Music Audio Spectrum - VU Meter

- Nume și prenume: Crețu Mihnea Tudor
- Grupa: 335CA

Introducere

Proiectul **Music Audio Spectrum - VU Meter** este un vizualizator audio care afișează spectrul sonor în timp real pe un display LED de tip matrice 8×32, iar în modul IDLE se comportă ca un ceas LED cu termometru (reprezentând ora și temperatura din cameră). Scopul său este de a crea o reprezentare vizuală atractivă a intensității și frecvenței semnalului audio captat din mediu. Ideea proiectului a pornit de la dorința de a combina electronica cu muzica și elementele vizuale, pentru a obține un efect interactiv și estetic plăcut. Este util atât pentru uz personal (ambianță luminoasă în funcție de muzică), cât și ca instrument educativ în învățarea componentelor hardware și programarea microcontrolerelor.

Descriere generală

Dispozitivul are două moduri principale de funcționare:

1. Idle – Afișează constant ora curentă și temperatura ambientală pe matricea LED.
2. Spectru Audio – Vizualizează frecvențele detectate de microfon sub formă de bare armonice, reprezentând spectrul audio în timp real.

Trecerea între cele două moduri se face printr-un singur buton de control, care comută secvențial între ele la fiecare apăsare.

Hardware Design

Listă de piese:

Componentă	Descriere

Arduino UNO R3	Microcontroller
LM386	Microfon cu amplificator
MAX7219 (8×32)	Matrice LED
DHT11	Senzor temperatură
DS1302	RTC
Button	Button pentru schimbare mod

Schemă electrică:

Pini folosiți:

Matrice LED MAX7219 (4x8x8):

- DIN (data in) → D11 (MOSI)
- CLK (clock) → D13 (SCK)
- CS (chip select) → D10

Modul microfon LM386:

- AOUT (semnal audio) → A0

RTC DS1302:

- CLK → D6
- DAT (I/O) → D7
- RST (CE) → D8

Senzor DHT11:

- DATA → D2

Buton mod:

- D5

Software Design

Librării folosite:

- arduinoFFT.h – pentru analiza semnalului audio prin transformata Fourier rapidă (FFT)
- DHT.h – pentru citirea temperaturii de la senzorul DHT11
- RtcDS1302.h + ThreeWire.h – pentru comunicarea cu modulul de ceas în timp real (RTC) DS1302
- MD_MAX72xx.h – pentru controlul matricei LED cu driverul MAX7219
- MD_Parola.h – pentru animații și afișaj text pe matricea LED
- SPI.h – pentru comunicația SPI cu MAX7219

Cum am folosit aceste librării:

Pentru afișajul principal, modul Spectru Audio, am utilizat biblioteca MD_MAX72xx, care permite controlul individual al fiecărei coloane din matricea LED (MAX7219). Această funcționalitate este esențială pentru redarea grafică în timp real a nivelului diferitelor frecvențe detectate din semnalul audio. Fiecare coloană a matricei corespunde unei benzi de frecvență, iar înălțimea coloanei este determinată în urma analizei FFT.

Pentru analiza semnalului audio am integrat biblioteca arduinoFFT, care transformă semnalul captat de microfon (pe pinul analog A0) din domeniul timp în domeniul frecvență. Rezultatul este procesat pentru a determina amplitudinile armonicilor principale, care sunt apoi mapate vizual în matricea LED. Am aplicat și o funcție de fereastră Hamming pentru a reduce erorile de margine și a obține un rezultat vizual mai stabil.

În celelalte moduri (temperatură și ceas), afișarea valorilor se face textual. Pentru acestea am utilizat biblioteca MD_Parola, care este optimizată pentru afișarea de texte animate și mesaje derulante pe afișaje cu MAX7219. Această bibliotecă este mult mai prietenoasă pentru text, dar nu permite un control detaliat la nivel de coloană LED, motiv pentru care nu a fost folosită și pentru spectrul audio.

Pentru a simplifica procesul de afișare a înălțimii coloanelor LED în spectru, am definit un vector numit spectralHeight[], care conține valori predefinite binar pentru fiecare nivel de amplitudine (0–8). Astfel, nu este nevoie să construim manual fiecare coloană, ci selectăm rapid modelul corespunzător.

Algoritmi și logică de bază:

- Citirea semnalului audio de la un microfon analogic prin pinul A0
- Aplicarea funcției de fereastră (Hamming) pentru reducerea distorsiunilor de margine în FFT
- Transformata Fourier rapidă (FFT) pentru a extrage componentele de frecvență din semnal
- Vizualizare spectru audio: valorile FFT sunt mapate pe 32 de coloane verticale ale matricei LED
- Schimbare moduri cu buton – dispozitivul comută între 3 moduri la fiecare apăsare:
  - Mod 0 – Spectru audio (FFT)
  - Mod 1 – Temperatură ambientală
  - Mod 2 – Ceas digital (ora:minute)
- Afișaj pe 4 module LED MAX7219 (32×8) controlate prin SPI

Explicare Spectru Audio (FFT)

ETAPA 1: Citire semnal audio brut și pregătire FFT

```
for (int i = 0; i < 64; i++) {
  realComponent[i] = analogRead(A0) / sensitivity;
  imagComponent[i] = 0;
}
```

Ce face:

- Citește de 64 de ori valoarea de tensiune analogică de la microfon (pinul A0).
- Salvează acele valori în vectorul realComponent[] (partea reală).
- Inițializează partea imaginară imagComponent[] cu 0 (necesar pentru FFT).
- Sensitivity este un divizor care reglează amplitudinea semnalului (valori mai mici = mai sensibil la sunete slabe).

ETAPA 2: Preprocesare și transformare FFT

```
FFT.windowing(realComponent, 64, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
FFT.compute(realComponent, imagComponent, 64, FFT_FORWARD);
FFT.complexToMagnitude(realComponent, imagComponent, 64);
```

Ce face:

- windowing(...) – aplică o funcție de fereastră Hamming:
  - Corectează distorsiunile care apar la margini (fenomenul de “leakage”).
- compute(...) – aplică FFT (Fast Fourier Transform):
  - Transformă semnalul din domeniul timp în domeniul frecvență.
  - Rezultatul este în realComponent[] și imagComponent[].
- complexToMagnitude(...) – calculează modulul numerelor complexe:
  - √(Re² + Im²) → adică obține amplitudinea fiecărei frecvențe.

ETAPA 3: Afișare pe matricea LED

```
for (int i = 0; i < 32; i++) {
  realComponent[i] = constrain(realComponent[i], 0, 80);
  realComponent[i] = map(realComponent[i], 0, 80, 0, 8);
  disp.setColumn(31 - i, spectralHeight[(int)realComponent[i]]);
}
```

Ce face:

- for (int i = 0; i < 32; i++)
  - FFT generează 64 de rezultate (pentru 64 de eșantioane), dar doar primele 32 sunt utile (frecvențe reale, partea “pozitivă” a spectrului).
- constrain(...):
  - Limitează valoarea FFT între 0 și 80 (nivel rezonabil de amplitudine).
  - Evită valori prea mari care pot ieși din gamă.
- map(...):
  - Convertește amplitudinea într-o valoare între 0 și 8.
  - Deoarece matricea are doar 8 rânduri pe coloană, fiecare bară poate avea max 8 “leduri aprinse”.
- disp.setColumn(31 - i, spectralHeight[(int)realComponent[i]])
  - Desenează bare verticale pe coloanele de la 31 în jos până la 0 (e invers pentru că afisajul merge de la dreapta la stânga).
  - Fiecare bară este un cod binar din spectralHeight[], de exemplu:
    - 0b11111100 aprinde 6 LED-uri din 8, de jos în sus.

Precizări:

- Memoria utilizată este de: ~89%.

Rezultate Obţinute

Deși am reușit să implementez ce mi-am propus, primele 2 coloane din matrice sunt mereu aprinse pe modul FFT, chiar și atunci când nu există zgomot ambiental (precum în imaginea atașată mai jos). Din păcate, nu am reușit să rezolv acest incovenient.

Urmează să incorporez și să lipesc circuitul într-o cutie pentru prezentarea finala, pentru a ascunde din fire și a face proiectul să arate mai plăcut din punct de vedere estetic.

Demo video:

https://youtu.be/GsM8L1uTOQw

Concluzii

Realizarea acestui proiect a fost, în primul rând, plină de trial & error, dar plăcută. . Sunt, desigur, numeroase direcții de perfecționare: un RTC mai precis (de ex. DS3231), o carcasă printată 3D care să mascheze cablurile și să îmbunătățească răcirea, un microfon mai sensibil ori animații suplimentare pe matrice.

Din punct de vedere al timpului, partea cea mai mare a mers pe documentare – am vrut să evit greșelile costisitoare și să nu „prăjesc” componente. Montajul hardware propriu-zis și design-ul circuitului s-au dovedit, în final, a fi mai ușor decât îmi închipuiam la început.

Surpriza plăcută a fost că, deși am petrecut multe ore în șir lucrând, am descoperit că îmi place procesul în sine – de la debugging la aranjat fire și conectat pini. Cu toate acestea, am aflat că îmi place și partea hardware a acestei facultăți, nu doar cea software, așa cum credeam înainte.

Proiectul rămâne un punct de plecare solid, iar îmbunătățirile viitoare vor fi cu siguranță mai rapide, având acum o bază hardware și software stabilă.

Download

vu_meter.zip

Bibliografie/Resurse

Resurse Hardware:

- [MAX7219 Datasheet](https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf)
- [DS1302 RTC Module Datasheet](https://www.analog.com/media/en/technical-documentation/data-sheets/ds1302.pdf)
- [Canal YouTube cu tutoriale despre conectat diferite module pe placă](https://www.youtube.com/@Core-Electronics)
- [Tutorial Audio Spectrum](

https://projecthub.arduino.cc/shajeeb/32-band-audio-spectrum-visualizer-analyzer-924af5)

Resurse Software:

- [Arduino FFT Library](https://github.com/kosme/arduinoFFT)
- [DHT Sensor Library](https://github.com/adafruit/DHT-sensor-library)
- [RTC Library](https://github.com/adafruit/RTClib)
- [Explicare FFT](
https://projecthub.arduino.cc/abhilashpatel121/approxfft-fastest-fft-function-for-arduino-f1b6ba)

Export to PDF