Sistem Radio

Toacă Alexandra Simona, 332CA

Introducere

Proiectul își propune realizarea unui sistem radio, ce permite selectarea unui post, afișarea detaliilor (precum numele postului și frecvența de emisie) pe un ecran și, evident, ascultarea muzicii difuzate de către acesta.

De ce un radio? Mi s-a părut interesant de interfațat un modul radio cu un microcontroler, în special pe partea de date de tip RDS pe care le poate recepționa acest modul.

Scopul și utilitatea proiectului: Când ai nevoie să te concentrezi și o faci mai bine cu muzică pe fundal, un radio este un mod foarte bun de a asculta muzică fără distrageri.

Descriere generală

Piesa de bază a acestui proiect este modulul radio, ce interacționează atât cu microcontrolerul, un Arduino Uno, cât și cu mediul extern, prin transmiterea de semnal audio către un amplificator, semnal ce este apoi redat utilizatorului prin difuzor.

Microcontrolerul face intermedierea între utilizator și modulul radio. Acesta primește comenzi de la utilizator, cum ar fi schimbarea frecvenței, și le semnalează modulului radio folosind protocolul I2C. Totodată, datele primite de către microcontroler din partea modulului radio (precum numele postului sau melodia redată) sunt afișate pe un ecran LCD ce utilizează SPI drept protocol de comunicație.

De asemenea, utilizatorul poate controla volumul muzicii fără a implica microcontrolerul, printr-un potențiometru amplasat la ieșirea din amplificator.

Hardware Design

Lista de componente:

- Arduino UNO R3
- Ecran LCD pe SPI ST7735
- Modul Encoder Rotativ

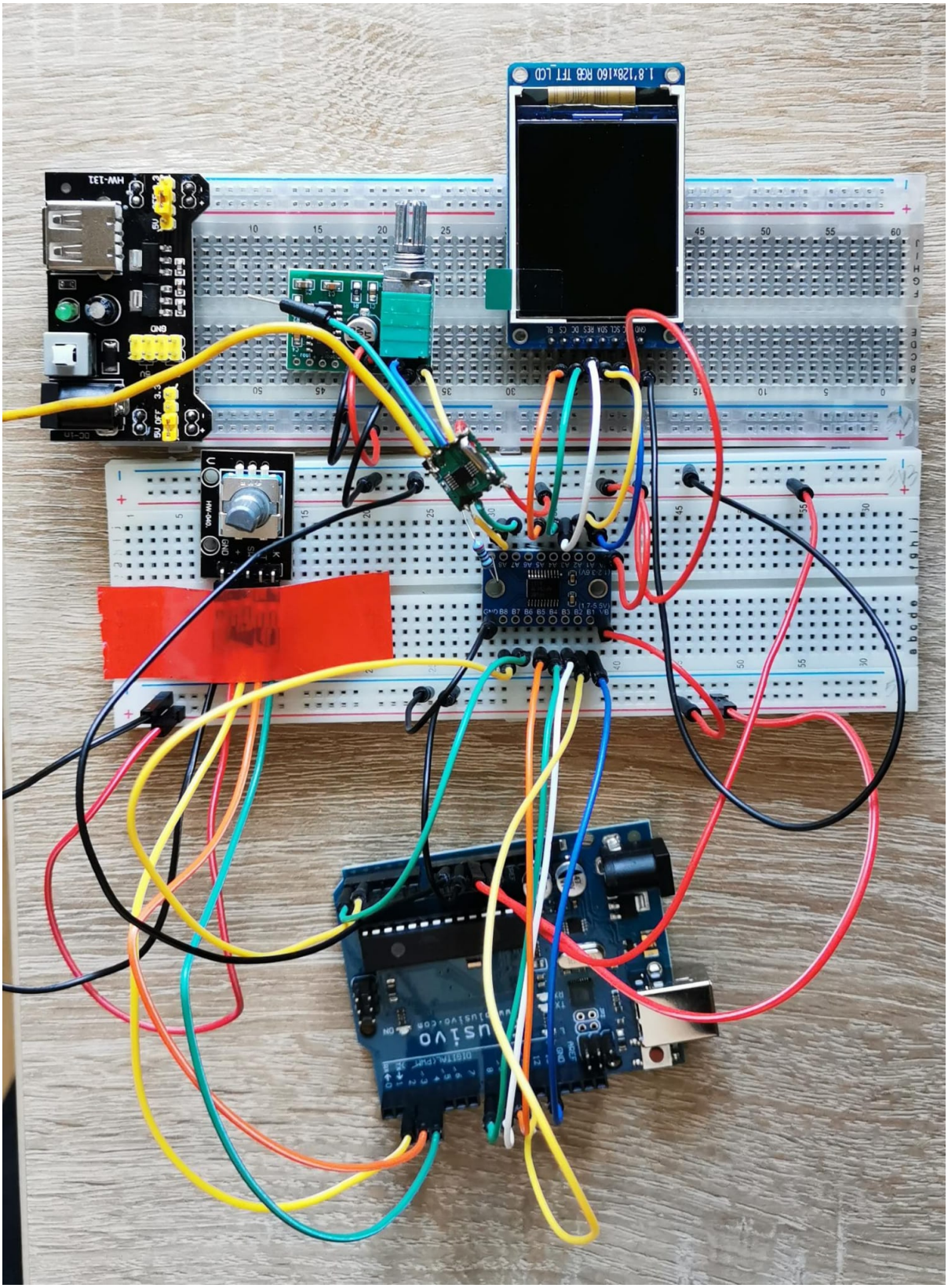
- Modul Radio RDA5807m
- Amplificator PAM7803 cu potentiometru
- Sursa de alimentare de 3v3/5v
- Convertor de nivele logice
- Difuzoare

Schema electrică:

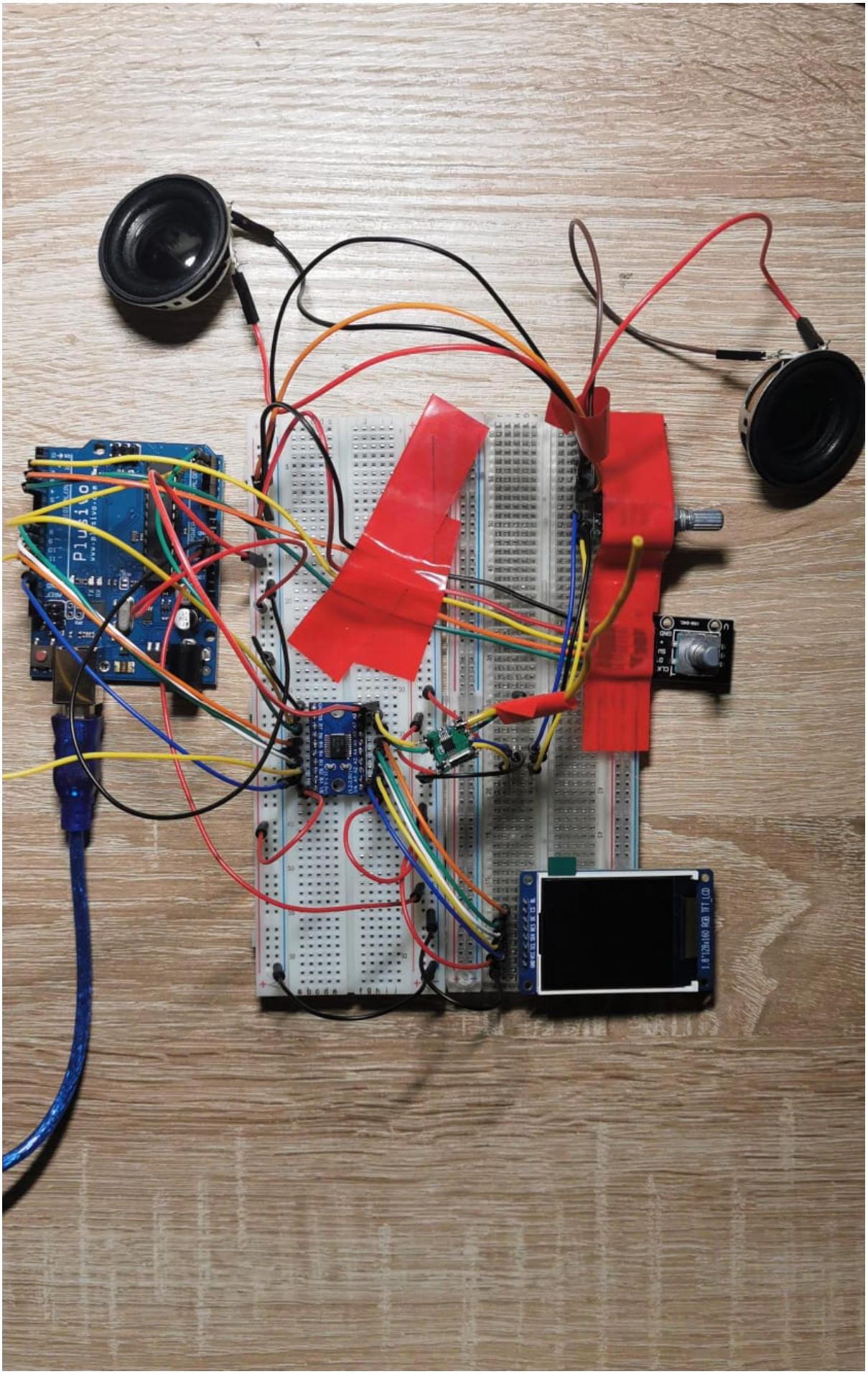
Conectarea componentelor și rolul fiecăreia:

- Ecranul LCD este conectat la pinii de SPI ai Arduino. Controllerul ST7735 foloseste interfata 4-wire, adică: SDA (de fapt linia de date MOSI pe Arduino, MISO nu este folosită), SCL (SCK), CS (Chip Select), DC (semnal digital ce selectează dacă ceea ce se trimite este date/comandă).
- Modulul Encoder Rotativ este conectat direct la Arduino, la pinii digitali 2, 3, 4. Acesta este folosit pentru a selecta frecvența.
- Modulul Radio comunică prin I2C și foloseste pinii SDA, SCL ai Arduino
- Convertorul de nivele logice preia semnalele de date de la microcontroler (SDA, SCL, MOSI, SCK, CS, DC), care au 1 logic echivalent cu 5V, și le transforma in semnale logice la 3v3, pentru că atât LCD-ul, cât și modulul radio funcționează la 3v3. Pinul OE este conectat la 5V.
- Pinii LOUT si ROUT ai modulului radio sunt conectați la amplificator, iar amplificatorul este conectat la difuzoare. Amplificatorul are deja conectat un potențiometru pentru controlul volumului.

Un prim montaj al componentelor pe breadboard (cu exceptia difuzoarelor, ce trebuie lipite):



Montaj final



In montajul final am spațiat mai mult componentele și am îndepărtat LCD-ul și amplificatorul de

modulul radio pentru a evita interferențele.

Software Design

Link cu repo-ul de GitHub: SistemRadio

Mediu de dezvoltare: PlatformIO

Cunoștințe din laboratoare: Întreruperi, I2C, SPI.

Biblioteci folosite

În dezvoltarea proiectului am folosit biblioteca externă RDA5807 pentru modulul radio. În rest, tot codul este scris de mine. Astfel, am scris driver-ul pentru comunicarea pe **SPI** și pentru comunicarea cu controller-ul displayului, **ST7735**. Am ales să fac acest lucru pentru că am vrut un codebase mai lightweight decât ce mi se oferea prin intermediul bibliotecii de la AdaFruit. Dacă mai aveam timp la dispoziție, aș fi scris codul pentru I2C și modulul radio. Codul pentru display permite scrierea a 2 tipuri de font-uri: font8x8.h și font16x16.h.

Structura codului

- Rotary Encoder-ul comunică cu Arduino folosind **întreruperi**. Astfel, am legat INT0 la pin-ul SW (apăsarea rotary encoder-ului schimbă modul de lucru de la selectarea frecvenței la selectarea unui post radio dintr-o listă) și INT1 la CLK pentru a detecta când este rotit encoder-ul. Din verificarea stării celuilalt pin (DT) se deduce direcția de rotație a encoder-ului. Dacă CLK declanșează întreruperea și pin-ul DT e 1, atunci înseamnă ca DT a fost primul care și-a schimbat starea.
- Radio-ul pornește cu o frecvență default și codul din **loop()** execută o funcție ce verifică dacă există date de tip RDS și le afișează pe LCD. Schimbarea funcției ce rulează în loop() se face prin intermediul unui pointer la funcție, **current_action**. Acesta ține mereu adresa funcției ce trebuie executată.
- În modul de selectare a frecvenței, rotația encoder-ului determină o întrerupere, ce mai departe este interpretată ca freq++ sau freq–, iar funcția din loop se schimbă în **update_freq()**, funcție ce comunică cu modulul radio pentru schimbarea frecvenței și cu LCD-ul pentru afișarea acesteia.
- La apăsarea encoder-ului se intră/iese din meniul de posturi radio și current_action devine **select_chan**. **select_chan** decide în funcție de tipul acțiunii (CHAN_SELECT_ENTER sau FREQ_CHANGE) dacă afișează meniul sau face update la frecvența radio.
- Tipul acțiunii determină și ce se întâmplă în întreruperea pentru rotația encoder-ului. În modul CHAN_SELECT_ENTER, rotația înseamnă scroll prin meniu.
- Pentru a salva posturile din meniu țin un array de structuri **chan_t**, care stochează frecvența și numele postului radio. Deocamdată acest array este static, dar ideea poate fi extinsă la ceva dinamic.
- Array-urile ce țin font-urile sunt salvate în memoria de program pentru a nu ocupa RAM-ul. Despre PROGMEM.
- Funcțiile ce fac afișări pe display fac mereu un check pentru a nu trece mai puțin de 200ms între afișări. Motivul? Proiectul este deja destul de demanding din punct de vedere al curentului consumat și afișarea continuă ar provoca glitch-uri.

Optimizări făcute

- Am folosit întreruperi pentru rotary encoder → am evitat polling = busy waiting
- Afișările pe display se fac doar când e nevoie, nu constant. De exemplu, cât timp utilizatorul nu interacționează cu radio-ul, frecvența de pe display nu este rescrisă.
- Când se face o afișare pe display, cursorul intern al acestuia este setat pentru a scrie fix pe porțiunea necesară, nu pe tot display-ul.
- Codul de display printează un string pe lcd de ”sus în jos” (primul rand de biti pt char1, primul rand pt char2...al doilea rand pt char1...), nu caracter cu caracter, pentru a putea face o singura scriere în registrele ce setează cursorul.
- O modificare mică: led-ul builtin este mereu stins pentru a nu consuma curent.

Note: Am vrut să folosesc și modurile de sleep ale avr-ului pentru a salva baterie, însă din ceva motiv asta facea ca proiectul să nu mai funcționeze cum ar fi trebuit. Sigur există o explicație logică pentru asta, dar mi-a lipsit timpul necesar pentru a face debug.

Rezultate Obţinute

Demo

- Radio-ul funcționează bine și fără delay-uri, însă nu dispune de foarte multe funcționalități. Recepția radio este bună și sunetul se aude clar.
- Datele RDS ajung corupte câteodată și nu este afișat corect postul, dar cred că asta depinde de interferențe (ar fi mers poate o antenă mai lungă).
- Am reușit să alimentez totul din Arduino, deși am multe module și amplificatorul consumă destul de mult.
- Codul, cu tot cu font-urile, folosește 40% din memoria Flash, iar din memoria RAM este folosită doar 29%. (ce mi-a zis PlatformIO).

Concluzii

Acest proiect, deși simplu la prima vedere, m-a învățat multe, inclusiv cum să lipesc componente (pini, difuzoare și fire). Până am scris codul mă rugam ca lipiturile mele să fie bune :)))

A fost o provocare să alimentez totul din Arduino și am dat de niște cazuri când Brown-Out-Detector-ul declanșa reset-ul.

Mi-ar fi plăcut să lipesc tot proiectul pe un perfboard, dar nu am găsit din timp unul destul de mare pentru a ține componentele cât de cât aerisite (modulul radio este afectat și de SPI și de amplificator).

De asemenea, a fost interesant să ascult muzică la proiect în timp ce scriam codul pentru el :)))

Download

Link cu repo-ul de GitHub: SistemRadio unde se găsește și schematic-ul proiectului.

Jurnal

- **5 mai** - adăugare documentație inițială
- **16 mai** - adăugare documentație hardware + prim montaj
- **24 mai** - adăugare documentație software + video + extra explicații hardware

Bibliografie/Resurse

- Datasheet Controller LCD ST7735
- Datasheet Modul Radio
- Datasheet convertor de nivele logice

Export to PDF