 unități se trasnmite schimbarea către Spotify.

Timer

- este folosit timer-ul 0 al plăcii ESP32, cu un interval de declanșare de 1 secundă, și o întrerupere care setează un flag verificar in main loop; astfel, la fiecare secundă este re-randat timestamp-ul în melodie și bara de progres, iar la fiecare 4 secunde se trimite o cerere la Spotify API la **/me/player** pentru reactualizarea datelor legate de player-ul curent.

Implementare completă disponibilă la:

GitHub Repo

Rezultate Obţinute

Rezultatul proiectului este un mini dispozitiv cât de cât util care poate comunica cu Spotify API și poate controla cu succes starea player-ului curent. Funcționalitatea este cea dorită, putând fi aduse, desigur, și alte îmbunătățiri ulterioare.

Există unele limitări ale performanței și responsivității din cauza request-urilor către Spotify care pot lua destul de mult timp, comparativ cu interacțiunile directe cu dispozitivul.

Un demo video al flow-ului de funcționare se găsește la adresa:

Spotify Remote Controller Demo

Concluzii

A fost un proiect foarte instructiv, a fost nevoie să mă familiarizez cu multe noțiuni și tehnici practice noi, dar a fost distractiv all in all. Debugging-ul hardware destul de complicat, și autentificarea în mai mulți pași pe API-ul de la Spotify a pus destul de multe probleme care nu au putut fi rezolvate decât prin rute de redirectare și scripturi de JS injectate în pagini HTML.

Jurnal

- 29 aprilie - creat pagină proiect
- 1 mai - primit piese (ecran, senzor, placă, expansion board)
- 3 mai - completat documentație milestone 1 + comandat piese noi (butoane noi și rotary encoder)
- 11 mai - completat documentație milestone 2 (hardware) + cablaj inițial pe breadboard si expansion board
- 24 mai - completat documentație milestone 3 (software) + funcționare parțială
- 26 mai - upload demo + rezultate + github
- 27 mai - GitHub public după git rebase + push –force pentru a scoate client_secret (pentru Spotify API) din commit-uri vechi

Bibliografie/Resurse

Pinout ESP32

Rotary Encoder Datasheet

ST7789 Display Pinout and Tutorial

ESP32 Button Intrerrupts Tutorial

WiFi Connection Tutorial

Spotify Web API Documentation

Export to PDF