Projets spéciaux

ELE-791

Sonar ultrasonique à réseau

Présenté par:	Sylvain Fortin
Matricule :	FORS22057101
Professeur superviseur :	Jean Belzile
Date :	6 août 1998

École de Technologie Supérieure

Département de génie électrique

Sommaire

L'objectif est de concevoir un système qui permet au robot Hydromas III de détecter la présence d'obstacles dans son environnement. Le système devra mesurer la distance et l'angle qui sépare le robot des différents obstacles présents à l'intérieur d'un certain rayon d'action. Le robot utilisera cette information pour modifier sa trajectoire afin de contourner les obstacles.

Pour réaliser ce projet de télédétection nous utiliserons le principe de réflexion des ultrasons. Comme nous savons que le son voyage dans l'air à une certaine vitesse alors nous utiliserons le temps de propagation qui existe entre l'émission d'un ultrason et la réception de l'écho provenant d'un obstacle. Les chauves souris utilisent ce principe pour naviger sans collisions.

La méthodologie utilisé consiste à mettre en application les connaissances acquises en génie électrique et à les utiliser pour améliorer les performances du robot Hydromas III lors de compétitions inter-universitaires. Les connaissances au niveau des circuits électroniques seront mise à profit pour conçevoir les modules de traitements analogiques nécessaire pour la mise en forme des faibles signaux d'écho. Un bloc numérique implanté à l'aide d'un microprocesseur sera responsable de calculer la distance et la direction des obstacles à partir du temps de délais et de la phase receuillis par les récepteurs d'ultrasons .

Table des matières

Page de titre....1
Sommaire....2
Table des matières....3
Remerciements....4
Liste des tableaux....5
Liste des figures....5
Liste des symboles et des abréviations....6
Introduction....7
Les réalisations du club Walking Machine....7
Revue de la documentation....8
Mise en situation....8
Méthodologie et élaboration de la solution....8
Réseau à n récepteurs idéaux....8
Patron de sensibilité d'un récepteur piézo réel....11
Réseau à n récepteurs réels....13
Paramétrisation de la directivité d'un corrélateur....14
Calcul des paramètres de direction....16
Simulation du corrélateur discrêt....18
Redressement et détection d'enveloppe....19
Implémentation matérielle du sonar....21
Logiciel embarqué sur le sonar....22
Programme de terminal pour PC....22
Traitement numérique du signal d'écho....22
Discussion et interprétation des résultats....23
Problèmes rencontrés....23
Conclusion....24
Recommandations....24
Références....25
Annexes....26

Remerciements

Je tiens à remercier monsieur Jean Belzile qui a accepté de me paraîner pour ce projet spécial. Les différents entretient que j'ai eu avec lui ont permis de déterminer l'orientation du projet, le choix des composants et les stratégies algorithmique pour résoudre certains problèmes.

Je remercie également le club Walking Machine de m'avoir accepté au sein de leur équipe et de m'avoir fournis des récepteurs et de des emmeteurs piézo-électrique. Je remercie le club de m'avoir fournis l'idée de développer un sonar pour le robot Hydromas III. J'espère que mon travail pourra améliorer les performances de ce robot qui est déjà très performant étant donné les nombreux prix qu'il à remporté.

Merci au département de génie électrique d'avoir accepté le projet spéciale malgré les nombreuses modifications qu'il a fallut apporter à la description initiale du projet.

Liste des tableaux

Tableau 1 : Exposants pour le corrélateur à 5 éléments $f_s = 80$ KHz.

Liste des figures

Figure 1 : Réseau à n récepteurs
Figure 2 : Patron de sensibilité $b(\theta)$ pour un réseau idéal à 5 récepteurs
Figure 3 et 4 : Patron de sensibilié $b(\theta)$ d'un piézo à 30 cm et 1m.
Figure 5 : Approximation du patron de sensibilité d'un piezo ultrasonique
Figure 6 : Patron de sensibilité $b_{réseauRéel}(\theta)$ à 5 récepteurs
Figure 7 : Corrélateur paramétrisé en direction
Figure 8 : Simulation de la fonction de corrélation discrête
Figure 9 : Simulation de corrélation avec enveloppe $f_s = 85$ kHz

Liste des symboles et des abréviations

AD	Analog Device
ADC	Analog to Digital Converter
ASM	Assembler
cm	Centimètre
CPU	Central Processing Unit
Hz	Hertz
IRQ	Interrupt ReQuest
MHz	Mega Hertz
p-p	Peak to Peak
PC	Personal Computer
PIC	Micro-contrôleur fabriqué par la compagnie Microchip
RAM	Random Acces Memory
SAE	Society of Automotive Engineering
SRAM	Statical Random Acces Memory
US	Ultra-Son
V	Volts

Introduction

Les réalisations du club Walking Machine

Le club du Walking Machine de l'ÉTS existe déjà depuis 5 ans et l'expérience acquise durant ces années permet au club d'aller de l'avant et de faire de ses robots des produits de haute technologie qui impressionnent par leur qualité de fabrication. La première machine réalisée fut Atomas, robot entièrement électrique à 18 moteurs asservis. Afin d'élargir son champ d'application en dehors de la compétition SAE, le club à choisi d'investir ses efforts dans la technologie hydraulique en raison de ses nombreux avantages, dont son rapport poids/puissance. Ainsi, la deuxième génération de robots a été baptisée Hydromas. Hydromas I et II furent les premiers de cette lignée et marchèrent chacun dans une compétition. Le club travaille actuellement sur la conception d'Hydraumas III, un produit beaucoup plus raffiné. Les applications possibles de cette lignée de machines sont nombreuses: industrie forestière, exploration spatiale, véhicules pour handicapés, etc...

La SAE organise des compétitions annuelles de design, qui permettent à ses membres de concevoir et de réaliser des projets pour faire progresser la recherche et le développement des technologies reliées aux domaines des transports terrestres et à l'industrie aérospatiale. Le décathlon SAE Walking Machine est une compétition internationale de machines marchantes. Chaque année, une quinzaine d'équipes compétitionnent aux points de vue design, vitesse, autonomie et manoeuvrabilité, et ce depuis 1987. Les 10 épreuves proposées ont lieu sur un terrain de compétition composé de 11 cases de 9m2 chacune. Il s'agit essentiellement de contourner et de franchir des obstacles, de détecter et ramasser des objets. La dernière compétition a eu lieu au Mexique, à l'Université panaméricaine de Mexico.

Le second robot de la série, Hydromas II, reprend le concept original d'Hydromas tout en y apportant les changements nécessaires à son évolution : des pattes totalement redessinées, une nouvelle approche au niveau hydraulique et un nouveau programme dans un nouvel automate. Les efforts de l'équipe d'Hydromas II ont été récompensés par une bourse de 5000$ décernée par l'Association Canadienne des Technologies de Pointe (ACTP). Cette bourse a été remise le 3 juin 1997 lors d'une cérémonie que se déroulait dans le cadre du congrès Globenet 97.

En s'appuyant toujours sur la puissance de l'énergie hydraulique, l'équipe du club Walking Machine ÉTS repart en neuf pour le nouveau robot Hydromas III qui sera présenté à la compétition 1999 et dont le concept semble des plus prometteurs.

Revue de la documentation

Nous entrerendrons de définir clairement les principes de bases qui seront utilisés pour réaliser un sonar. Ensuite nous effecturons des simulations pour tester les équations qui servirons à traiter les échos ultrasoniques. L'architecture matérielle proposé pour acquérir et traiter les signaux sera présenté ainsi que les différentes couches de logiciel permettant l'exploitation du sonar. Une discussion sur les résultats obtenus nous amènera à conclure notre démarche qui vise à démontrer que notre sonar est utilisable.

Mise en situation

Nous désirons conçevoir et construire un système sonar qui permet de localiser différents obstacles à partir d'une onde ultrasonique. Pour accomplir ce projet nous devons choisir les principes qui seront utilisés. Nous ferons usage de réseau de récepteurs, de techniques d'échantillonage et de traitement numérique du signal. Des simulations seront faites à l'aide du logiciels MATLAB pour nous aider à faire les choix reliés à la conception et observer nos résultats. Ensuite nous passerons à l'implémentation matérielle du système et nous développerons les différentes couches de logiciels.

Réseau à n récepteurs idéaux

Lorsque l'on dispose un certain nombre de récepteurs linéairement espacés le long d'une droite, on peut observer des propriétés de directivités de réseau. Considérons un front d'onde sinusoïdale incident avec une pression unitaire possédant une fréquence f et un angle d'arrivé θ par raport à la normale d'une droite formée par n récepteurs. La longueur du réseau est l et chaque élément est espacé de la distance m. Toutes les tensions de sortie de réception sont envoyés sur un sommateur qui fournit le résultat V.

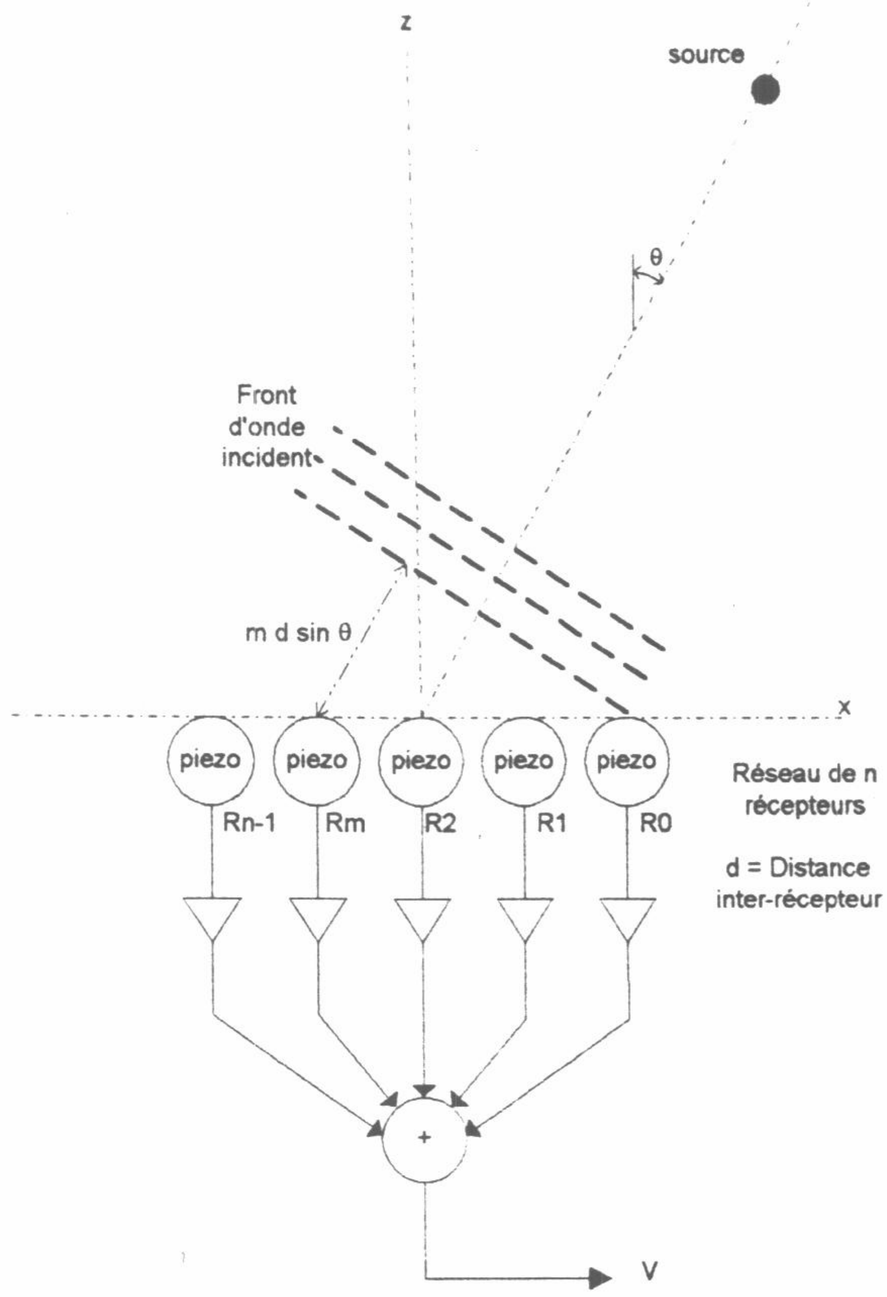


Figure 1 : Réseau à n récepteurs

En regardant la figure 1 nous pouvons réaliser que la puissance maximale du signal de sortie sera atteinte lorsque l'angle incident du front d'onde est $\beta=0$. Le patron de sensibilité d'un réseau linéaire régulièrement espacé et non déphasé peut être exprimé sous forme d'une équation. La sortie en tension de l'élément R*m* relatif à l'élément R0 sera retardé de la quantitée de temps nécessaire pour que le son voyage le long de la distance l_m = m d sin θ. Le délais de phase correspondant à un son de longueur d'onde λ pour une fréquence $\omega = 2\pi f$,

$$u_m = \frac{2\pi}{\lambda} l_m = mu \quad (1)$$

le délais de phase exprimé en radian entre deux éléments adjacents est,

$$u = \frac{2\pi d}{\lambda} \sin\theta$$

La tension de sortie du $m^{\text{ème}}$ élément qui possède un gain R*m* sera,

$$v_m = R_m \cos(\omega t + mu)$$

et le voltage du réseau sera la somme de chaque terme:

$$V = R_0 \cos\omega t + R_1 \cos(\omega t + u) + \ldots + R_m \cos(\omega t + mu) + \ldots + R_{n-1} \cos(\omega t + (n-1)u)$$

Nous pouvons écrire cette dernière équation en notation complexe,

$$V = (R_0 + R_1 e^{iu} + R_2 e^{2iu} + \ldots + R_{n-1} e^{(n-1)iu}) e^{i\omega t}$$

Si tous les éléments ont une réponse unitaire alors nous pouvons laisser tomber les R et écrire

$$V = (1 + e^{iu} + e^{2iu} + \ldots + e^{(n-1)iu}) e^{i\omega t}$$

En multipliant par e^{iu} et en soustrayant on trouve

$$V = \frac{e^{inu} - 1}{e^{iu} - 1} e^{i\omega t}$$

Après manipulation et le rejet du terme oscillant nous obtenons,

$$V = \frac{\sin(nu/2)}{\sin(u/2)}$$

Finalement on exprime u en terme de θ, on obtient le patron de sensibilité qui est le carré de cette fonction normalisée à l'unité à θ=0.

$$b_{réseau\,idéal}(\theta) = \left(\frac{V}{n}\right)^2 = \left[\frac{\sin\left(\frac{n\pi\, d \sin\theta}{\lambda}\right)}{n \sin\left(\frac{\pi\, d \sin\theta}{\lambda}\right)}\right]^2$$

Sur la figure 2 nous avons le graphique $b_{réseau\,idéal}(\theta)$ pour un réseau de piézo possédant les caractéristiques suivantes : n=5, d=1.77cm et en considérant une onde sonore se propagant dans l'air selon : c=340 m/s et f=40kHz. Ce graphique est obtenu en exécutant le programme `Patern.m` fournis en annexe.

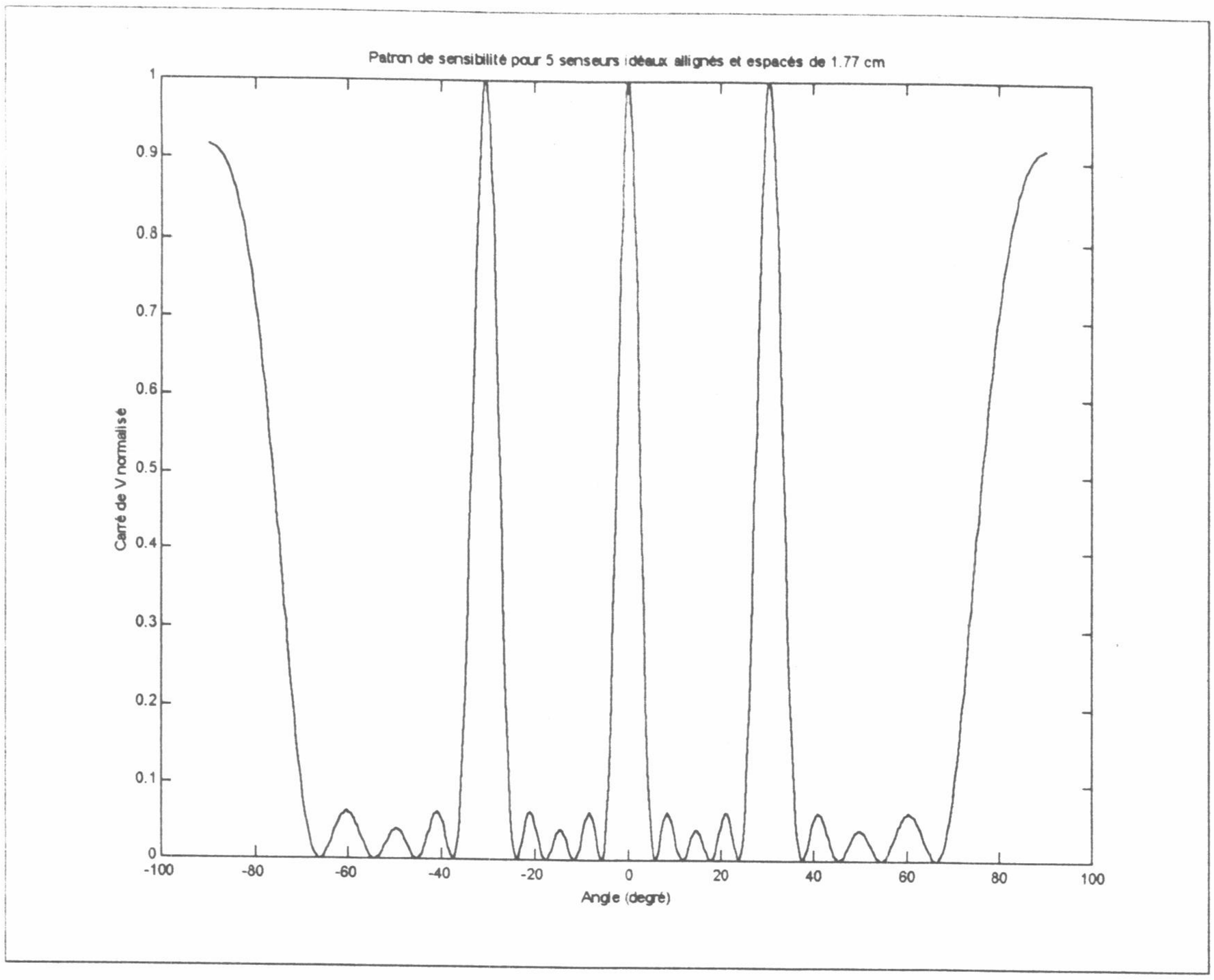


figure 2 : Patron de sensibilité $b(\theta)$ pour un réseau idéal à 5 récepteurs

L'observation du graphique de la figure 2 nous montre que le réseau possède une réponse maximale en amplitude à 0 degrés comme nous avions prévus. Toutefois nous avons deux autres régions de sensibilité tout aussis importantes à -32^0 et $+32^0$. Ces zones sont indésirables pour notre application car il rendent notre faisceau non unidirectionnel. Nous pourrions changer la position des zones de sensibilité indésirable en les déplacant vers les bords mais pour celà il faut soit diminuer la distance l entre les récepteurs ou diminuer la fréquence de l'onde sonore. Il existe sur le marché des récepteurs possédant différentes caractéristiques mais actuelement leur coût est trop élevé pour notre application. Nous sommes donc contraint par les dimensions actuelle de nos piezo ainsi que leur fréquence de résonance de 40 kHz non modifiable.

Patron de sensibilité d'un récepteur piézo réel

Nous disons du réseau précédent qu'il est idéal car nous avons considéré que chaque récepteur possède une réponse omnidirectionnelle. En réalité nos récepteurs ultrasonique sont loin d'être idéaux. L'amplitude des signaux électrique varie en fontion de l'angle incident que forme l'onde sonore par raport à

la normale du récepteur. Pour déterminer le patron de sensibilité du récepteur nous avons placé un récepteur sur une base fixe et fait déplacer une source tout en mesurant l'angle incidant. L'amplitude du signal source étant fixe nous mesurons pour chaque angle l'amplitude du signal électrique reçue tout en gardant constante la distance qui sépare l'émmeteur du récepteur. Les tensions sont ensuite converties en dB par rapport à la tension maximale mesuré à 0 degré. Les mesures expérimentales sont fournies sur le tableau intitulé mesure de la distribution spatiale du champ de pression acoustique d'un emmeteur piezo ultrasonique qui est fournie en annexe. Un programme MATLAB permet de représenter sur un diagramme polaire l'aspect de la courbe de sensibilité, le code source `Champ.m` est fournie en annexe. Voici sur les figures 3 et 4 nos patrons de sensibilités pour une distance de 30 cm et 1m. Sur ces figures le signal à 0 degré est normalisé à 0 dB.

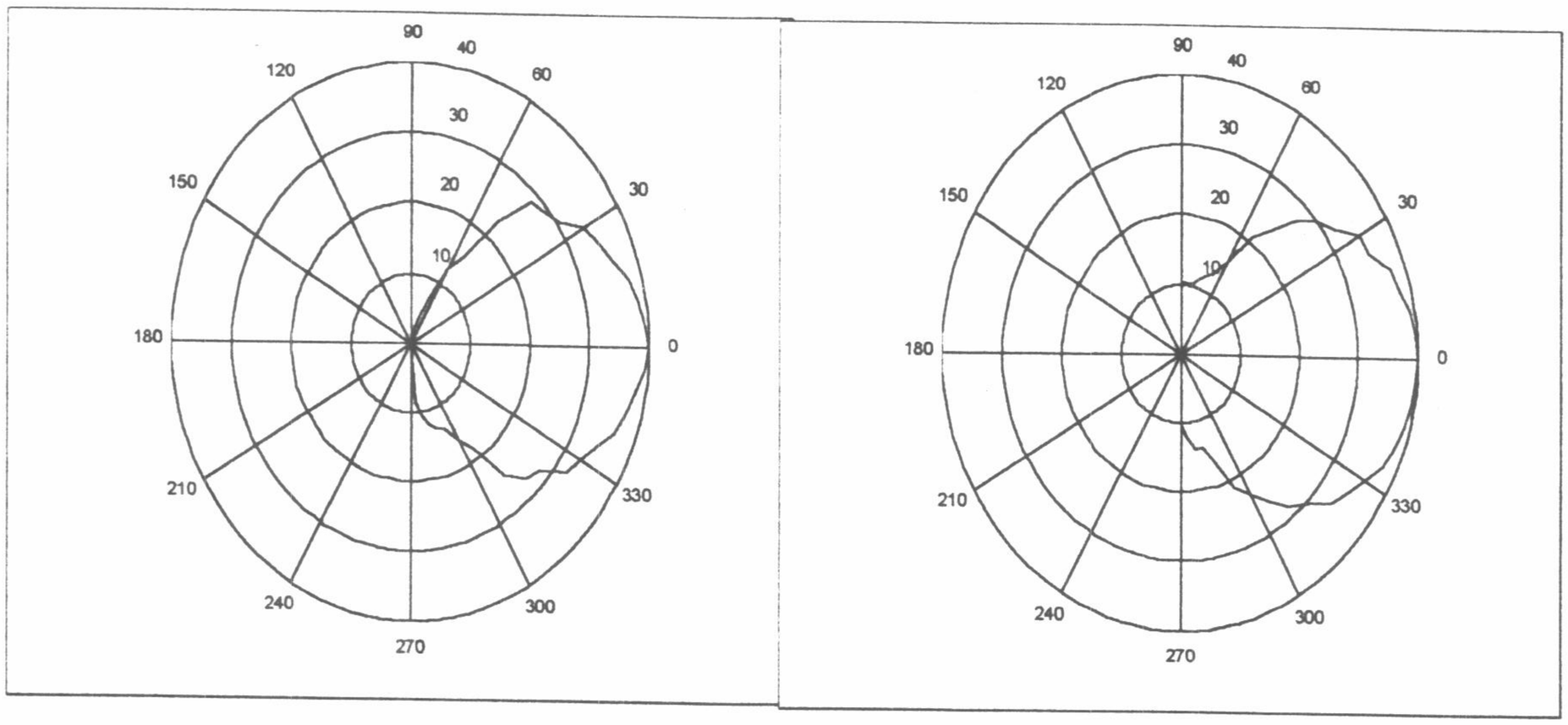


Figure 3 et 4 : Patron de sensibilié $b(\theta)$ d'un piézo à 30 cm et 1m.

Une approximation du patron de sensibilité peut être déterminée à partir d'une fonction analytique de la forme suivante,

$$b_{piézo}(\theta) = e^{-a\theta^2}.$$

Considérons que l'angle est exprimé en radian. Pour déterminer la valeur du coefficient a nous superposons sur un même graphique les données expérimentales et notre fonction analytique. Les données receuillies à 1 m sont choisies pour l'approximation. Le programme `aprox.m` inséré en annexe, est utilisé pour tracer notre fonction. Après quelques itérations sur a on obtient l'aproximation illustré sur la figure 5. Sur cette figure on voit que la valeur de a=0.5 founie une aproximation acceptable car l'erreur est assez faible.

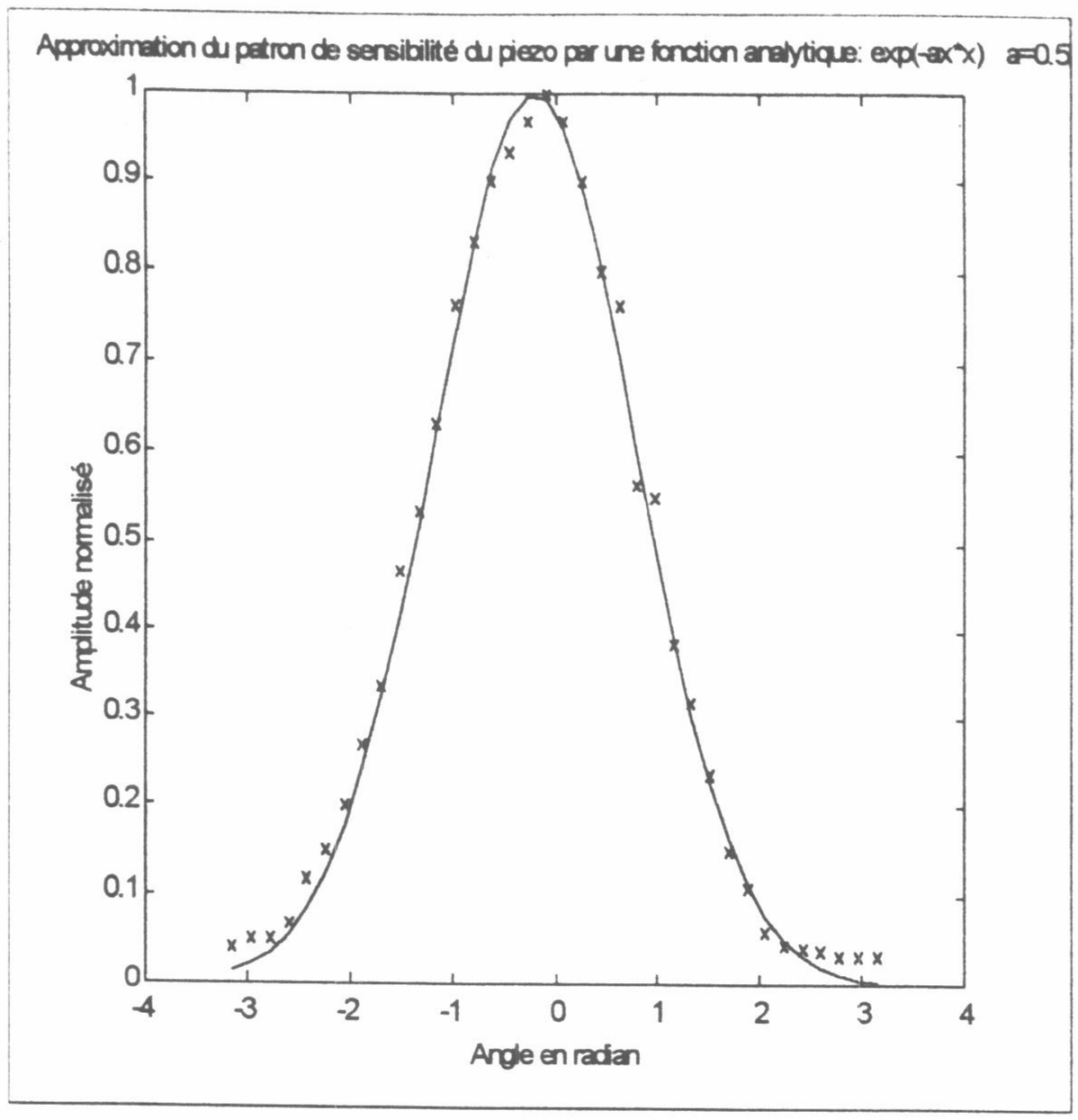


Figure 5 : Approximation du patron de sensibilité $b_{piézo}(\theta)$

Réseau à n récepteurs réels

Nous savons que chaque élément du réseau est composé d'éléments qui se sont pas omnidirectionnels.

Soit,

$b_{piezo}(\theta)$	:	Patron de sensibilité d'un piezo ultrasonique
$V_{réseau\ idéal}(\theta)$	:	Voltage en fontion de l'angle d'un réseau à n récepteurs idéaux, (signal incident unitaire)

Nous voulons déterminer,

$b_{réseau\ réel}(\theta)$	:	Patron de sensibilité d'un réseau à n récepteurs réel.

Alors,

$$b_{réseauRéel}(\theta) = \left(\frac{V_{réseauIdéal}(\theta) * b_{piézo}(\theta)}{n}\right)^2 = \left[\frac{\sin\left(\frac{n\pi\ d\sin\theta}{\lambda}\right)}{n\sin\left(\frac{\pi\ d\sin\theta}{\lambda}\right)} * e^{-0.5\theta^2}\right]^2$$

Sur la figure 6 nous avons le graphique $b_{réseauRéel}(\theta)$ pour un réseau de piézo possédant les caractéristiques suivantes : n=5, d=1.77cm et en considérant une onde sonore se propagant dans l'air selon : c=340 m/s et f=40kHz. Le graphique de la figure 6 est obtenus en exécutant le programme `Pat_res.m` fournis en annexe. On remarque que les lobes latéraux ont été atténués et que le réseau montre une directivité à 0 degré.

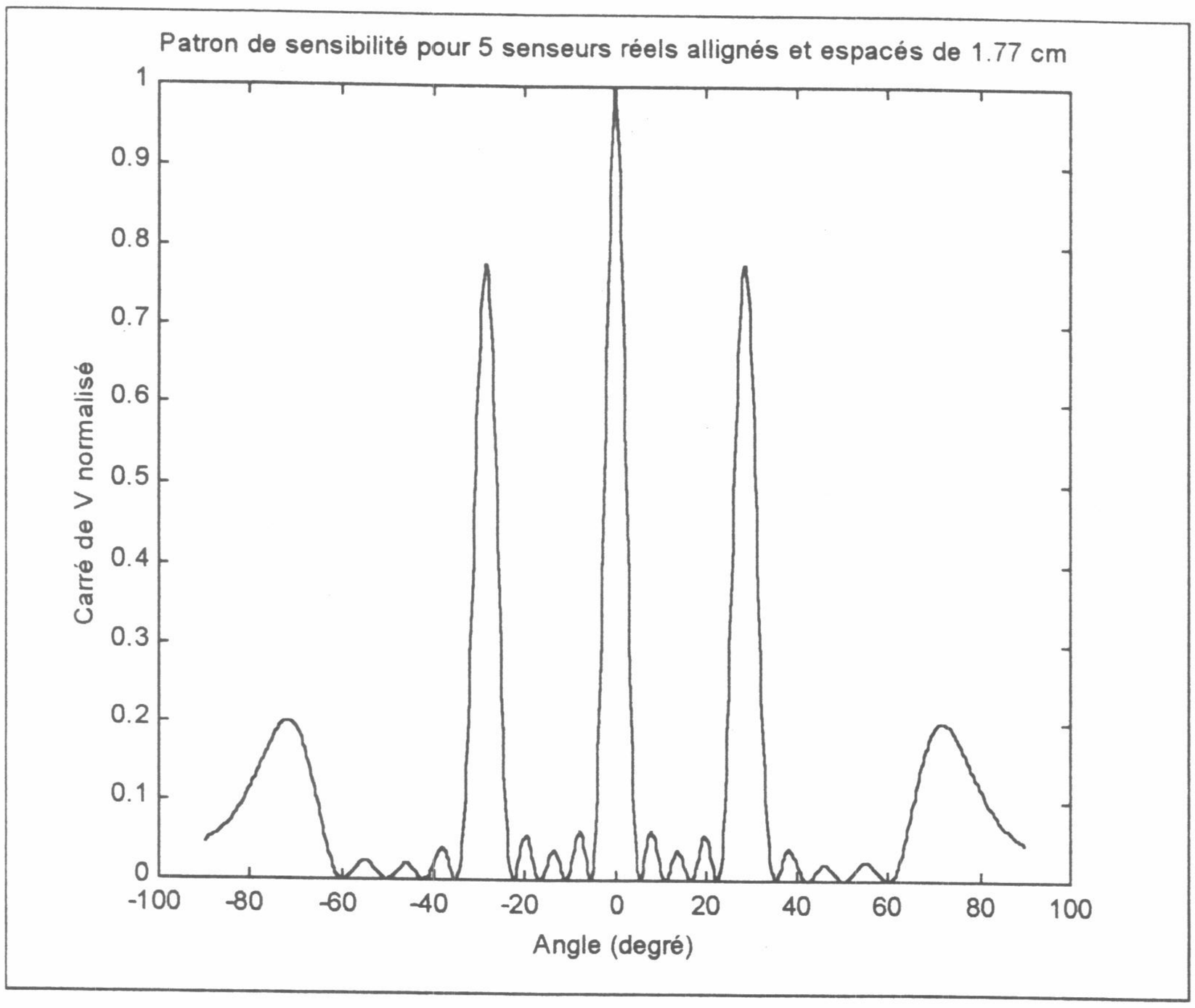


figure 6 : Patron de sensibilité $b_{réseauRéel}(\theta)$ à 5 récepteurs

Paramétrisation de la directivité d'un corrélateur

Pour réaliser notre sonar nous utiliserons un corrélateur paramétrisé en direction, ce corrélateur est illustré sur la figure 7. Nous entreprendrons d'implanter cette corrélation dans le domaine numérique pour générer un balayage angulaire. Un processeur de traitement numérique effectuera le traitement des signaux d'écho. Suite à l'émission d'une impulsion formée de quelques cycles à une fréquence de 40 kHz nous obtenons une succession d'écho provenant des différentes surfaces réfléchissante présente dans l'environnement sondé. Le processeur se charge de faire l'enregistrement numérique de ces échos. Le réseau de récepteurs est formé de 5 canaux chacun échantillonnés à une fréquence minimale de 80 kHz.

Considérons les 5 signaux suivants correspondant aux échos enregistrés simultanément par chacun des récepteurs ultrasonique.

$$f_0(n)\ ,\ f_1(n)\ ,\ f_2(n)\ ,\ f_3(n) \text{ et } f_4(n)$$

Nous appliquons le filtre numérique de direction qui retourne le signal s(n) corespondant à une mesure de corrélation entre les signaux.

$$s(n) = f_0(n)\, z^{-a} + f_1(n)\, z^{-b} + f_2(n)\, z^{-c} + f_3(n)\, z^{-d} + f_4(n)\, z^{-e}$$

En utilisant la notation de la transformé en z on représente des unités de retard dans le domaine numérique. Les valeurs des exposants a, b, c ,d et e doivent être linéairement espacés étant donné que le réseau est constitué de récepteurs régulièrement espacés et disposés le long d'une droite. Le signal s(n) résultant correspond à l'écho filtré pour une direction; cette direction forme un angle θ par rapport à la perpendiculaire au réseau. Si tous les exposants des z sont égaux alors le s(n) correspond au signal d'écho provenant d'un angle de 0 degrés.

L'équation du filtre est appelé successivement avec de nouveaux exposants a,b,c,d et e pour effectuer un balayage angulaire de la zone observée. Les signaux s(n) calculés pour chaque angle sont stockés en mémoire RAM dans le but de faire une reconstitution visuelle de l'environnement sondé.

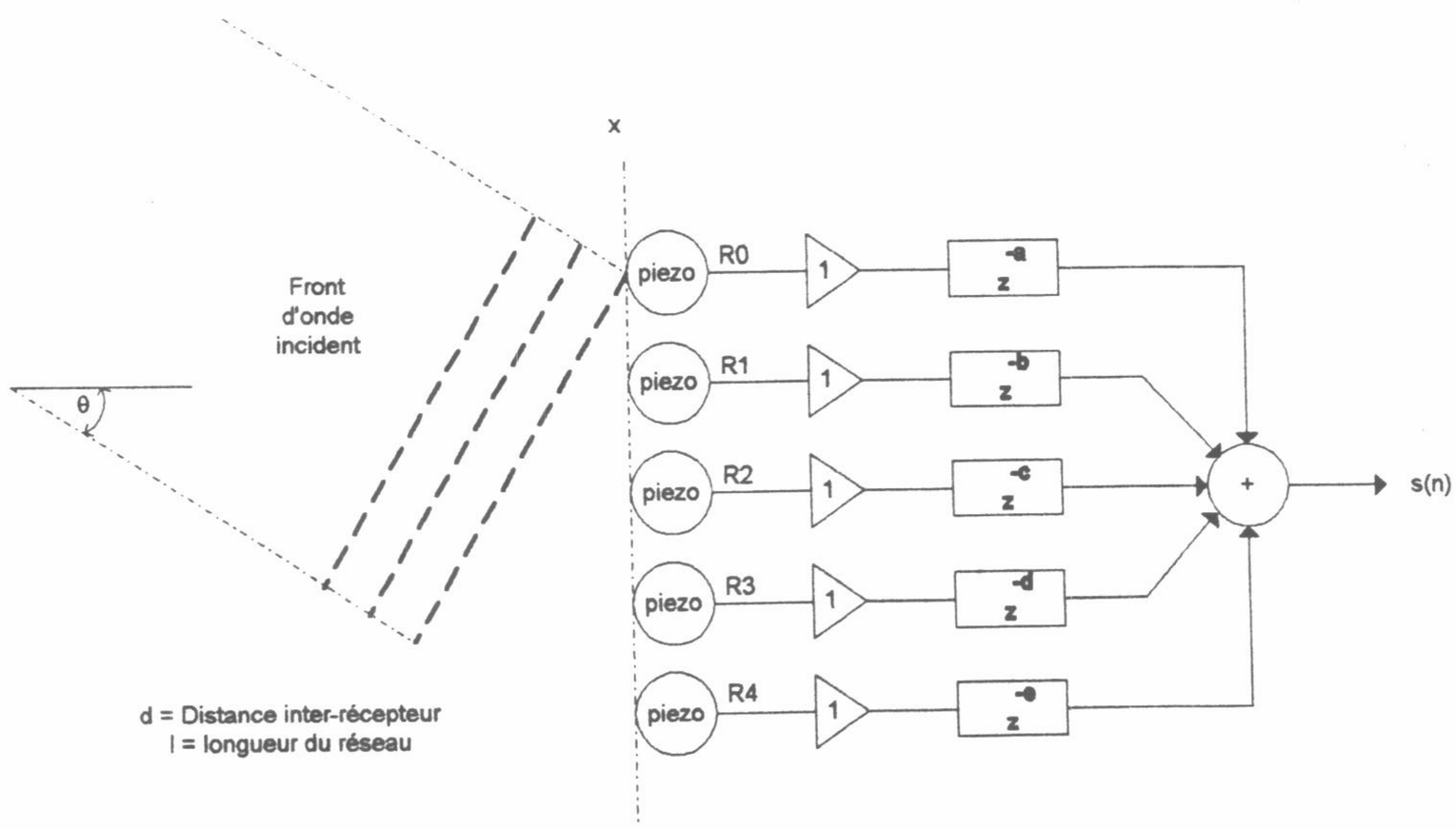


Figure 7 : Corrélateur paramétrisé en direction

Calcul des paramètres de direction

Nous voulons déterminer la relation qui nous donne les valeurs des exposants de Z en fonction de l'angle θ de direction. Considérons les variables suivante,

d : Distance entre chaque élément du réseau
ℓ : Longeur du réseau
c : Vitesse du son dans l'air
f : Fréquence du signal ultrasonique
λ : Longueur d'onde

Dans un premier temps nous déterminons combien de longueur d'onde sont comprises sur une distance correspondante à la longueur du réseau. Ensuite nous posons N_ℓ correspondant au temps exprimé en nombre d'échantillons pris pour que le son parcours la distance de la longueur du réseau.

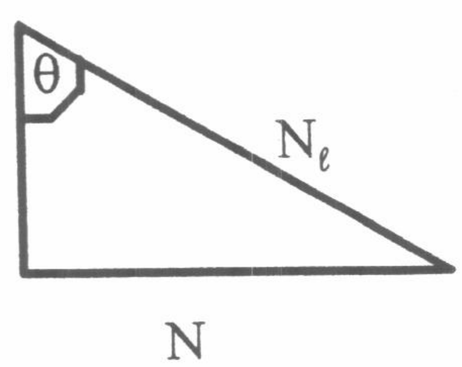
$$\frac{\ell}{\lambda} = \frac{\ell f}{c}$$

$$N_l = \frac{f_s \ell}{f \lambda}$$

$$\lambda = \frac{c}{f}$$

$$N_l = \frac{f_s \ell}{c}$$

Nous posons que N correspond au retard exprimé en échantillons qu'il faut partager entre les éléments du réseau pour obtenir une directivité maximale du réseau pour l'angle θ. Traçons le rectangle suivant,



$$\sin\theta = \frac{N}{N_\ell}$$

$$\theta = \arcsin\left(\frac{N c}{f_s \ell}\right)$$

$$N = \frac{f_s \ell \sin\theta}{c}$$

Soit i l'indice du récepteur considéré dans le réseau. i = [0 , n-1] , n étant le nombre de récepteurs dans le réseau, et θ l'angle de direction du réseau exprimé en degré. D (i , θ) représente le coefficient de retard pour l'élément i pour une directivité de θ. Cependant cette valeur de retard doit être arrondie à l'entier le plus près car nous désirons implémenter ce retard dans le domaine discrêt. Cet arrondissement peut causer des erreurs importantes sur la directions si la fréquence d'échantillonage est trop basse.

$$D(i,\theta) = i\, f_s\, \ell \, \frac{\sin\left(\pi \frac{\theta}{180}\right)}{4c}$$

Voici sur le tableau 1 les valeurs du calcul des exposants de Z pour chaque direction d'un réseau à cinq éléments.

Fréquence d'échantillonage (Hz): 80000
Longueur du réseau (m) : 0.071
Vitesse du son (m/s) : 340

Indice de l'exposant du récepteur	4	3	2	1	0
Angle (Degré)					
0	0.00	0.00	0.00	0.00	0.00
5	1.46	1.09	0.73	0.36	0.00
10	2.90	2.18	1.45	0.73	0.00
15	4.32	3.24	2.16	1.08	0.00
20	5.71	4.29	2.86	1.43	0.00
25	7.06	5.30	3.53	1.77	0.00
30	8.35	6.26	4.18	2.09	0.00
35	9.58	7.19	4.79	2.40	0.00
40	10.74	8.05	5.37	2.68	0.00
45	11.81	8.86	5.91	2.95	0.00
50	12.80	9.60	6.40	3.20	0.00
55	13.68	10.26	6.84	3.42	0.00
60	14.47	10.85	7.23	3.62	0.00
65	15.14	11.36	7.57	3.79	0.00
70	15.70	11.77	7.85	3.92	0.00
75	16.14	12.10	8.07	4.03	0.00
80	16.45	12.34	8.23	4.11	0.00
85	16.64	12.48	8.32	4.16	0.00
90	16.71	12.53	8.35	4.18	0.00

Valeurs arrondies

4	3	2	1	0
0	0	0	0	0
1	1	1	0	0
3	2	1	1	0
4	3	2	1	0
6	4	3	1	0
7	5	4	2	0
8	6	4	2	0
10	7	5	2	0
11	8	5	3	0
12	9	6	3	0
13	10	6	3	0
14	10	7	3	0
14	11	7	4	0
15	11	8	4	0
16	12	8	4	0
16	12	8	4	0
16	12	8	4	0
17	12	8	4	0
17	13	8	4	0

Tableau 1 : Exposants pour un corrélateur à 5 éléments f_s = 80 KHz.

Simulation du corrélateur discrêt

La simulation du corrélateur pour 5 signaux discrèts est représenté sur la figure 8. Nous effectuons cette simulation du réseau en générant cinq signaux sinusoïdaux dont la phase varie selon l'angle incident. Les délais sont simulés en variant la phase des signaux qui arrivent au réseau. Nous interpolons les délais ce qui nous donne l'équivalent de délais fractionnaires ainsi notre courbe de réponse est plus lisse. Nous sommes particulièrement intéressé à faire ressortir les images discrètes dupliqués par raport à la normale au réseau qui sont visibles pour des délais entre 7 et 8 ainsi que 15 retard d'échantillons pour f_s=75 KHz. Le programme de simulation est fournie dans l'annexe F sous le nom `Corel.m`. Sur ce graphique nous n'avons pas simulé l'atténuation causé par les récepteurs non-omnidirectionnels, on peut considérer que les images parasites seront atténuées comme illustrée sur la figure 6.

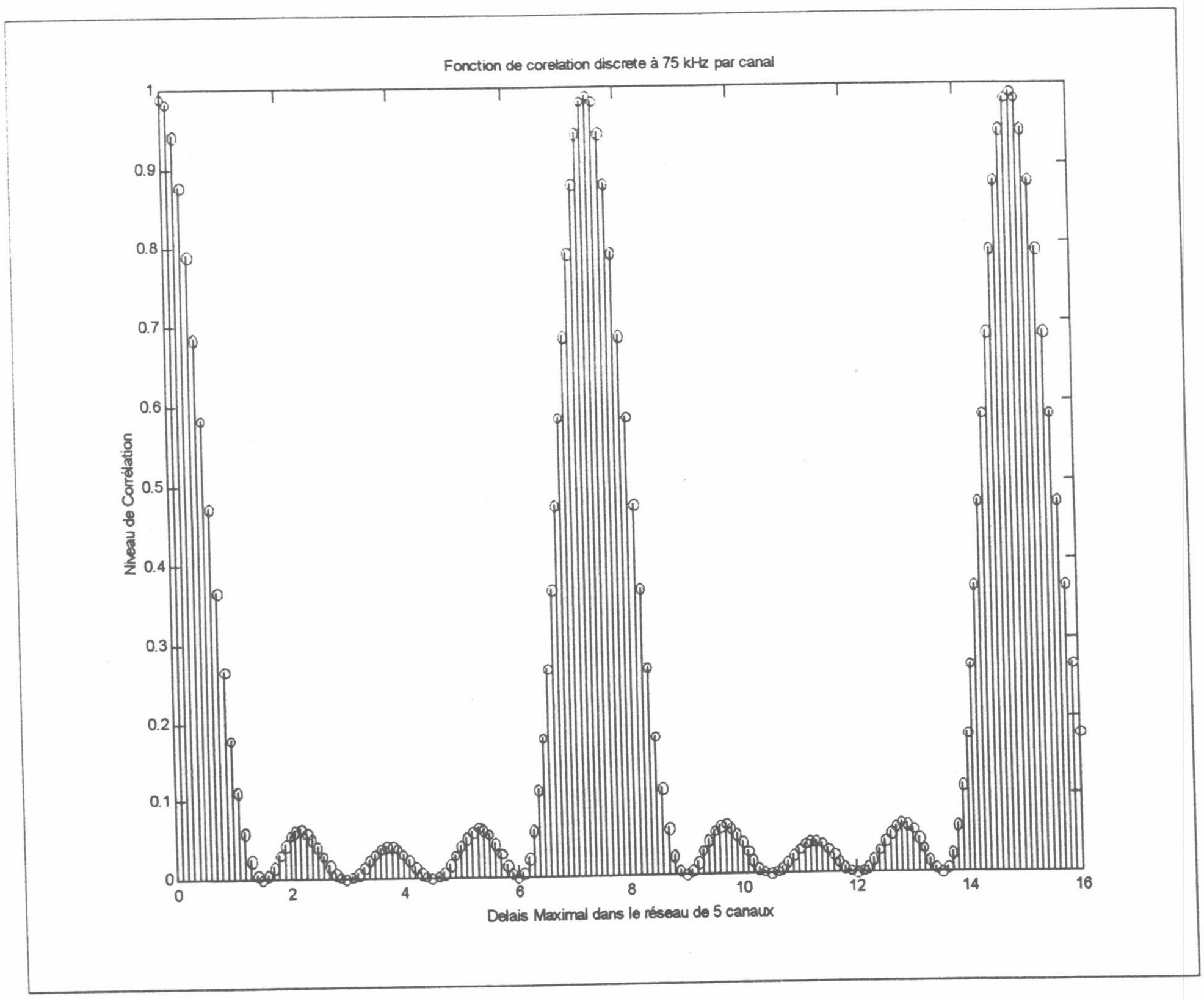


Figure 8 : Simulation de la fonction de corrélation discrète

Redressement et détection d'enveloppe

Le signal corrélé doit être traité pour extraire l'information de la distance et de l'angle d'arrivé du signal d'écho. Soit s(n) le signal de corrélation pour n récepteurs,

$$s(n) = \frac{1}{n}\sum_{i=0}^{n} \sin(\varpi + \theta_n)$$

Cette somme de sinus peut être redressé en utilisant la fontion ABS qui fait la valeur absolue ce qui est l'équivalent d'un resresseur à diode idéal. En réalité nous élevons au carré pour obtenir le signal de puissance $s_p(n)$, ainsi nous obtenons également l'effet de la valeur absolue.

$$s_p(n) = s^2(n) = \frac{1}{n^2}\sum_{i=0}^{n} \sin^2(\varpi + \theta_n)$$

Le signal $s_p(n)$ présente l'apparence de bosses à alternance positive. On peut réaliser le détecteur d'enveloppe en utilisant le principe d'un circuit R-C réalisé à l'aide de l'algorithme itératif suivant,

```
% Détecteur d'enveloppe
%
% Entree : cor(dir,i)
% Signal de correlation en fonction de l'angle et du temps i (échantillon)
%
% Sortie : env(dir,i)
% Signal du détecteur d'enveloppe en fonction de la direction et du temps i

for dir=1:length(angles);
  for i=2:length(f0),
    env(dir,1)=0;                                      % Départ de l'enveloppe à 0
    if cor(dir,i)>env(dir,i-1)
      env(dir,i)=0.6*env(dir,i-1)+0.4*cor(dir,i); % Charge le condensateur
    else
      env(dir,i)=0.998*env(dir,i-1);               % Décharge le condensateur
    end;
  end;
end;
```

Les coefficient choisis permettent de charger rapidement le condensateur lorsque le signal d'entrée est plus grand que le seuil actuel. Le temps de décharge est plus lent que le temps de charge, ceci permet d'atténuer fortement les ondulations. Les coefficients sont déterminés expérimentalement pour s'approcher de la résolution optimale.

Les résultats de simulation du corrélateur avec détecteur d'enveloppe pour une fréquence de 85 KHz sont donnés à la figure 9. Le code est founis en annexe G. Le tracé tridimensionnel est construit à partir de l'indice de l'échantillon courrant sur l'axe des x, qui peut être considéré comme le temps. L'angle de directivité du réseau exprimé en degré est fournis sur l'axe y et l'intensité de corrélation est sur l'axe z. En premier lieu, le lecteur doit savoir que pour cette expérience le signal source est sinusoïdale et il est appliqué en phase égale sur les cinq entrées. Nous voyons le travail effectué par le détecteur d'enveloppe qui tend à lisser les courbes. Nous constatons que le réseau possède toujours une sensibilité maximale à 0 degré avec des bandes latérales images. Pour clarifier l'affichage des résultats, l'algorithme du filtre ne commence à calculer qu'à partir du nombre d'échantillons suffisant pour lancer la corrélation, on observe ceci par une zone morte pour les premiers échantillons. La simulation nous montre des bandes très nettes avec des zones de silence (intensité quasi nulle) bien définies pour les angles intermédiaires. Les résultats que nous obtenons nous encouragent à continuer notre implémentation du sonar.

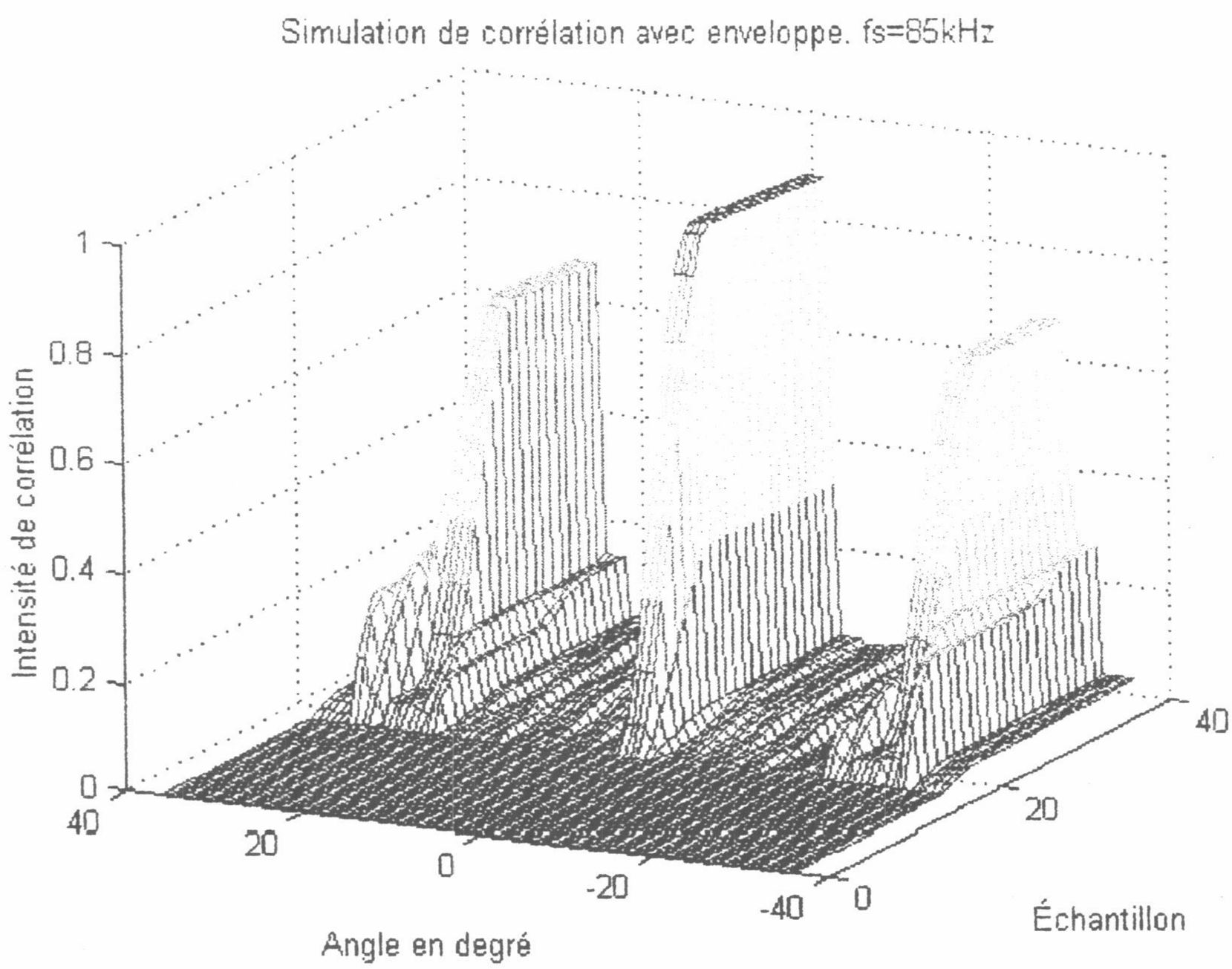


Figure 9 : Simulation de corrélation avec enveloppe $f_s = 85$ kHz

Implémentation matérielle du sonar

Le système d'acquisition d'écho est illustré à la figure 10. Notre architecture est composé d'un convertisseur analogue à digial AD7891-1 qui permet théoriquement d'atteindre une fréquence d'échantillonage de 454.5 kHz car son temps d'acquisition est de 600 ns et le temps de conversion est de 1.6 us. Nous disposons de 8 canaux mais nous n'en utilisons que 5, la fréquence maximale d'échantillonage par canal est par conséquent de 90.9 kHz. Le processeur utilisé est le PIC17C44 de la compagnie Microchip et ce processeur est piloté par un oscillateur de 32 MHz. Étant donné que pour ce microcontroleur chaque instruction demande 4 cycle d'horloge alors chaque instruction prends 125 ns à s'exécuter. Nous disposons de 8K de EPROM sur le microcontrôleur pour le programme machine embarqué sur le sonar. Deux mémoire SRAM de 32K*8 bits sont utilisés mais seulement 12 bits sont reliés, la mémoire RAM est disposé de manière parallèle pour former 32K*12 bits. La communication se fait par un lien sériel RS232, le coupleur MAX232 permet l'interface facile avec un PC.

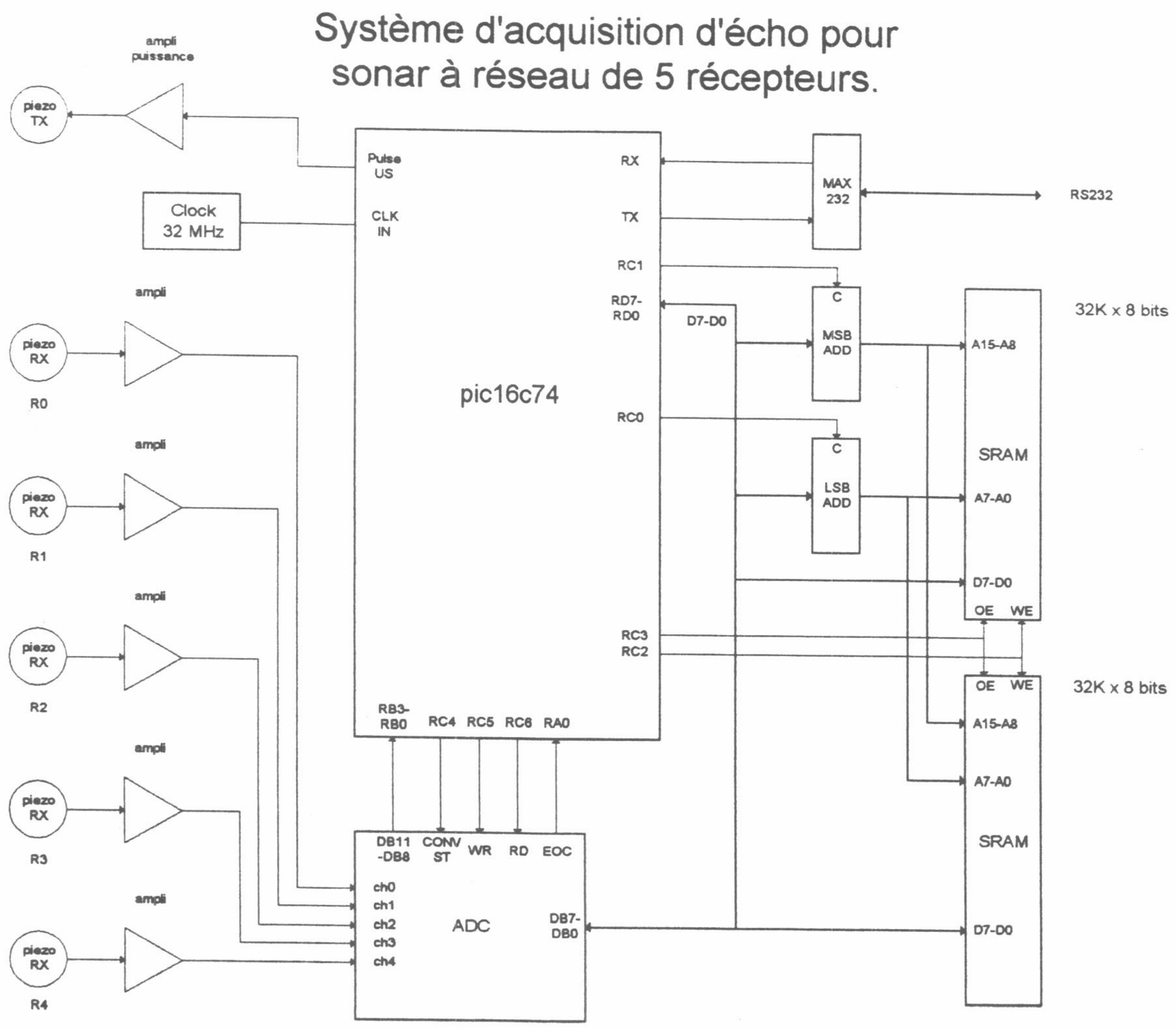


Figure 10: Schémas bloc du sonar

Logiciel embarqué sur le sonar

Le logiciel embarqué de gestion du convertisseur AD, de la mémoire et de l'interface sériel est développé en assembleur pour optimiser la vitesse d'exécution. Ce logiciel propose un menu qui permet de lancer l'acquisition des échantillons et également il permet d'envoyer ces échantillons au PC à l'aide de l'interface sérielle RS232. Le code source du logiciel SONAR.ASM est fournie en annexe. Le logiciel est particulièrement optimisé dans la boucle d'acquisition des échantillons pour rouler le plus rapidement possible tout en conservant une précision acceptable sur les mesures.

Programme de terminal pour PC

Ce programme est essentiellement un terminal utilisant le port sériel d'un PC. Le code source du terminal pour le sonar est fournis en annexe sous le nom MONSONAR.C Ce terminal permet de recevoir les échantillons du système d'acquisition d'écho du sonar et de les enregistrer sur le disque dur. Ce fichier de données sera utilisé par un programme écrit en MATLAB qui réalise le traitement numérique des signaux. Le protocole de communication utilisé est très simple, tout caractère ASCII est affiché lorsque le terminal est dans son mode normal. La réception du caractère '%' signale l'arrivée des échantillons et le terminal crée un fichier dans lequel les échantillons seront écrits à mesure qu'ils sont reçus. Le caractère '&' est réservé pour indiquer la fin de la réception d'un groupe d'échantillons, ainsi le le programme termine l'écriture du fichier et le terminal reviens à son mode normal. Pour réaliser le terminal nous avons utilisé la technique des interruptions, chaque caractère reçu sur le port sériel génère une interruption.

Traitement numérique du signal d'écho

Nous présentons ici l'implémentation du corelateur pour les 5 signaux discrèts echantillonés provenant du systeme d'acquisition d'écho. Nous avons vu comment la corelation s'effectue pour différent angles, l'équation du filtre directionel utilise des retard et une somation. Le signal de puissance est envoyé dans un détecteur d'enveloppe. Le fichier TABLE.SMP comprend tous les échantillons que le terminal MONSONAR a recu. Le fichier est formé par une alternance des échantillons pour chaque canal. Chaque échantillon est exprimé sous un mot de 16 bits signés. Les bytes les moins significatifs arrivent avant les bytes les plus sigificatifs. L'alternance des canaux est comme suit: 0,1,2,3,4,0,1,2,3,4,0,1... jusqu'à la fin du fichier. Le nombre d'échantillons total dans le fichier peut varier et le programme SIG.M ne sera pas affecté. Le programme SIG2.M est

responsable d'implémenter tout le traitement numérique et l'affichage sous forme de graphique des résultats.

Nous avons réalisé l'expérience suivante qui consiste à relier un générateur sinusoïdale sur nos 5 canaux, toutes les entrées sont alimentées en phase. À l'aide du menu sur le terminal nous lançons l'acquisition des échantillons et nous démarrons le programme SIG2.M qui effectue le traitement. Le graphique du résultat est présenté à l'annexe I.

Discussion et interprétation des résultats

Le graphique de l'annexe I est très révélateur sur les performances de notre système d'acquisition et de traitement des signaux d'écho. Nous voyons la raie principale centrée à 0 degré indiquant la présence de signaux en phase sur l'axe perpendiculaire. Nous pouvons constater la présence de zones de silences tout autour de la raie centrale. Les raies image sont présente comme prévue dans nos calculs et nos simulations. On note cependant des amplitudes différentes pour les raies images. Ceci est difficile à expliquer et il faudrait investir plus de temps pour analyser le problème.

Problèmes rencontrés

Nous avons réussis à obtenir la fréquence d'échantillonage minimale de Nyquist mais ce fût au coût dramatique d'une perte importante de précision des échantillons. Pour augmenter la vitesse d'acquisition nous retranchons du temps du temps minimale pour effectuer la conversion complète. Normalement il faut attendre 1.6 us pour obtenir une valeur sur 12 bits mais comme le ADC utilise une technique d'aproxomation successive les bits MSB sont disponibles très tôts sur le bus de données. Alors nous n'attendons pas toute cette durée et prenons la valeur sur le bus ce qui nous fait perdre de la précision. Sans ce sacrifice nous ne pouvons pas obtenir une fréquence d'échantillonage supérieure à 80 kHz par canal. Cette condition est nécessaire pour que le traitement numérique soit valable.

Conclusion

L'architecture proposé d'un sonar ultrasonique à réseau est assez complexe à implanter. La rapidité d'échantillonage combiné à un processeur rapide et un nombre de récepteurs suffisant, sont des éléments critiques pour obtenir des performances acceptables. Pour obtenir des résolutions angulaires plus raffinées, il faut augmenter le nombre de senseurs composant le réseau pour diminuer la largeur du faisceau de balayage. Les données expérimentales nous ont montrés que le procédé que nous avons appliqué pour notre sonar est fonctionnel et qu'il est utilisable. Le procédé que nous avons utilisé est déja utilisé pour les systèmes d'échographie prénatale ainsi que les sonar militaire utilisés pour la défense.

Recommandations

Un sonar à réseau de 5 récepteurs est une application qui convient à un robot qui peut utiliser l'information pour obtenir une vue grossière de l'environnement sondé, et ainsi déterminer la trajectoire à emprunter. Pour une application en temps réel et autonôme, il faut transposer les différents algorithmes de traitement vers le processeur embarqué, ceci demande un travail supplémentaire à effectuer.

Références

1. Robert J.Urick: Principles of Underwater Sound, 3d edition, McGraw-Hill, 1983.

2. Glen Wade: Acoustic Imaging, Plenum Press, 1976.

3. Force Armées Canadiennes: Receiver Group Sonar, OA—116/SQS-505, 1984

4. Forces Armées Canadiennes: AN/SQS-505 Theory of operation and funtional flow diagram, CBCN 5828, 1984.

5. Forces Armées Canadiennes: AN/SQS-505 Theory of operation and funtional flow for the I.T.E., CBCN 5828(2), 1984.

6. Forces Armées Canadiennes: AN/SQS-505 Shematics., CBCN 5828(3), 1984.

ANNEXES

A- Programme MATLAB de détermination du patron de sensibilité pour un réseau de n senseurs idéaux.

B- Table des mesures expérimentales de la distribution spatiale du champ de pression acoustique d'un emmeteur piezo ultrasonique.

C- Programme MATLAB du traçage du diagramme polaire de réponse en amplitude en fontion de l'angle d'observation d'un piézo réel.

D- Programme MATLAB d'approximation de la réponse du piezo à partir d'une relation analytique.

E- Programme MATLAB de détermination du patrons de sensibilité pour n senseurs réels espacés d'une distance d.

F- Programme MATLAB de simulation du corrélateur pour 5 signaux discrêts.

G- Programme MATLAB de simulation de corrélation avec détecteur d'enveloppe.

H- Programme MATLAB de traitement numérique des signaux du sonar.

I- Graphique de corrélation des 5 signaux. fs=85.1 kHz

J- Programme source MONSONAR.C

K- Programme source SONAR.ASM

L- Shémas électronique du système d'acquisition d'écho du sonar

ANNEXE A

```
% ***********************************************************
% FICHIER : Patern.m
%
% Auteur  : Sylvain Fortin
% Date    : 23 Juin 1998
%
% DESCRIPTION :
% Determine les patrons de sensibilité pour n senseurs
% idéaux et espacés de 1.77 cm.
% Nous voulons obtenir la réponse en amplitude en fonction
% de l'angle incident au réseaux linéaire de piezo
% recepteurs d'ultrasons.
%

% Vitesse du son en m/s
c=360;
% Frequence de l'ultrason
f=40000;
% Nombre de piezo mis en ligne
n=5;
% Distance entre chaque piezo (mètres)
d=0.0177;

% Nombre de points pour tracer graphique
NbPoints=600;

% Calcul de la longueur d'onde
lambda=c/f;

% Balayage de teta entre 0 -pi/2 et pi/2
teta=linspace(-pi/2,pi/2,NbPoints);
% Calcul de l'amplitude en fonction de l'angle
for i=1:NbPoints,

V(i)=(sin(n*pi*d*sin(teta(i))/lambda)/(n*sin(pi*d*sin(teta(i))/lambda)))
;
  V(i)=V(i)*V(i);
  i=i+1;
end;
degre=180*teta/(pi);
plot(degre,V);
TITLE('Patron de sensibilité pour 5 senseurs idéaux allignés et espacés
de 1.77 cm')
YLABEL('Carré de V normalisé');
XLABEL('Angle (degré)');
```

ANNEXE B

Mesure de la distribution spatiale du champ de pression acoustique d'un emmeteur piezo ultrasonique

Voltage appliqué sur l'emmetteur :	10 V p-p
Fréquence appliquée sur le piézo:	40 kHz

Angle	Distance 1	Atténuation	Distance 2	Atténuation
	30 cm.		1 m.	
	Voltage crête		Voltage crête	
degré	(Volts)	(dB)	(Volts)	(dB)
-90	0.02	-39.55	0.002	-29.54
-85	0.05	-31.60	0.0024	-27.96
-80	0.065	-29.32	0.003	-26.02
-75	0.075	-28.07	0.003	-26.02
-70	0.085	-26.99	0.004	-23.52
-65	0.09	-26.49	0.007	-18.66
-60	0.12	-23.99	0.009	-16.48
-55	0.16	-21.49	0.012	-13.98
-50	0.32	-15.47	0.016	-11.48
-45	0.44	-12.71	0.02	-9.54
-40	0.5	-11.60	0.028	-6.62
-35	0.75	-8.07	0.032	-5.46
-30	0.85	-6.99	0.038	-3.97
-25	1	-5.58	0.046	-2.31
-20	1.25	-3.64	0.05	-1.58
-15	1.4	-2.65	0.054	-0.92
-10	1.55	-1.77	0.056	-0.60
-5	1.8	-0.47	0.058	-0.29
0	1.9	0.00	0.06	0.00
5	1.8	-0.47	0.058	-0.29
10	1.65	-1.23	0.054	-0.92
15	1.45	-2.35	0.048	-1.94
20	1.2	-3.99	0.046	-2.31
25	1.05	-5.15	0.034	-4.93
30	0.9	-6.49	0.033	-5.19
35	0.64	-9.45	0.023	-8.33
40	0.56	-10.61	0.019	-9.99
45	0.52	-11.26	0.014	-12.64
50	0.26	-17.28	0.009	-16.48
55	0.11	-24.75	0.0065	-19.30
60	0.08	-27.51	0.0036	-24.44
65	0.045	-32.51	0.0026	-27.26
70	0.03	-36.03	0.0024	-27.96
75	0.024	-37.97	0.0022	-28.71
80	0.02	-39.55	0.002	-29.54
85	0.02	-39.55	0.002	-29.54
90	0.02	-39.55	0.002	-29.54

ANNEXE C

```
% ***********************************************************
% FICHIER : Champ.m
%
% Auteur  : Sylvain Fortin
% Date    : 23 Juin 1998
%
% DESCRIPTION :
% A partir des donnes de mesure de la distribution spatiale du
% champs de pression acoustique d'un emmeteur piezo ultrasonique,
% nous affichons le diagramme polaire de réponse en amplitude en
% fontion de l'angle d'observation.
% La première étape consite à normaliser les données pour obtenir
% une amplitude maximale de 1.

% Nombre de points de données pour tracer le diagramme polaire
NbPoints=37;

% Donnes expérimentales receuillis à partir d'un oscillateur,
oscilloscope et piezo.
% D1 est la tension lue à une distance de 30 cm.
D1=[ 0.02 0.05 0.065 0.075 0.085 0.09 0.12 0.16 ...
0.32 0.44 0.5 0.75 0.85 1 1.25 1.4 1.55 ...
1.8 1.9 1.8 1.65 1.45 1.2 1.05 0.9 ...
0.64 0.56 0.52 0.26 0.11 0.08 0.045 0.03 ...
0.024 0.02 0.02 0.02 ];

% D2 est la tension lue à ine distance de 1 m.
D2=[ 0.002 0.0024 0.003 0.003 0.004 0.007 0.009 0.012 ...
0.016 0.02 0.028 0.032 0.038 0.046 0.05 0.054 0.056 0.058 ...
0.06 0.058 0.054 0.048 0.046 0.034 0.033 0.023 0.019 0.014 ...
0.009 0.0065 0.0036 0.0026 0.0024 0.0022 0.002 0.002 0.002 ];

% Balayage de teta entre -pi/2 et pi/2
teta=linspace(-pi/2,pi/2,NbPoints);

% Normalisation de l'amplitude des données
% On veut que la valeur maximale soit de 1.
Max_D1=max(D1);
D1_Norm=D1/Max_D1;

Max_D2=max(D2);
D2_Norm=D2/Max_D2;

% Calcul du gain en db pour tracer le diagramme polaire
D1_db=20*log10(D1_Norm)+40;
D2_db=20*log10(D2_Norm)+40;

polar(teta,D2_db);

%TITLE('Distribution spatiale du champs de pression acoustique pour
emmeteur piezo ultrasonique d=1m')
```

ANNEXE D

```
% ***********************************************************
% FICHIER : aprox.m
%
% Auteur  : Sylvain Fortin
% Date    : 24 Juin 1998
%
% DESCRIPTION :
%Approximation de la réponse du piezo à partir d'une relation analytique
% qui se raproche le plus possible de la forme du patron
% observé à l'aide des données expérimentales.
% F=exp(-ax*x)

clear;

% Nombre de points de données recuillis experimentalement
NbPointsExp=37;
% Donnes expérimentales receuillis à partir d'un oscillateur,
oscilloscope et piezo.
% Dexp est la matrice des tension lue à une distance de 1 m.
Dexp=[ 0.002 0.0024 0.003 0.003 0.004 0.007 0.009 0.012 ...
0.016 0.02 0.028 0.032 0.038 0.046 0.05 0.054 0.056 0.058 ...
0.06 0.058 0.054 0.048 0.046 0.034 0.033 0.023 0.019 0.014 ...
0.009 0.0065 0.0036 0.0026 0.0024 0.0022 0.002 0.002 0.002 ];
Dexp=[Dexp,0];

% Normalisation de l'amplitude des données
% On veut que la valeur maximale soit de 1.
Max_D=max(Dexp);
D=Dexp/Max_D;

% En extrapolant nous engendrons 16 fois le nombre de points
experimentales
n=1;
NbPointsMult=n*NbPointsExp;
% Balayage de teta entre -pi et pi
teta=linspace(-pi,pi,NbPointsMult-1);
for i=1:NbPointsMult-1,
  pente=(D(floor(i/n)+2)-D(floor(i/n)+1))/n;
  G(i)=D(floor(i/n)+1)+(i-n*floor(i/n))*pente;
end

% Fonction qui aproxime la suite de donné mesurés par oscillo et piezo.
a=0.5;
t=0.2;        % Le piezo n'était pas centré à un angle de 0 degré !
f=exp(-a*((teta+t).*(teta+t)));
plot(teta,G,'X',teta,f,'g');
Title('Approximation du patron de sensibilité du piezo par une fonction
analytique: exp(-ax*x)   a=0.5');
YLABEL('Amplitude normalisé');
XLABEL('Angle en radian');
```

ANNEXE E

```
% ***********************************************************
% FICHIER : Pat_res.m
%
% Auteur  : Sylvain Fortin
% Date    : 24 Juin 1998
%
% DESCRIPTION :
% Determine les patrons de sensibilité pour n senseurs
% reels et espacés d'une distance d.
% Produit de la fonction d'aproximation du champ de sensibilité du piezo
% avec la reponse en amplitude du reseau a n piezo
% Nous voulons obtenir la réponse en amplitude en fonction
% de l'angle incident au réseaux linéaire de piezo
% recepteurs d'ultrasons.

% Vitesse du son en m/s
c=340;
% Frequence de l'ultrason
f=40000;
% Nombre de piezo mis en ligne
n=5;
% Distance entre chaque piezo (mètres)
d=0.0177;

% Nombre de points pour tracer graphique
NbPoints=600;

% Calcul de la longueur d'onde
lambda=c/f;

% Balayage de teta entre 0 -pi/2 et pi/2
teta=linspace(-pi/2,pi/2,NbPoints);

% Aproximation de la reponse en amplitude d'un piezo en fonction de
l'angle teta.
a=0.5;
ARAP=exp(-a*((teta).*(teta)));

% Calcul de l'amplitude en fonction de l'angle V(i) voltage lue par
reseau ideal
for i=1:NbPoints,

V(i)=(sin(n*pi*d*sin(teta(i))/lambda)/(n*sin(pi*d*sin(teta(i))/lambda)))
;
% Produit de la fonction d'aproximation du champ de sensibilité du piezo
% avec la reponse en amplitude du reseau a n piezo
  AmpResReel(i)=V(i)*ARAP(i);
  PowResReel(i)=AmpResReel(i)*AmpResReel(i);
  i=i+1;
end;
degre=180*teta/(pi);
plot(degre,PowResReel);
TITLE('Patron de sensibilité pour 5 senseurs réels allignés et espacés
de 1.77 cm')
YLABEL('Carré de V normalisé');
XLABEL('Angle (degré)');
```

ANNEXE F

```
% ***********************************************************
% FICHIER : Corel.m
%
% Auteur  : Sylvain Fortin
% Date    : 16 juillet 1998
%
% DESCRIPTION :
% Simulation du corelateur pour 5 signaux discrêts.
% Nous effectuons cette simulation du réseau en générant
% cinq signaux sinusoïdaux dont la phase varie selon l'angle
% incident. Les délais sont simulés en variant la phase des
% signaux arrivant au réseau. Nous interpolons les délais
% ce qui nous donne l'équivalent de délais fractionnaires
% ainsi notre courbe de réponse est plus lisse. Nous sommes
% particulièrement intéressé à faire ressortie les images
% discrètes dupliqués par raport à la normale au réseau.
% ***********************************************************
clear;
% Frequence du signal ultrasonique
f=40000;
% Frequence d'échantillonnage / canal
fs=75000;
% Longueur du réseau (m)
l=0.071;
% Vitesse du son (m/s)
c=340;
% Delais maximal entre les recepteurs extremes
delai_max=ceil(fs*l/c)
% Facteur de d'interpolation des delais, ainsi
% nous obtenons nos délais fractionaires.
fact=10;

% Calcul itératif des max du corelateur
% Creation de 5 signaux déphasés linéairement
% 100 échantillons pour chaque canal
n=linspace(1,100,100);
cste=2*pi*f/fs;

% Pour tous les angles possibles en fonction du délais maxial
% implantable entre les récepteurs extrémums, effectue la corrélation.

for da=0:delai_max*fact,
  d=da/fact;              % C'est ici que les 'd' simulent le délais

  f1=sin(cste*(n+0*d));         % La phase de chaque sin est linéairement
  f2=sin(cste*(n+0.25*d));      % espacé pour simulé le temps de délais.
  f3=sin(cste*(n+0.5*d));
  f4=sin(cste*(n+0.75*d));
  f5=sin(cste*(n+1.00*d));

  f_corel=(1/5)*abs(f1+f2+f3+f4+f5);
  f_corel=f_corel.*f_corel;
%  stem(f_corel)
%  pause
  % Trouve la valeur maximale du signal corrélé
  y(da+1)=max(f_corel);
```

```
end

% Vecteur horizontal pour le graphique
hor=linspace(0,delai_max,delai_max*fact+1);

stem(hor,y);
 title('Fonction de corelation discrete à 75 kHz par canal');
 xlabel('Delais Maximal dans le réseau de 5 canaux');
 ylabel('Niveau de Corrélation')
```

ANNEXE G

```
% ***********************************************************
% FICHIER : delais.m
%
% Auteur  : Sylvain Fortin
% Date    : 24 juillet 1998
%
% DESCRIPTION :
% Corelateur pour 5 signaux discrêts sinusoïdaux dont la
% phase varie selon l'angle incident. Les delais sont
% implantés pour effectuer ultérieurement la somme.
% Nous n'implémentons pas les délais causés
% par l'échantillonage multipexé des canaux.
% Nb. Il y a 12.4 us de delais entre l'échantillonage
% entre le canal 0 et le canal 4 ceci doit être considéré
% car ce temps représente un demi cycle à 40 kHz.
% ***********************************************************
clear;
% Frequence du signal ultrasonique
f=40000;
% Frequence d'échantillonnage / canal
fs=85000;
% Longueur du réseau (m)
l=0.071;
% Vitesse du son (m/s)
c=340;
% Delais maximal entre les recepteurs extremes
delai_max=round(fs*l/c)
% Vecteur d'angles en degrés
angles=linspace(-36,36,100);
% Calcul des délais pour chaque recepteur et chaque angle
d0= round( abs( ( 0*fs*l*sin(pi*angles/180))/(4*c) ) );
d1= round( abs( ( 1*fs*l*sin(pi*angles/180))/(4*c) ) );
d2= round( abs( ( 2*fs*l*sin(pi*angles/180))/(4*c) ) );
d3= round( abs( ( 3*fs*l*sin(pi*angles/180))/(4*c) ) );
d4= round( abs( ( 4*fs*l*sin(pi*angles/180))/(4*c) ) );
% Création de 5 signaux à analyser par le corrélateur
% 50 échantillons pour chaque canal

% Retard de phase sur à l'échantillonage série est
% calculé par Ret_Phase=arcsin(ct/d) où t = 12.4 us
retph=0
n=linspace(1,40,40);
cste=2*pi*f/fs;
f0=sin(cste*n);
f1=sin(cste*n);
f2=sin(cste*n);
```

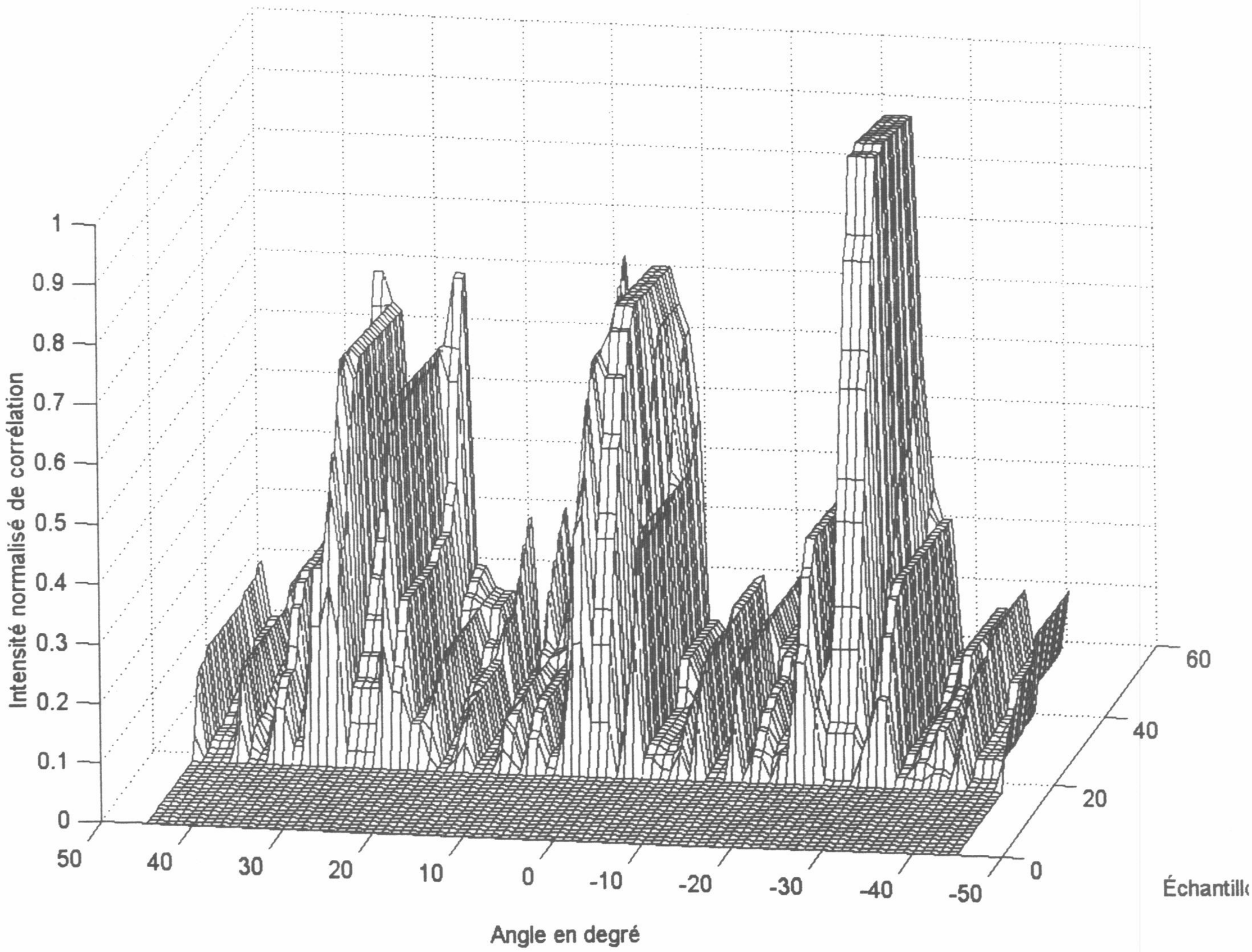
```
f3=sin(cste*n);
f4=sin(cste*n);
% Applique la correlation pour chaque direction
for dir=1:length(angles);
      for i=delai_max+1:length(f0),
      temp = 0.2*abs( f0(i-d0(dir)) + f1(i-d1(dir)) + f2(i-d2(dir)) +
f3(i-d3(dir)) + f4(i-d4(dir)) );
      temp=temp*temp;
      cor(dir,i) = temp;
      end;
end;

% Détecteur d'enveloppe
% Entree : cor(dir,i)     Signal de correlation en fonction de l'angle et
du temps i (échantillon)
% Sortie : env(dir,i)     Signal du détecteur d'enveloppe en fonction de
la direction et du temps i

for dir=1:length(angles);
  for i=2:length(f0),
    env(dir,1)=0;                                  % Départ de l'enveloppe à 0
    if cor(dir,i)>env(dir,i-1)
      env(dir,i)=0.6*env(dir,i-1)+0.4*cor(dir,i);% Charge condensateur
    else
      env(dir,i)=0.998*env(dir,i-1);               % Décharge condensateur
    end;
  end;
end;

% COnversion pour affichage MESH
AxeX=linspace(0,length(f0),length(f0));
mesh( AxeX, angles, env);
title('Simulation de corrélation avec enveloppe. fs=85kHz');
xlabel('Échantillon');
ylabel('Angle en degré');
zlabel('Intensité de corrélation');

view(-64,20);
```

ANNEXE H

```
%*****************************************************************************
% FICHIER : sig2.m
%
% Auteur  : Sylvain Fortin
% Date    : 21 juillet 1998
%
% DESCRIPTION :
% Corelateur pour 5 signaux discrêts echantillones provenant du systeme
% d'acquisition d'écho. La corelation s'effectue pour différent angles.
% L'équation du filtre directionel utilise des retard et une somation.
% Le signal de puissance est envoyé dans un détecteur d'enveloppe.
% Le fichier TABLE.SMP comprend tous les échantillons que le terminal
MONSONAR
% a recu. Le fichier est formé par une alternance des échantillons pour
chaque
% canal. Chaque échantillon est exprimé sous un mot de 16 bits signés. Les
% bytes les moins significatifs arrivent avant les bytes les plus
sigificatifs.
% L'alternance des canaux est comme suit: 0,1,2,3,4,0,1,2,3,4,0,1... jusqu'à
% la fin du fichier. Le nombre d'échantillons total dans le fichier peut
% varier et le programme SIG.M ne sera pas affecté.
%
*****************************************************************************
clear;
% Frequence du signal ultrasonique
f=40000;
% Frequence d'échantillonnage / canal
fs=85106;
% Longueur du réseau (m)
l=0.071;
% Vitesse du son (m/s)
c=340;
% Delais maximal en nombre d'échantillons entre les
% recepteurs extremes formant le réseau
delai_max=ceil(fs*l/c)

% Generer les delais en fonction des dirrerents angles en degrés
% Effectue le balayage pour les deux quadrans et l'inversion
% qui en découle sur l'allignement retard.
angles=linspace(-45,45,91);
for i=1:45,
      d4(i)=round( abs( ( 0*fs*l*sin(pi*(46-i)/180))/(4*c) ) );
      d3(i)=round( abs( ( 1*fs*l*sin(pi*(46-i)/180))/(4*c) ) );
      d2(i)=round( abs( ( 2*fs*l*sin(pi*(46-i)/180))/(4*c) ) );
      d1(i)=round( abs( ( 3*fs*l*sin(pi*(46-i)/180))/(4*c) ) );
      d0(i)=round( abs( ( 4*fs*l*sin(pi*(46-i)/180))/(4*c) ) );
end;
for i=46:91,
      d0(i)=round( abs( ( 0*fs*l*sin(pi*(i-46)/180))/(4*c) ) );
      d1(i)=round( abs( ( 1*fs*l*sin(pi*(i-46)/180))/(4*c) ) );
      d2(i)=round( abs( ( 2*fs*l*sin(pi*(i-46)/180))/(4*c) ) );
      d3(i)=round( abs( ( 3*fs*l*sin(pi*(i-46)/180))/(4*c) ) );
      d4(i)=round( abs( ( 4*fs*l*sin(pi*(i-46)/180))/(4*c) ) );
end;

FID1=fopen('c:\sonar\tc\table.smp','r');  % Fichier d'échantillons
[Smpl, Nb_total]=fread(FID1,'short');
```

```
Nb_Echant_Canal=floor(Nb_total/5);  % Calcul le nombre d'échantillons par
canal
Nb_Echant_Canal=floor(Nb_Echant_Canal/16);  % Reduit le nombre d'échantillons
pour traitement plus rapide

% Création d'un vecteur d'identification du numéro d'échantillon du canal
n=LINSPACE(0,Nb_Echant_Canal-1,Nb_Echant_Canal);

for i=1:Nb_Echant_Canal,                % Extraction des échantillons pour chaque
canal
  s0(i)=Smpl(5*i-4);
  s1(i)=Smpl(5*i-3);
  s2(i)=Smpl(5*i-2);
  s3(i)=Smpl(5*i-1);
  s4(i)=Smpl(5*i-0);
end;

 % Applique la correlation pour chaque direction
for dir=1:length(angles);
   for i=delai_max+1:length(s0),
   temp = abs( s0(i-d0(dir)) + s1(i-d1(dir)) + s2(i-d2(dir)) + s3(i-d3(dir))
          + s4(i-d4(dir)) );
   temp=temp/10240;                    % Division par 5*(2^12)/2 'normalisation'
   cor(dir,i) = 1.7*temp*temp;                % Puissance et gain

   end;
end;

% Détecteur d'enveloppe
% Entree : cor(dir,i)    Signal de correlation en fonction de l'angle et du
temps i (échantillon)
% Sortie : env(dir,i)    Signal du détecteur d'enveloppe en fonction de la
direction et du temps i

for dir=1:length(angles);
  for i=2:length(s0),
    env(dir,1)=0;                      % Départ de l'enveloppe à 0
    if cor(dir,i)>env(dir,i-1)
      env(dir,i)=0.6*env(dir,i-1)+0.4*cor(dir,i);
    else
      env(dir,i)=0.998*env(dir,i-1);
    end;
  end;
end;


% COnversion pour affichage MESH
AxeX=linspace(0,length(s0),length(s0));
mesh( AxeX, angles, env);

view(-80,20);

title('Corrélation des 5 canaux. fs=85.1 kHz');
xlabel('Échantillon');
ylabel('Angle en degré');
zlabel('Intensité normalisé de corrélation');
```

Corrélation des 5 canaux. fs=85.1 kHz



ANNEXE J

Programme MONSONAR.C

```
/*****************************************************************************
*
 Fichier     : MONSONAR.C
 Auteur      : Sylvain Fortin
 Date        : 19 juillet 1998
 Description:
 Programme qui implémente un terminal sur le port sériel d'un PC. Ce terminal
 permet d'acquérir les échantillons du système de sonar et de les enregister
 sur le disque dur. Ce fichier est utilisé par MATLAB pour le traitement
 numérique du signal.
 Le protocole est très simple, tout caractère ASCII est affiché en mode
 normal.
 Le caractère '%' signale l'arrivée des échantillons et le programme crée un
 fichier sur le disque dur et écrit les échantillons reçus à mesure qu'ils
 arrivent.
 Le caractère '&' Indique la fin du groupe d'échantillons et le programme
 termine l'écriture du fichier et retourne en mode de terminal normal.
 Le programme utilise l'interruption du port sérielle pour recevoir chaque
 byte.
*****************************************************************************
/

/******** COM_3.C *********/

#include "dos.h"
#include "c:\sonar\tc\com_3.h"     /* comm prototypes */
#include "stdio.h"


#define TCICOMM                    /* avoid extern declaration in... */
#define OUT          -100          /* sortie demand, par l'utilisateur */
#define Taille 4000                /* Taille du buffer circulaire */

/** variables globales */

int XonXoff;                       /* auto xon/xoff support flag
                                   * XonXoff  = 0 ==> direct control
                                   * XonXoff != 0 ==> utilise XonXoff
                                   */
int xofsnt;                        /* XOFF transmitted flag */
int xofrcv;                        /* XOFF received flag */
int flag =0;
short int Mot;

/** variables du buffer circulaire **/
unsigned char tableau[Taille];
unsigned char *sortie;
unsigned char *entree;
int quantite;
int dimension;                     /** Taille du buffer **/


/** globals not externally accessible ****************************************/

static int   dat8250;              /* 8250 data register */
static int   stat8250;             /* 8250 line-status register */
static int   com8250;              /* 8250 line-control register */
static char  en8259;               /* 8259 IRQ enable mask */
static char  dis8259;              /* 8259 IRQ disable mask */
static unsigned int intv;          /* interrupt number to usurp */
```

```
static void interrupt ( far *oldvec ) ( );
/* vector of previous comm interrupt */
void InitTampon(void)
{
  sortie=tableau;
  entree=tableau;
  quantite=0;
  dimension=Taille;
}
/*******************
Fonction Write tampon
Entree : Valeur a ecrire dans le tampon
Sortie : Renvoie -1 si le buffer est plein
********************/
int WriteTampon(unsigned char d)
{
  if((sortie==entree)&&quantite!=0)
     return (-1);                /* Buffer plein */
  *entree = d;                   /* Place donne dans cellule */
  quantite++;
  entree++;
  if(entree==tableau+dimension)
     entree=tableau;             /* Retour au debut du tableau */
  return(0);
}
/***************************
Fonction Read tampon
Sortie: Renvoie -1 si le buffer est vide sinon c'est une data
****************************/
int ReadTampon(void)
{
  int d;
  if((sortie==entree)&&quantite==0)
     return(-1);                 /* Buffer vide */
  d=*sortie;
  quantite--;
  sortie++;
  if(sortie==tableau+dimension)
     sortie=tableau;
  return(d);
}

/*******************************
Handler d'interuption sur IRQ5 (COM3)
*************************************/
void interrupt serint ( void )
{                                /* ISR to receive character */
  unsigned char car;
  disable();
//  flag=1;
  car = inportb(dat8250);
  /*appeler la fonction qui ecrit dans le tampon /* echant -> tampon_circ!*/
  WriteTampon(car);
  enable();
  outportb ( 0x20, 0x20 );       /* Generic EOI to 8259 */
}


void comm_close ( void )
{                                /* restore previous settings of 8259 */
  outportb ( INTCONT, dis8259 | inportb ( INTCONT ) );
```

```
  /* Disable com interrupt at 8259 */
  setvect ( intv, oldvec );      /* Reset original interrupt vector */
  xofrcv = 0;                    /* commence autre travaille sans attendre de
  XON */
}

void dobaud ( long baudrate)
{
                                 /* parses baud integer to mask,
                                 * re-inits port accordingly */
  unsigned char portval;
  unsigned char blo, bhi;
  switch ( baudrate )
  {
                                 /* Baud rate LSB's and MSB's */
  case 50:
    bhi = 0x9;
    blo = 0x00;
    break;
  case 75:
    bhi = 0x6;
    blo = 0x00;
    break;
  case 110:
    bhi = 0x4;
    blo = 0x17;
    break;
  case 150:
    bhi = 0x3;
    blo = 0x00;
    break;
  case 300:
    bhi = 0x1;
    blo = 0x80;
    break;
  case 600:
    bhi = 0x0;
    blo = 0xC0;
    break;
  case 1200:
    bhi = 0x0;
    blo = 0x60;
    break;
  case 1800:
    bhi = 0x0;
    blo = 0x40;
    break;
  case 2000:
    bhi = 0x0;
    blo = 0x3A;
    break;
  case 2400:
    bhi = 0x0;
    blo = 0x30;
    break;
  case 4800:
    bhi = 0x0;
    blo = 0x18;
    break;
  case 9600:
    bhi = 0x0;
    blo = 0x0C;
    break;
```

```
    case 19200:
      bhi = 0x0;
      blo = 0x06;
      break;
    case 38400:
      bhi = 0x0;
      blo = 0x03;
      break;                        /* ajout Sept 1990 */
    case 57600:
      bhi = 0x0;
      blo = 0x02;
      break;                        /* ajout Sept 1990 */
    case 115200:
      bhi = 0x0;
      blo = 0x01;
      break;                        /* ajout Sept 1990 */
    default:
      return;
    }
  portval = inportb ( com8250 );          /* read Line-Control Reg val */
  outportb ( com8250, portval | 0x80 ); /* set high bit for baud init */
  outportb ( dat8250, blo );              /* Send LSB for baud rate */
  outportb ( dat8250 + 1, bhi );          /* Send MSB for baud rate */
  outportb ( com8250, portval );          /* Reset initial value at LCR */
}


/* installs comm interrupts */

int comm_open ( char portid,
                long speed,               /* 50 … 115200 bauds */
                char parity,              /* None, Event, Odd (1iŠre lettre) */
        char databits,                    /* mots de 5 … 8 bits de long */
        char stopbit)                     /* 1 ou 2 stop bits */
{
  char controle;
  unsigned int be;

    /* PARITY :
    * bit 3 ==> si = 1 enable parity, si 0 desable parity
    * bit 4 ==> si = 0 parit, impaire (odd), si = 1 parit, paire (event)
    */
  if((parity == 'E') || (parity == 'e')) parity = 0x18;
  else if((parity == 'O') || (parity == 'o')) parity = 0x08;
  else parity = 0x00;          /* par d,faut ==> pas de parit, */

    /* DATABITS :
    *           bit 1 et bit 0 ==> databits
    *             0        0   ==> 5 bits
    *             0        1   ==> 6 bits
    *             1        0   ==> 7 bits
    *             1        1   ==> 8 bits
    */
  if(databits == 5) databits = 0;
  else if(databits == 6) databits = 1;
  else if(databits == 7) databits = 2;
  else databits = 3;                /* par défaut ==> 8 bits par mots */

    /* STOPBIT :
    *          bit 3 ==> si = 0 ==> 1 stopbit,
    *                    si = 1 ==> 2 stopbit (1« si 5 databits)
```

```
  */
if(stopbit == 2) stopbit = 0x04;
else stopbit = 0x00;            /* par d,faut ==> 1 stop bit */
controle = parity | databits | stopbit;

be = biosequip ( );             /* to get # installed serial ports */
be <<= 4;                       /* shift-wrap high bits off */
be >>= 13;                      /* shift down to low bits */

printf("Nombre de ports disponible : %d\n",be);
printf("Connection sur le port COM%d\n",portid);

if ( be >= portid )
{
  if ( portid == 1 )
  {
    dat8250 = MDMDAT1;
    stat8250 = MDMSTS1;
    com8250 = MDMCOM1;
    dis8259 = MDMINTC;
    en8259 = MDMINTO;
    intv = MDMINTV;
  }
  else if ( portid == 2 )
  {
    dat8250 = MDMDAT2;
    stat8250 = MDMSTS2;
    com8250 = MDMCOM2;
    dis8259 = MDINTC2;
    en8259 = MDINTO2;
    intv = MDINTV2;
  }
  else if ( portid == 3 )
  {
        dat8250 = MDMDAT3;
        stat8250 = MDMSTS3;
        com8250 = MDMCOM3;
        dis8259 = MDINTC3;
        en8259 = MDINTO3;
        intv = MDINTV3;
  }
  else if ( portid == 4 )
  {
    dat8250 = MDMDAT4;
    stat8250 = MDMSTS4;
    com8250 = MDMCOM4;
    dis8259 = MDINTC4;
    en8259 = MDINTO4;
    intv = MDINTV4;
  }
  else
    return ( 0 );
  dobaud ( speed );                 /* set baud */
  oldvec = getvect ( intv );        /* Save old int vector */
  disable();
  setvect ( intv, serint );         /* Set up SERINT as com ISR */
  enable();
  outportb ( com8250, controle);    /* Set databits, stopbit, parity */
  outportb ( com8250 + 1, 0xb );    /* Assert OUT2, RTS, and DTR */
  inportb ( dat8250 );
  outportb ( dat8250 + 1, 0x1 );    /* Receiver-Data-Ready int */
```

```
    outportb ( INTCONT, en8259 & inportb ( INTCONT ) );
    /* Enable 8259 interrupts */
  }
  else
    be = 0;
  return ( be );
}

int comm_putc ( unsigned char c )        /* sends char out port */
{
    unsigned char status;

    status = inportb ( stat8250 );       /* Wait til transmitter is ready */
    while ( (status & 0x20 ) == 0 )
    {
      status = inportb ( stat8250 );     /* Wait til transmitter is ready */
    }
    outportb ( dat8250, c );             /* then send it */
     return status;
}

/* Routine de conversion ASCII HEX. en valeur decimal */

int Conv(char caract)
{
    switch(caract)
        {
        case '0':return 0;
        case '1':return 1;
        case '2':return 2;
        case '3':return 3;
        case '4':return 4;
        case '5':return 5;
        case '6':return 6;
        case '7':return 7;
        case '8':return 8;
        case '9':return 9;
        case 'A':return 10;
        case 'B':return 11;
        case 'C':return 12;
        case 'D':return 13;
        case 'E':return 14;
        case 'F':return 15;
        default :return ']';
        }
}                          /*fin de Conv */
/*****************************************************************************/

main()
{
  char key;
  char RxData;
  int  WrFich;                /* 1==>Ecriture fichier  0==>Ecriture … l'ecran */
  int   Quartet;              /* MSB==> 0,1,2,3 <==LSB */
  FILE *fptr;

  InitTampon();               /* Prepare le buffer circulaire */
  XonXoff = 0;                /* XonXoff == 0, controle direct */
                              /* XonXoff != 0, controle par XonXoff */
  printf("\n\nTerminal pour sonar … r,seau\n");
  comm_open (COM3, 19200, 'N', 8, 1);
```

```
printf("Appuyer sur ESCAPE pour quitter\n\n");


WrFich=0;                    // Par defaut echo sur l'ecran
while ( key != ESC )
   {
   while (-1!=(RxData=ReadTampon()))      // Tant que tampon n'est pas vide
      {
      if (RxData=='%')                    // Commande d',criture de fichier
      ?
        {
        if ((fptr=fopen("table.smp","wb"))==NULL)
           {
           printf("Pas capable d'ouvrir le fichier d',criture");
           exit();
           }
        WrFich=1;                         // Drapeau ecriture de fichier
        Quartet=0;
        Mot=0;
        printf("\nR,ception des ,chantillons...\n");
        }
      else if (RxData=='&')               // Commande de fermeture de
      fichier ?
        {
        fclose(fptr);
        WrFich=0;          // Drapeau d'echo console
        printf("\nFin de l',criture du fichier\n");
        }

      else if (WrFich==0)                 // Echo sur l'ecran
        {
        putchar(RxData);
        }
      else                                // Ecriture dans fichier
        {
        if (Quartet==0)
           {
           Mot=Conv(RxData)*4096;
           }
        else if (Quartet==1)
           {
           Mot=Mot+(0x7 & Conv(RxData))*256;
           Mot=Mot-(0x8 & Conv(RxData))*256;  // Complement 2 a 12 bits
           }
        else if (Quartet==2)
           Mot=Mot+Conv(RxData)*16;
        else
           {
           Mot=Mot+Conv(RxData);
           fwrite(&Mot,sizeof(Mot),1,fptr);
           printf("%d ",Mot);
           }
        Quartet=Quartet+1;
        if (Quartet==4)
           {
           Quartet=0;
           }
//      putchar('.');
        }
      }
   if ( kbhit() )
```

```
            {
            key = (unsigned char) getch();
            comm_putc(key);
            }
        }
    comm_close();

    return(0);
}
```

ANNEXE K

Programme SONAR.ASM

```
        LIST    p = 17C44, f= INHX32, r=DEC
        PROCESSOR 17C44
        __config H'FFE3'; Microcontroleur, WDT Disable, EC Oscillator
;*****************************************************************
*
; Fichier          : SONAR.ASM
; Auteur           : Sylvain Fortin
; Date             : 1 aout 1998
; Description      :
; FIRMWARE pour le systeme de sonar ultrasonique à réseau de n senseurs.
; Implémentation d'un menu par interface sérielle permettant de faire
; l'acquisition des échantillons d'écho lue par le convertisseur A/D.
;*****************************************************************
*
        include "P17C44.INC"
;*****************************************************************
*
; Constantes pour l'initialisation du l'UART
;*****************************************************************
*
#define SPBRG_INIT      0x19    ; 19200 Baud
#define TXSTA_INIT      0x20    ; 8-bits, Tx Enable, Asynchrous,
#define RCSTA_INIT      0x90
;************** ALU
*****************************************************
Carry           equ     0
;************** Registres de la banque 0
*************************************
Can0Cfg         equ     0x1A    ; Patern de configuration du ADC
Can1Cfg         equ     0x1B
Can2Cfg         equ     0x1C
Can3Cfg         equ     0x1D
Can4Cfg         equ     0x1E

loopext         equ     0x20    ; Compteur de boucle externe de delais
loopint         equ     0x21    ; Compteur de boucle interne de delais
SerOutData      equ     0x22    ; Caractere a envoyer sur le UART
SerInData       equ     0x23    ; Caractere recu sur le UART
ByteHEX         equ     0x24    ; Byte à afficher sur l'écran en HEX
MSB_Data_RAM    equ     0x25    ; Data, Ecriture et lecture en RAM
LSB_Data_RAM    equ     0x26
MSB_Adr_RAM     equ     0x27    ; Adresse, Ecriture et lecture en RAM
LSB_Adr_RAM     equ     0x28
BinAscV         equ     0x29    ; Quartet convertis en ASCII
Pat_RAM         equ     0x2A    ; Patern de test de la RAM
Compteur        equ     0x2F
CompBloc        equ     0x30    ; Compteur de bloc de 256 Sample
;************** Bits du PORTC de contrôle
*************************************
LSB_SRAM        equ     0       ; Ligne de chargement du LSB Adresse de la
SRAM
MSB_SRAM        equ     1       ; Ligne de chargement du MSB Adresse de la
SRAM
WE_SRAM         equ     2       ; Ligne d'écriture en SRAM (Actif bas)
OE_SRAM         equ     3       ; Ligne de lecture de la SRAM (Actif bas)
CONVST          equ     4       ; Début de conversion A/D
WR_AD           equ     5       ; Ecriture au convertisseur A/D
RD_AD           equ     6       ; Lecture sur le convertisseur A/D
;*************  Bits du PORTB
*************************************************
;RB0-RB3 Bits 2^8 à 2^11 du bus du convertisseur A/D
```

```
;************** Définition des vecteurs
*************************************
                org     0x00    ; Le vecteur RESET
                goto    start
;               org     0x08    ; Le vecteur Interrupt de la pin INT
;               goto    Interupt
;*****************************************************************************
*
        org     0x10    ; Le code de départ
start
;       call    wait
;       call    wait
;       call    wait
;*****************************************************************************
*
; Code du registre de configuration de convertisseur A/D
;*****************************************************************************
*
        movlw   0x01            ; Formons 0 0 A2 A1 A0 SWCONV SWSTBY FORMAT
        movwf   Can0Cfg         ; Pour nos 5 canaux
        movlw   0x09
        movwf   Can1Cfg
        movlw   0x11
        movwf   Can2Cfg
        movlw   0x19
        movwf   Can3Cfg
        movlw   0x21
        movwf   Can4Cfg
;*****************************************************************************
*
; Initialisation du l'UART
;*****************************************************************************
*
Setup_Async_Mode
        movlb   0               ; SPBRG, TXSTA & RCSTA are in bank 0
        movlw   SPBRG_INIT      ; Regle la vitesse de transfert (baud)
        movwf   SPBRG           ; baud rate generator is reset & initialised
        movlw   TXSTA_INIT
        movwf   TXSTA           ; 8-bits transmission, asyn mode
        movlw   RCSTA_INIT
        movwf   RCSTA           ; 8-bit reception, enable serial port,
                                ; enable reception
;*****************************************************************************
*
; Message de demarrage
;*****************************************************************************
*
        movlw   LOW  MessBoot
        movwf   TBLPTRL
        movlw   HIGH MessBoot
        movwf   TBLPTRH
        call    Aff_Chaine      ; Affiche message de démarrage
;*****************************************************************************
*
; Initialisation du PORTB
; RB0-RB3 : bits 2^8 à 2^11 du bus de data du convertisseur A/D (Entree)
;*****************************************************************************
*
        movlb   0               ; DDRB is in Bank 0
        movlw   0xFF            ; Valeur initiale sur le PORTB
        movwf   PORTB
```

```
        movlw   0xFF            ; PORTB en entree  0=Output, 1=input
        movwf   DDRB
;*****************************************************************************
*
; Initialisation du PORTC
; Broches de controle de la DRAM et du convertisseur A/D
; RC3=OE , RC2=WE , RC1=latch MsbAdd , RC0=latch LsbAdd ( Sorties )
; RC4=CONVST, RC5=WR_AD, RC6=RD_AD ( Sorties )
;*****************************************************************************
*
        movlb   1               ; DDRC is in Bank 1
        movlw   0xFC            ; Initialise PORTC WE_SRAM->1, OE_SRAM->1
        movwf   PORTC           ;       Latch Msb->0, Latch Lsb->0
                                ; CONVST->1,WR__AD->1,RD_AD->1
        clrf    DDRC,f          ; PORTC en sortie     0=output 1=input
;*****************************************************************************
*
; Initialisation du bus de sortie PORTD reliant la SRAM et le ADC
;*****************************************************************************
*
        movlb   1               ; DDRD is in Bank 1
        movlw   0xFF            ; PORTD entree
        movwf   DDRD
;*****************************************************************************
*
; Test de la SRAM
;*****************************************************************************
*
        movlw   LOW RamTMes     ; Message de debut du test de SRAM
        movwf   TBLPTRL
        movlw   HIGH RamTMes
        movwf   TBLPTRH
        call    Aff_Chaine
        movlb   0
        clrf    TBLPTRH,f       ; Pointe au debut de la SRAM
        clrf    TBLPTRL,f
        clrf    Pat_RAM,f       ; Initialise Patron de test avec 0
RamWrT  movfp   Pat_RAM,WREG    ; Prend patern
        movpf   WREG,LSB_Data_RAM
        movpf   WREG,MSB_Data_RAM
        call    WrRAM           ; Ecris patern sur bus 12 bits Data SRAM
        movlw   0
        incf    TBLPTRL,f       ; Prochaine adresse
        addwfc  TBLPTRH,f
        movlb   0               ; Pat_RAM  Bank 0
        incf    Pat_RAM,f       ; Calcule prochain patron
        movlw   0x80            ; Adresse_RAM = 0x8000 ?
        cpfseq  TBLPTRH
        goto    RamWrT
                                ; Fin ecriture patron de test
        clrf    TBLPTRH,f       ; Pointe au debut de la SRAM
        clrf    TBLPTRL,f
        clrf    Pat_RAM,f       ; Initialise Patron de test avec 0
RamRdT
        call    RdRAM           ; Lecture SRAM 12 bits
        movlb   0               ; PatRam,LSB_Data_RAM,MSB_Data_RAM   Bank 0
        movfp   Pat_RAM,WREG    ; Prend patern calculé
        cpfseq  LSB_Data_RAM    ; Compare LSB avec le patrons
        goto    FailRAM
        andlw   0x0F
        cpfseq  MSB_Data_RAM    ; Compare MSB avec patrons
```

```
        goto    FailRAM
        incf    TBLPTRL,f           ; Prochaine adresse
        btfss   ALUSTA,Carry
        goto    NoRetTR             ; Pas de bit de retenue
        incf    TBLPTRH,f           ; On a un bit de retenue
        movlw   "."                 ; Affiche qu'un bloc de 256 Byte est OK
        movpf   WREG,SerOutData
        call    Send_Char
NoRetTR movlb   0                   ; Par_RAM  Bank 0
        incf    Pat_RAM,f           ; Calcule prochain patron
        movlw   0x80                ; Adresse_RAM = 0x8000 ?
        cpfseq  TBLPTRH
        goto    RamRdT
        movlw   LOW RamPass         ; SUCCES du test de SRAM
        movwf   TBLPTRL
        movlw   HIGH RamPass
        movwf   TBLPTRH
        call    Aff_Chaine          ; Message de succes du test
        goto    SRAM_OK
FailRAM movlw   LOW RamFail         ; ECHEC du test de SRAM
        movwf   TBLPTRL
        movlw   HIGH RamFail
        movwf   TBLPTRH
        call    Aff_Chaine          ; Message d'ECHEC du test
MortRAM goto    MortRAM             ; Boucle sans fin
SRAM_OK
;*****************************************************************
*
; Initialisation des interruptions
; Noter que la ligne EOC du ADC est relié sur la pin INT du microcontroleur
;*****************************************************************
*
;       bcf     T0STA,7             ; Front baissant sur pin INT => interruption
;       bsf     CPUSTA,4            ; Disable global interupt (GLINT)
;       bcf     INTSTA,4            ; Clear Interupt sur INT (INTF)
;       bsf     INTSTA,0            ; Enable interupt for INT pin
;       bcf     CPUSTA,4            ; Enable global interupt (GLINT)
;*****************************************************************
*
; Affiche le menu d'aide
;*****************************************************************
*
        movlw   LOW  MenuTxt
        movwf   TBLPTRL
        movlw   HIGH MenuTxt
        movwf   TBLPTRH
        call    Aff_Chaine          ; Affiche le menu d'aide
;*****************************************************************
*
; Programme principal de gestion du menu et appelle les fonctions sélectionnés
;*****************************************************************
*
MenuLoop
        movlb   0                   ; SerOutData dans banque 0
        movlw   "\n"                ; Force un changement de ligne
        movwf   SerOutData
        call    Send_Char
        movlw   "\r"
        movwf   SerOutData
        call    Send_Char
        movlw   ">"                 ; Affiche Prompt d'attente d'une touche
```

```
        movwf   SerOutData
        call    Send_Char
        call    Get_Char        ; Attente d'une touche au clavier
        movfp   SerInData,WREG
        movwf   SerOutData      ; Echo du caractère vers PC
        call    Send_Char
        movlw   LOW  NextLine   ; Cariage return et Line Feed
        movwf   TBLPTRL
        movlw   HIGH NextLine
        movwf   TBLPTRH
        call    Aff_Chaine
        movlb   0               ; SerInData is in bank 0
        movlw   "a"             ; Touche du Menu d'aide ?
        CPFSEQ  SerInData
        goto    Men1
        goto    Aff_Menu        ; Va au programme d'affichage du menu
Men1    movlw   "m"
        CPFSEQ  SerInData
        goto    Men2
        goto    ModCanal        ; Va au programme Modifie Canal A/D
Men2    movlw   "c"
        CPFSEQ  SerInData
        goto    Men3
        goto    AffCanal        ; Va au programme Affiche Canal A/D
Men3    movlw   "l"
        CPFSEQ  SerInData
        goto    Men4
        goto    LireEcha        ; Programe de lecture d'un échantillon sur A/D
Men4    movlw   "d"
        CPFSEQ  SerInData
        goto    Men5
        goto    DumpMem         ; Va au programme de Dump Memoire
Men5    movlw   "f"
        CPFSEQ  SerInData
        goto    Men6
        goto    DumpFich        ; Va au programme de Dump Memoire
Men6

goto    MenuLoop
;*****************************************************************************
*
; Gestion Touche d'affichage du Menu d'aide
;*****************************************************************************
*
Aff_Menu
        movlw   LOW  MenuTxt
        movwf   TBLPTRL
        movlw   HIGH MenuTxt
        movwf   TBLPTRH
        call    Aff_Chaine
        goto    MenuLoop
;*****************************************************************************
*
; Gestion Touche de modification du canal A/D
;*****************************************************************************
*
ModCanal
        movlw   LOW EntreCan    ; Message Entrer no de canal
        movwf   TBLPTRL
        movlw   HIGH EntreCan
        movwf   TBLPTRH
```

```
        call    Aff_Chaine
        call    Get_Char            ; Attente d'une touche au clavier
        movlb   0                   ; SerInData, SerOutData in bank 0
        movfp   SerInData,WREG
        movwf   SerOutData          ; Echo du caractère vers PC
        call    Send_Char
        movlw   LOW  NextLine       ; Cariage return et Line Feed
        movwf   TBLPTRL
        movlw   HIGH NextLine
        movwf   TBLPTRH
        call    Aff_Chaine
        movlb   0                   ; SerInData is in bank 0

        goto    MenuLoop
;*****************************************************************************
*
; Routine d'affichage du canal AD courant
;*****************************************************************************
*
AffCanal
        movlw   0
        movpf   WREG,Compteur
LopComp
        movfp   Compteur,WREG
        movpf   WREG,ByteHEX
        call    AffByte
        incf    Compteur,f
        goto    LopComp


        goto    MenuLoop
;*****************************************************************************
*
; Routine qui lance le transfert de groupe d'échantillons du A/D vers les SRAM
; Débit de 500 kHz voulu...  125 ns / instructions  (32 MHz)
;*****************************************************************************
*
LireEcha
        movlb   0                   ; PORTB, DDRB Bank 0
        movlw   0xFF                ; PORTB 3-0 en entree  0=Output, 1=input
        movwf   DDRB
        movlw   0x0E                ; Initialise le nombre de blocs a lire
        movwf   CompBloc            ; 14 X bloc de 256 sample
        clrf    TBLPTRH,f           ; Pointe le premier bloc de 256 echantillons

        movlb   1                   ; DDRD,PORTD,PORTC,FinWrSmp   Bank 1
        clrf    TBLPTRL,f           ; Pointe au debut d'un bloc de 256 Sample
        movpf   TBLPTRH,PORTD       ; Envoie l'adresse MSB du bloc actuel
        clrf    DDRD,f              ; PORTD en sortie
        bsf     PORTC,MSB_SRAM      ; Chargement du MSB Adresse (Bloc)
        bcf     PORTC,MSB_SRAM
        movfp   Can0Cfg,PORTD       ; Commande au ADC

Smpl Can0
        bcf     PORTC,WR_AD         ; *** Conversion CANAL 0 ***
        clrf    DDRD,f              ; PORTD en sortie
Cont4   bsf     PORTC,WR_AD         ; désactiver le signal d'écriture au ADC
                                    ;-- 600 ns Track/Hold Acquisition Time
        movpf   TBLPTRL,PORTD       ; Met LSB Address sur le LATCH LSB
        bsf     PORTC,LSB_SRAM      ; Chargement du LSB Adresse
        bcf     PORTC,LSB_SRAM
```

```
        bcf     PORTC,CONVST      ;-- Apres 5 instruct, On lance la conversion
        bsf     PORTC,CONVST      ;-- Resultat dans 1.6 us (13 instructions)
        movlw   0xff
        movwf   DDRD
        incf    TBLPTRL,f         ; Prochaine adresse
        btfsc   ALUSTA,Carry      ; Fin du bloc ?
        goto    Break0
        nop
        movfp   Can1Cfg,PORTD     ; Commande au ADC pour prochaine conversion
        bcf     PORTC,WE_SRAM     ; Ecrire dans la SRAM
        bcf     PORTC,RD_AD       ; Lire l'échantillon
        bsf     PORTC,WE_SRAM
        bsf     PORTC,RD_AD

        bcf     PORTC,WR_AD       ; *** Conversion CANAL 1 ***
        clrf    DDRD,f            ; PORTD en sortie
Cont0   bsf     PORTC,WR_AD       ; désactiver le signal d'écriture au ADC
                                  ;-- 600 ns Track/Hold Acquisition Time
        movpf   TBLPTRL,PORTD     ; Met LSB Address sur le LATCH LSB
        bsf     PORTC,LSB_SRAM    ; Chargement du LSB Adresse
        bcf     PORTC,LSB_SRAM
        bcf     PORTC,CONVST      ;-- Apres 5 instruct, On lance la conversion
        bsf     PORTC,CONVST      ;-- Resultat dans 1.6 us (13 instructions)
        movlw   0xff
        movwf   DDRD
        incf    TBLPTRL,f         ; Prochaine adresse
        btfsc   ALUSTA,Carry      ; Fin du bloc ?
        goto    Break1
        nop
        movfp   Can2Cfg,PORTD     ; Commande au ADC pour prochaine conversion
        bcf     PORTC,WE_SRAM     ; Ecrire dans la SRAM
        bcf     PORTC,RD_AD       ; Lire l'échantillon
        bsf     PORTC,WE_SRAM
        bsf     PORTC,RD_AD

        bcf     PORTC,WR_AD       ; *** Conversion CANAL 2 ***
        clrf    DDRD,f            ; PORTD en sortie
Cont1   bsf     PORTC,WR_AD       ; désactiver le signal d'écriture au ADC
                                  ;-- 600 ns Track/Hold Acquisition Time
        movpf   TBLPTRL,PORTD     ; Met LSB Address sur le LATCH LSB
        bsf     PORTC,LSB_SRAM    ; Chargement du LSB Adresse
        bcf     PORTC,LSB_SRAM
        bcf     PORTC,CONVST      ;-- Apres 5 instruct, On lance la conversion
        bsf     PORTC,CONVST      ;-- Resultat dans 1.6 us (13 instructions)
        movlw   0xff
        movwf   DDRD
        incf    TBLPTRL,f         ; Prochaine adresse
        btfsc   ALUSTA,Carry      ; Fin du bloc ?
        goto    Break2
        nop
        movfp   Can3Cfg,PORTD     ; Commande au ADC pour prochaine conversion
        bcf     PORTC,WE_SRAM     ; Ecrire dans la SRAM
        bcf     PORTC,RD_AD       ; Lire l'échantillon
        bsf     PORTC,WE_SRAM
        bsf     PORTC,RD_AD

        bcf     PORTC,WR_AD       ; *** Conversion CANAL 3 ***
        clrf    DDRD,f            ; PORTD en sortie
Cont2   bsf     PORTC,WR_AD       ; désactiver le signal d'écriture au ADC
                                  ;-- 600 ns Track/Hold Acquisition Time
        movpf   TBLPTRL,PORTD     ; Met LSB Address sur le LATCH LSB
```

```
        bsf     PORTC,LSB_SRAM  ; Chargement du LSB Adresse
        bcf     PORTC,LSB_SRAM
        bcf     PORTC,CONVST    ;-- Apres 5 instruct, On lance la conversion
        bsf     PORTC,CONVST    ;-- Resultat dans 1.6 us (13 instructions)
        movlw   0xff
        movwf   DDRD
        incf    TBLPTRL,f       ; Prochaine adresse
        btfsc   ALUSTA,Carry    ; Fin du bloc ?
        goto    Break3
        nop
        movfp   Can4Cfg,PORTD   ; Commande au ADC pour prochaine conversion
        bcf     PORTC,WE_SRAM   ; Ecrire dans la SRAM
        bcf     PORTC,RD_AD     ; Lire l'échantillon
        bsf     PORTC,WE_SRAM
        bsf     PORTC,RD_AD

        bcf     PORTC,WR_AD     ; *** Conversion CANAL 4 ***
        clrf    DDRD,f          ; PORTD en sortie
Cont3   bsf     PORTC,WR_AD     ; désactiver le signal d'écriture au ADC
                                ;-- 600 ns Track/Hold Acquisition Time
        movpf   TBLPTRL,PORTD   ; Met LSB Address sur le LATCH LSB
        bsf     PORTC,LSB_SRAM  ; Chargement du LSB Adresse
        bcf     PORTC,LSB_SRAM
        bcf     PORTC,CONVST    ;-- Apres 5 instruct, On lance la conversion
        bsf     PORTC,CONVST    ;-- Resultat dans 1.6 us (13 instructions)
        movlw   0xff
        movwf   DDRD
        incf    TBLPTRL,f       ; Prochaine adresse
        btfsc   ALUSTA,Carry    ; Fin du bloc ?
        goto    Break4
        nop
        movfp   Can0Cfg,PORTD   ; Commande au ADC pour prochaine conversion
        bcf     PORTC,WE_SRAM   ; Ecrire dans la SRAM
        bcf     PORTC,RD_AD     ; Lire l'échantillon
        bsf     PORTC,WE_SRAM
        bsf     PORTC,RD_AD
        goto    Smpl Can0

Break0                          ; Fin d'un bloc de 256 Sample
        dcfsnz  CompBloc,f      ; A't-on acquis tous les blocs ?
        goto    FinLiEc         ; Oui Terminé
        incf    TBLPTRH,f       ; Non , Pointe prochain bloc.
        nop
        nop
        nop
        bcf     PORTC,WE_SRAM   ; Ecrire dans la SRAM
        bcf     PORTC,RD_AD     ; Lire l'échantillon
        bsf     PORTC,WE_SRAM
        bsf     PORTC,RD_AD
        clrf    DDRD,f          ; PORTD en sortie
        movpf   TBLPTRH,PORTD   ; Envoie l'adresse MSB du bloc actuel
        bsf     PORTC,MSB_SRAM  ; Chargement du MSB Adresse (Bloc)
        bcf     PORTC,MSB_SRAM
        bcf     PORTC,WR_AD     ; *** Conversion CANAL 1 ***
        movfp   Can1Cfg,PORTD   ; Commande au ADC
        goto    Cont0           ; et recommence pour 256 Echantillons
Break1                          ; Fin d'un bloc de 256 Sample
        dcfsnz  CompBloc,f      ; A't-on acquis tous les blocs ?
        goto    FinLiEc         ; Oui Terminé
        incf    TBLPTRH,f       ; Non , Pointe prochain bloc.
```

```
        nop
        nop
        nop
        nop
        bcf     PORTC,WE_SRAM   ; Ecrire dans la SRAM
        bcf     PORTC,RD_AD     ; Lire l'échantillon
        bsf     PORTC,WE_SRAM
        bsf     PORTC,RD_AD
        clrf    DDRD,f          ; PORTD en sortie
        movpf   TBLPTRH,PORTD   ; Envoie l'adresse MSB du bloc actuel
        bsf     PORTC,MSB_SRAM  ; Chargement du MSB Adresse (Bloc)
        bcf     PORTC,MSB_SRAM
        bcf     PORTC,WR_AD     ; *** Conversion CANAL 2 ***
        movfp   Can2Cfg,PORTD   ; Commande au ADC
        goto    Cont1           ; et recommence pour 256 Echantillons
Break2                          ; Fin d'un bloc de 256 Sample
        dcfsnz  CompBloc,f      ; A't-on acquis tous les blocs ?
        goto    FinLiEc         ; Oui Terminé
        incf    TBLPTRH,f       ; Non , Pointe prochain bloc.
        nop
        nop
        nop
        nop
        bcf     PORTC,WE_SRAM   ; Ecrire dans la SRAM
        bcf     PORTC,RD_AD     ; Lire l'échantillon
        bsf     PORTC,WE_SRAM
        bsf     PORTC,RD_AD
        clrf    DDRD,f          ; PORTD en sortie
        movpf   TBLPTRH,PORTD   ; Envoie l'adresse MSB du bloc actuel
        bsf     PORTC,MSB_SRAM  ; Chargement du MSB Adresse (Bloc)
        bcf     PORTC,MSB_SRAM
        bcf     PORTC,WR_AD     ; *** Conversion CANAL 3 ***
        movfp   Can3Cfg,PORTD   ; Commande au ADC
        goto    Cont2           ; et recommence pour 256 Echantillons
Break3                          ; Fin d'un bloc de 256 Sample
        dcfsnz  CompBloc,f      ; A't-on acquis tous les blocs ?
        goto    FinLiEc         ; Oui Terminé
        incf    TBLPTRH,f       ; Non , Pointe prochain bloc.
        nop
        nop
        nop
        nop
        bcf     PORTC,WE_SRAM   ; Ecrire dans la SRAM
        bcf     PORTC,RD_AD     ; Lire l'échantillon
        bsf     PORTC,WE_SRAM
        bsf     PORTC,RD_AD
        clrf    DDRD,f          ; PORTD en sortie
        movpf   TBLPTRH,PORTD   ; Envoie l'adresse MSB du bloc actuel
        bsf     PORTC,MSB_SRAM  ; Chargement du MSB Adresse (Bloc)
        bcf     PORTC,MSB_SRAM
        bcf     PORTC,WR_AD     ; *** Conversion CANAL 4 ***
        movfp   Can4Cfg,PORTD   ; Commande au ADC
        goto    Cont3           ; et recommence pour 256 Echantillons
Break4                          ; Fin d'un bloc de 256 Sample
        dcfsnz  CompBloc,f      ; A't-on acquis tous les blocs ?
        goto    FinLiEc         ; Oui Terminé
        incf    TBLPTRH,f       ; Non , Pointe prochain bloc.
        nop
        nop
        nop
        nop
```

```
        bcf     PORTC,WE_SRAM   ; Ecrire dans la SRAM
        bcf     PORTC,RD_AD     ; Lire l'échantillon
        bsf     PORTC,WE_SRAM
        bsf     PORTC,RD_AD
        clrf    DDRD,f          ; PORTD en sortie
        movpf   TBLPTRH,PORTD   ; Envoie l'adresse MSB du bloc actuel
        bsf     PORTC,MSB_SRAM  ; Chargement du MSB Adresse (Bloc)
        bcf     PORTC,MSB_SRAM
        bcf     PORTC,WR_AD     ; *** Conversion CANAL 0 ***
        movfp   Can0Cfg,PORTD   ; Commande au ADC
        goto    Cont4           ; et recommence pour 256 Echantillons

FinLiEc goto    MenuLoop        ; Terminé
;*****************************************************************************
*
; Dump une partie de la memoire en format d'affichage pour l'écran
; Seulement deux blocs de 256 Échantillons.
;*****************************************************************************
*
DumpMem
        clrf    TBLPTRH,f       ; Pointe au debut de la SRAM
        clrf    TBLPTRL,f
Lp_dump
        call    RdRAM           ; Lecture SRAM 12 bits
        movlb   0               ; MSB_Data_RAM, ByteHEX  Bank 0
        movfp   MSB_Data_RAM,WREG
        movwf   ByteHEX
        call    AffByte
        movlb   0               ; LSB_Data_RAM, ByteHEX , Bank 0
        movfp   LSB_Data_RAM,WREG
        movwf   ByteHEX
        call    AffByte
        movlw   " "
        movwf   SerOutData
        call    Send_Char
        incf    TBLPTRL,f       ; Prochaine adresse
        btfss   ALUSTA,Carry    ; Bit de retenue ?
        goto    Lp_dump         ; Non, Boucle
        incf    TBLPTRH,f       ; Oui, Ajoute bit de retenue sur MSB
        movlw   0x02            ; Adresse_RAM (MSB) = ?
        cpfseq  TBLPTRH
        goto    Lp_dump
                                ; Terminé ! le dump
        goto    MenuLoop
;*****************************************************************************
*
; Dump le contenu memoire et déclenche l'écriture du fichier au terminal
; '%' Caractère de début du fichier
; '&' Caractere de fin du ficher
;*****************************************************************************
*
DumpFich
        clrf    TBLPTRH,f       ; Pointe au debut de la SRAM
        clrf    TBLPTRL,f
        movlw   "%"             ; Envoie caractere de debut de fichier
        movwf   SerOutData
        call    Send_Char
LpDFich
        call    RdRAM           ; Lecture SRAM 12 bits
        movlb   0               ; MSB_Data_RAM, ByteHEX  Bank 0
        movfp   MSB_Data_RAM,WREG
```

```
        movwf   ByteHEX             ; Envoie MSB echantillon au terminal
        call    AffByte
        movlb   0                   ; LSB_Data_RAM, ByteHEX , Bank 0
        movfp   LSB_Data_RAM,WREG
        movwf   ByteHEX             ; Envoie LSB au terminal
        call    AffByte
        incf    TBLPTRL,f           ; Prochaine adresse
        btfss   ALUSTA,Carry        ; Bit de retenue ?
        goto    LpDFich             ; Non, Boucle
        incf    TBLPTRH,f           ; Oui, Ajoute bit de retenue sur MSB
        movlw   0x0E                ; *****  14 eme blocs ? *****
        cpfseq  TBLPTRH
        goto    LpDFich
                                    ; On a terminé l'envoie des échantillons
        movlw   "&"                 ; Envoie caractere de fin de fichier
        movwf   SerOutData
        call    Send_Char
        goto    MenuLoop
;*****************************************************************************
*
; Routine d'affichage de chaine de caracter
; Input : TBLPTRL, TBLPTRH
; Routine appelé : Send_Char
; Note. Le caractere '$' determine la fin de la chaine de caracteres
;*****************************************************************************
*
Aff_Chaine
ChainLp movlb   0                   ; SerOutData is in bank 0
        tablrd  0,0,SerOutData      ; load latch
        tlrd    1,SerOutData        ; load high byte
        tstfsz  SerOutData          ; Caractere de fin de chaine ?
        goto    SuitCh1
        goto    FinChain
SuitCh1 call    Send_Char
        tablrd  0,1,SerOutData      ; load low byte and increment TBLPTR
        tstfsz  SerOutData          ; Caractere de fin de chaine ?
        goto    SuitCh2
        goto    FinChain
SuitCh2 call    Send_Char
;       incf    TBLPTRL,f           ; Prochain doublet de caracteres
;       btfss   0,ALUSTA            ; Retenue ?
;       goto    PasRetCh
;       incf    TBLPTRH,f           ; Oui on incremente MSB
;PasRetCh
        goto    ChainLp
FinChain
        return
;*****************************************************************************
*
; Routine d'affichage d'un Byte en Hexadécimal
; Input  : ByteHEX (Byte que l'on veut afficher à l'écran)
; Output : Byte affiché en Hexadécimal envoyé  sur le port série
;*****************************************************************************
*
AffByte movlw   0xF0                ; Lire le quartet le plus significatif
        movlb   0                   ; ByteHEX, BinAscV et SerOutData => Banque 0
        andwf   ByteHEX,WREG
        rrncf   WREG,w
        rrncf   WREG,w
        rrncf   WREG,w
        rrncf   WREG,w
```

```
        movpf   WREG,BinAscV      ; Convertir le quartet en ASCII
        call    BinASCII
        movfp   BinAscV,WREG
        movpf   WREG,SerOutData ; Envoyer à l'écran
        call    Send_Char
        movlw   0x0F              ; Lire le quartet le moins significatif
        andwf   ByteHEX,WREG
        movpf   WREG,BinAscV      ; Convertir le quartet en ASCII
        call    BinASCII
        movfp   BinAscV,WREG
        movpf   WREG,SerOutData ; Envoyer à l'écran
        call    Send_Char
        return
;*****************************************************************************
*
; Routine de conversion d'un quartet binaire en sa valeur ASCII
; Input  : BinAscV, Valeur binaire de 4 bits à convertir en ASCII
; Output : BinAscV, Résultat 8 bits du code ASCII.
;*****************************************************************************
*
BinASCII
        movlb   0                 ; BinAscV => Banque 0
        movlw   0x0A
        cpfslt  BinAscV
        goto    Tr_A_F
        movlw   0x30              ; Valeur de 0-9
        addwf   BinAscV,f         ; Convertie en ASCII
        goto    FinCAsc
Tr_A_F  movlw   0x37              ; Valeur de A-F
        addwf   BinAscV,f         ; Convertie en ASCII
FinCAsc return
;*****************************************************************************
*
; Routine d'envoie d'un caractère sur l'UART
; Input : SerOutData
;*****************************************************************************
*
Send_Char
        movlb   1                 ; PIR is in bank 1
PollTXIF
        btfss   PIR,1             ; chek the TXIF bit of PIR register in bank 1
        goto    PollTXIF          ; loop until char transmitted, assume WDT id
        off
        movlb   0                 ; TXREG is in bank 0
        movfp   SerOutData,TXREG
        return
;*****************************************************************************
*
; Routine qui attend et lis un caractère sur l'UART
; Output : SerInData
;*****************************************************************************
*
Get_Char
        movlb   1                 ; PIR is in bank 1
PollRCIF
        btfss   PIR,0             ; chek the RCIF bit of PIR register in bank 1
        goto    PollRCIF          ; loop until char received
        movlb   0                 ; RCREG is in bank 0
        movpf   RCREG,SerInData
        return
;*****************************************************************************
```

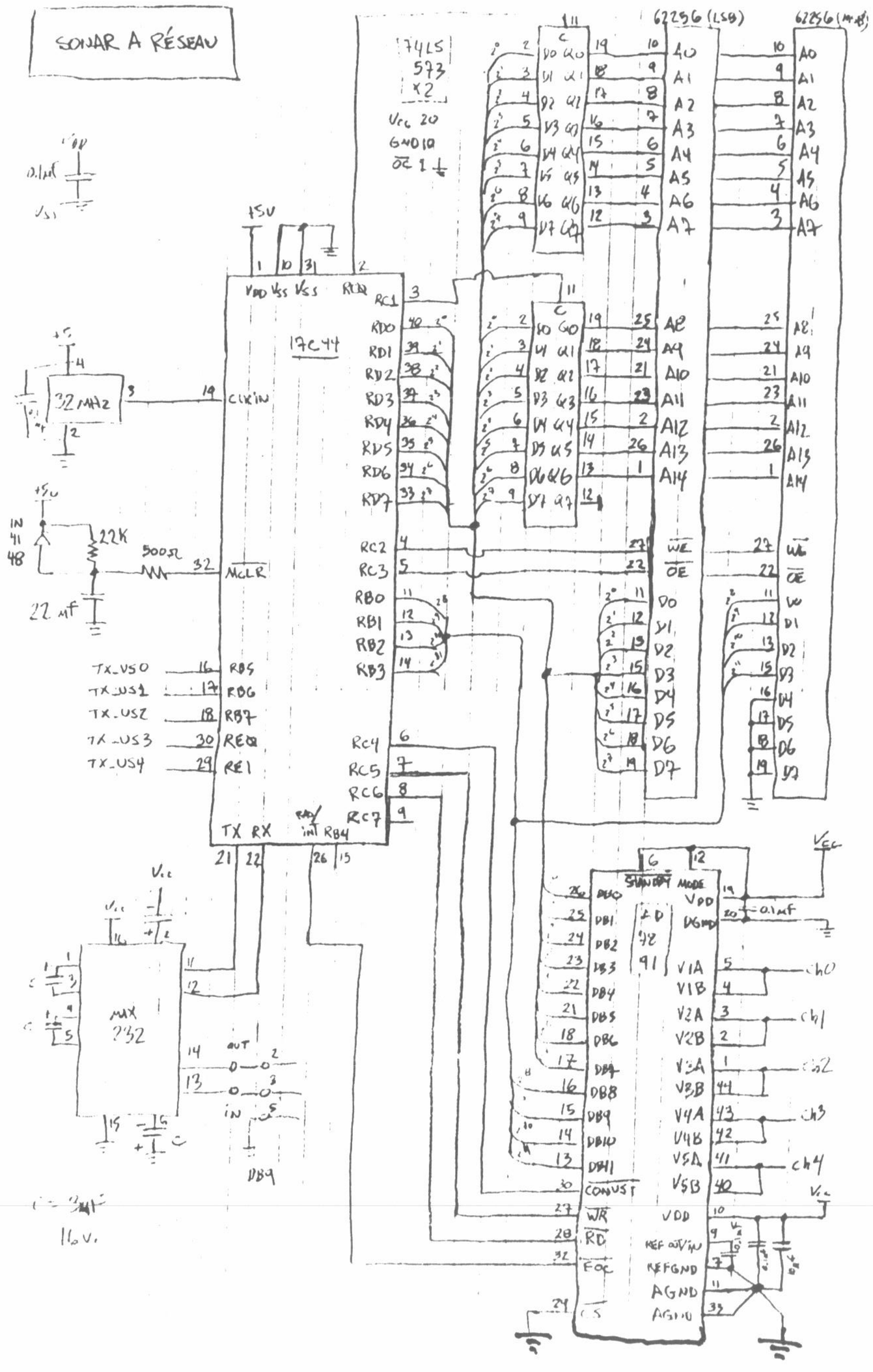
```
*
; Routine Conversion Minuscule à majuscule
; Entree : SerInData
; Sortie : SerInData
;*****************************************************************************
*
CnvMaj
                return
;*****************************************************************************
*
; Routine d'écriture d'un byte dans la SRAM
; Entree :      LSB_Data_RAM = Byte LSB à écrire en RAM
;               MSB_Data_RAM = Byte MSB à écrire en RAM
;               tblptrl = Adressé LSB
;               tblptrh = Adresse MSB
;*****************************************************************************
*
WrRAM
        movlb   1                       ; DDRD and PORTD, PORTC are in Bank 1
        movfp   TBLPTRL,WREG            ; Met LSB Address sur le LATCH LSB
        movpf   WREG,PORTD
        movlw   0                       ; Put PORTD in output
        movwf   DDRD
        bsf     PORTC,LSB_SRAM          ; Chargement du LSB Adresse
        bcf     PORTC,LSB_SRAM
        movfp   TBLPTRH,WREG            ; Met MSB Address sur le LATCH MSB
        movpf   WREG,PORTD
        bsf     PORTC,MSB_SRAM          ; Chargement du MSB Adresse
        bcf     PORTC,MSB_SRAM
        movlb   0                       ; LSB_Data_RAM  Bank 0
        movfp   LSB_Data_RAM,WREG        ; Met LSB_Data sur l'entrée de la
SRAM
        movlb   1                       ; PORTD Bank 0
        movpf   WREG,PORTD
        movlb   0                       ; MSB_Data_RAM,PORTB,DDRB  Bank 0
        movfp   MSB_Data_RAM,WREG
        movwf   PORTB
        movlw   0xF0                    ; PORTB 3-0 en sortie  0=Output,
1=input
        movwf   DDRB
        movlb   1                       ; PORTC,DDRD Bank1
        bcf     PORTC,WE_SRAM           ; Ecriture dans la SRAM
        nop
        nop
        bsf     PORTC,WE_SRAM
        clrf    DDRD,1                  ; PORTD en entrée
        movlb   0                       ; DDRB Bank 0
        movlw   0xFF                    ; PORTB 3-0 en entree  0=Output,
1=input
        movwf   DDRB
        return
;*****************************************************************************
*
; Routine de lecture d'un byte dans la SRAM
; Entree :      tblptrl = Adresse LSB
;               tblptrh = Adresse MSB
; Sortie :      MSB_Data_RAM = MSB Byte lue en RAM
;               LSB_Data_RAM = LSB Byte lue en RAM
;*****************************************************************************
*
```

```
RdRAM
        movlb   1                   ; DDRD,PORTD,PORTC are in Bank 1
        movfp   TBLPTRL,WREG        ; Met LSB Address sur le LATCH LSB
        movpf   WREG,PORTD
        movlw   0                   ; Put PORTD in output
        movwf   DDRD
        bsf     PORTC,LSB_SRAM      ; Chargement du LSB Adresse
        bcf     PORTC,LSB_SRAM
        movfp   TBLPTRH,WREG        ; Met MSB Address sur le LATCH MSB
        movpf   WREG,PORTD
        bsf     PORTC,MSB_SRAM      ; Chargement du MSB Adresse
        bcf     PORTC,MSB_SRAM
        movlw   0xFF                ; Put PORTD in input
        movwf   DDRD
        movlb   0                   ; DDRB is in Bank 0
        movlw   0xFF                ; PORTB 3-0 en entree  0=Output, 1=input
        movwf   DDRB
        movlb   1                   ; PORTC,PORTD, Bank 1
        bcf     PORTC,OE_SRAM       ; Output Enable SRAM
        nop
        nop
        nop
        movfp   PORTD,WREG          ; Lis LSB data
        movlb   0                   ; LSB_Data_RAM,PORTB,MSB_Data_RAM  Bank 0
        movwf   LSB_Data_RAM
        movpf   PORTB,WREG          ; Lis MSB data
        andlw   0x0F
        movwf   MSB_Data_RAM
        movlb   1                   ; PORTC  Bank 1
        bsf     PORTC,OE_SRAM       ; Disable sortie de la SRAM
        return
;*****************************************************************************
*
; Routine de delais
;*****************************************************************************
*
wait    movlb   0       ; loopint et loopext sont dans la banque 0
        movlw   0x50
        movwf   loopext ; initialise la boucle externe
top     movlw   0xff    ; initialise la boucle interne
        movwf   loopint
top2    nop
        nop
        nop
        nop
        nop
        decfsz  loopint ; boucle interne complete ?
        goto    top2    ; non, boucle encore
        decfsz  loopext ; boucle externe complete ?
        goto    top     ; non, boucle encore
        retlw   0       ; oui, retour au programme appelant
;*****************************************************************************
*
; Handler de gestion des interrupt de la ligne INT
;*****************************************************************************
*
Interupt
        retfie
;*****************************************************************************
*
; Messages pour communication serie
```

```
;*****************************************************************************
*
NextLine        DATA    "\n\r\x00"
MessBoot        DATA    "\n\r"
                DATA    "Systeme sonar \n\r"
                DATA    "Version 1.1 \n\r"
                DATA    "\n\r\x00"
RamTMes         DATA    "RAM Test 384 KBits \n\r\x00"
RamPass         DATA    "\n\rRAM Test Correct \n\r\x00"
RamFail         DATA    "\n\rRAM Test Echec \n\r\n\r\x00"
MenuTxt         DATA    "Commande    Description \n\r"
                DATA    "--------    ----------- \n\r"
                DATA    "   A        Afficher Menu d'aide\n\r"
                DATA    "   M        Modifier no du Canal A/D\n\r"
                DATA    "   C        Afficher no du Canal A/D\n\r"
                DATA    "   L        Lire un echantillon \n\r"
                DATA    "   D        Dump Ecran\n\r"
                DATA    "   F        Dump Fichier \n\r\x00"
                DATA    "\n\r\x00"
EntreCan        DATA    "Entrer le No de Canal (0-7) :  \x00"
        END
```

ANNEXE L

Shémas électronique du système d'acquisition d'écho du sonar.


SONAR A RÉSEAU
74LS 573 x2
62256 (LSB)
62256 (MSB)
17C44
32 MHz
22K
500Ω
22 µf
MCLR
CLKIN
TX_US0
TX_US1
TX_US2
TX_US3
TX_US4
TX RX
MAX 232
STANDBY MODE
CONVST
WR
RD
EOC
CS
REFGND
AGND
DB9