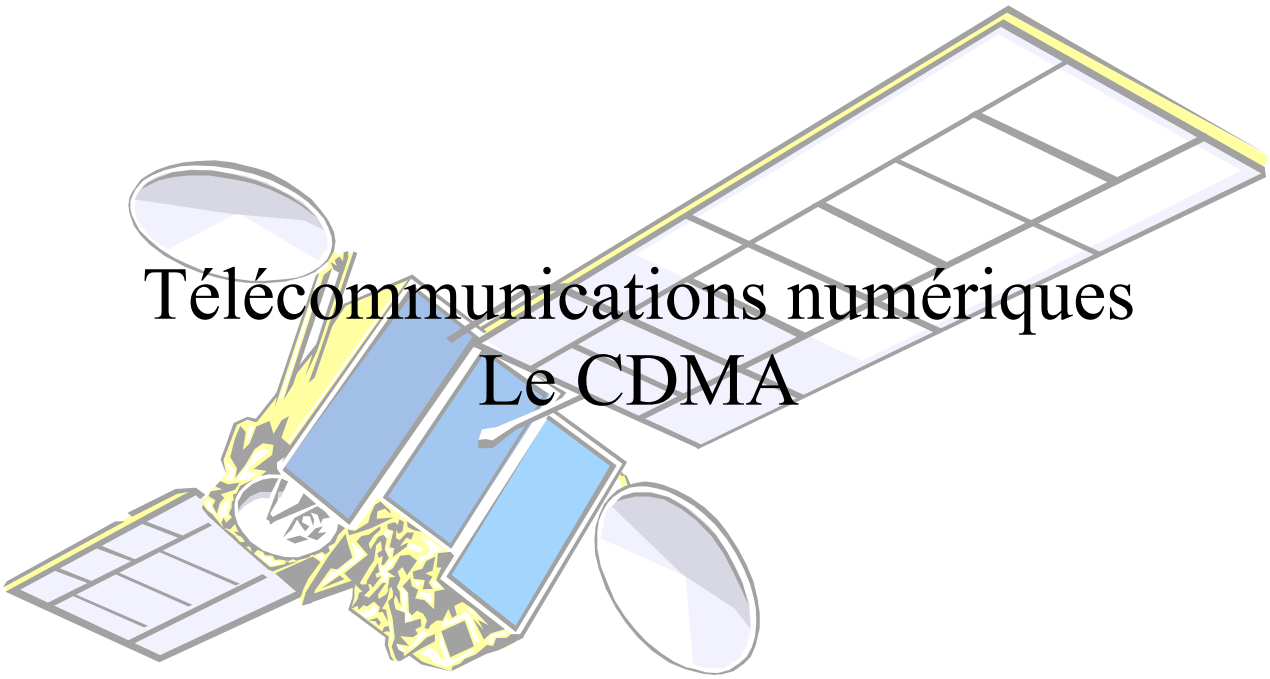
Année 2000-2001 MODEX



Télécommunications numériques
Le CDMA

Jeremy LAINE & Philippe VIGNERON

Sommaire

III. Implémentation du CDMA: le logiciel CDMAster

Introduction

L'aventure des télécommunications a commencé avec l'invention du télégraphe. La téléphonie a fait ensuite son apparition. On transportait alors la voix humaine de manière analogique. La troisième révolution industrielle correspond à l'avènement de l'informatique et l'expansion simultanée des télécommunications. Avec la technologie du numérique transporter de la voix, de l'image ou des données informatiques relève du même procédé. Dès lors qu'il existe un canal de communication entre deux utilisateurs, ils peuvent échanger les données qu'ils désirent.

Ce canal de communication peut revêtir différentes formes : le traditionnel fils de cuivre pour la téléphonie fixe les ondes hertziennes pour la téléphonie mobile ou les fibres optiques… Il n'est généralement pas possible d'attribuer à chaque utilisateur son propre canal, d'autant plus que le nombre de personnes interconnectées ne cesse de croître. Il est donc absolument indispensable d'avoir recours au multiplexage. Le CDMA, pur produit de l'ère numérique, est une des méthodes les plus utilisées en ce moment. Il fait, de plus, l'objet de nombreuses études visant à l'optimiser, notamment dans le domaine de la téléphonie cellulaire.

Nous avons fait une étude théorique et expérimentale de cette technologie. La théorie nous a permis de comprendre les avantages qu'elle apportait. L'expérience consistait à réaliser la transmission d'information multiplexée avec le CDMA. C'est par le biais d'un ordinateur que nous avons pu réaliser cette communication.

Après une présentation générale des télécommunications nous aborderons les aspects théoriques du CDMA. Nous verrons alors le logiciel réalisant le multiplexage/démultiplexage que nous avons mis au point et un exemple de son utilisation.

I

-

Généralités sur les télécommunications

1. Télécommunications, principes

1.1. Les systèmes de communication

L'objectif des télécommunications est de permettre l'échange, à distance, d'informations. Tout système de télécommunication, qu'il soit numérique ou analogique, qu'il transporte de la voix, de l'image ou de quelconques données sous forme de bits, peut se résumer par le schéma ci-dessous (**fig1**).

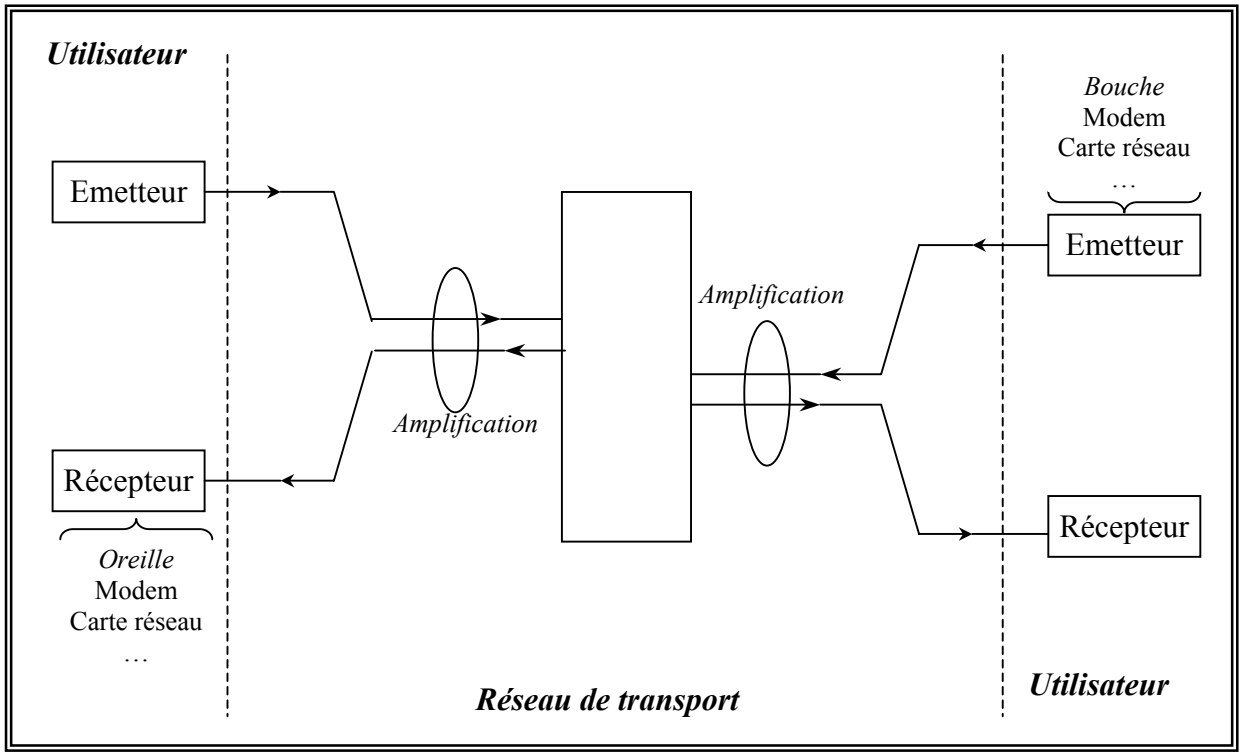


Fig1. *Un système de communication*

1.2. La téléphonie analogique

a) Les fréquences transmises

Considérons plus particulièrement le cas de la téléphonie analogique. Le récepteur est bien entendu l'oreille. Sachant que l'homme ne peut entendre que des sons compris entre 20Hz et 20kHz, on se contente a priori de ne transporter que dans signaux qui se situe dans cette gamme de fréquence. Cependant, le téléphone sert essentiellement à la conversation (on écoute que rarement de l'opéra par le biais d'un téléphone !), on se limite alors aux **fréquences comprises entre 300 et 3400Hz**.

b) Les fonctions à assurer

Pour que la communication ait lieu, il faut qu'un certain nombre de fonctions soient assurées : les fils ne servent pas qu'à transporter de la voix. Dans un premier temps, il faut alimenter le téléphone (48V). Pour rentrer en contact avec un correspondant, il faut pouvoir indiquer au central les « coordonnées » de cette personne.

Dans un premier temps il y avait des opératrices. Mais des commutateurs automatiques ont fait leur apparition. Les fils doivent alors servir à transporter ces « coordonnées ». Le premier système (par impulsion) générait de brève coupure de la ligne (une pour le 1, deux pour le 2…). Avec l'utilisation de commutateur numérique, à chaque numéro correspond une tonalité : somme de deux fonctions sinusoïdales de fréquences propres. C'est le réseau qui assure le transport de ce signal.

Il faut ensuite pouvoir déclencher la sonnerie. Pour cela, un signal de 50Hz indique qu'une personne souhaite rentrer en communication. La ligne assure aussi le transport d'une signalisation de l'état du réseau : 440Hz pour dire que la ligne est disponible ; « sonnerie occupée »… Tous ces signaux doivent être transporter sur les mêmes fils qui servent au transport de la conversation.

c) Le convertisseur 4 fils /2 fils

A priori, pour l'émission comme pour la réception, on a besoin de 2 fils à chaque fois, soit 4 fils en tout. Or les réseaux de communication s'étendent la plupart du temps sur de grandes distances et le matériau utilisé, le cuivre, est assez coûteux. De plus, plus il y a de fils, plus les branchements au central sont complexes. Il serait donc intéressant de pouvoir diviser par 2 le nombre de fils nécessaires au transport des communications. C'est ce que réalise le convertisseur 4 fils / 2 fils (**fig2**).

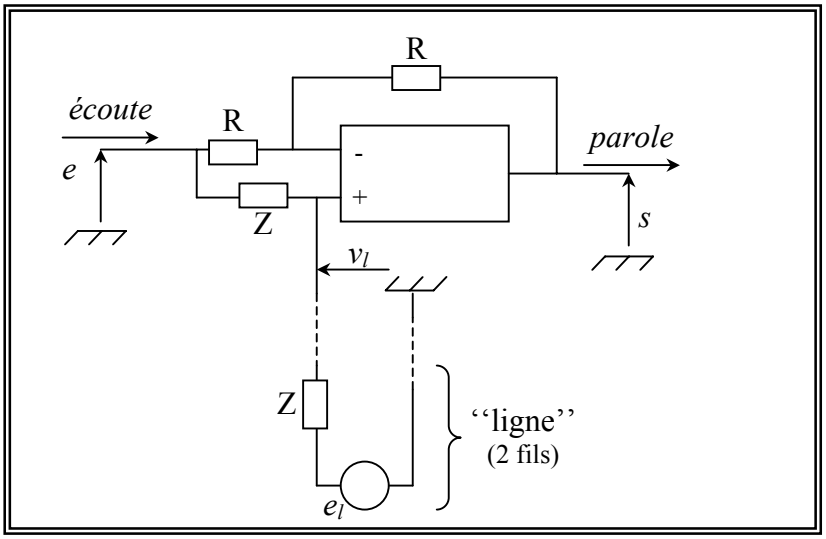


Fig2.
Convertisseur 4 fils / 2 fils

2. Communications numériques

2.1. Principe de fonctionnement

Pour la téléphonie numérique, les principes fondamentaux restent les mêmes. Ce système possède de nombreux avantages sur la téléphonie analogique : meilleur filtrage du bruit, compatibilité avec la transmission de données informatiques… Cette fois, le signal n'est pas transmis directement mais il est d'abord numérisé. Etant donné que l'on se limite à des fréquences inférieures à 4000Hz, d'après le théorème de Shannon, on peut prendre une fréquence d'échantillonnage de 8000Hz. Chaque échantillon est codé sur 8 bits, ce qui donne un débit de 64kbits/s.

2.2. Intérêt du numérique face à l'analogique

Le grand intérêt est la qualité de la transmission vocale. En effet, une conversation n'a pas de propriété particulière ce qui rend difficile la distinction entre le bruit et le signal désiré lors de la réception. Par contre un signal numérique possède des caractéristiques précises. Par exemple, la régularité dans le temps de l'envoi de paquets de bits permet de corriger certaines erreurs : quand on observe deux variations de 0 à 1 trop proches, on supprime la moins probable… Ceci permet de retrouver quasiment un signal identique entre l'émission et la réception. De plus de systèmes de correction d'erreurs minimise encore les déformations.

En télécommunication numérique, on peut également implémenter des algorithmes de compression de données et des méthodes de multiplexage plus performantes et ainsi optimiser l'usage de la bande passante disponible. C'est un autre avantage qui justifie le remplacement de l'analogique par le numérique.

Enfin, le passage au numérique permet d'assurer un meilleur interfaçage avec les données issues d'un ordinateur. Ainsi, avec un même appareil, on peut à la fois converser et s'échanger des fichiers, des images… C'est donc une force supplémentaire du numérique face à l'analogique.

3. Travail réalisé en laboratoire

3.1. Téléphonie analogique

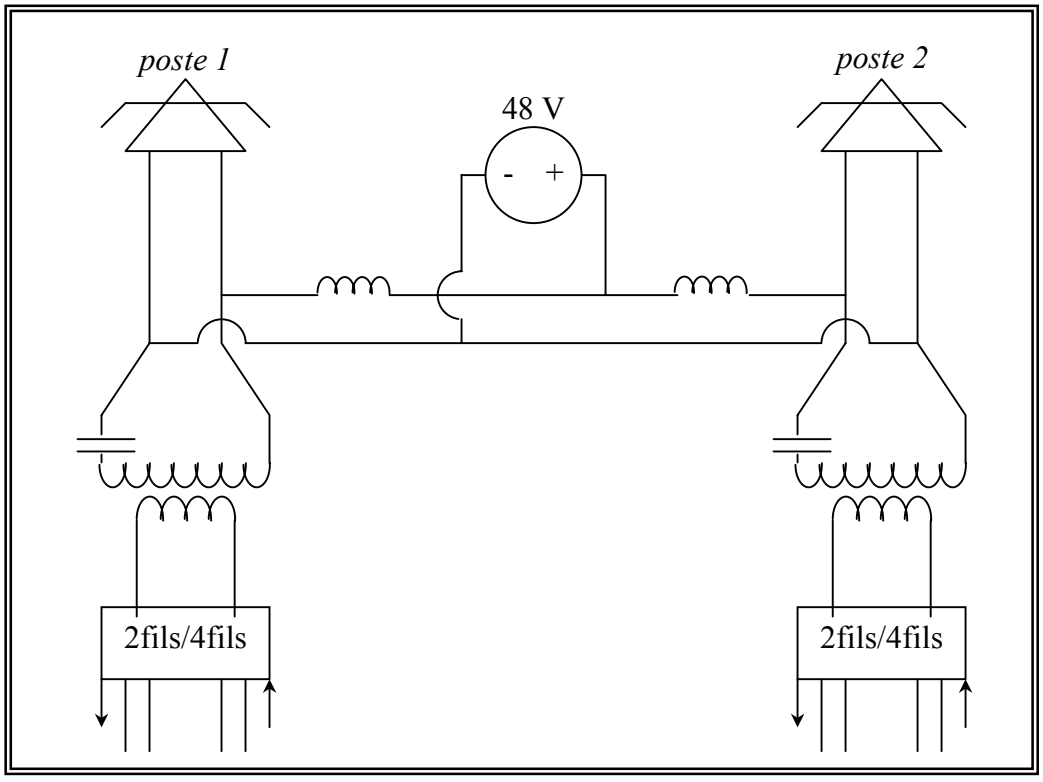


Fig3. *Une première ligne téléphonique*

Le premier travail fut de réaliser la ligne téléphonique la plus simple : celle qui connecte deux utilisateurs (**fig3**). Il n'était donc pas nécessaire de réaliser un système de commutation. Notre montage se constituait d'une alimentation commune aux deux postes et d'un convertisseur 2fils/4fils pour chaque poste. Ce dernier a posé quelques problèmes de mise en

œuvre, car, comme on peut le voir figure 1, les deux impédances Z doivent avoir la même valeur. Or l'impédance de la ligne n'est pas connue car c'est la résistance interne du poste. Il a donc fallu utiliser un potentiomètre que l'on a réglé en minimisant la valeur de *s* lorsque que personne ne parlait.

Le résultat fut satisfaisant, nous avons pu alors aborder le problème de la numérisation.

3.2. Téléphonie numérique

A la sortie du convertisseur 2fils/4fils (**fig3**), nous avons implémenté un convertisseur analogique numérique (CAN) travaillant avec une fréquence d'échantillonnage de 8kHz. On transmet alors les données résultant de cette numérisation. A la réception, un convertisseur numérique analogique (CNA) retransforme les données en un signal analogique. Celui-ci est alors traité par un filtre passe-bas pour éliminer les fréquences supérieures à 4 kHz. Le dispositif est résumé sur la **figure 4**.

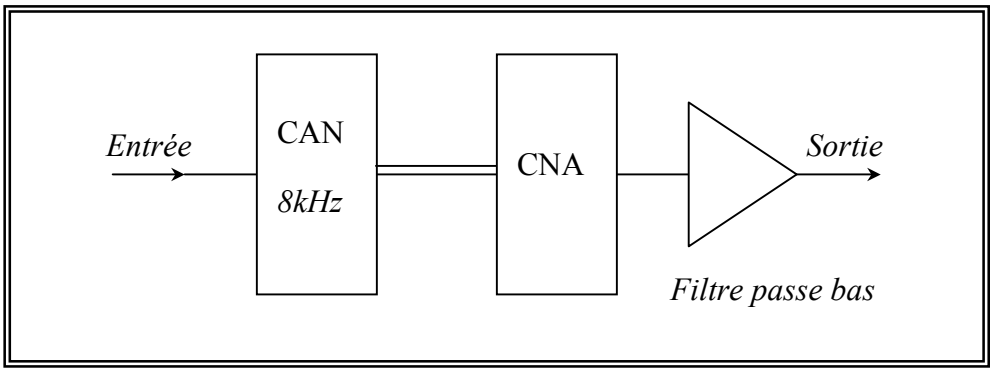


Fig4. *numérisation*

Pour continuer à utiliser le minimum de fils, il faut envoyer en série les bits qui codent un échantillon. Ceci est réalisé en intercalant un système de sérialisation / désérialisation utilisant un registre à décalage.

La numérisation/sérialisation se faisant en plusieurs étapes, des problèmes d'horloges peuvent se poser. Nous avons envisagé d'utiliser une PLL (phase-locked loop) afin de rendre indépendantes les horloges d'émission et de réception en extrayant l'horloge du signal. Il faut cependant déterminer expérimentalement le délai à introduire pour s'affranchir du décalage introduit entre la conversion et la sérialisation.

Toutefois, n'ayant pas besoin de ce montage précis dans le cadre de notre projet, nous n'avons pas cherché à régler ces problèmes d'horloge. Nous avons préféré consacrer plus de temps à notre implémentation du CDMA. Comme nous le verrons par la suite (cf. III) nous avons été amenés à mettre en place un mécanisme de synchronisation dans le cadre de ce projet.

II

-

Multiplexage, CDMA

L'objet du multiplexage est la transmission de plusieurs communications sur un même canal (**Fig5**). Il existe plusieurs méthodes. Nous avons choisi d'étudier plus particulièrement le CDMA (Code Division Multiple Access). Ce qui a motivé notre choix est le fait que cette technologie est actuellement la plus en vogue car elle possède certains avantages. Nous aborderons cette question à la fin de ce chapitre.

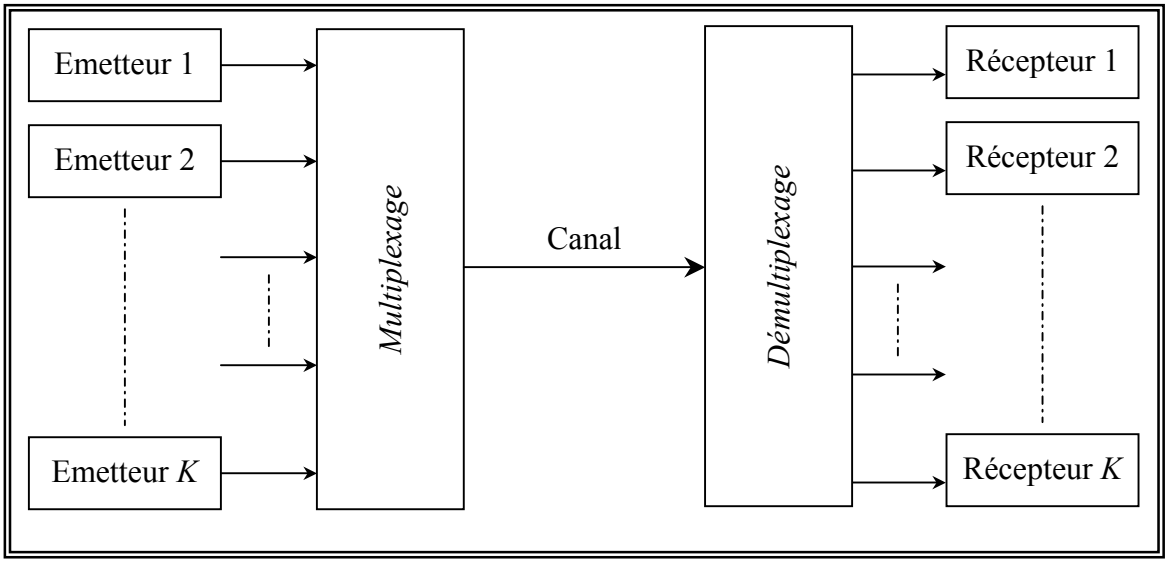


Fig5.*Un système multiplexé*

1. Différents types de multiplexage

1.1. Multiplexage temporel : TDMA

Le TDMA (Time Division Multiple Access) ou multiplexage temporel utilise le fait que la fréquence d'envoi des bits d'information est plus faible que la capacité du réseau (**fig9**). On peut donc profiter des « temps morts » pour transmettre une deuxième communication.

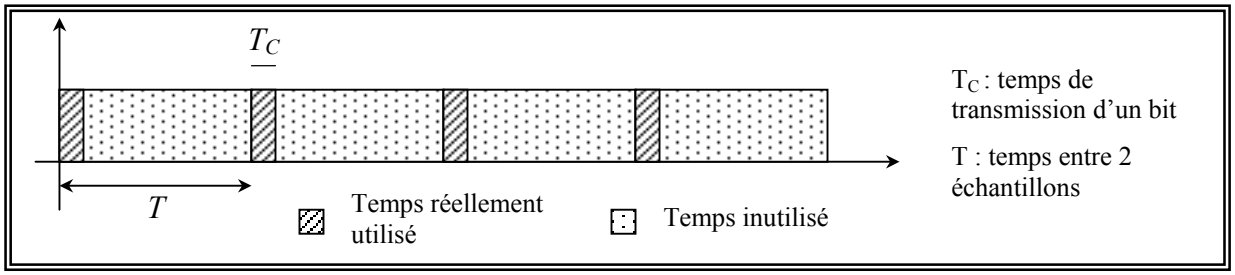


Fig9. *non-optimisation des capacités du réseau*

Dans le cas du schéma ci-dessus, on peut transmettre jusqu'à 8 communications à la fois (**fig10**). A la réception, il suffit d'utiliser un commutateur qui passe d'un utilisateur au suivant tous les T_C (**fig11**).

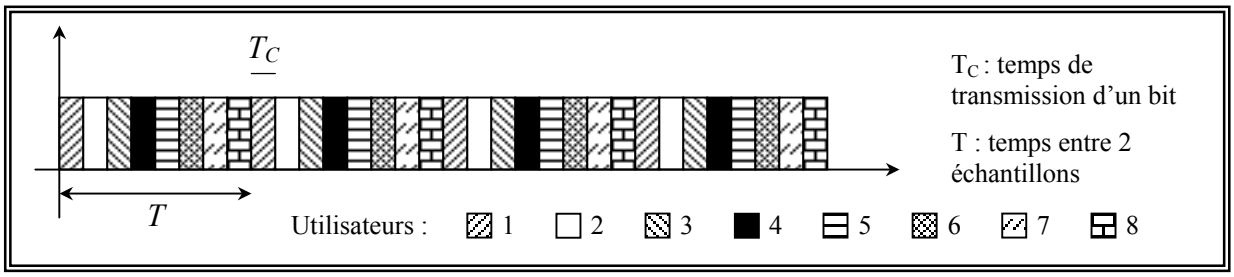


Fig10. *multiplexage temporel*

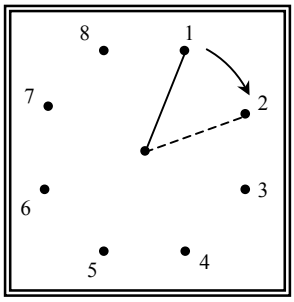


Fig11 *découplage à la réception*

1.2. Multiplexage fréquentiel : FDMA

Le FDMA (Frequency Division Multiple Access) consiste à diviser la bande passante du canal en *K* (nombre d'utilisateurs) bandes de fréquences d'intersection nulle. Cette méthode est illustrée **fig12**. Il faut donc translater les différents utilisateurs sur ces bandes de fréquences.

Supposons que la bande de fréquence occupée par un utilisateur seul soit [0,*F*]. On peut pour simplifier considérer un signal du type $s_i = A_i \cos(2\pi f_i t)$, avec $f_i<F$. Si on le multiplie par $2\cos(2\pi v_i t)$, on obtient alors un signal $S_i = A_i \cos[2\pi(f_i + v_i)t] + A_i \cos[2\pi(f_i - v_i)t]$ qui (en éliminant le deuxième terme par filtrage) est le signal s_i translaté sur la fréquence v_i. Donc, répétant cette opération pour chaque utilisateur i, avec la loi $v_i = i \cdot F$, on réalise le multiplexage désiré.

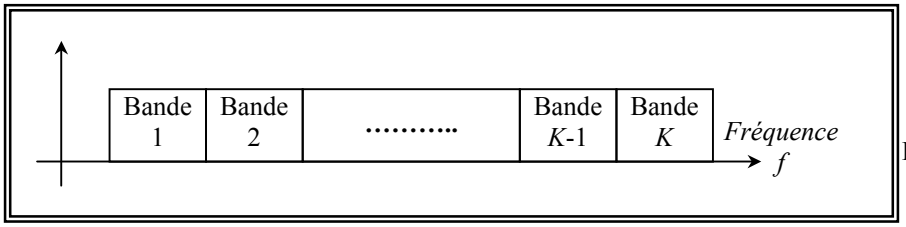


Fig12. *multiplexage fréquentiel*

1.3. Multiplexage codé : CDMA

Le CDMA appartient à la classe des multiplexages dits à étalement de spectre. En effet, comme nous allons le voir, chaque utilisateur émet sur toute la largeur de bande du canal de communication. Le principe est le suivant : à chaque utilisateur correspond une clé (ou code) à l'aide de laquelle son message est codé avant d'être émis.

Il existe deux principales variétés de CDMA :

- FH-CDMA (Frequency Hop). Dans ce système, on fait de l'évasion de fréquence : la clé de chaque utilisateur code pour une suite de fréquences qui feront alternativement office de porteuse. Ce système ressemble à un multiplexage fréquentiel dans lequel l'attribution des fréquences varierait rapidement (par rapport au débit d'informations à transmettre).
- DS-CDMA (Direct Sequence). C'est à ce type de CDMA qu'on fait généralement référence quand on parle de CDMA, et c'est celui que nous avons étudié aussi bien théoriquement qu'expérimentalement. Ici, on multiplie directement le message à transmettre par une le code (séquence pseudo-aléatoire). L'étalement spectral du signal codé vient de ce que la fréquence du code est largement supérieure à la fréquence d'envoi des données.

2. CDMA : émission

Comme toujours en communication, on commence par la transmission du signal. On se place dans la situation suivante : K utilisateurs souhaitent transmettre des informations via un même câble. Chaque information est modélisée par une suite de ±1 : $\mathbf{b}_k = [b_k(1), \ldots, b_k(N)]$. On désigne par k le $k^{ème}$ utilisateur.

Le principe du CDMA consiste en l'utilisation de codes. Chacun utilise un code propre, de la forme(**Fig6**) :

$$g_k(t) = \sum_{n=0}^{L-1} a_k(n)\, p(t - nT_c), \quad 0 \le t \le T$$

$\{a_k\}$ est un « pseudo-noise (PN) code sequence », chaque a_k vaut ±1.
$p(t)$ est une pulsation de durée T_c. On a donc $T=LT_c$.

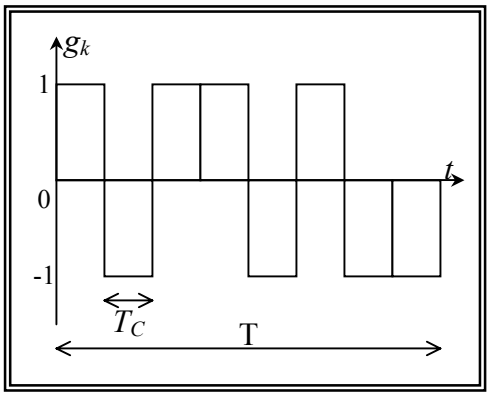


Fig6. *Exemple de code* $\{a_k\}=\{1,-1,1,1,-1,1,-1,-1\}$

On peut donc considérer les g_k comme des vecteurs de $\{-1,1\}^L$.

On considère pour la suite (afin d'alléger les notations) que l'on normalise les g_k :

$$\int_0^T g_k^2(t)\, dt = 1 .$$

On définit la fonction ρ qui permet de déterminer une corrélation entre les codes g_k :

$$\rho_{ij}(\tau) = \int_0^T g_i(t) g_j(t-\tau)\, dt , \quad i \le j$$

$$\rho_{ji}(\tau) = \int_0^T g_i(t) g_j(t+T-\tau)\, dt , \quad i \le j$$

$\rho_{ij}(0) = \delta_{i,j}$, dans le cas de codes orthogonaux.

On multiplexe les informations en les combinant chacune avec un 'vecteur' g_k. Ainsi, le paquet de bits de longueur N : $\mathbf{b}_k=[b_k(1), \ldots, b_k(\mathrm{N})]^t$, devient :

$$\boxed{s_k(t) = \sqrt{\mathscr{E}_k}\sum_{i=1}^{N} b_k(i)g_k(t-iT)} \; ; \quad \mathscr{E}_k \text{ représente l'énergie du signal par octet.}$$

On couple alors tous les utilisateurs et le signal émis devient :

$$s(t) = \sum_{k=1}^{K} s_k(t-\tau_k)$$

$$= \sum_{k=1}^{K}\sqrt{\mathscr{E}_k}\sum_{i=1}^{N} b_k(i)g_k(t-iT-\tau_k) \qquad \text{avec } 0\le\tau_k<T \text{ pour } 1\le k\le K.$$

τ_k représente le délai de transmission pour l'utilisateur k.

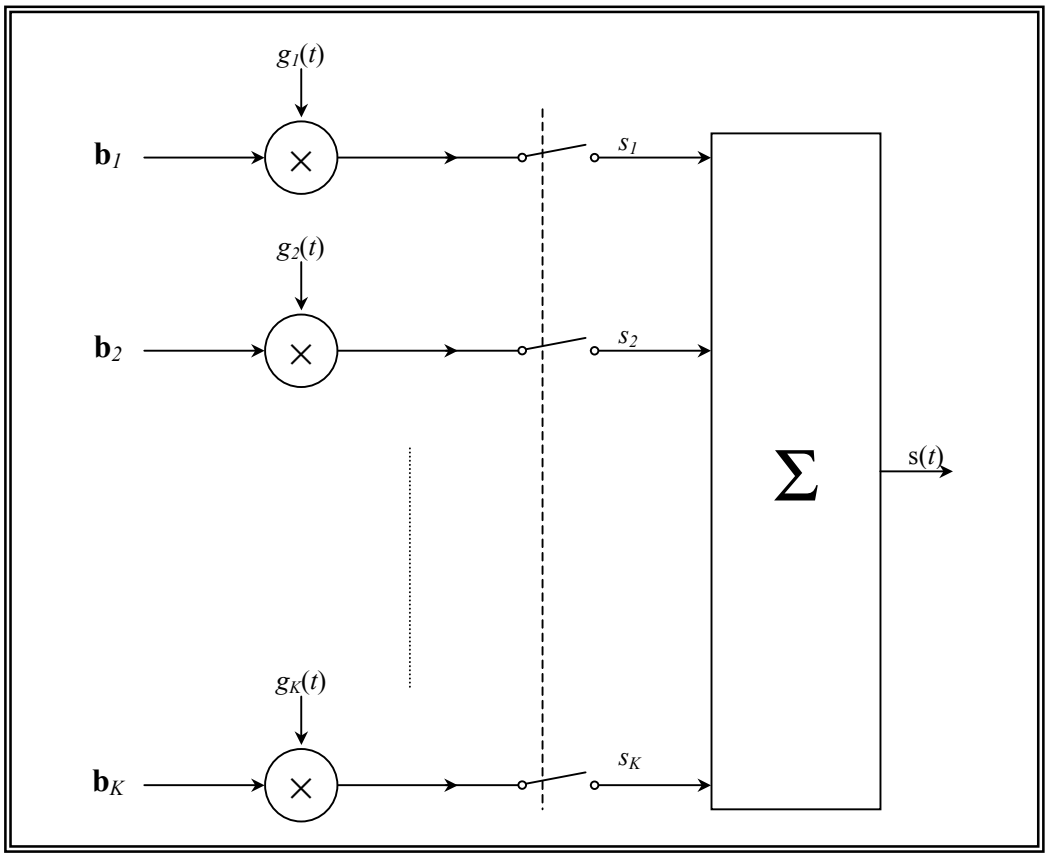


Fig7. *Dispositif de multiplexage*

3. CDMA : réception

On étudie maintenant la façon de traiter le signal reçu afin de découpler des informations transmises. On étudie d'abord le cas d'une transmission parfaite avant d'étudier les méthodes permettant traités les signaux bruités.

3.1. Transmission idéale

On suppose que la transmission s'est effectuée sans ajout de bruit, donc le signal reçu est de la forme: $r(t)=s(t) = \sum_{k=1}^{K} \sqrt{\mathscr{E}_k} \sum_{i=1}^{N} b_k(i) g_k(t - iT - \tau_k)$. On suppose également que les codes choisis sont orthogonaux.

Dans ce cas, si les signaux sont synchrones, il suffit de faire K produits scalaires pour obtenir les K informations de départ :

$$<r(t),g_n(t-iT)> = \int_{iT}^{(i+1)T} \left(\sum_{k=1}^{K} \sqrt{\mathscr{E}_k} \sum_{j=1}^{N} b_k(j) g_k(t - jT) \right) . g_n(t - iT) dt$$

or, on ne regarde que le bit i alors :

$$<r(t),g_n(t-iT)> = \int_{iT}^{(i+1)T} \left(\sum_{k=1}^{K} \sqrt{\mathscr{E}_k} b_k(i) g_k(t - iT) \right) . g_n(t - iT) dt = \sum_{k=1}^{K} \left(\sqrt{\mathscr{E}_k} b_k(i) \int_0^T g_k(t) . g_n(t) dt \right)$$

les codes étant orthogonaux :

$$\boxed{<r(t),g_n(t-iT)> = \sqrt{\mathscr{E}_n}\ b_n(i)}$$

On peut donc ainsi récupérer les informations de chaque utilisateur séparément.

Si maintenant les signaux ne sont pas synchrones, il faut transmettre à un moment les $\{\tau_k\}$ (déphasage entre les utilisateurs). Ensuite, on détermine : $<r(t),g_n(t-iT-\tau_k)> = \sqrt{\mathscr{E}_n}\ b_n(i)$.

3.2. Le récepteur optimal

Le signal reçu est susceptible d'être bruité. On modélise ce bruit par du AWGN (Additional White Gaussian Noise), noté $n(t)$. On a maintenant $\boxed{r(t) = s(t)+n(t)}$. D'une manière générale, en présence de bruit ou quand les codes ne sont pas orthogonaux, on aborde le problème de la manière suivante. On définit la fonction :

$$\Lambda(\mathbf{b}) = \int_0^T \left[r(t) - \sum_{k=1}^{K} \sqrt{\mathscr{E}_k} b_k(1) g_k(t) \right]^2 dt$$

On cherche alors la suite $\{b_k(1), 1 \le k \le K\}$ qui minimise $\Lambda(\mathbf{b})$. Si on développe son expression, on obtient :

$$\Lambda(\mathbf{b}) = \int_0^T r^2(t) dt - 2 \sum_{k=1}^{K} \sqrt{\mathscr{E}_k} b_k(1) \int_0^T r(t) g_k(t) dt + \sum_{j=1}^{K} \sum_{k=1}^{K} \sqrt{\mathscr{E}_j \mathscr{E}_k} b_j(1) b_k(1) \int_0^T g_k(t) g_j(t) dt$$

Le premier terme, étant indépendant de **b**, ne joue pas de rôle dans la minimisation de Λ : on peut donc le négliger. Si on pose alors $r_k = \int_0^T r(t)g_k(t)dt$ et $\rho_{jk}(0) = \int_0^T g_j(t)g_k(t)dt$, on obtient une nouvelle fonction à minimiser :

$$C(\mathbf{r}_K, \mathbf{b}_K) = 2\sum_{k=1}^{K}\sqrt{\mathscr{E}_k}\,b_k(1)r_k - \sum_{j=1}^{K}\sum_{k=1}^{K}\sqrt{\mathscr{E}_j\mathscr{E}_k}\,b_j(1)b_k(1)\rho_{jk}(0)$$

ou, sous une autre forme : $\boxed{C(\mathbf{r}_K, \mathbf{b}_K) = 2\mathbf{b}_K^t\mathbf{r}_K - \mathbf{b}_K^t\mathbf{R}_S\mathbf{b}_K}$

avec $\mathbf{r}_K = [r_1, r_2, ..., r_K]^t$, $\mathbf{b}_K = \left[\sqrt{\mathscr{E}_1}b_1(1), ..., \sqrt{\mathscr{E}_K}b_K(1)\right]^t$ et $(\mathbf{R}_s)_{ij} = \rho_{jk}(0)$.

Il apparaît donc, que pour pouvoir déterminer la bonne suite $\mathbf{b}_K$, il faut transmettre au récepteur la valeur des énergies des signaux émis par chaque utilisateur. Ensuite, le récepteur recherche parmi les 2^K séquences possibles, celle qui correspond le mieux au signal, c'est-à-dire, celle qui minimise $C(\mathbf{r}_k, \mathbf{b}_k)$. K étant le nombre total d'utilisateurs, on comprend qu'il soit impossible d'utiliser ce type de récepteur pour un réseau de taille normal.

De plus, on montre que lorsque la transmission est asynchrone, le récepteur optimal doit réaliser 2^{KN} tests de corrélation (N est la longueur des codes). Il est donc indispensable d'utiliser des récepteurs non-optimaux mais bien plus rapides.

3.3. Les récepteurs non optimaux

a) Le détecteur conventionnel, mono-utilisateur

Le récepteur optimal a une complexité qui croît exponentiellement avec le nombre d'utilisateurs K. On cherche maintenant à déterminer des récepteurs dont la complexité croît linéairement avec K.

Le détecteur le plus simple est celui que nous avons utilisé dans le cas de la transmission idéale (cf. II.2.1). Il s'agit donc de faire des produits scalaires.

On pose $\langle r(t), g_k(t-iT)\rangle = \int_{iT}^{(i+1)T} r(t)g_k(t-iT)dt$,

alors $r_k(i) = \langle r(t), g_k(t-iT)\rangle = \sqrt{\mathscr{E}_k}\,b_k(1) + \sum_{\substack{j=1 \\ j\neq k}}^{K}\sqrt{\mathscr{E}_j}\,b_j(1)\rho_{jk}(0) + n_k(1)$,

en posant $n_k(1) = \int_0^T n(t)g_k(t)dt$

On trouve que si les codes sont orthogonaux, alors le terme central devient nul et ce détecteur est optimal. Par contre, si ce n'est pas le cas, les interférences avec les autres utilisateurs peuvent ne pas être négligeables. Ceci est d'autant plus vrai que les énergies sont assez différentes : les signaux peu puissants risquent d'être mal décodés.

De plus, même dans le cas de codes orthogonaux, si les signaux ne sont pas synchrones, on a de fortes chances de perdre l'orthogonalité et donc de voir la qualité du récepteur se détériorer.

b) Un détecteur multi-utilisateurs

Le détecteur présenté au paragraphe précédent a une complexité linéaire, mais présente une faiblesse dans le cas où les utilisateurs émettraient avec des puissances assez différentes. On va donc étudier un autre détecteur qui tout en restant de complexité linéaire, limite les problèmes d'interférences inter-utilisateurs.

Dans le cadre d'une transmission où les utilisateurs sont synchronisés, si l'on décompose le signal reçu sur la base des $\{g_k\}$ on obtient le vecteur :

$$\mathbf{r}_K = \mathbf{R}_s\mathbf{b}_K + \mathbf{n}_K \text{ ; en posant } (\mathbf{R}_s)_{ij} = \rho_{ij}(0)$$

avec : $\mathbf{b}_K = \left[\sqrt{\mathscr{E}_1}b_1(1),...,\sqrt{\mathscr{E}_K}b_K(1)\right]^t$ et $\mathbf{n}_K = \left[n_1(1), n_2(1),..., n_K(1)\right]^t$ est le bruit gaussien (sa matrice de covariance $E(\mathbf{n}_K\mathbf{n}^t{}_K) = \mathbf{R}_S$).

Or, puisque le bruit est gaussien, et le signal $\mathbf{r}_K$ est d'espérance $\mathbf{R}_s\mathbf{b}_K$ et de covariance $\mathbf{R}_s$ alors,

$$p(\mathbf{r}_K|\mathbf{b}_K) = \frac{1}{\sqrt{(2\pi)^K \det(\mathbf{R}_s)}} \exp\left[-\frac{1}{2}(\mathbf{r}_K - \mathbf{R}_s\mathbf{b}_K)^t \mathbf{R}_s^{-1}(\mathbf{r}_K - \mathbf{R}_s\mathbf{b}_K)\right]$$

donc, la meilleure approximation linéaire de $\mathbf{b}_K$ est le vecteur $\mathbf{b}_K$ qui minimise la fonction :

$$\boxed{\Lambda(\mathbf{b}_K) = (\mathbf{r}_K - \mathbf{R}_s\mathbf{b}_K)^t \mathbf{R}_s^{-1} (\mathbf{r}_K - \mathbf{R}_s\mathbf{b}_K)}$$

On montre que Λ est minimale pour le vecteur $\mathbf{b}_K^0 = \mathbf{R}_s^{-1}\mathbf{r}_K$ le résultat final étant :

$$\hat{\mathbf{b}}_K = \text{signe}(\mathbf{b}_K^0)$$

$\mathbf{b}_K^0$ étant déterminé par une opération linéaire, la complexité des calculs est linéaire en K. On a schématisé le fonctionnement du détecteur **fig8**.

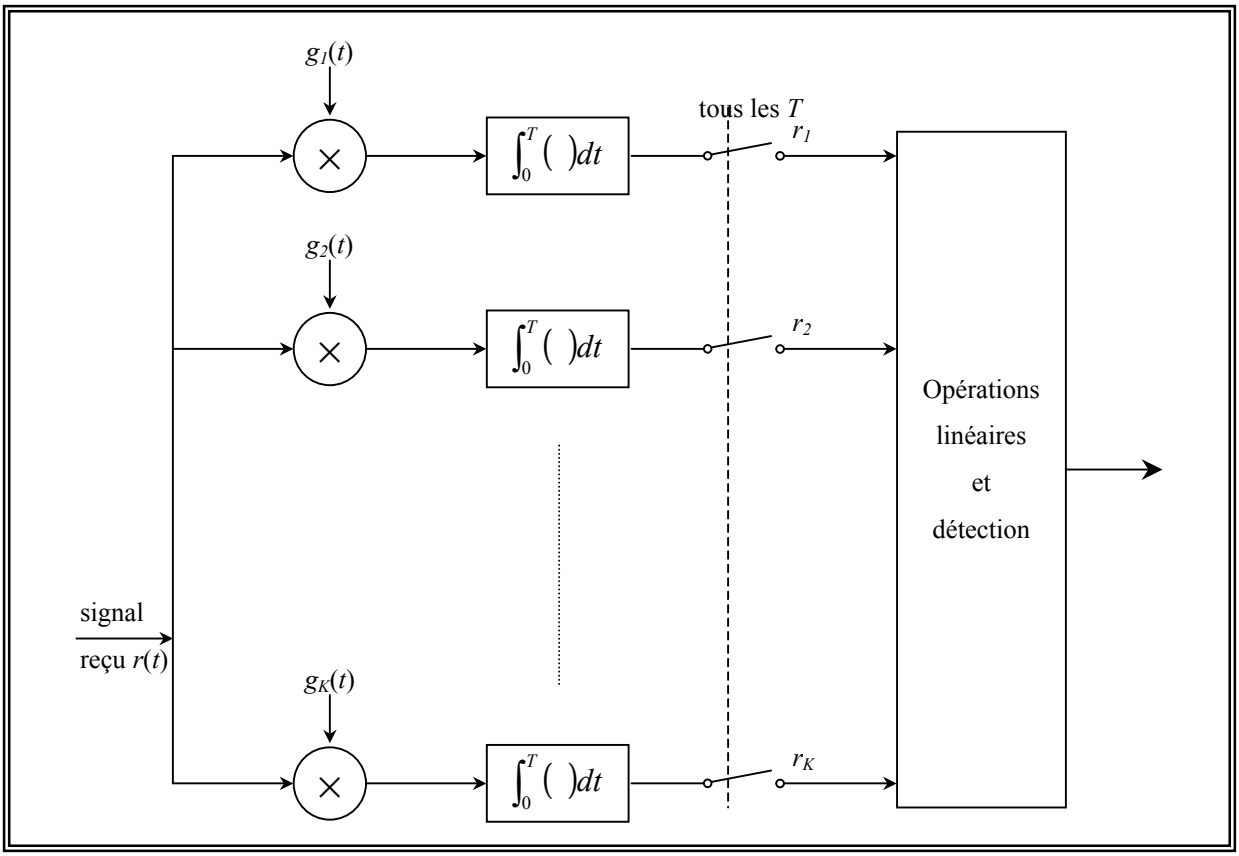


Fig8. *Structure d'un récepteur multi-utilisateur*

Prenons pour illustrer cette démonstration, étudions le cas de deux utilisateurs. On a alors :

$$\mathbf{R}_s = \begin{pmatrix} 1 & \rho_{12}(0) \\ \rho_{21}(0) & 1 \end{pmatrix}, \text{ on rappelle que } \rho_{jk}(0) = \int_0^T g_j(t) g_k(t) dt$$

$$\text{notons } \rho_{12}(0) = \rho_{21}(0) = \rho$$

$$\text{on a également } \mathbf{R}_s^{-1} = \frac{1}{1-\rho^2} \begin{pmatrix} 1 & -\rho \\ -\rho & 1 \end{pmatrix}$$

Le signal reçu est donc de la forme $r(t) = \sqrt{\mathscr{E}_1} b_1 g_1(t) + \sqrt{\mathscr{E}_2} b_2 g_2(t) + n(t)$, ce qui donne :

$$\mathbf{r}_2 = \begin{bmatrix} \sqrt{\mathscr{E}_1} b_1 + \rho \sqrt{\mathscr{E}_2} b_2 + n_1 \\ \rho \sqrt{\mathscr{E}_1} b_1 + \sqrt{\mathscr{E}_2} b_2 + n_2 \end{bmatrix}$$

$$\text{il vient, } \mathbf{b}_2^0 = \mathbf{R}_s^{-1} \mathbf{r}_2 = \begin{bmatrix} \sqrt{\mathscr{E}_1} b_1 + (n_1 - \rho n_2)/(1-\rho^2) \\ \sqrt{\mathscr{E}_2} b_2 + (n_2 - \rho n_1)/(1-\rho^2) \end{bmatrix}$$

Il est intéressant de remarquer que les 2 signaux sont maintenant décorrélés, ce qui signifie que l'on n'a plus les problèmes qui pouvaient apparaître dans le cas où les énergies des différents signaux seraient assez différentes.

On peut aussi noter que dans le cas où g_1 et g_2 sont orthogonaux ($\rho = 0$), on retrouve alors le même résultat qu'avec le détecteur mono-utilisateur. De plus, on peut montrer que cette méthode est toujours valable dans le cas où les signaux ne seraient plus synchrones. Ce détecteur permet donc d'éliminer les problèmes d'interférences des signaux entre eux. C'est pourquoi on lui donne aussi le nom de *decorrelating detector*.

4. Les avantages du CDMA

4.1. Résistance aux interférences

Historiquement, le CDMA est issu de programmes de recherche militaires qui avaient pour but de protéger les transmissions d'information contre le brouillage, c'est à dire une forme d'interférence volontaire. Le CDMA, étant une méthode de multiplexage à étalement de spectre, le brouillage efficace doit se faire sur toute la bande de fréquences utilisées, ce qui n'est pas envisageable car cela consommerait une puissance colossale.

Dans les applications civiles, la résistance à un brouillage intentionnel n'est pas un critère déterminant dans le choix de la technologie de multiplexage. On cherche cependant à rendre le système de communication résistant à des interférences non volontaires : les interférences entre utilisateurs, les interférences liées à des phénomènes de réflexion et la présence d'un bruit additif. Le premier type d'interférences est bien toléré par le CDMA, par construction, puisque les codes utilisés sont faiblement corrélés. Au vu des publications auxquelles nous avons eu accès, la résistance au deuxième type d'interférences est également assurée, même si nous n'avons pas pu la mettre en évidence par nous-même. Quant à la résistance au bruit additif, nos résultats expérimentaux nous ont montré la performance très satisfaisante du CDMA (cf. annexe I).

Toutes ces considérations font que le CDMA permet de garantir une téléphonie haute fidélité.

4.2. Confidentialité (faible probabilité d'interception)

Pour les applications militaires comme civiles, la confidentialité est un atout important pour un système de communication. Dans le cas du CDMA, le signal émis ressemble beaucoup à du bruit parce que l'on utilise des codes longs pseudo-aléatoires.

Le signal est étalé uniformément sur un large spectre : on ne détecte aucun pic en amplitude pour une fréquence donnée. Ceci permet de masquer la présence ou non d'une communication. Quand bien même on détecterait l'existence d'une communication, il est très difficile de l'intercepter si on n'a pas accès aux codes utilisés. C'est une des raisons qui font que l'armée, ainsi que les opérateurs téléphoniques utilisent cette méthode.

4.3. Un multiplexage adapté au système cellulaire

Les réseaux de téléphonie mobile actuels sont tous basés sur le concept de cellules. Une cellule correspond à une zone géographique dans laquelle les utilisateurs transitent tous par le même relais. Il se pose deux problèmes : celui de la réutilisation des fréquences et celui du passage d'un utilisateur d'une cellule à une autre.

Du point de vue de la réutilisation des fréquences, le CDMA déplace le problème puisqu'il s'agit de codes et non plus de fréquences. Cet aspect donne lieu soit à des analyses caricaturales (dans les publications quasi publicitaires d'entreprises), soit à des études dont le niveau nous dépassait. Il semblerait que le CDMA soit plus performant que les autres méthodes de multiplexage au niveau des zones de recouvrement des cellules. Nous n'avons pas pu cependant contrôler cette affirmation.

Comme l'étude l'a montré (cf. 3.3.b), la qualité de transmission en CDMA n'est que faiblement affecté par les différences d'amplitude signaux des différents utilisateurs. Ceci permet dans la pratique d'augmenter la taille des cellules. On réduit alors la fréquence de passage d'une cellule à l'autre (pour des utilisateurs en mouvement) et on réduit d'autant les risques de décrochage.

4.4. Une faible consommation

Le CDMA nécessite moins de puissance que les technologies concurrentes. Ce gain est présent en conversation ou non. Ceci permet l'augmentation de l'autonomie des téléphones portables ou bien la diminution de la taille des batteries donc des combinés.

III

-

Implémentation du CDMA: le logiciel CDMAster

L'étude théorique du CDMA nous a fait comprendre qu'il s'agit d'une méthode de multiplexage performante, et nous avons donc décidé d'en réaliser une implémentation pour pouvoir étudier les problèmes pratiques qui peuvent se poser à la réception :

- Détecter le début d'une émission (synchronisation initiale).
- Maintenir la synchronisation avec le signal au cours du démultiplexage si les horloges d'émission et de réception ne sont pas parfaitement identiques.
- La tolérance au bruit est traitée en annexe.

1. Choix techniques

1.1. Méthode de multiplexage

Nous avons opté pour du **DS-CDMA** « Direct Sequence CDMA », car cette variété de CDMA est la plus simple à mettre en œuvre et est effectivement utilisée dans des applications de téléphonie mobile.

Rappelons le principe du DS-CDMA : chaque utilisateur du canal de communication se voit attribuer un code composé d'une série de bits. Un « 1 » dans le message de l'utilisateur se traduit alors par la clé dans le signal codé, un « 0 » se traduit par l'inverse de la clé.

Les clés utilisées sont des **codes de Gold orthogonaux**, l'orthogonalité facilitant grandement le démultiplexage.

1.2. Méthode de modulation

La modulation se fait en amplitude : si pour un « chip » donné les 4 utilisateurs émettent un « 1 », on a une demi-sinusoïde positive d'amplitude 4, si 4 utilisateurs émettent un « 0 », une demi-sinusoïde d'amplitude -4, si 3 utilisateurs émettent un « 1 » et le dernier un « 0 », une demi-sinusoïde d'amplitude 2, etc…

1.3. Une solution logicielle basée sur la carte son d'un ordinateur

Ayant déjà réalisé des circuits de numérisation, sérialisation et désérialisation, il nous a paru intéressant de passer à un niveau d'abstraction plus élevé, en utilisant un convertisseur A/N et N/A existant et contrôlable à volonté par logiciel : la carte son d'un ordinateur. Nous avons réalisé un logiciel intitulé CDMAster qui est capable d'effectuer les opérations de génération de clés puis de multiplexage/démultiplexage CDMA, le canal de transmission étant soit un câble reliant deux cartes sons, soit les ondes acoustiques en se munissant d'un haut-parleur et d'un microphone.

Cette solution logicielle correspond assez bien à ce qui se fait dans l'industrie puisque les équipements de communication récents sont basés sur des DSP (Digital Signal Processor) et autres composants programmables.

Ce système comporte plusieurs avantages :

- Une carte son est un composant peu cher et très répandu.
- La carte son peut servir d'oscilloscope numérique à mémoire grâce à un logiciel de capture de son.
- On peut enregistrer le signal émis, le modifier (en ajoutant du bruit par exemple), et le renvoyer de façon à tester la tolérance du système à la déformation ainsi introduite.

Nous avons opté pour une programmation en Visual C++, alliant ainsi la rapidité du C++ et une conception facilitée d'interface graphique pour plateforme Windows.

1.4. Capacité du canal de transmission

Nous avons travaillé le plus souvent avec les données suivantes :

- Codes orthogonaux de 8 bits, permettant d'avoir un maximum de huit utilisateurs.
- Un utilisateur réservé à l'horloge, pour maintenir la synchronisation au moment du démultiplexage.
- Une fréquence d'échantillonnage de 44.1kHz qui correspond au maximum offert par les cartes sons les plus courantes.
- Une définition de 32 échantillons par « chip » (demi-sinusoïde)

Ceci nous donne donc la capacité suivante pour le canal de transmission :

- 172 bits/s par utilisateur
- 7 utilisateurs « utilisables » pour un total de 1200 bits/s

En pratique, nous n'utilisons que 3 véritables utilisateurs, pour plus de clarté lors de l'analyse des signaux multiplexés.

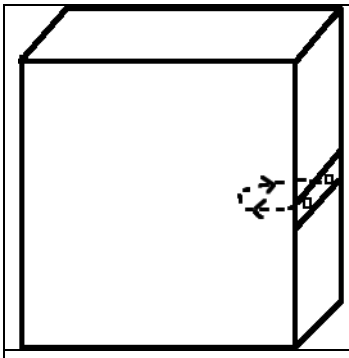
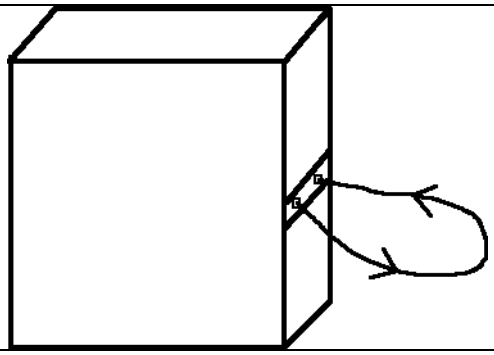
Le débit obtenu peut paraître très faible, mais il faut garder à l'esprit que dans les applications de téléphonie numérique, la fréquence d'échantillonnage est de l'ordre de plusieurs gigahertz, soit quasiment un million de fois plus élevée, ce qui même en rallongeant les codes donne des débits beaucoup plus importants.

2. Organisation et utilisation du logiciel CDMaster

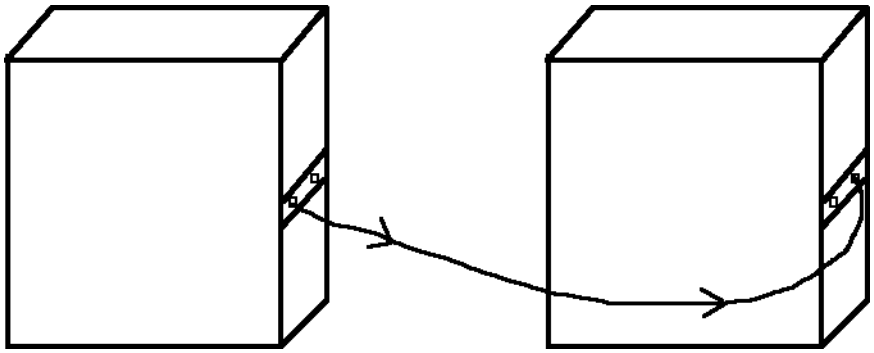
2.1. Organisation

Nous avons scindé CDMaster en deux programmes : CDMAClient et CDMAServer, le premier étant chargé de l'émission, le deuxième de la réception. Ceci permet de travailler avec plusieurs configurations :

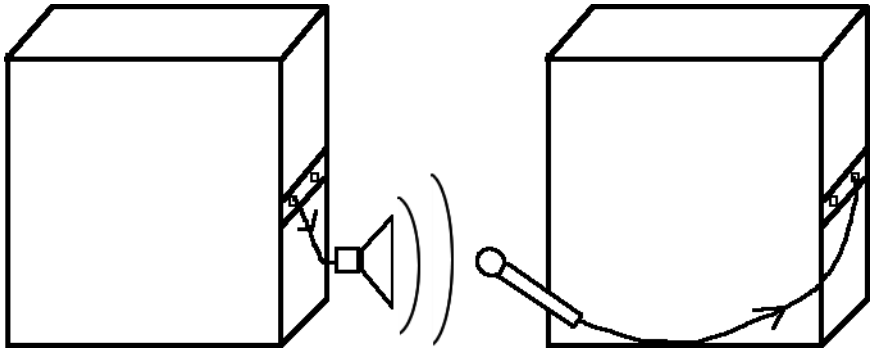
Avec un seul ordinateur :

	
Ré-acquisition directe du signal (tout numérique)	Conversion N/A et A/N sur le même ordinateur

Avec deux ordinateurs :



Connexion par câble

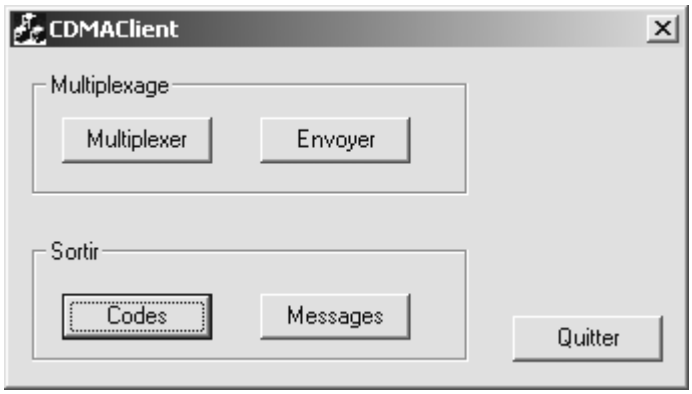


Utilisation d'un haut-parleur et d'un microphone

Selon la configuration adoptée, il faut ajuster les volumes de l'émetteur et du récepteur pour avoir un niveau correct à la réception (au moins 50% du maximum autorisé, sans pour autant saturer).

2.2. CDMAClient (émission)

Ce programme s'occupe de coder des messages (séquences d'octets saisies sous forme de chaîne de caractère) fournis par l'utilisateur, de les moduler, de les sommer et de les envoyer sur la carte son. Les codes de chaque utilisateur sont générés dès qu'on lance le programme et sont stockés en mémoire.



- **Multiplexer**

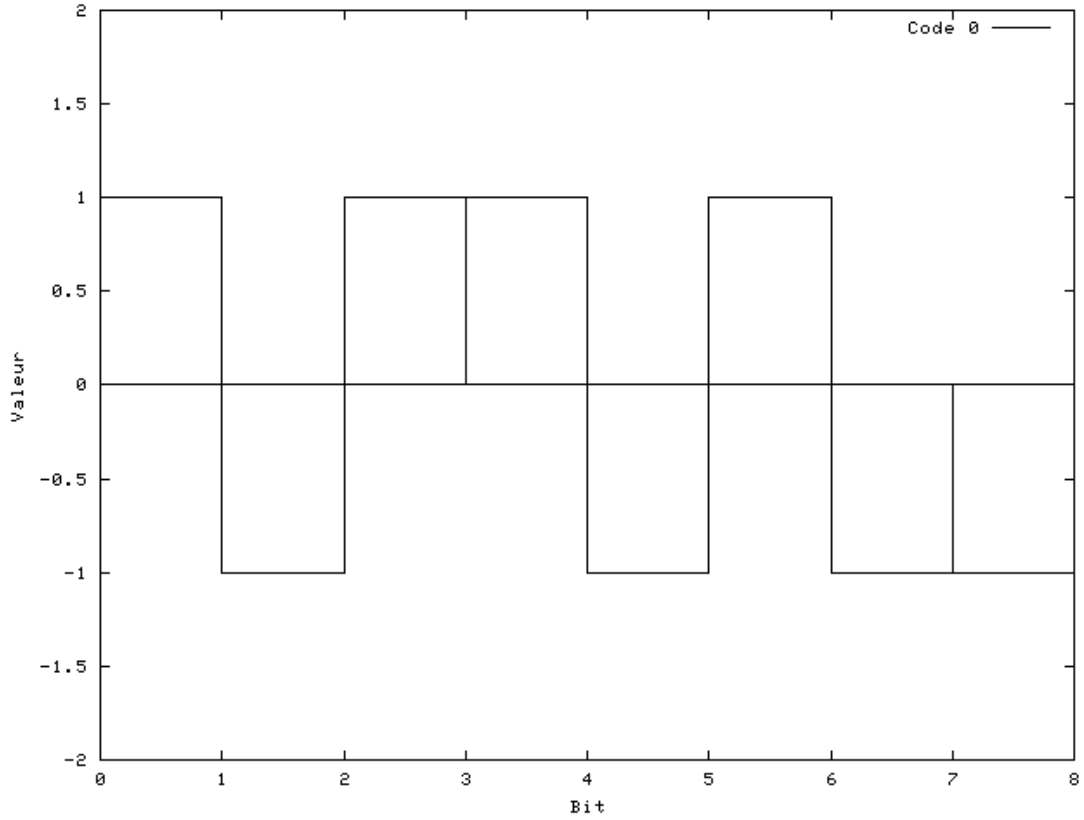
Quand on clique sur « Multiplexer », CDMAClient demande de saisir trois chaînes de caractères qui correspondent aux messages envoyés par les trois utilisateurs.

- **Envoyer**

Ce bouton est initialement grisé et est déverrouillé dès qu'on a multiplexé les messages à envoyer. En cliquant dessus, on déclenche l'émission du signal sonore.

- **Codes**

Ce bouton permet d'écrire les codes utilisés dans un fichier texte, de façon à les visualiser avec un logiciel de tracer de courbes tel que ***gnuplot***.

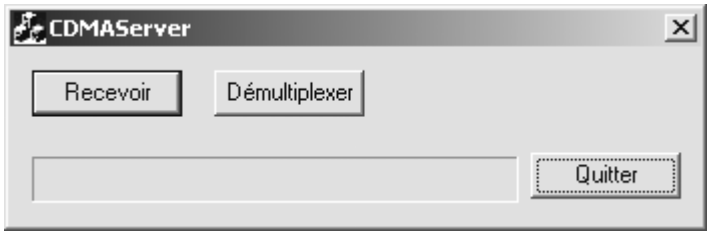


- **Messages**

Cette option permet d'écrire dans un fichier le message codé de chaque utilisateur.

2.3. CDMAServer (réception)

CDMAserver reçoit le signal résultant du multiplexage, l'enregistre puis se charge de repérer le début de l'émission, d'effectuer le démultiplexage et de restituer les messages correspondant à chaque émetteur. Au démarrage, CDMAServer génère une copie des codes de chaque utilisateur pour être en mesure de démultiplexer le signal reçu.



- **Recevoir**

Dès qu'on appuie sur ce bouton le programme se met à enregistrer ce que reçoit la carte son, et ce jusqu'à remplir un tampon qui fait la longueur maximale de message autorisée plus cinq secondes. Ces cinq secondes sont prévues pour laisser le temps à l'utilisateur de déclencher l'émission sous CDMAClient.

- **Démultiplexer**

En cliquant sur ce bouton, on lance le traitement du signal reçu. Ce traitement se fait en plusieurs temps :

a) détection du début de l'émission en balayant le tampon de réception ;
b) calibrage des échelles d'amplitude ;
c) démultiplexage à partir du début de l'émission, avec maintient de la synchronisation au fur et à mesure qu'on avance.

3. Les briques élémentaires du programme

3.1. Génération des codes

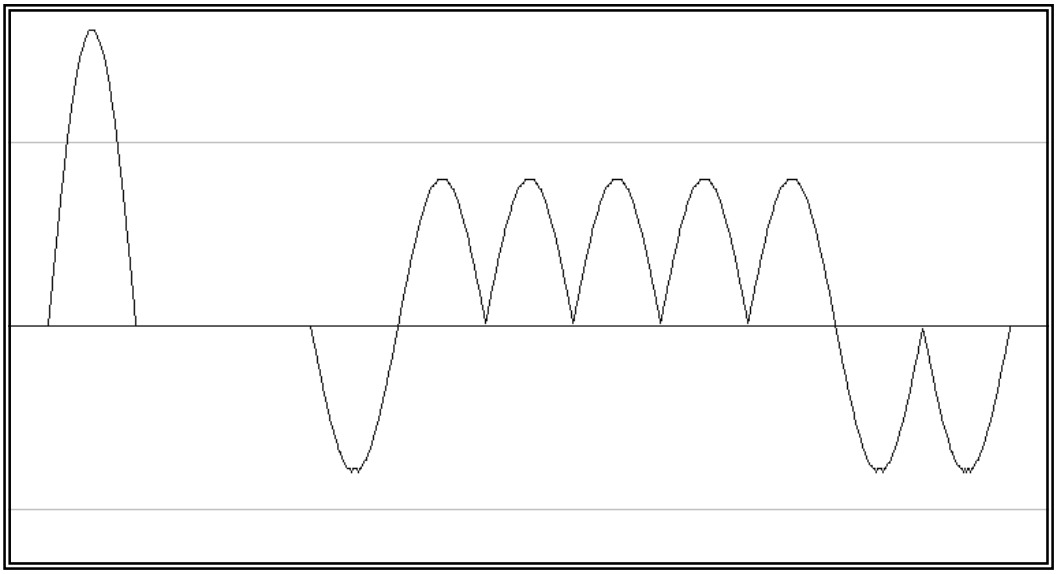
Les deux programmes CDMAClient et CDMAster commencent par générer les codes de Gold orthogonaux de 8 bits à l'aide d'une routine qui leur est commune. Cette routine s'appuie sur un morceau de code capable de générer des codes Gold, code mis en logiciel libre par Byougjo Choi de l'Université de Southampton. Les codes de Gold sont obtenus en effectuant des combinaisons linéaires particulières de codes appelés *m-sequences* ou *maximal length sequences* décalés l'un par rapport à l'autre. Nous avons écrit un programme auxiliaire qui cherche les bons paramètres pour obtenir des codes de Gold qui sont de plus orthogonaux.

Une fois ces codes générés, on stocke la clé de chaque utilisateur. Nous avons travaillé le plus souvent avec des codes de 8 bits car il ne nous a pas semblé intéressant de travailler avec plus de trois utilisateurs effectifs (plus un pour l'horloge) sur le canal de communication. Cependant, la longueur des clés est stockée dans une constante globale commune aux deux programmes, donc passer à des clés de 16 bits ou plus est une opération qui peut se faire aisément.

3.2. Du signal carré à un signal sinusoïdal

Les calculs de codage se font sur des signaux numériques, c'est-à-dire de suites de bits qui peuvent prendre pour valeur 0 ou 1. En réalité, nous travaillons plutôt avec des suites de -1 et +1 car il nous faut fréquemment faire des multiplications sur ces séquences (XOR entre séquences, génération de demi-sinusoïdes d'amplitude -1 ou +1) et il est alors confortable d'avoir des valeurs symétriques par rapport à zéro.

Les opérations d'émission (sommation des signaux émis par chaque utilisateur) et de démultiplexage sont quant à elles effectuées sur des séquences d'échantillons codés chacun sur plusieurs bits. Nous avons décidé de travailler avec une profondeur d'échantillonnage de 8 bits même si la plupart des cartes sons sont capables de gérer du 16 bits parce que la manipulation de séquences d'octets est plus commode que la manipulation de séquences de paires d'octets.



Les signaux émis sont des signaux composés d'une suite de demi-sinusoïdes d'amplitudes différentes, chaque demi-sinusoïde étant décrite par 32 échantillons. Nous avons donc écrit la procédure *WriteHalfSine* qui génère une demi-sinusoïde et l'écrit dans un emplacement mémoire.

```
void WriteHalfSine(BYTE* buffer, int coeff) {
      const float cf = 3.1415927f/HSINE_POINTS;
      float ampli = float(HSINE_AMPLI * coeff);

      int zero = 1 << 7;

      for(int i = 0; i < HSINE_POINTS; i++) {
            buffer[i] = BYTE(ampli*sin(i*cf)+zero);
      }
}
```

HSINE_POINTS (32) : points par demi-sinusoïde
HSINE_AMPLI (127/USERS) : amplitude d'une sinusoïde émise par un seul utilisateur
coeff : amplitude de la demi-sinusoïde par rapport à l'amplitude pour un seul utilisateur
buffer : emplacement mémoire où l'on souhaite stocker la demi-sinusoïde

Cette procédure est utilisée à de nombreuses reprises, pour l'émission comme pour la réception

3.3. Calculs de corrélation

Comme nous l'avons vu, le démultiplexage de signaux DS-CDMA passe par des calculs de corrélation entre le signal reçu et un signal attendu : l'équivalent sinusoïdal de la clé de chaque utilisateur. Ces calculs de corrélation sont des produits scalaires de fonctions, mais par la nature échantillonnée des signaux traités, il s'agit en fait de sommes discrètes.

Nous avons écrit la procédure *CalcCorrel* qui permet de calculer le coefficient de corrélation entre deux signaux sur un nombre de « blocs » que l'on passe en paramètre. Un

bloc désigne une séquence de KEY_LENGTH chips où KEY_LENGTH est le nombre de bits composant un code (8 dans notre cas).

```
double CalcCorrel(BYTE* buffer1, BYTE* buffer2,
double prod_ampli, DWORD n /*=1*/)
{
      long correl = 0;
      double correl2;
      DWORD dwPos = 0;

      for (dwPos = 0; dwPos < n*KEY_LENGTH*HSINE_POINTS; dwPos++) {
            correl += (buffer1[dwPos]-128) * (buffer2[dwPos]-128);
      }

      correl2 = 2*double(correl) /prod_ampli /HSINE_POINTS /KEY_LENGTH /n;
      return correl2;
}
```

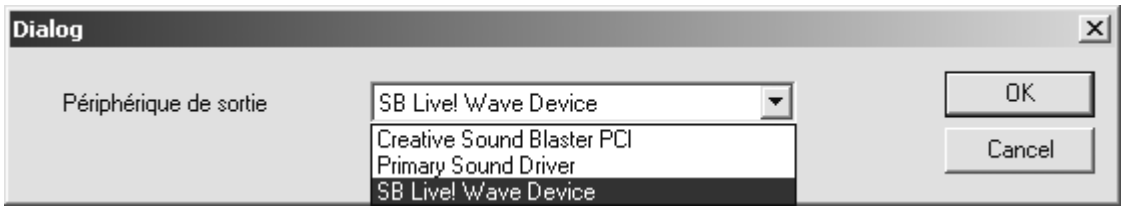
KEY_LENGTH (8) : longueur d'une clé
buffer1 : emplacement mémoire du premier signal
buffer2 : emplacement mémoire du deuxième
n : nombre de blocs sur lequel effectuer le calcul
prod_ampli : produit *attendu* des amplitudes

prod_ampli mérite une explication supplémentaire qui est donnée en 4.2.

3.4. Pilotage de la carte son

Il existe de nombreuses cartes sons disponibles sur le marché, chacune ayant des caractéristiques et des mécanismes de commande propres. Cependant, écrire un logiciel destiné à un seul type de carte son en aurait limité la portée et nous avons donc eu recours à une couche d'abstraction par rapport au matériel. Nous avons opté pour la librairie DirectSound de Microsoft qui est téléchargeable librement.

Au démarrage de CDMAClient comme de CDMAServer, il est demandé à l'utilisateur de préciser quel périphérique d'acquisition ou de sortie son il souhaite utiliser si son ordinateur en comporte plusieurs.



a) dans CDMAClient

Dès que l'utilisateur à choisi son périphérique de sortie, on a :

- initialisation d'un objet « carte son »
- création d'un tampon mémoire dans lequel la carte son viendra lire les échantillons à jouer

Quand on effectue le multiplexage (bouton « multiplexer »), on procède en trois étapes :

- verrouillage du tampon
- écriture des échantillons multiplexés dans le tampon
- déverrouillage du tampon
- lecture du tampon par la carte son (émission)

b) dans CDMAServer

Dès que l'utilisateur à choisi son périphérique d'acquisition, on a :

- initialisation d'un objet « carte son »
- création d'un tampon mémoire d'acquisition dans lequel la carte écrira les échantillons reçus

Lors de la réception (bouton « recevoir »), on a successivement :

- verrouillage du tampon
- suppression du contenu du tampon
- déverrouillage du tampon
- passage en mode acquisition de la carte son

Enfin, la procédure de démultiplexage se déroule ainsi :

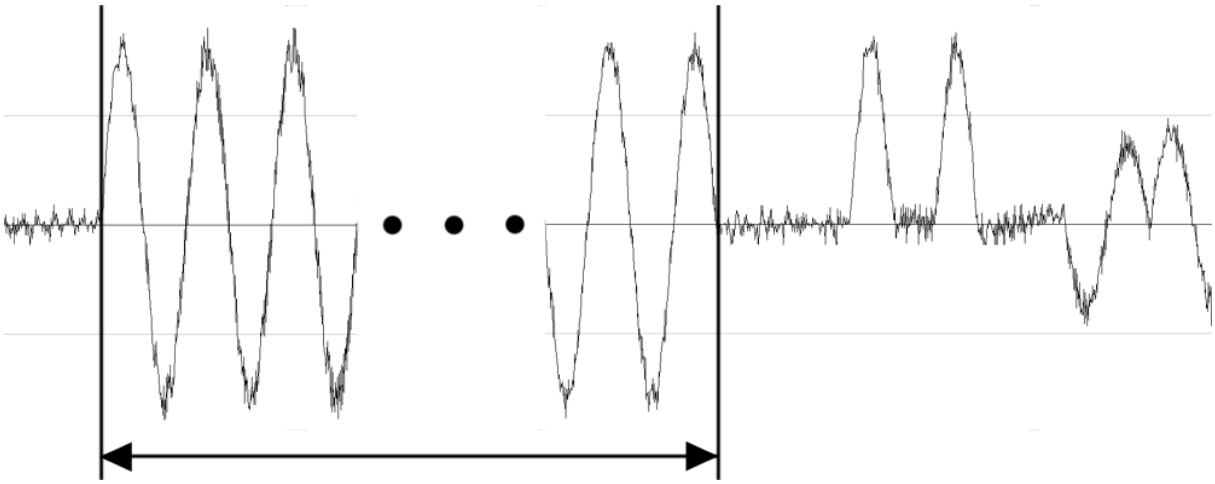
- verrouillage du tampon
- traitement des échantillons (recherche du début de l'émission, démultiplexage)
- déverrouillage du tampon

4. Mécanismes de la transmission

4.1. Synchronisation initiale (détection du début de l'émission)

Notre système de communication étant asynchrone, il nous faut signaler le début de l'émission et savoir détecter à la réception. Pour cela, avant de commencer à émettre des données, nous envoyons une séquence caractéristique qui sert de marqueur.

La séquence qui nous sert de marqueur de départ dure 8 blocs, la première moitié étant une sinusoïde, la deuxième une sinusoïde en opposition de phase avec la première. Il est très peu probable d'avoir ce signal autre part dans le signal reçu, voire impossible à partir du moment où tous les utilisateurs n'émettent pas (on a alors une base incomplète).



Au démultiplexage, pour détecter la présence et l'emplacement dans le tampon du marqueur de début d'émission, CDMAServer commence par générer une copie du marqueur. Comme on dispose de la fonction *WriteHalfSine*, c'est très simple :

```
dwPos = 0;
for (i = 0; i < HDR_LEN / 2; i++) {
      WriteHalfSine(pStartBlock+dwPos, (2*(i % 2) -1) * USERS);
      dwPos += HSINE_POINTS;
}
for (i = 0; i < HDR_LEN / 2; i++) {
      WriteHalfSine(pStartBlock+dwPos, (1- 2*(i % 2)) * USERS);
      dwPos += HSINE_POINTS;
}
```

`pStartBlock` : l'emplacement en mémoire où on stocke le marqueur

CDMAServer ensuite parcourt le tampon de réception et effectue des calculs de corrélation entre le signal reçu et le marqueur sur 8 blocs, en décalant les deux signaux l'un par rapport à l'autre. On mémorise la position où le coefficient de corrélation est maximal, et si celui-ci est au-dessus d'un seuil arbitraire, on considère qu'on a trouvé le début de l'émission.

```
maxcorrel = 0;
dwPos =0;
for (dwPos=0; dwPos<dwLength - SND_BUFF_MSG-SND_BUFF_HDR; dwPos+=SND_SYNC)
{
      correl = CalcCorrel(lpvRead+ dwPos,pStartBlock,127*127, HDR_LEN /
                                                              KEY_LENGTH);
      if (fabs(correl) > fabs(maxcorrel)) {
            maxcorrel = correl;
            dwMaxPos = dwPos;
      }
}
```

`lpvRead` : pointeur sur le tampon de réception

4.2. Calibrage en amplitude

Le marqueur de départ remplit une deuxième fonction : il nous permet de faire un calibrage en amplitude. Le marqueur de départ est émis avec l'amplitude maximale autorisée, 127. On passe 127*127 comme valeur du paramètre *prod_ampli* (cf. 3.3) à la fonction *CalcCorrel*, et la valeur du coefficient de corrélation maximal obtenu nous donne accès à

l'amplitude. En effet, *CalcCorrel* est écrite de façon à nous renvoyer exactement 1 si le produit des amplitudes correspond effectivement à *prod_ampli*.

On peut alors se servir du coefficient obtenu pour corriger tous les calculs de corrélation qui suivent.

4.3. Démultiplexage, maintien de la synchronisation

Le démultiplexage se fait selon la méthode décrite au II.3.3.a), il s'agit d'un « détecteur conventionnel mono-utilisateur », soit un détecteur qui calcule (produit scalaire) un par un les messages émis. Les produits scalaires sont encore une fois effectués à l'aide la fonction *CalcCorrel*.

Lors de la mise au point de CDMAster nous avons eu des difficultés à transmettre des messages au-delà d'une certaine longueur. Nous avons donc pensé qu'il y avait un décalage au cours du temps. Le choix d'une solution informatique s'est alors avéré être une bonne solution, car nous avons pu faire des captures du tampon et les visualiser. Pour une raison quelconque, les fréquences d'échantillonnage de la carte son à l'émission et à l'acquisition ne sont pas strictement identiques et on avait bien un phénomène de décalage, comme les captures nous l'ont prouvé. Le décalage était de l'ordre de 0.1%.

Pour résoudre ce problème, nous avons introduit un utilisateur factice, l'horloge. Cet utilisateur a ceci de particulier qu'il émet toujours le même message. Sachant ceci, au début de chaque nouvel octet, on fait un calcul similaire à celui de la synchronisation. Si l'octet bloc précédent commençait à un position donnée, le suivant devrait commencer 2048 échantillons plus loin. Pour déterminer le décalage qui a été introduit depuis le dernier octet, on fait un calcul de corrélation au voisinage de la position théorique du nouvel octet, et on retient la position qui maximise le coefficient de corrélation.

Les différents problèmes étant résolus, nous avons pu tester efficacement notre programme. Les résultats sont assez bons, les messages sont bien transmis et la résistance au bruit est très satisfaisante. Un exemple d'utilisation de notre programme est détaillé en annexe (I).

Conclusion

Le CDMA est une technologie à la fois performante en terme de rapidité (complexité linéaire) et en terme de qualité. Sa résistance au bruit, aux interférences inter-utilisateurs… semble très satisfaisante d'un point de vue théorique. Sur le plan expérimental, nos attentes n'ont pas été déçues.

On pourrait apporter à notre programme un certain nombre d'améliorations qui le rendraient plus performant. Il serait par exemple possible de se dispenser du signal d'horloge et d'extraire l'information d'horloge directement du signal émis par un utilisateur quelconque. En effet, même si on ne sait pas à l'avance quelle séquence de bits celui-ci va envoyer, connaissant les codes utilisés, on connaît des propriétés du signal émis (présence soit du code de l'utilisateur, soit de son opposé). On libèrerait ainsi une place pour un utilisateur supplémentaire.

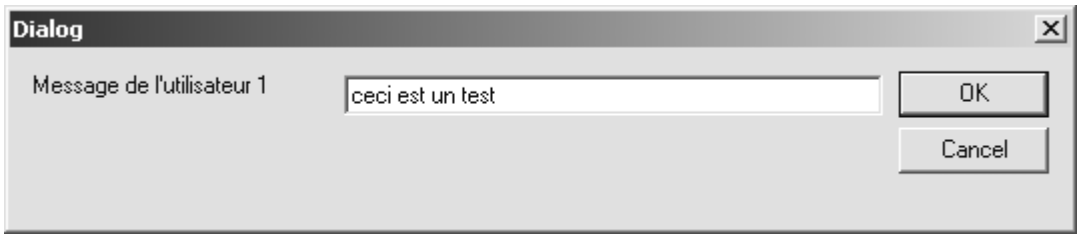
Le cahier des charges du logiciel n'a pas été facile à définir. Le CDMA est une technologie déjà bien maîtrisée et l'essentiel des publications concerne des raffinements et des optimisations pour des cas de figure bien particuliers. Il nous a donc fallu faire un important travail de synthèse pour dégager l'essentiel de cette technologie afin de pouvoir nous fixer des objectifs réalisables dans le cadre du projet.

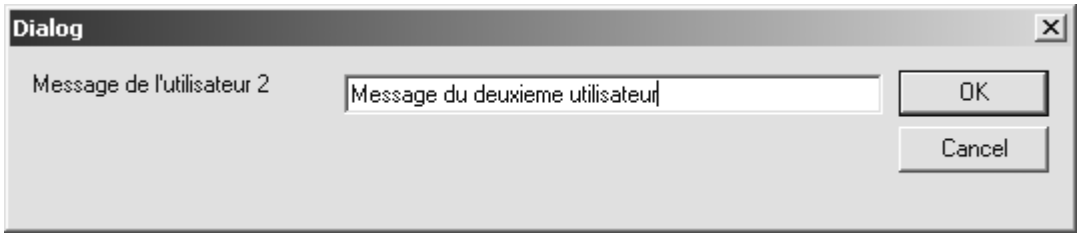
Malgré ces difficultés, le fait d'avoir pu choisir et mener à bien un projet personnel est pour nous une grande source de satisfaction.

Annexes

Un exemple d'utilisation du programme

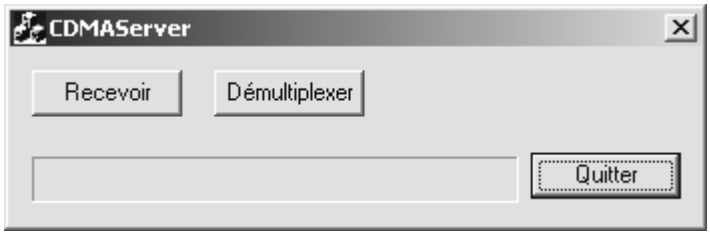
On commence par rentrer les 3 messages :





Puis on envoie le résultat du multiplexage.

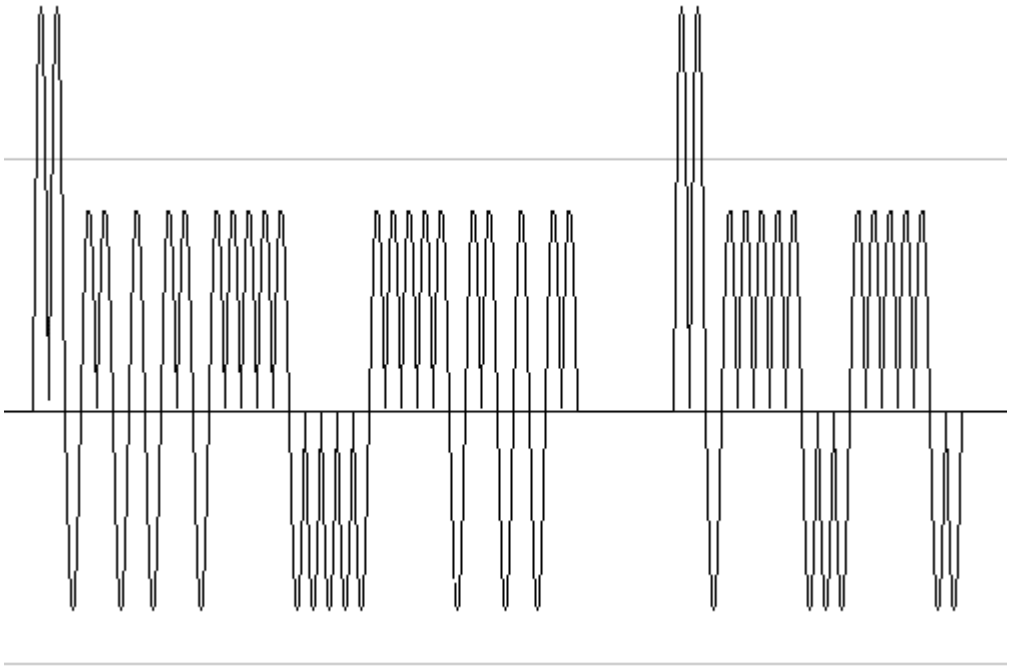
On lance ensuite le démultiplexage :



On obtient alors le résultat suivant :

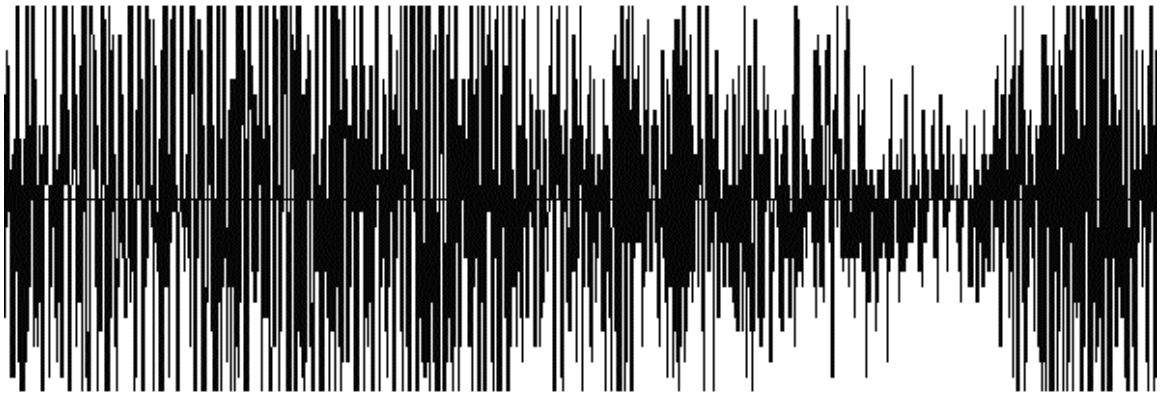
Ceci nous indique que la transmission a réussi.

Lorsque que l'on transmet le message par un câble (le bruit est quasiment nul, on n'a pas de déformation), on obtient le signal suivant :

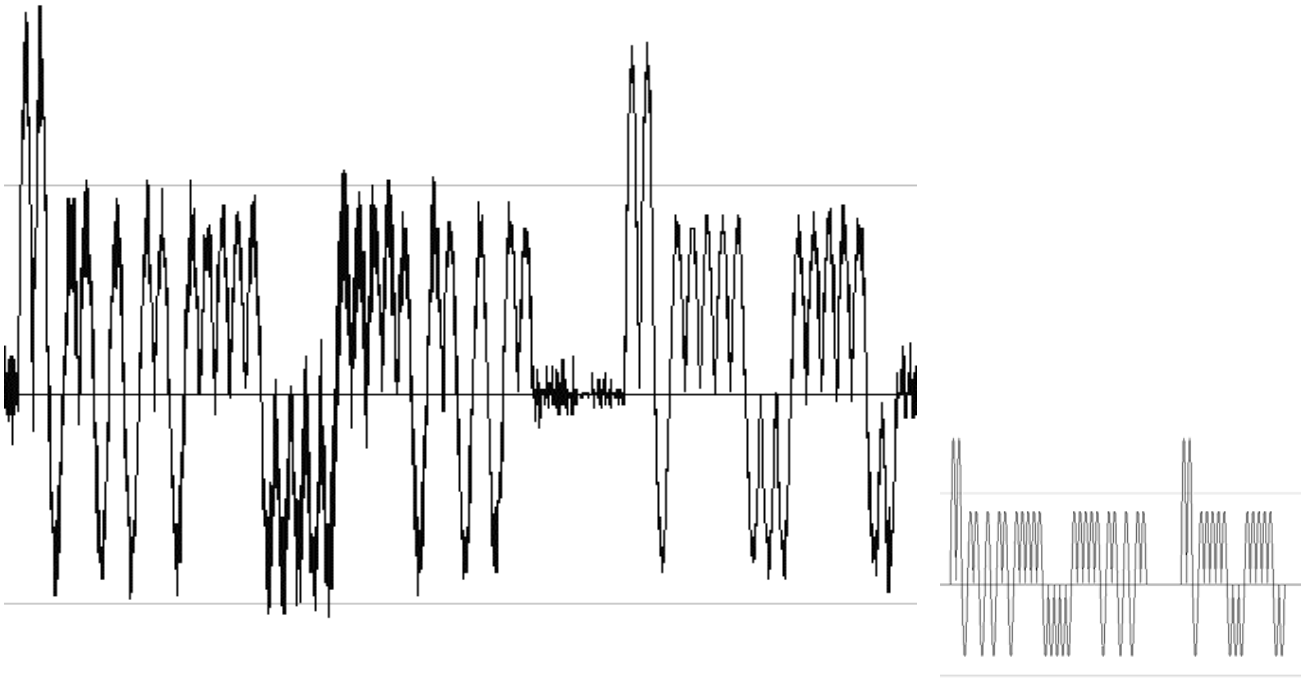


On obtient bien ensuite les bons messages après démultiplexage.

On teste donc le comportement du programme lorsque que l'on ajoute du bruit. Le bruit a cette allure :

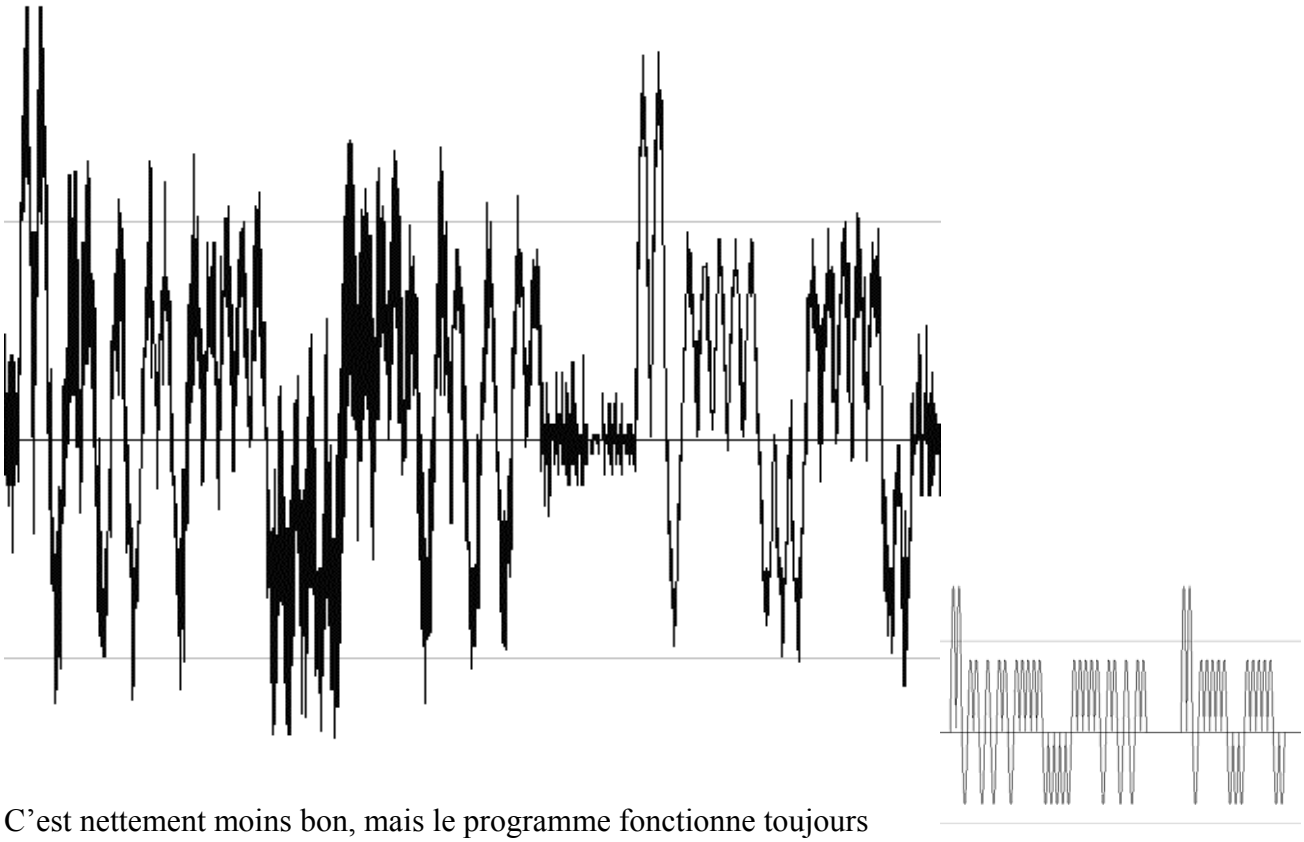


On a fait alors plusieurs tests en fonction de l'amplitude que l'on affecte à ce bruit. A faible amplitude, on obtient ce signal :



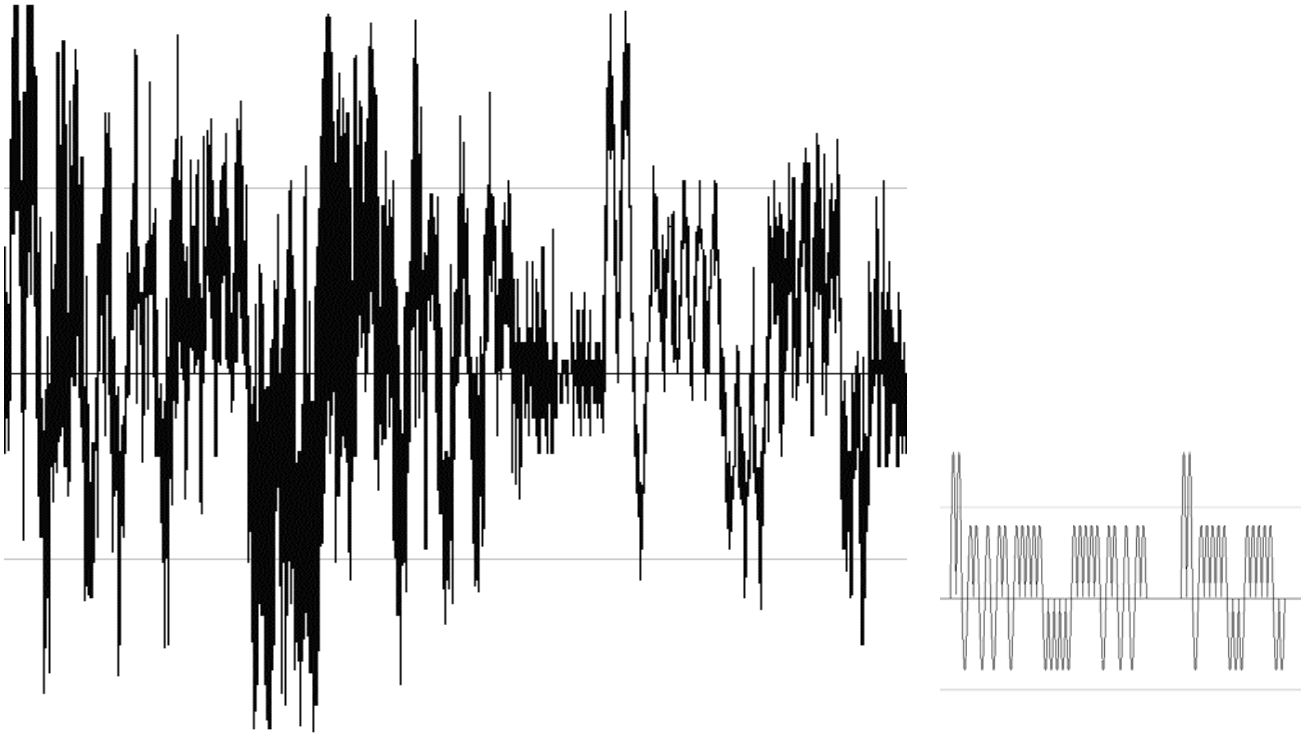
Visuellement on retrouve facilement les pics et leur amplitude. Le programme nous redonne également les bons messages.

En mettant le bruit un peu plus fort on obtient :

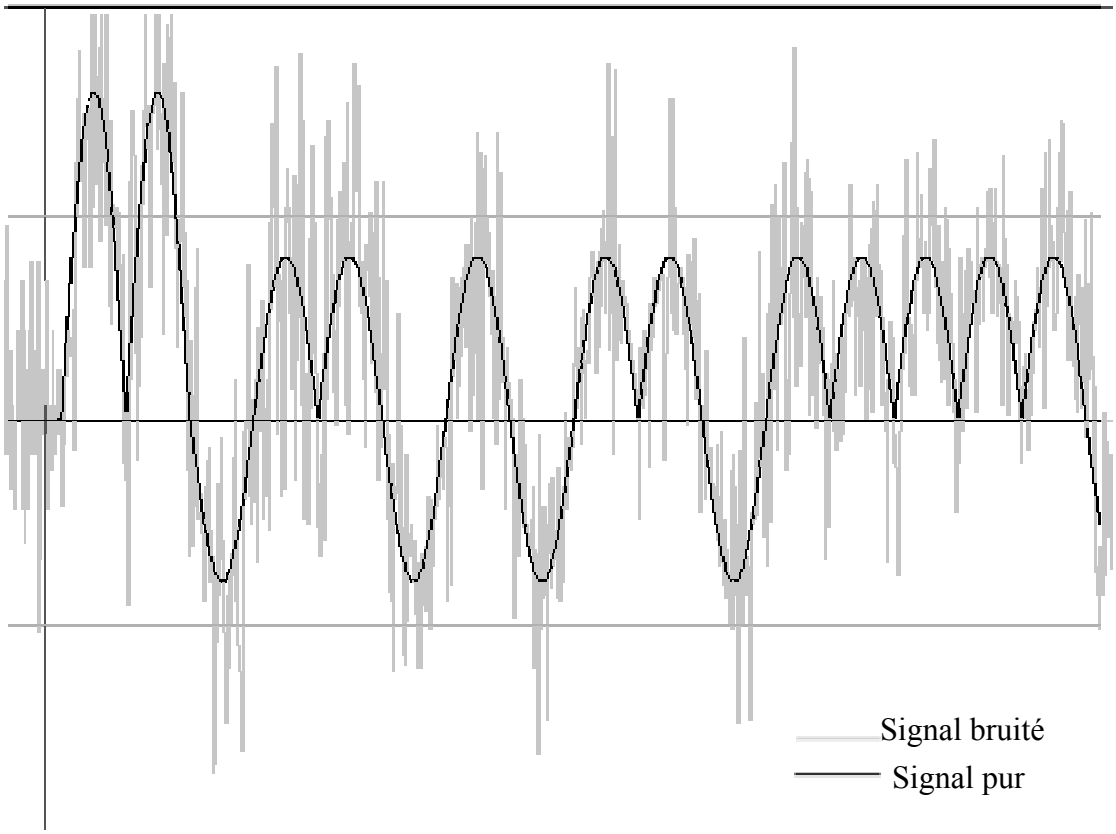


C'est nettement moins bon, mais le programme fonctionne toujours

Augmentons encore l'amplitude du bruit :



Cette fois, tout le signal ressemble vraiment à du bruit : on a du mal à distinguer les pics et surtout les zones de blancs n'apparaissent plus. C'est intéressant de le remarquer, car c'était un des avantages évoqués pour la confidentialité (II.4.2). En faisant un zoom, on peut quand même retrouver les caractéristiques du signal de départ :



Maintenant (connaissant le résultat), il est plus facile de repérer les pics. Nous avons été initialement surpris par le fait que le programme donnait encore les bons messages pour chaque utilisateur. Mais, après réflexion, ce résultat était prévisible en raison du type de bruit utilisé. Il s'agit d'un bruit de « fréquence » plus élevée que la fréquence des « chips ». Par conséquent, le bruit, qui est de moyenne nulle est éliminé lorsqu'on fait les calculs de corrélation. Qualitativement, le bruit déplace le signal tantôt vers le haut, tantôt vers le bas, ceci un grand nombre de fois par bloc, et la résultante est nulle.

Bibliographie et remerciements

- *Digital Communications*, John G. Proakis – third edition

- *The ARRL Handbook for Radio Amateurs* - Seventy-Eigth Edition

- *An Overview of CDMA Evolution toward Wideband CDMA,*
  Ramjee Prasad; Tero Ojanperä.
  http://www.comsoc.org/pubs/surveys/4q98issue/prasad.html

- *Spreading Sequences,* Byoungjo Choi
  http://www-mobile.ecs.soton.ac.uk/bjc97r/pnseq-1.1/index.html

- *Signal Processing for Communications*
  http://cas.et.tudelft.nl/education/et4147/

- CDMA development group
  http://www.cdg.org/

Nous tenons à remercier tout particulièrement M. Byoungjo CHOI. pour son programme disponible sur Internet qui génère les codes de Gold. Nous le remercions également pour les réponses qu'il a eu l'amabilité de nous donner.

Listing du programme

(joint dans la pochette)

Les modules que nous avons indiqués en gras sont les plus intéressants.

Code commun

- **common.h , common.cpp** : fonctions partagées par CDMAClient et CDMAServer pour la génération des codes, calculs de corrélation, écriture de demi-sinusoïdes.
- defs.h **:** définitions de constantes.
- mseq.h, mseq.cpp, gold.h, gold.cpp, ogold.h, ogold.cpp : utilisés lors de la génération des codes.

Code de CDMAClient

- **CDMAClientDlg.h, CDMAClientDlg.cpp** : fenêtre principale avec toutes les opérations relatives au multiplexage.
- CDMAclient.h, CDMAClient.cpp : lancement du programme.
- DeviceDlg.h, DeviceDlg.cpp : selection du périphérique de sortie.
- PlaySound.h, PlaySound.cpp : initialisation du périphérique de sortie, création du tampon.

Code de CDMAServer

- **CDMAServerDlg.h, CDMAServerDlg.cpp** : fenêtre principale, avec toutes les opérations relatives au démultiplexage.
- CDMAServer.h, CDMAServer.cpp : lancement du programme.
- DeviceDlg.h, DeviceDlg.cpp : selection du périphérique de sortie.
- CaptureSound.h, CaptureSound.h : initialisation du périphérique d'acquisition, création de tampon.