Universitatea Tehnică "Gheorghe Asachi" din Iaşi
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Procesarea Statistică a Semnalelor

Lucrarea de laborator nr. 2

Implementarea unui filtru de semnal ca un plugin Winamp

1 Obiectivul lucrării

Lucrarea urmărește implementarea unui filtru de procesare de semnal sub forma unui *plugin* pentru popularul program audio Winamp. Implementarea se va face în limbajul C, pe baza ecuației cu diferenţe a filtrului dorit.

2 Noțiuni teoretice

2.1 Ce este un plugin?

În terminologia utilizată în dezvoltarea de software, un plugin reprezintă o componentă opțională care adaugă o nouă funcționalitate unui program existent.

Exemple de plugin-uri:

- diverse filtre și transformări de imagini, în aplicații de prelucrare de imagini.
- diverse efecte audio, în *media-playere*.

În general, plugin-urile sunt implementate sub forma unui fișier de tip `.dll` aflat într-o locație prestabilită și având o anumită structură impusă.

2.2 Ce este un fișier `.dll`?

Un fișier cu extensia `.dll` (Dynamic Link Library) reprezintă o bibliotecă de funcții gata compilate și link-editate. O parte din funcții, care sunt declarate corespunzător, pot fi utilizate de un alt program. Se spune că acestea sunt *exportate* de către DLL. Funcțiile care nu sunt exportate nu sunt accesibile din exterior.

Fișierul `.dll` conține doar codul-obiect al funcțiilor, fără alte informații referitoare la tipul și numărul parametrilor. De aceea, pentru a folosi în practică funcțiile disponibile într-un DLL, este necesar a se include la compilare un fișier header (*.h*) care să conțină declarațiile funcțiilor disponibile în fișierul DLL.

2.3 Structura unui plugin de prelucrare a semnalelor în Winamp

Programul Winamp acceptă mai multe tipuri de fișiere *plugin*, pentru diverse funcționalități (pentru citirea a noi tipuri de fișiere, pentru ieșirea audio etc.). În cele ce urmează ne vom referi la un *plugin* destinat prelucrării semnalului audio.

Un astfel de *plugin* poate conține mai multe *module*, reprezentând efecte și funcționalități diferite implementate fizic în același fișier. *Plugin*-ul trebuie să implementeze o serie de funcții specifice predefinite pentru a putea interopera cu Winamp.

Un *plugin* trebuie să declare o variabilă de tip structură cu un format predefinit. Această structură conține, printre altele, numele *plugin*-ului și numele unei funcții care returnează modulele implementate. Această structură trebuie returnată de singura funcție exportată din DLL, `winampDSPGetHeader2()`, care este apelată de către Winamp la selectarea *plugin*-ului în fereastra de preferințe.

Un *modul* reprezintă un efect sau filtru implementat într-un *plugin*. Este caracterizat de o structură cu un format predefinit, care conține numele modulului precum și numele a patru funcții definite în fișier, pe care Winamp le apelează la momente specifice: la inițializarea modulului, la configurarea sa, atunci când se rulează un cântec, și la dezactivarea sa. Toate operațiile pe care le efectuează modulul sunt concentrate în aceste patru funcții, pe care le scrie programatorul în funcție de scopul urmărit. Dintre cele patru funcții, acționarea propriu-zisă asupra eșantioanelor semnalului se realizează în cea de-a treia.

În concluzie, din punct de vedere practic, pentru a realiza un modul în cadrul unui astfel de plugin pentru Winamp este necesar a scrie aceste patru funcții și a completa câteva variabile tip structură după un format prestabilit.

2.4 Etapele apelării unui *plugin* de prelucrare a semnalului

1. Plugin-ul trebuie sa fie numit “dsp_*.dll” și să se găsească în directorul “Plugins”.
2. Winamp găsește toate fișierele “dsp_*.dll” din directorul respectiv, și pentru fiecare apelează funcția `winampDSPGetHeader2()` (singura funcție exportată din DLL).
3. Funcția `winampDSPGetHeader2()` returnează către Winamp structura care conține numele plugin-ului și numele (adresa) unei funcții care returnează modulele implementate.
4. Winamp apelează repetat funcția de mai sus și primește structurile care cuprinde informațiile fiecărui modul. Acestea conțin, printre altele:
   (a) Numele modulului
   (b) O funcție de inițializare (`init()`), care se va apela atunci când utilizatorul selectează modulul în fereastra *Preferences*
   (c) O funcție `quit()` care se apelează la de-selectarea modulului (pentru dealocări de memorie etc.)
   (d) O funcție de configurare (`config()`) care se apelează atunci când utilizatorul apasă butonul de configurare
   (e) O funcție de procesare (`process()`) care primește ca parametru eșantioanele audio înainte de a fi trimise spre redare.

5. Utilizatorul selectează plugin-ul şi modulul dorit din fereastra *Preferences*. În acest moment Winamp va executa funcția `init()` a modulului.

2.5 Formatul datelor prelucrate

2.5.1 Funcția de procesare a datelor

În momentul redării unui fișier, Winamp apelează funcția de procesare a datelor (`process()`) indicată în structura modulului selectat. Această funcție primește următorii parametri de intrare, care conțin toate informațiile necesare pentru a prelucra eșantioanele:

```
int process(struct winampDSPModule *this_mod, short int *samples,
                  int numsamples, int bps, int nch, int srate)
{
        ...
}
```

Descrierea parametrilor:

`struct winampDSPModule *this_mod`
: Pointer către structura modulului, care conține numele modulului și adresele tuturor funcțiilor.

`short int *samples`
: Un vector de tipul `short int` care conține eșantioanele (`samples[0]`, `samples[1]` etc.). Dacă fișierul este stereo, vor exista eșantioane pentru ambele canale stânga / dreapta. Un variabilă de tip `short int` este o variabilă de tip întreg care ocupă 2 octeți.

`int numsamples`
: Numărul de eșantioane disponibile într-un canal. Atenție: nu este vorba de dimensiunea vectorului `samples`, care depinde de numărul de biți pe eșantion și de numărul de canale.

`int bps`
: Numărul de biți pe eșantion (8, 16 etc.).

`int nch`
: Numărul de canale (1=mono, 2=stereo).

`int srate`
: Frecvența de eșantionare.

2.5.2 Formatul datelor prelucrate

Pentru a prelucra datele, este necesar a înțelege formatul în care sunt stocate datele în vectorul de eșantioane `samples` primit de funcția de procesare.

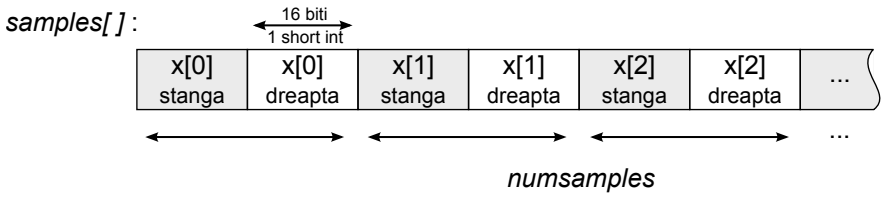

Figura 1: Formatul vectorului de date `samples[]` în cazul unui fișier cu două canale (stereo), eșantionat pe 16 biți. Un bloc reprezintă un element din vectorul de date (1 short int).

Numărul de canale

Eșantioanele sunt stocate în ordine, dar, dacă fișierul este stereo, sunt dispuse alternativ pentru canalul stâng / drept (Fig.1).

Biți pe eșantion

Vectorul de date este de tip `short int`, adică fiecare element ocupă 2 octeți. Dacă sunt 16 biți pe eșantion, un element al vectorului reprezintă exact un eșantion. Dacă sunt 8 biți pe eșantion, atunci fiecare element al vectorului cuprinde două eșantioane succesive (fișier mono - două eșantioane succesive; fișier stereo - eșantioanele stânga și dreapta de la același moment de timp).

Numărul de eșantioane

Din motive de eficiență, Winamp nu prelucrează întreg fișierul odată, ci operează pe ferestre de `numsamples` eșantioane (uzual 1152 pentru fișiere `*.mp3`). Funcția de procesare se va apela repetat pentru fiecare fereastră succesivă de 1152 eșantioane. Aceasta impune **atenție** la implementarea operației de filtrare, întrucât pentru filtrare sunt necesare de obicei ultimele n eșantioane ale intrării și ale ieșirii, care însă nu sunt disponibile la începutul unei astfel de ferestre (vezi Fig.3).. De aceea se impune salvarea în niște vectori auxiliari a ultimelor eșantioane ale intrării și ieșirii din cadrul unei ferestre pentru a fi disponibile în cadrul ferestrei următoare.

Pentru cazul unui semnal stereo, cu eșantioanele cuantizare pe 16 biți, formatul datelor este prezentat în Fig.1.

3 Desfășurarea lucrării

Aplicaţiile practice se efectuează utilizând programele software atașate lucrării:

- Compilatorul TCC (*Tiny C Compiler*), apelat prin script-ul `compile.bat`.
- Programul Winamp
- Programul Notepad++ (pentru editarea codului C)
- *Kit*-ul de programare pentru Winamp (*Winamp SDK*) necesar compilării unui *plugin*.

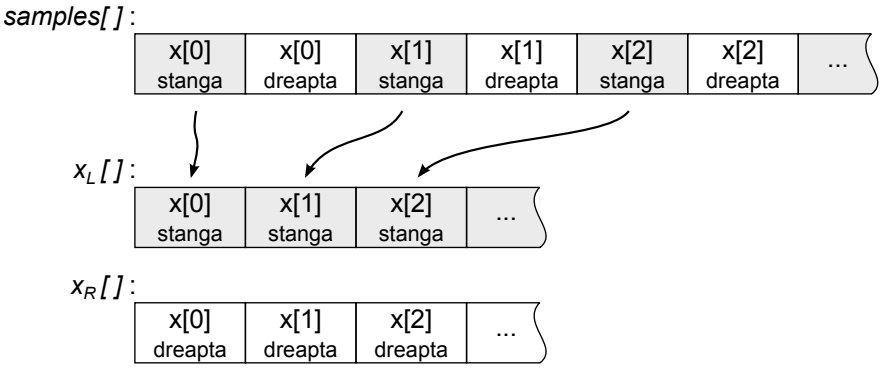

Figura 2: Separarea datelor din vectorul de intrare în canalele stânga/dreapta.

Exercițiul 1 *Plugin*-uri în Winamp

Deschideți programul Winamp. Deschideți meniul de selectare a *plugin*-urilor (Preferences (Ctrl-P) → Plugins → DSP/Effect). Selectați și deselectați plugin-ul `dsp _pss.dll` și modulul implementat, apăsați butonul de configurare. Ce se întâmplă la fiecare din aceste operații?

Exercițiul 2 Structura unui plugin

Deschideți fișierul sursă `dsp_pss.c` folosind, de preferință, programul Notepad++.

a). Identificați funcția exportată de DLL şi structura *header* a pluginului. Modificați descrierea plugin-ului în "Plugin-ul PSS". Ce reprezintă parametrul `getModule` care urmează descrierii?

b). Identificați funcția de returnare a modulelor şi structura modulului disponibil. Modificați descrierea modulului în "Filtru 1". Ce reprezintă parametrii `config`, `init`, `process`, `quit`?

c). Identificați funcțiile `config()`, `init()`, `process()`, `quit()`. Care este efectul apelării funcției `MessageBox()`? Modificați unul din șirurile de caractere furnizate ca parametri.

d). Compilați fișierul sursă prin rularea `compile.bat`. Copiați fișierul `.dll` rezultat în directorul *Winamp* *Plugins*. Reporniți Winamp și intrați în meniul pentru *plugin*-uri (Preferences (Ctrl-P) → Plugins → DSP/Effect). Care este efectul modificărilor făcute mai sus?

Exercițiul 3 Proiectarea unui filtru

a). Proiectați un filtru IIR trece-jos de ordin 3, cu frecvența de tăiere de 1 kHz la o frecvență de eșantionare de 44.1 kHz, folosind interfața grafică Matlab (`fdatool`). Afișați coeficienții formei directe II.

b). Identificați în fișierul sursă `dsp_pss.c` definirea coeficienților `a` și `b` ai filtrului. Introduceți valorile filtrului proiectat mai sus.

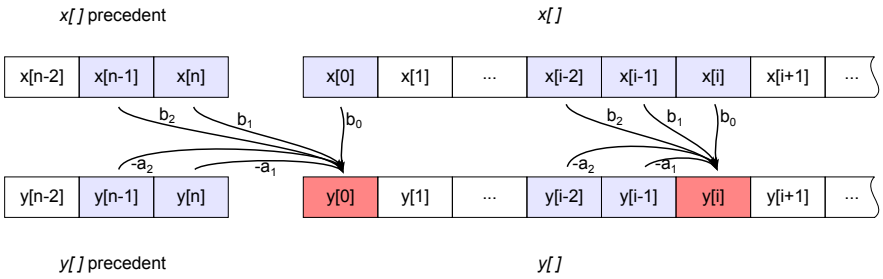

Figura 3: Implementarea ecuației cu diferențe pentru un filtru de ordin 2. Se remarcă faptul că este nevoie de intrarea şi ieșirea din blocul anterior.

Exercițiul 4 Filtrarea datelor

Identificați funcția `process()` din fișierul sursă `dsp_pss.c`. Se va considera numai cazul unui fișier stereo, cu eșantioane pe 16 biți.

a). Scrieți codul pentru a separa datele din vectorul de intrare în doi vectori corespunzători celor două canale stânga/dreapta, conform Fig.2.

b). Studiați Fig.3. Scrieți codul pentru a calcula y[0], y[1] şi y[2] pe baza ecuației cu diferențe. Scrieți apoi codul pentru a calcula y[i] în general, pentru $i \geq 3$.

c). Scrieți codul pentru a salva ultimele valori ale intrării şi ale ieșirii, în vederea utilizării lor în blocul următor.

d). Scrieți codul pentru a recombina valorile semnalelor de ieșire pentru cele două canale în vectorul de intrare (operație inversă celei din Fig.2).

Exercițiul 5 (TEMĂ) Fișiere mono

Exercițiul 4 funcționează numai pentru cazul unui fișier audio cu două canale (stereo), cu eșantionare pe 16 biți. Implementați, pe lângă acesta, și cazul mono/16 biți.

* Exercițiul 6 (opțional) Eșantionare pe 8 biți

Exercițiul 4 funcționează numai pentru cazul unui fișier audio cu două canale (stereo), cu eșantionare pe 16 biți. Implementați și cazurile stereo/8 biți, mono/8 biți.

* Exercițiul 7 (opțional) Filtru cu ordin general

În această lucrare s-a implementat un filtru de ordin 3. Implementați cazul general pentru un filtru de ordin `ord`, pentru o valoare `ord` oarecare.

Exercițiul 8 (opțional) Citire coeficienți dintr-un fișier

Realizați citirea coeficienților `a` și `b` dintr-un fișier text de pe disc. În acest fel se poate modifica filtrul implementat fără a mai fi necesară recompilarea, doar prin schimbarea valorilor din acest fișier.

** Exercițiul 9 (opțional) Implementare în spațiul stărilor

Implementați filtrarea folosind ecuațiile din spațiul stărilor tip I sau II. (*Hints*: definiți `v1`, `v2`, `v3`; la fiecare iterație calculați `y[i]` conform ecuației de ieșire, apoi actualizaţi starea;

starea poate fi păstrată între două apeluri succesive ale funcției dacă variabilele sunt declarate `static`).

4 Întrebări finale

1. Ce se modifică în cazul implementării unui filtru FIR?
2. TBD
3. TBD