III.1. Introduction

Ce chapitre présente les différentes parties pour l'acquisition et l'affichage des données sur ordinateur qui sont les suivantes :

- ❖ Mise en forme de signal avant la carte son.
- ❖ La carte son (sound card).
- ❖ Programmation sous DELPHI.
- ❖ Le programme de l'affichage.

III.2. Mise en forme de signal d'entrée

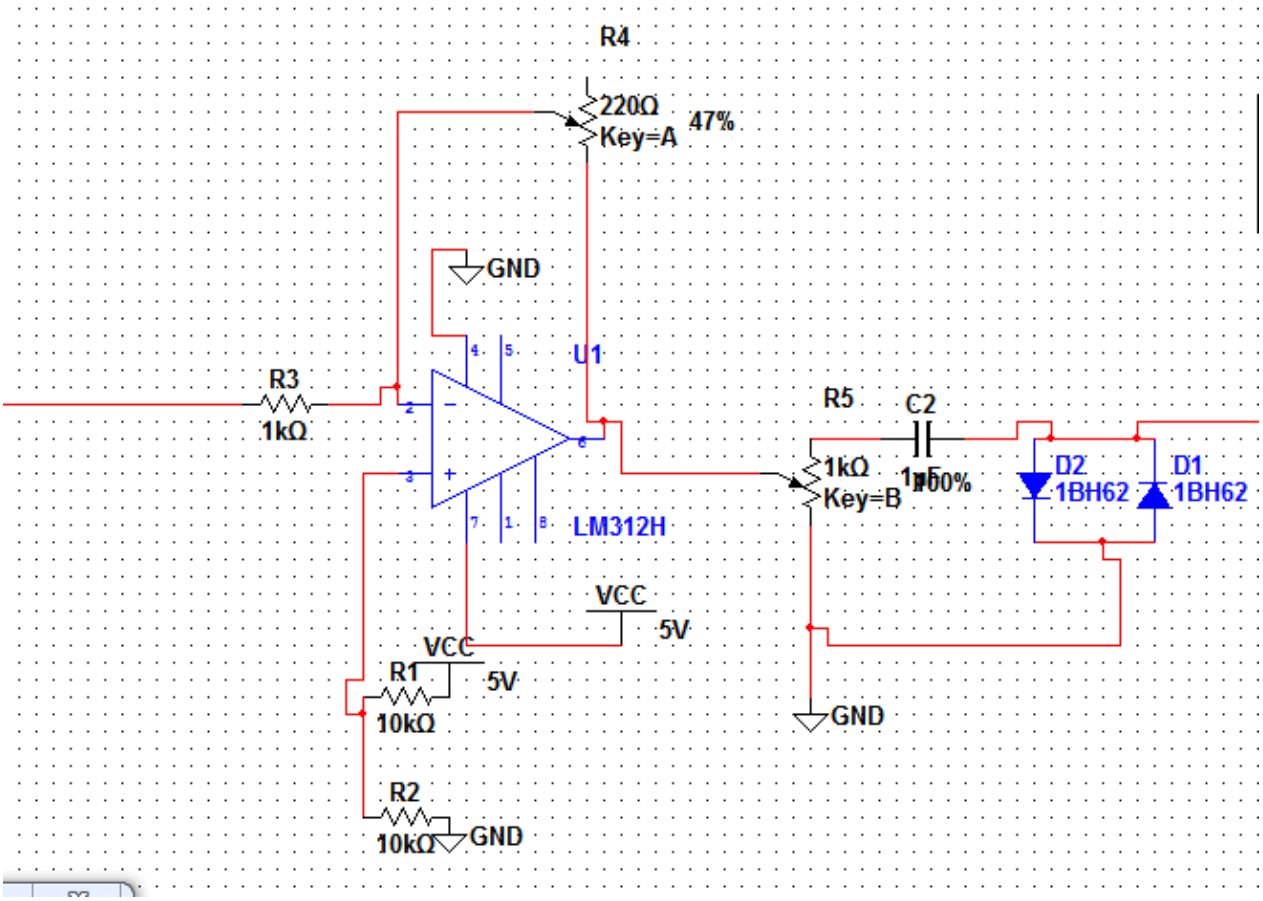

Figure 19 : Circuit d'un régulateur de tension.

On sait que la tension à injecter à la carte son du micro-ordinateur ne doit pas dépasser 1vcc donc il nous faudra un circuit qui gardera cette tension toujours au-dessous de cette maximale et qui le protégera en cas de dépassement.

Si on prend e+=e- et i+=i- et que l'AOP est parfait alors : Applique le théorème de Millman :

$e+=\frac{\frac{5}{R2}}{\frac{1}{R2}+\frac{1}{R1}}$

$e-=\frac{\frac{S}{Rv1}+\frac{e}{R1}}{\frac{1}{Rv1}+\frac{1}{R3}}$

Puisque R2=R1 alors

$S=\frac{5}{2}-\frac{Rv1}{R3}(e-\frac{5}{2})$

Dans cette équation on Remarque que la tension de sortie est en fonction de deux paramètres e (tension d'entrée) et R_{v1} (potentiomètre).

Remarque

- Si on diminue R_{v1} alors on remarque que la tension de sortie augmente.
- Si on augmente R_{v1} alors la tension de sortie démunie.
- S change de valeur avec e mais en opposé.
- Le circuit manipule le signal en signe (il joue le rôle d'un inverseur) et en amplitude (un abaisseur de tension).
- À la sortie de circuit on a posé un passe bas pour éliminer tous les bruit dû au circuit.
- pour protéger la carte son on a utilisé deux diodes pour limiter la tension entre [-0.6-+0.6]

III.3. La carte son

Une carte son est une carte d'extension d'ordinateur. La principale fonction de cette carte est de gérer tous les sons émis, quelle envoie vers les haut-parleurs ou reçus par l'ordinateur. Elle se présente sous la forme d'un périphérique que l'on peut connecter à l'ordinateur sur un bus PCI, PCI Express, PCMCIA (pour ordinateur portable), USB ou Firewire.1.

Figure 20 : Carte son ordinateur.

III.3.1. Architecture

La carte son repose généralement sur un processeur DSP (Digital Signal Processor) pour le traitement des signaux audio, qui communique avec le processeur central (CPU) via le bus d'extension de l'ordinateur (PCI ou PCI-E). Elle est équipée de convertisseurs analogique/numérique pour numériser des signaux externes (micro ...), et de convertisseurs numérique/analogique pour restituer les signaux audibles vers les enceintes ou le casque (nous on s'intéresse seulement aux entrées). La plupart possèdent également une interface MIDI pour communiquer avec des synthétiseurs, également utilisé pour connecter un joystick.

Les DSP des cartes son, étant spécialisés pour le traitement des signaux sonores, sont souvent appelés APU (Audio Processing Unit). Pour un besoin d'efficacité, les APU accèdent à la mémoire centrale (RAM) par un bus DMA pour ne pas avoir à surcharger le processeur central.

Certaines cartes hautes gamme comportent plusieurs processeurs DSP, de la mémoire additionnelle, des entrées-sorties numériques, ou encore un boîtier de connexion externe (contenant les convertisseurs). D'autres, de base gamme, sont directement intégrées à la carte mère.

Il faut savoir qu'une carte son fonctionne la plupart du temps en mode numérique, cela veut donc dire que le signal qui est lu l'est dans la plupart des cas sous forme numérique. L'unité de base d'une carte son est donc l'échantillon.

Les cartes son sont généralement classées suivant deux critères principaux : la résolution et l'échantillonnage. La résolution correspond au niveau de détail d'un échantillon, plus la résolution est élevée, plus le son sera précis et fin. Actuellement les cartes son grand public fonctionnent en 16 ou 24 bits, cela signifie que lors de la numérisation, le signal analogique peut être codé sur 16 ou 24 bits, c'est à dire codé respectivement sur 65536 valeurs ou sur un petit peu plus de 16 millions de valeurs.

Le deuxième critère de sélection est l'échantillonnage, à ce critère correspond une fréquence exprimée en hertz ou en kilohertz. Cette fréquence correspond au nombre d'échantillons qui seront produits à la seconde lors de l'échantillonnage. Les cartes son actuelles présentent des fréquences d'échantillonnage de l'ordre de 44100 Hz à 192 kHz. Plus l'échantillonnage est élevé, plus le son est détaillé.

III.3.2. Les composants de la carte son et leurs fonctionnements

Le processeur de la carte son le DSP

Chaque carte son possède son processeur le DSP (Digital Signal Processor).Cette puce va s'occuper de transcrire les signaux numériques qui proviennent du processeur et les transformer en sons audibles. Les DSP les plus évolués permettent de rajouter de l'écho, de la distorsion. C'est aussi lui qui distribue les différents sons sur les sorties. Le DSP va donc prendre en charge la plupart des calculs audio, le reste sera laissé au processeur de l'ordinateur. Plus le DSP sera puissant, et moins le CPU de l'ordinateur travaillera.

Beaucoup de cartes mères intègrent maintenant des cartes son intégrées. Les meilleures cartes son intégrées ne sont pas au niveau de celles sur port PCI. Cependant, certaines offrent 3 sorties disponibles, et donc, permettent de décoder le son 5.1 des films DVD. La plupart restent en stéréo, et n'offrent pas beaucoup de possibilités. Il est bien sûr possible de les désactiver si l'on veut brancher une carte son PCI, ce qui est préférable pour les jeux ou autre activité comme dans notre cas qui dépasse le domaine de la bureautique.

Les convertisseurs

Il y a deux types de convertisseurs analogiques-numériques (CAN) pour enregistrer le son, et numériques-analogiques (CNA) pour le restituer afin de pouvoir l'entendre.

Comme la plupart des musiciens utilisent la carte son pour enregistrer des instruments, la caractéristique la plus importante est certainement la qualité des CAN. En effet, ce sont ces convertisseurs qui sont chargés de transformer une oscillation (le son d'origine) en une suite de nombres. Les CNA, quant à eux, servent généralement à écouter le résultat d'un mix par exemple, mais n'altèrent aucunement le son enregistré. Ils ont donc une importance non négligeable mais moins grande que les CAN, à moins de réutiliser les sorties de la carte pour enregistrer le son sur un autre support. Dans tous les cas, la qualité des deux types de convertisseurs pour une même carte est la même et les caractéristiques d'un convertisseur sont :

- Son pas de quantification (appelé aussi dynamique ou résolution).
- Sa fréquence d'échantillonnage ($f>=2*f_{max}$).
- Son facteur de suréchantillonnage (ou "oversampling").
- .Son rapport signal / bruit.

Générateur interne

Plutôt rares sur les cartes son semi-professionnelles et professionnelles, les générateurs sonores internes à la carte son peuvent être intéressants dans certains cas. Il s'agit soit d'un expandeur interne, soit de la gestion de banques de son propres à la carte, comme les Sound Fonts (format propriétaire de Sound Blaster).

Aujourd'hui, la création de synthétiseurs et de sampleurs virtuels rendent les cartes son à table d'onde un peu désuètes dans le cas d'une configuration informatique puissante. En effet, si l'on a un synthétiseur à table d'ondes sur la carte, le jeu des sons est entièrement géré par celle-ci, alors que dans le cas des synthétiseurs et sampleurs virtuels, c'est le processeur de l'ordinateur qui travaille.

La latence

La latence d'une carte son est un élément de plus en plus important aujourd'hui où l'on utilise des synthétiseurs virtuels et des effets en temps réel. En effet, la latence désigne, pour simplifier, le temps mis par la carte audionumérique entre le moment où l'on veut qu'elle émette un son et celui où elle l'émet réellement. Cela se caractérise par le retard entre l'appui d'une touche du clavier MIDI et l'émission du son voulu par le synthétiseur virtuel.

Entrées analogiques

L'entrée analogique nous permet de transporter le signal audio d'un instrument musical ou un autre signal comme dans notre cas les signaux électro-physiologiques qui se trouve sous forme d'une fiche jacke soit stéréo ou mono.

Le signal est amplifiéavec une très grande capacité et l'amplitude de l'entrée est limité à 1Volt.[13]

III.4. Programmation sous DELPHI [14][15]

Delphi est un environnement de développement de type RAD (Rapid Application Development) basé sur le langage Pascal. Il permet de réaliser rapidement et simplement des applications Windows. Cette rapidité et cette simplicité de développement sont dues à une conception visuelle de l'application. Delphi propose un ensemble très complet de composants visuels prêts à l'emploi incluant la quasi-totalité des composants Windows (boutons, boîtes de dialogue, menus, barres d'outils) ainsi que des experts permettant de créer facilement divers types d'applications et de librairies. Pour maîtriser le développement d'une application sous Delphi, il est indispensable d'aborder les trois sujets suivants :

- Le langage Pascal et la programmation orientée objet.
- L'Environnement de Développement Intégré (EDI) de Delphi.
- Les objets de Delphi et la hiérarchie de classe de sa bibliothèque.

III.4.1. Le langage pascal

Le langage de programmation Pascal a été conçu en 1968 par Niklaus Wirth.

III.4.1.1. Éléments du langage :

Identificateurs et instructions :

Un identificateur est un nom permettant au compilateur d'identifier un objet donné. Les noms de variables, par exemple, sont des identificateurs. Un identificateur doit commencer par une lettre. Les caractères suivants peuvent être des lettres, des chiffres ou le caractère _. Majuscules et minuscules ne sont pas différenciées. Seuls les 63 premiers caractères d'un identificateur sont pris en considération par le compilateur. Identificateur valide :

Ma_variable01 **Identificateur invalide** : 9variable.

Outre les variables, constantes, etc., un programme Pascal contient des mots réservés que l'utilisateur ne peut pas employer. Ce groupe d'identificateurs particuliers correspond aux composants du langage Pascal.

Opérateurs

Affectation :

Ex. resultat:=100;

Opérateurs arithmétiques :

- Multiplication.
- Division entière : div.
- Division.
- Modulo : mod.
- Addition.
- Soustraction.

Opérateurs logiques :

- Et logique : and.
- Ou logique : or.
- Ou exclusif : xor.

- Négation : not.

Opérateurs de relation :

- Égal.
- Différent : <>.
- Supérieur/Supérieur ou égal : > >=.
- Inférieur/Inférieur ou égal : < <=.
- Appartenance à un ensemble : in.

Commentaires :

- (* … *) ou { … }.

III.4.1.2. Types de données :

Types prédéfinis :

Tableau 2 : différents type de données pascale :

Types entiers	Domaine
Byte	0..255
Shortint	-128..127
Integer	-32768..32767
Word	0..65535
Longint	-2147483648..2147483647

Types réels	Domaine
Single	$1,5.10^{-45}..3,4.10^{38}$
Real	$2,9.10^{-39}..1,7.10^{38}$
Double	$5,0.10^{-324}..1,7.10^{308}$
Extended	$3,4.10^{-4951}..1,1.10^{4932}$

Type booléen	Domaine
Boolean	True | False

Types caractères	Domaine
Char	Caractère alphanumérique
String[n]	Chaîne de n caractères (n = 255 au maximum)
String	Chaîne de 255 caractères

Types tableaux
Array[imin..imax, …] of <type>

Types personnalisés :

La déclaration d'un type utilisateur s'effectue dans une clause **type**.

Squelette d'un programme Pascal :

```
Program Nom_du_programme;
Uses {unités}
Const {Déclaration de constantes}
Type {Déclaration de types personnalisés}
Var {Déclaration de variables}
{Procédures et fonctions}
Begin
  {Bloc principal du programme}
End.
```

III.4.2. L'EDI de Delphi :

III.4.2.1. L'interface de Delphi

La figure ci-dessous représente l'interface typique de Delphi. Elle est composée de :

- La barre de menus (en haut),
- La barre d'icônes (à gauche sous la barre de menus),

- La palette de composants (à droite sous la barre de menus),
- Le concepteur de fiche (au centre),
- L'éditeur de code (au centre sous le concepteur de fiche),
- L'inspecteur d'objets (à gauche).

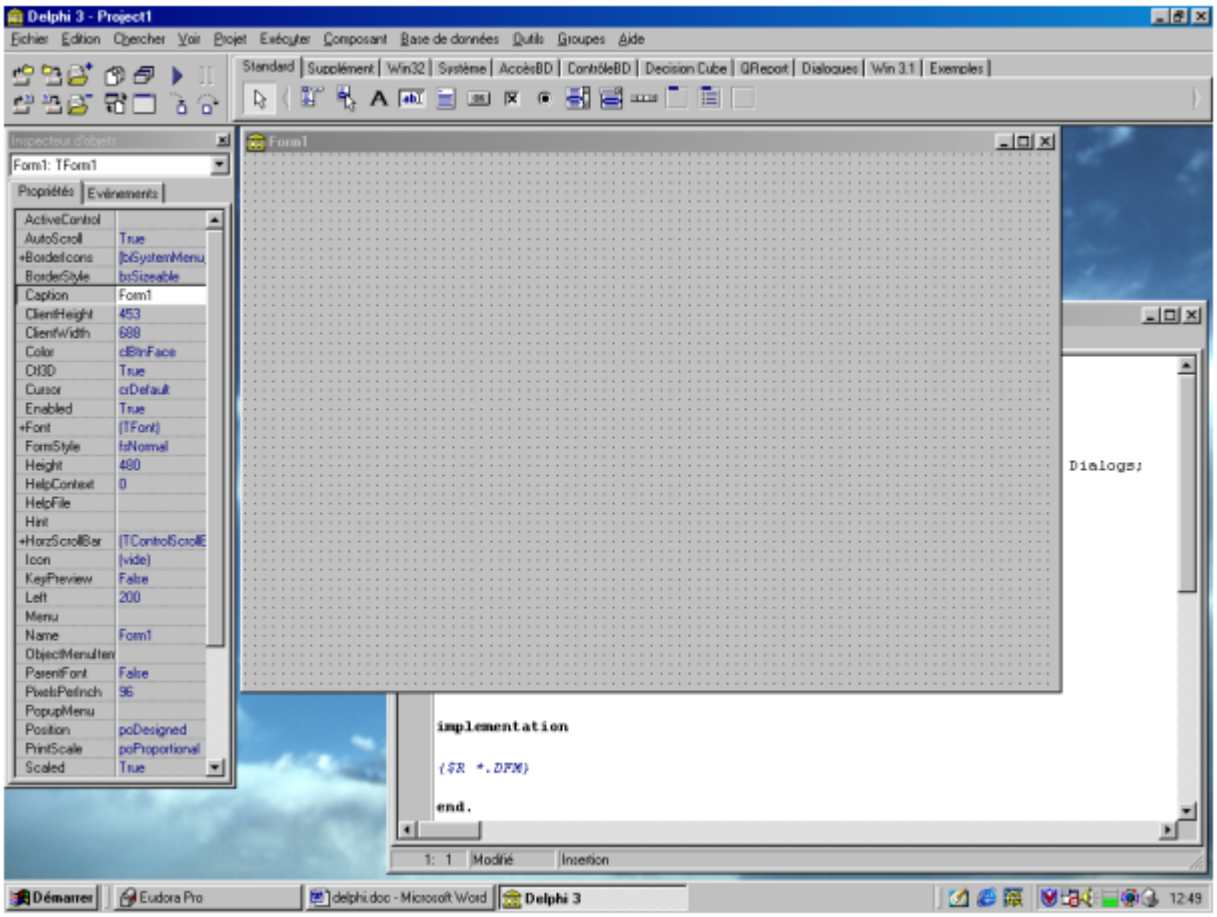

Figure 21 : L'interface de Delphi[15].

Conception de fiches : la palette des composants :

Une fiche constitue l'interface (ou une partie de l'interface) d'une application. Pour concevoir une fiche, il suffit d'y insérer des contrôles (ressources Windows prêtes à l'emploi : boutons de commande, listes, menus…) listés dans la palette des composants. Un clic sur le contrôle, puis un autre sur la fiche cible suffisent (un double clic insère le composant au milieu de la fiche active). La palette des composants réunit plusieurs volets. Les principaux sont listés ci-dessous.

Figure 22 : composants standards.

Figure 23 : composants supplémentaires.

Figure 24 : composants boites de dialogues.

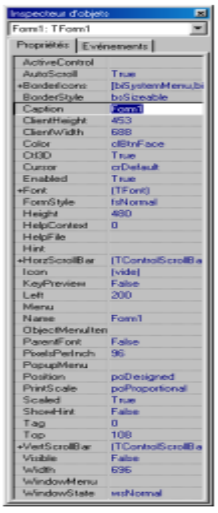

Figure 25 : l'inspecteur d'objet.

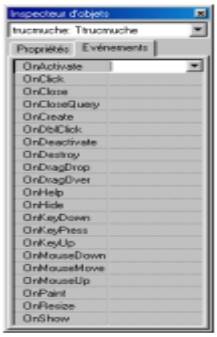

Figure 26 : évènements.

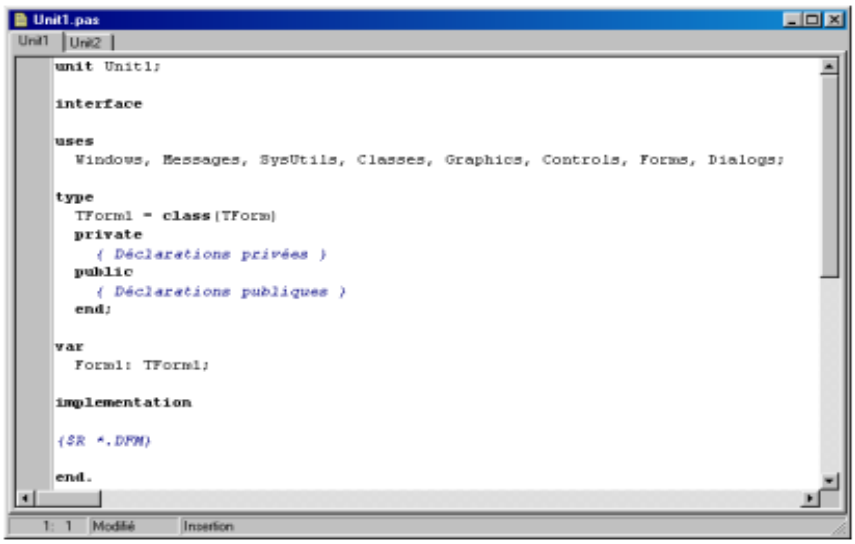

Figure 27 : éditeur de codes.

III.5. Le programme de l'affichage :

Pour l'enregistrement de son la carte saisit l'information et l'écrit sous la forme d'un bloc de données qui enregistre les données selon le format wav ou comme une image. On dispose, pour le paramétrage de l'enregistreur, du type TPCM Wave Format. Il travaille avec le format PCM déjà évoqué et que nous avons également rencontré avec l'enregistreur de sons. On y indique, entre autres, le nombre de bits par échantillon et le nombre d'échantillons par seconde utilisé. Cela donne un enregistrement (record) du type TPCMW wave Format dans lequel il est mentionné que nous utilisons le format PCM, le nombre d'échantillons par seconde et quelques autres paramètres.

III.5.1. Le code :

Le code-source de l'exemple est subdivisé en 3 blocs, unités, chacun d'entre eux représentant un bloc fonctionnel du programme. Unit1 comporte la partie visuelle dans laquelle se trouve la forme visualisée. Unité2 intègre tout ce qu'il faut pour créer la forme lors du démarrage, pour y afficher et y exécuter des tâches une fois que le programme est en cours d'exécution. On crée l'arrière-plan, y superpose le signal. Unit3, pour finir, rassemble tout ce qui concerne l'acquisition de données. Nous retrouvons MMSystem dans la ligne « Uses » et découvrons plusieurs routines définies

: LogInit, LogClose, LogStart et LogStop. Elles servent à la gestion de la saisie des données.

III.5.2. Le programme

Unit1 constitue la partie la plus importante, l'écran y est créé et le pilotage de tous les processus y prend place, mais on y fait également appel à des routines implantées dans les 2 autres Unités. Ceci permet d'avoir un programme lisible, ce qui est plus particulièrement important lorsqu'un projet commence à prendre un certain embonpoint. Sous Delphi, démarrons un nouveau projet par : Fichier/Nouveau/ Application. Nous voyons une forme vierge. Une unité standard se compose de 2 ensembles (l'unité proprement dite dans laquelle sont placé le code et une Forme sur laquelle sont placés les objets à utiliser pour le programme). Il nous faut en outre réutiliser Unit2 du projet et nous allons ajouter les routines de la 3ème unité. La caractéristique de ces 2 dernières unités est qu'elles sont constituées à 100% de code, ne comportant par conséquent pas d'objet. D'où l'absence de Forme. Nous commençons avec Unit1 qui nous sert à créer le GUI (Graphical User Interface) et là par l'arrière-plan de la forme. Nous allons utiliser, pour accélérer la création de l'écran, un bitmap placé d'un coup sur l'écran à chaque fois qu'il faut redessiner l'écran. Nous prenons une TPaintBox (de l'onglet Système de la Palette Composants) et la collons dans la forme. (TPaintBox fournit un canevas que les applications peuvent utiliser pour y plaquer une illustration). Une PaintBox a l'avantage de dessiner directement sur le canevas de la forme, mais pour ainsi dire dans une partie encadrée. Si l'on y charge alors une image, cette dernière est placée en une opération sur le canevas, ce qui élimine tout problème de scintillement. Il ne faut pas oublier que le canevas possède des coordonnées que nous choisissant selon notre fenêtre et voici l'image final de notre afficheur.

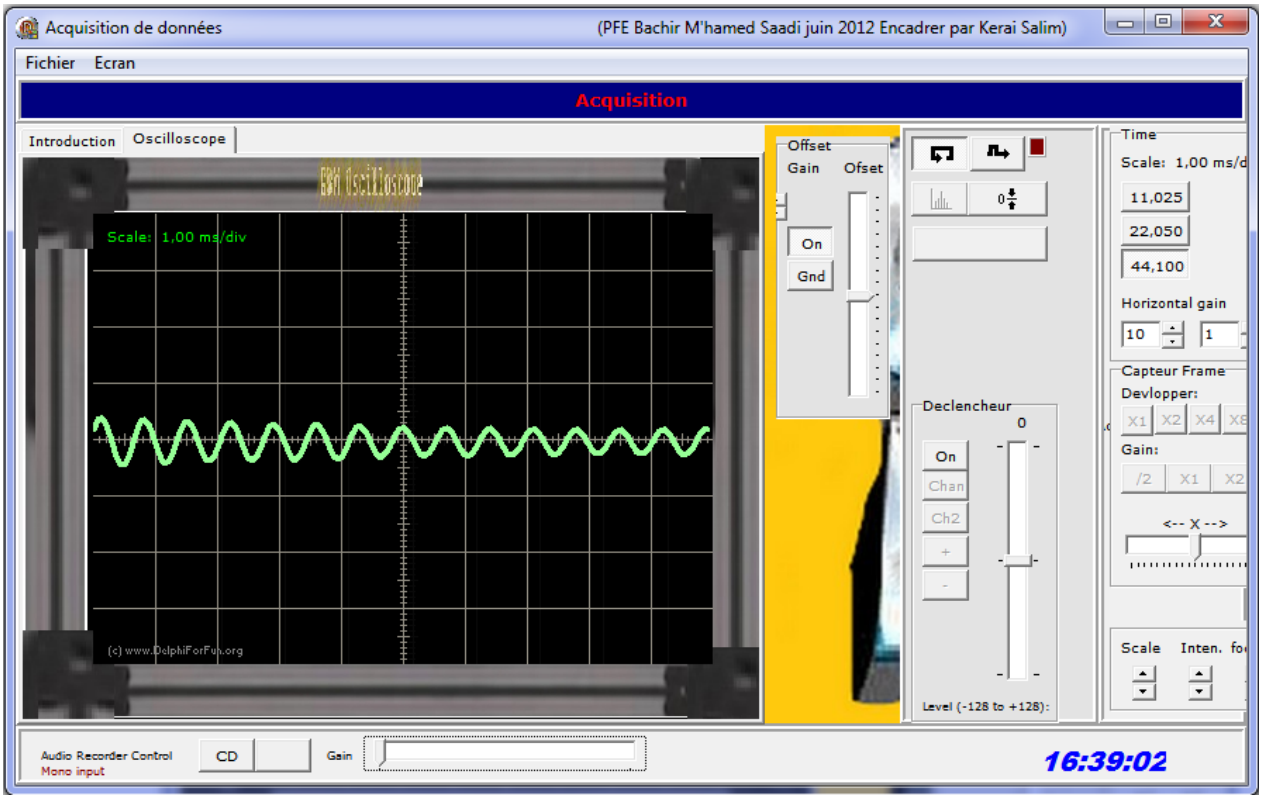

Figure 28 : Afficheur de données

L'acquisition :

Voici une procédure qui va nous permet de lancer l'acquisition et d'afficher si la réception des données est commencé ou non.

```
Procedure TfrmMain.Start; {Début de l'acquisition}
begin
  memo1.color:=clWindow;
  memo1.Clear;
  memo1.lines.add('Messages');
  Setup; {Ovrir wavein et prépare le buffers}
  singleframe := false;
  triggered   := false;
  if wavein.recordactive then
  begin
    wavein.stopinput;
    application.processmessages;
```

```
  end;
  Wavein.StartInput; {Début d'aquisition}
  statustext.caption := 'Acquisition';
end;
```

Après avoir lancé l'acquisition il faut s'assurer d'enregistrer les données et pour ça nous avons mis en place cette procédure qui va nous permettre d'enregistrer les données sous deux forme : image ou son. En effet, on peut les utiliser sur d'autres logiciels de traitement de signal comme matlab et il est très facile de faire appel à ce type de fichier.

```
procedure TfrmMain.BtnOneFrameClick(Sender: TObject);
begin
  if BtnOneFrame.Down then
  begin
    StoredCH1Offs :=trOfsCh1.Position;
    StoredCH2Offs :=trOfsCh2.Position;
    SetButtonstate;
    Recalc;
    btnExpand1.Down := True;
    btnGain1.Down   := True;
    memo1.color := clWindow;
    memo1.Clear;
    memo1.lines.add('Messages');
    Setup; {ovrire wavein et prepare le buffers}
    singleframe  := true;
    {if triggerRgrp.itemindex=1 then triggerrgrp.itemindex:=0; }
    triggered    := false;
    triggerindex := 0;
    if assigned(wavein) and wavein.recordactive then
    begin
      Wavein.stopinput;
```

```
    application.processmessages;
   end;
   {debug}
   Buffer1found := false;
   if assigned(wavein) then
    Wavein.StartInput; {debut d'enregistrement}
   statustext.caption := 'Image capturer- attendre';
  end
  else
  begin
   singleframe  := false;
   if assigned(wavein) and wavein.recordactive then
   begin
    Wavein.stopinput;
    application.processmessages;
   end;
   SetButtonstate;
  end;
end;
```

Pour le changement de l'échelle on va choisir 3 valeurs avec cette procédure.

```
Procedure TForm1.DoSample (Sender : TObject);
Var SampleRate : Integer;
Begin
Button2.Enabled := False;
Case RadioGroup1.ItemIndex Of
  SampleRate :=  11025 ;
  SampleRate :=  22050 ;
  SampleRate :=  44100 ;
End;
```

```
// Initialisation carte-son
LogInit (Form1.Handle, SampleRate);
LogStart (@WaveData, SizeOf (WaveData));
End;
```

Pour le gain horizontal qui va faire apparaitre les plus petites fréquences et afficher plus de détail dans le domaine temporel le programme suivant va nous permettre de changer le temps jusqu'à 10 fois plus.

```
procedure TfrmMain.SweepEdtChange(Sender: TObject);
begin
  xinc := UpDown1.position;
  SetMaxPtstoavg;
  ShowScaleValue;
  SetOscState;
end;
```

Dans notre afficheur le domaine d'amplitude est très important à connaitre avec précision et de faire le moins d'erreur et pour cela on utilise un gain vertical, cette fonction va nous permettre de réaliser cette option.

```
procedure TfrmMain.ShowStored;
var
  myBeamA: array of TPoint;
  myBeamB: array of TPoint;
  Loop:integer;
  Gain :double;
  ofs:integer;
begin
  if singleframe then
  begin
```

```
  if btnDual.Down then
  begin
    SetLength(myBeamA,high(BeamA));
    SetLength(myBeamB,high(BeamA));
  end
  else
  begin
    SetLength(myBeamA,high(BeamA));
    SetLength(myBeamB,1);
  end;
  Gain := GetGain;
  StoredExpand := GetExpand;
  //Adjust Y ofset
  if Gain = 0.5 then
    ofs := Trunc(frmOscilloscope1.imgScreen.Height /4)
  else if Gain = 2 then
    ofs := Trunc(frmOscilloscope1.imgScreen.Height/4)*-2
  else
    ofs := 0;
  //ReCalc Beeam
  for Loop:=0 to high(BeamA)-1 do
  begin
    myBeamA[Loop].X := (BeamA[Loop].X +trStartPos.Position) * StoredExpand   ;
    myBeamA[Loop].Y := Trunc(BeamA[Loop].Y *Gain)-StoredCH1Offs +
trOfsCh1.Position+ofs;
    if btnDual.Down then
    begin
      myBeamB[Loop].X := (BeamB[Loop].X +trStartPos.Position) * StoredExpand;
      myBeamB[Loop].Y := Trunc(BeamB[Loop].Y*Gain)-StoredCH2Offs +
trOfsCh2.Position +ofs ;
    end;
```

```
    end;
    //Draw beam
    frmOscilloscope1.BeamData(myBeamA,myBeamB);
  end;
end;
```

A la fin d'une manipulation après avoir enregistré on doit quitter l'interface alors on va utiliser un programme simple car on va faire une procédure comme suit.

```
procedure TfrmMain.menuExitClick(Sender: TObject);
begin
  Close;
end;
```

Pour simplifier la manipulation on peut utiliser des menu et des boutons qu'on va installer et qui sont simples à programmer. Nous allons utiliser plusieurs codes couleurs pour bien visualiser en différents thèmes.