**Audio Frequency Visualizer**

**Documentatie**

**Oros Timotei**

**Grupa 30238**

Cuprins:

1. Prezentare generala

2. Montaj

3. Implementare

4. Dificultati

5. Dezvoltari ulterioare

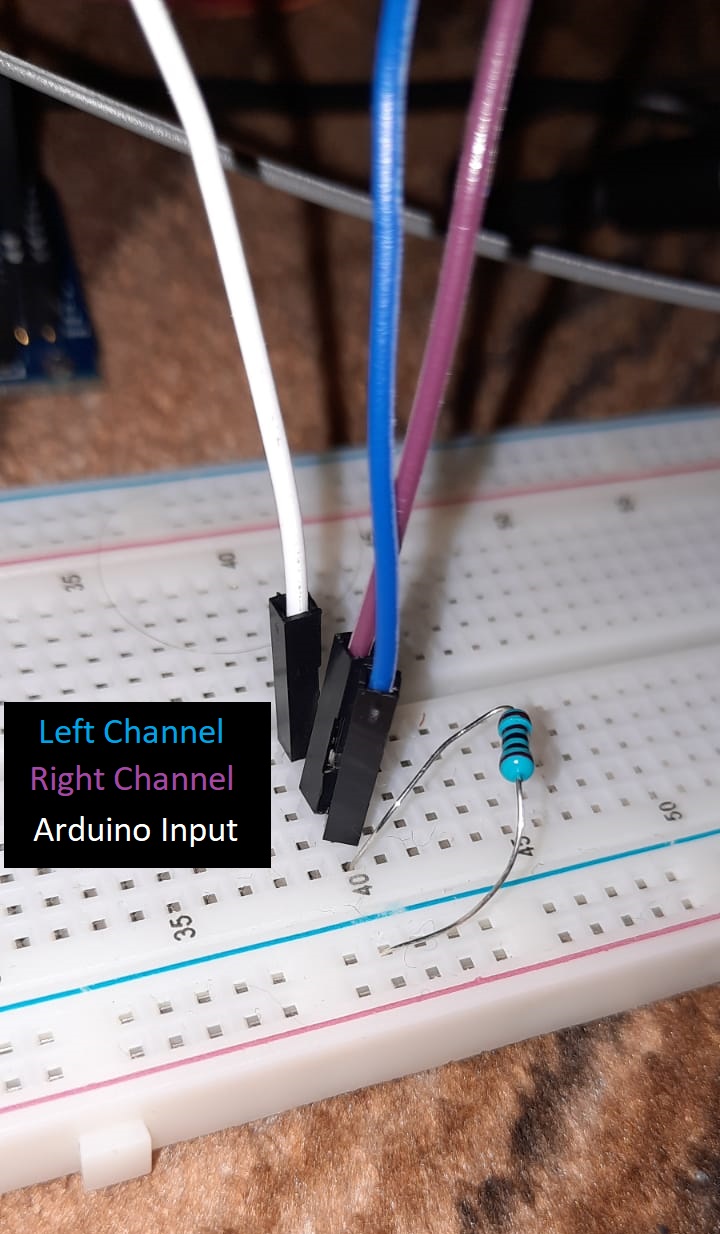
1. **Prezentare generala:**

Acest proiect are ca idee principala punerea unei melodii/sunet de pe un sistem desktop, si afisarea frecventelor pe o matrice de led-uri 32x8. Coloanele matreicei din dreapta vor reprezenta frecventele joase, coloanele din mijloc vor reprezenta frecventele medii, iar coloanele din stanga sunt alocate pentru frecventele inalte.

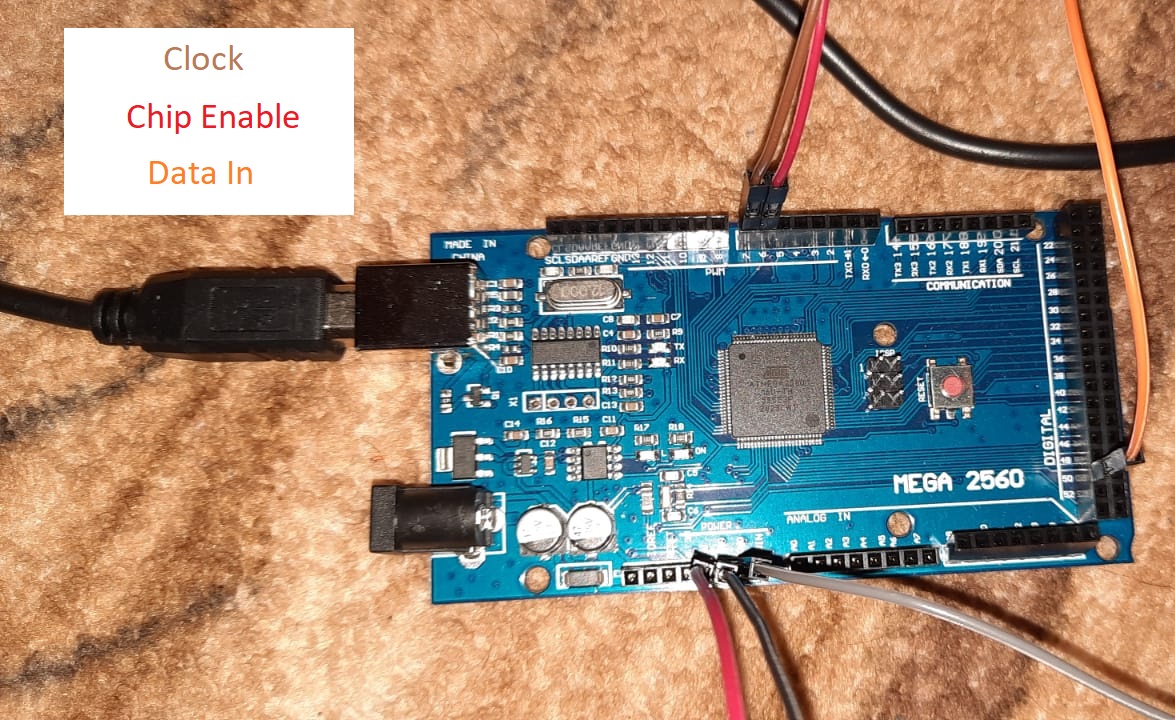
1. **Montaj:**

Pentru a putea avea real-time input in arduino, am ales sa folosesc un cablu jack 3.5mm. Acest cablu este unul cu 2 iesiri, pentru a putea da input la arduino, si pentru a putea asculta muzica/sunetul dat de pe desktop la casti/boxe. Pentru firele care merg spre arduino, am taiat cablul, si am infasurat cele 3 fire (ground, left channel, right channel) de fire tata-tata, le-am cositorit, si le-am izolat. In acest mod, am 3 fire pe care imi vine input in timp real, si pe care le pot pune in arduino/breadboard.

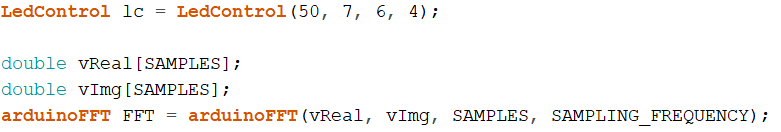


Ground-ul cablului jack merge la ground-ul de pe arduino, iar firele channel-ului stang si drept merg puse in breadboard pe aceeasi linie, pentru a fi combinate, si dupa trimise catre arduino. Folosesc o rezistenta de 1k ohmi pentru a filtra fluctuatiile fantoma nedorite are voltajului. Acest rezistor este foarte folositor, fara el input-ul este cateodata imprevizibil.

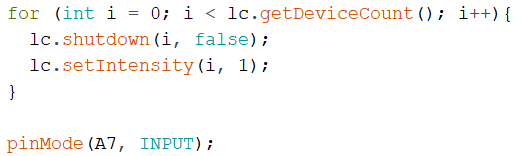
Mai departe, prezint firele care vin de la matricea de led-uri, acestea find vcc si ground-ul, care merg pune pe breadboard, iar in rest in arduino vin firele de clock, chip enable si data in.



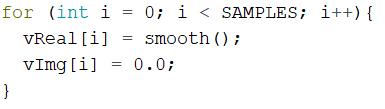
1. **Implementare:**

Pentru inceput, vom initializa doua variabile. Una care va controla comunicarea cu matricea de led-uri, iar cealalta care ne va ajuta sa calculam transformata fourier. 

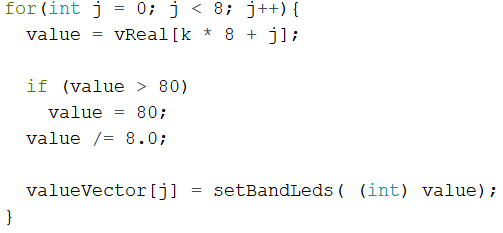
Matricea de led-uri este compusa din 4 matrici 8x8. Fiecare dintre acestea incep in modul sleep, deci trebuie ‘trezite’. In acelasi timp, la fiecare matrice setam intensitatea led-urilor. Setam pinul A7 ca fiind pin de input, de unde vom primit datele (muzica/sunetele) in real-time.



In continuare, in fiecare loop, citim de 64 de ori (numarul de sample-uri), si punem aceste date in vectorul de coeficienti reali, iar cel de coeficienti imaginari ramane setat pe 0, deoarece nu avem nevoie de el, nu lucram cu numere imaginare. Functia smooth() face 5 citiri, le aduna, si dupa le imparte tot la 5 (medie aritmetica), pentru a avea date mai coerente. Aceasta functie incetineste putin programul, dar rezultatul final este mai bun. Dintre cele 64 de sample-uri, doar primele 32 sunt bune. FFT trece celelalte 32 valori prin fază (phase), si sunt valori negative, si nu avem nevoie de ele. Din aceasta cauza am ales 64 de sample-uri, pentru ca raman cu 32, si se potriveste perfect cu matricea mea, care are 32 de led-uri lungime. Asa ca fiecare intrare din cele 32 de valori mi se potriveste unei coloane pe matrice.



Dupa ce avem cele 32 de valori, trebuie sa le transformam in date pe care le putem afisa pe o coloana. Transformata fourier a separat semnalul de intrare in frecvente. Fiecare valoare din cele 32 este un interval de frecventa. Am observat ca valorile transformatei fourier imi sunt intre 0 si 80-90-100. In continuare am zis ca valoarea maxima sa-mi fie 80. Acum am o valoare intre 0 si 80, si impartind la 8, o sa imi deie o valoare intre 0 si 8, care este un interval perfect, deoarece coloanele de pe matrice sunt de 8 led-uri inaltime. Aceasta valoare o trec prin functia setBandLeds(), care imi va da inapoi un byte care va reprezenta care led-uri sa fie aprinse si care sa fie stinse pe coloana.



Pot sa setez pe matrice led-uri cu ajutorul a 3 functii: setLed, setColumn si setRow. La prima vedere, functia setColumn este cea mai buna, deoarece eu trebuie sa setez coloane. Problema este ca aceasta functie este extraordinar de inceata, reusind sa comunic cu matricea aproximativ de 3 ori pe secunda, ceea ce este foarte lent. In acest caz, va trebuie sa transform datele in ‘date rand’, pentru a putea folosi functia setRow, care este mult mai rapida, reusind sa comunic cu matricea de aproximativ 10 ori pe secunda.

In valueVector[] am byte-urile care reprezinta led-urile de pe o coloana. Voi exemplifica ceea ce trebuie sa fac:

0 1 0 1

1 0 0 1

0 0 1 1

1 1 1 1

= = = =

V[0] v[1] v[2] v[3]

In v[0] de exemplu am “1010”, iar acesta dupa transormare va trebui sa devina “0101”:

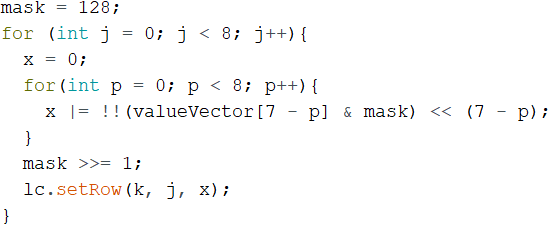
0 1 0 1 = v[0]

1 0 0 1 = v[1]

0 0 1 1 = v[2]

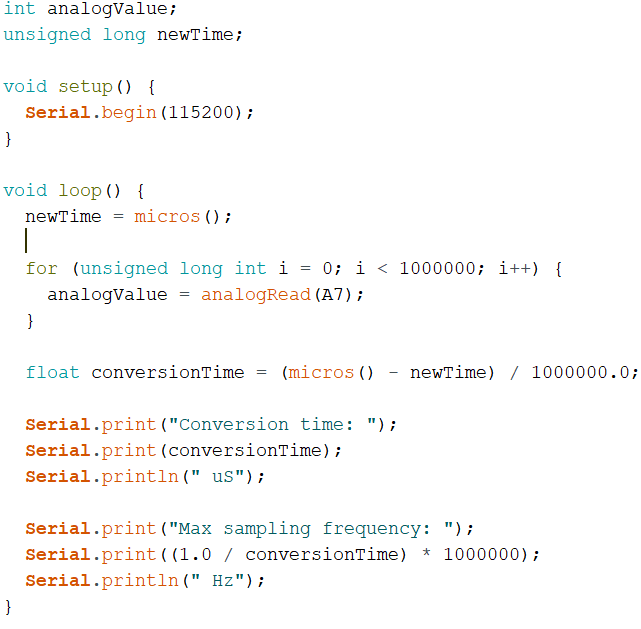
1 1 1 1 = v[3]

Aceasta transformare o voi face cu operatii pe biti. Am o masca care la inceput are valoarea 128 (b10000000), si o variabila x in care voi construi randul. Voi itera prin fiecare bit al valorilor din valueVector[], si voi face o operatie SAU cu masca, pentru a lua cei mai semificativi biti din fiecare valoare din vector, si pentru a le pune in x, una dupa alta. Dupa aceea voi shifta masca la dreapta, pentru a avea b01000000, si voi repeta acelasi proces. Dupa aceea voi seta randul de led-uri cu valoarea x.

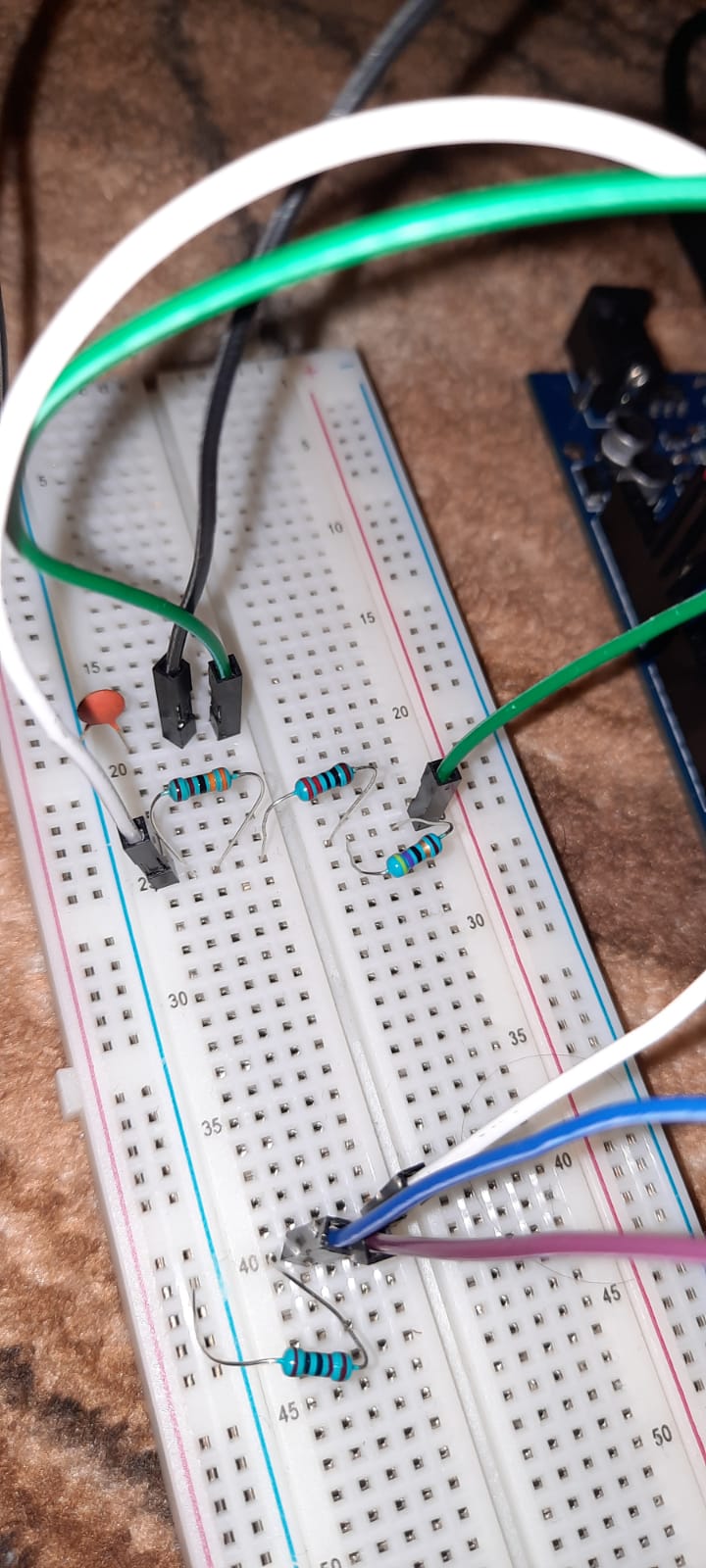


1. **Dificultati:**

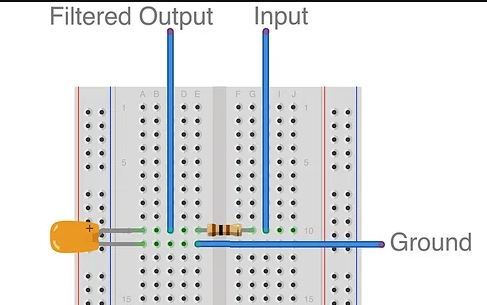
**a)** Exista o teorema, teorema lui Nyquist (Nyquist theorem), care spune ca frecventa maxima la care pot sa-i dau sample, este jumatate din rata de sampling. Deci daca vreau sa procesez frecvente de pana la 20kHz, trebuie sa am o rata de sampling de 40kHz. Problema majora este ca clock-ul de pe arduino este mult prea mic pentru a ajunge la 20kHz. La 16Mhz de pe arduino, rata maxima de sampling imi este 8928.56 Hz, adica pot procesa frecvente de maxim 4464.28 Hz. Avand in vedere ca in cod fac 2 lucruri foarte lente, adica transformata fourier si comunicarea cu matricea de led-uri, imi mai raman putine cicluri de ceas pentru a putea da sample la 4464 Hz, asa ca in practica pot detecta frecvente de maxim 450Hz, asa cum am aratat in video-ul de prezentare. Codul prin care am dedus ca rata de sampling a placii mele este de 8928.56Hz este urmatorul:



**b)** A doua dificultate cu care m-am intalnit este noise-ul. In mod normal, cand conectezi casti cu jack 3.5mm la un laptop sau telefon, nu ai noise in casti. Cablul meu jack bifurcat (la ca o bifurcatie merge catre arduino, iar cealalta imi merge in castile mele) are foarte mult noise pe care il aud in casti. Ma gandesc ca este de vina modul in care am taiat cablul si modol in care am legat firele de cupru de firele tata-tata, dar nu inteleg de ce ar avea atata noise, cat timp firele sunt cositorite si izolate. Am mai incercat sa mai tai un cablu nou, si si acela a avut la fe de mult noise. Am zis sa pun un filtru trece-jos, deoarece in casca am auzit noise-ul ca fiind o multitudine de frecvente acute. Am calculat cu formula de cutoff fc = 1/2piRC, ca RC trebuie sa fie in jur de 1/2800 (0.0003571) penntru ca fc sa fie aproximativ 450hz, adica sa tai frecventele care sunt mai inalte decat pot eu sa procesez. Cel mai mare condensator pe care l-am avut a fost de 100nF, asa ca aveam nevoie de o rezistenta in jur de 3571 ohmi, asa ca am pus in serie 3 rezistente, una de 3.3k, alta de 220 si alta de 47, ca sa mi deie aproape de 3571ohmi (3.3k + 220 + 47 = 3567). Din pacate, filtrul meu nu a functionat deloc. Dimpotriva, am avut mai mare noise, nu inteleg de ce. Acesta este montajul de la filtru:



Firul alb este semnalul de intrare, dupa vin 3 rezistori conectati in serie. Cablul verde ia semnalul trecut prin rezistori si il pune la un condensator conectat la ground. Output-ul meu este firul negru, care este intre firul verde si condensator. Am folosit urmatoarea schema pentru a ma ghida:



1. **Dezvoltari ulterioare:**

Prima idee ar fi bineinteles upgrade-ul hardware-ului, pentru a avea un clock are sa permita sampling ul frecventelor de pana la 20Mhz. Dupa aceea, pentru a scapa definitiv de noise, o idee ar fi sa folosesc un modul bluetooth pentru a transmite muzica/sunetele.