AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Approche déterministe de l'acquisition comprimée et la reconstruction des signaux issus de capteurs intelligents distribués

THÈSE

présentée et soutenue publiquement le 9 novembre 2015

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention systèmes électroniques)

par

Andrianiaina Ravelomanantsoa

Composition du jury

Président :	M. Patrick GARDA, Professeur, LIP6, Paris
Rapporteurs :	M. Christophe LALLEMENT, Professeur, ICube, Strasbourg
	M. Ian O'CONNOR, Professeur, INL, Lyon
Examinateurs :	M. Hassan RABAH, Professeur, IJL, Université Lorraine, Directeur de thèse
	M. Amar ROUANE, Professeur, IJL, Université de Lorraine, Co-Directeur de thèse
Invité :	M. Nicolas FERVEUR, Ingénieur en électronique, TEA, Nancy

Laboratoire Institut Jean Lamour (UMR7198) - Equipe MAE 406

Département 4 : Nanomatériaux, Électronique et Vivant

Remerciements

Je tiens à remercier tout particulièrement mes directeurs de thèse, Monsieur Hassan RABAH et Monsieur Amar ROUANE, pour la confiance qu'ils m'ont témoignée tout au long de ces années de travail, pour leurs participations, pour leurs encouragements quotidiens et pour leurs précieux conseils au niveau scientifique.

J'adresse aussi mes sincères remerciements à Monsieur Christophe LALLEMENT, Professeur à l'ICube Strasbourg, ainsi qu'à Monsieur Ian O'CONNOR, Professeur à l'INL Lyon, pour l'attention qu'ils ont accordée à la lecture de ce mémoire de thèse, et pour avoir accepté d'en être les rapporteurs. Je tiens également à remercier Monsieur Patrick GARDA, Professeur au LIP6 Paris, pour avoir accepté d'examiner ce travail et de participer au jury.

Je remercie l'entreprise TEA (Technologie Ergonomie Appliquées) et son équipe pour leur collaboration, en particulier Monsieur Nicolas FERVEUR et Monsieur Landry COLLET.

Je remercie également tous les membres et collègues de l'équipe MAE 406 pour leur accueil, leur amabilité et surtout la convivialité dont ils ont fait preuve à mon égard, en particulier, Patrice ROTH, Pierre SCHMITT et Christine DAUDENS pour leurs conseils administratifs. Et bien sûr, Yves BERVILLER et Slaviša JOVANOVIĆ pour leur écoute et leurs conseils.

Pour finir, je remercie mon épouse, mes trois filles et ma famille pour tous leurs aides et leurs soutiens qui m'ont servi pour réussir cette thèse. Sans oublier mes amis proches, en particulier Edward (Ted) MCRAE et Jean-François LUTZ.

Andrianiaina RAVELOMANANTSOA
Octobre 2015

Je dédie cette thèse
à mon épouse, Vola, et mes trois filles Felana, Malala et Sarah.

Sommaire

Table des figures **ix**

Liste des tableaux **xiii**

Acronymes et abréviations **xv**

Introduction générale

1 Contexte et problématique 1
2 Contributions 3
3 Plan du manuscrit 3

Chapitre 1
Réseau de capteurs sans fil sur le corps humain

1.1 Introduction 6
1.2 Applications 6
1.3 Architecture d'un réseau de capteurs sans fil sur le corps humain 7
1.3.1 Les nœuds 8
1.3.2 Le nœud central ou coordinateur 16
1.4 Discussion 16
1.5 Conclusion 19

Chapitre 2
L'Acquisition comprimée

2.1 Introduction 22
2.2 Parcimonie 22
2.3 Acquisition et reconstruction d'un signal avec l'acquisition comprimée 23
2.3.1 Phase d'acquisition 24
2.3.2 Phase de reconstruction 24
2.3.3 Discussion 25
2.4 Matrices de mesure 25
2.4.1 Propriétés 25
2.4.2 Exemples de matrices de mesure 28
2.5 Algorithmes de reconstruction 28
2.5.1 Relaxation convexe 28

2.5.2 Poursuite gloutonne . . . 29
2.5.3 Relaxation convexe vs. poursuite gloutonne . . . 29
2.5.4 Quelques algorithmes gloutons . . . 29
2.6 Critères de solvabilité . . . 31
2.6.1 Critère de solvabilité en fonction de la cohérence . . . 31
2.6.2 Critère de solvabilité en fonction du restricted isomtry property . . . 32
2.6.3 Critère de solvabilité en fonction de la cohérence mutuelle . . . 32
2.7 Domaine de parcimonie . . . 32
2.7.1 Transformée en cosinus discrète . . . 33
2.7.2 Transformée en ondelettes . . . 34
2.7.3 Seuillage . . . 35
2.8 Métriques d'évaluation de la qualité de reconstruction . . . 35
2.9 Chaîne d'acquisition comprimée . . . 36
2.9.1 Encodeur analogique . . . 37
2.9.2 Encodeur numérique . . . 43
2.10 Conclusion . . . 45

Chapitre 3
Approche déterministe de l'acquisition comprimée

3.1 Introduction . . . 50
3.2 Matrice de mesure proposée . . . 51
3.2.1 Construction de la matrice de mesure proposée . . . 51
3.2.2 Complexité en termes de calcul et allocation mémoire . . . 54
3.2.3 Performances de la matrice de mesure . . . 55
3.3 Algorithme de reconstruction proposé . . . 64
3.3.1 Description de l'algorithme . . . 64
3.3.2 Performances de l'algorithme . . . 66
3.4 Comparaison entre la méthode proposée et le codage par transformation . . . 72
3.4.1 Complexité de l'encodage . . . 73
3.4.2 Qualité de reconstruction . . . 74
3.5 Conclusion . . . 76

Chapitre 4
Évaluation système - Réseau de capteurs à base de l'acquisition comprimée

4.1 Introduction . . . 80
4.2 Modélisation du réseau de capteurs utilisant l'acquisition comprimée . . . 80
4.2.1 Modèle du nœud . . . 81
4.2.2 Modèle SPICE de l'encodeur . . . 90

4.2.3 Routeur . . . 91
4.2.4 Décodeur . . . 92
4.3 Résultats de simulation . . . 93
4.3.1 Évaluation de la qualité de reconstruction . . . 93
4.3.2 Validation du modèle SPICE de l'encodeur . . . 97
4.3.3 Influence de la valeur de $\mathcal{M}$ sur la qualité et le temps de reconstruction 98
4.3.4 Impact de la résolution du convertisseur analogique numérique sur la qualité de reconstruction . . . 100
4.3.5 Évaluation de la bande passante et de la consommation . . . 101
4.4 Conclusion . . . 104

Chapitre 5
Validation expérimentale

5.1 Introduction . . . 106
5.2 Encodeur numérique . . . 106
5.2.1 Implémentation de l'encodeur . . . 107
5.2.2 Implémentation de l'algorithme proposé . . . 110
5.2.3 Résultats . . . 111
5.3 Encodeur analogique . . . 115
5.3.1 Implémentation de l'encodeur . . . 116
5.3.2 Signaux de contrôle . . . 116
5.3.3 Convertisseur analogique numérique . . . 117
5.3.4 Reconstruction des signaux . . . 117
5.3.5 Résultats . . . 118
5.4 Conclusion . . . 121

Conclusion générale et perspectives

Annexes

Annexe A
Modèle SystemC-AMS

A.1 Modèle SystemC-AMS du capteur . . . 127
A.2 Modèle SystemC-AMS de l'amplificateur . . . 129
A.3 Modèle SystemC-AMS du convertisseur analogique numérique . . . 130
A.4 Modèle SystemC-AMS de l'encodeur analogique . . . 132
A.4.1 Modèle du sommateur non inverseur . . . 132
A.4.2 Modèle du suiveur . . . 134
A.4.3 Modèle ELN de l'encodeur . . . 135

A.4.4 Modèle TDF de l'encodeur 137
A.5 Modèle SystemC-AMS du microcontrôleur 139
A.5.1 Type de données représentant un paquet 139
A.5.2 Modèle du générateur de signaux de contrôle 142
A.5.3 Modèle du microcontrôleur 144
A.6 Modèle SystemC-AMS du module RF 148
A.7 Modèle SystemC-AMS d'un nœud 150
A.8 Modèle SystemC-AMS du routeur 155
A.9 Modèle SystemC-AMS décodeur 158

Publications personnelles

Bibliographie

Table des figures

1.1 Architecture typique d'un réseau de capteurs sans fil sur le corps humain. . . 8
1.2 Architecture typique d'un nœud du réseau. 9
1.3 Différentes topologies des nœuds. 12
1.4 Consommation d'énergie moyenne de quelques microcontrôleurs et modules radiofréquences en fonction du débit des données [HPB+09]. 15

2.1 Acquisition et reconstruction d'un signal avec l'acquisition comprimée. 23
2.2 Signal ECG et sa transformée DCT. 34
2.3 Signal EMG et sa transformée DCT. 34
2.4 Transformée en ondelettes. 35
2.5 Chaîne d'acquisition de données classique. µP : microprocesseur [Ere05]. f_{ACQ} : fréquence d'acquisition. f_N : fréquence de Nyquist. 37
2.6 Place d'un encodeur analogique dans une chaîne d'acquisition de données. . . 37
2.7 Architecture du random demodulator. GNP : générateur de nombre pseudo aléatoire. 38
2.8 Architecture du random modulation pre-integrator comportant 4 canaux [YBM+12]. 39
2.9 Principe de l'échantillonneur non uniforme. 40
2.10 Architecture du Compressed Sensing Analog Front-End [GAD+14]. 42
2.11 Place d'un encodeur numérique dans une chaîne d'acquisition de données. . . 43
2.12 Bloc diagramme et implémentation de l'encodeur numérique proposé par Chen *et al.* [CCS12]. 44

3.1 Cohérence μ entre la matrice Ψ_{IDCT} et quelques matrices de mesure, pour $\mathcal{N} = 500$ et $\mathcal{M}$ variant de 25 à 475. 53
3.2 Cohérence μ entre la matrice Ψ_{IHWT} et quelques matrices de mesure, pour $\mathcal{N} = 500$ et $\mathcal{M}$ variant de 25 à 475. 54
3.3 Compression et reconstruction de la transformée DCT α générée aléatoirement. 56
3.4 Taux de succès pour $\mathcal{M} = 25$. 56
3.5 Taux de succès pour $\mathcal{M} = 35$. 57
3.6 Taux de succès pour $\mathcal{M} = 45$. 57
3.7 Nombre de mesures $\mathcal{M}$ nécessaires pour reconstruire la transformée DCT avec une erreur $\varepsilon \leq 10^{-6}$, pour une valeur constante de $\mathcal{N}$ égale à 500. 58
3.8 Compression et reconstruction du signal ECG provenant de la base de donnée MIT-BIH Arrhythmia du site Physionet [Phy]. 59
3.9 SNR en fonction du taux de compression. 60

3.10 PRD en fonction du taux de compression. 60
3.11 Signal ECG original (noir) et celui reconstruit (rouge) avec un taux de compression de 80 % ($\mathcal{M} = 100$ et $\mathcal{N} = 500$). Le signal était compressé avec la matrice de mesure déterministe proposée. 61
3.12 Signal ECG original (noir) et celui reconstruit (rouge) avec un taux de compression de 90 % ($\mathcal{M} = 50$ et $\mathcal{N} = 500$). Le signal était compressé avec la matrice de mesure déterministe proposée. 61
3.13 Compression et reconstruction du signal EMG provenant de la base de donnée Physionet [Phy]. 62
3.14 SNR en fonction du taux de compression. 63
3.15 PRD en fonction du taux de compression. 63
3.16 Signal EMG original (noir) et celui reconstruit (rouge) avec un taux de compression de 70 % ($\mathcal{M} = 150$ et $\mathcal{N} = 500$). Le signal était compressé avec la matrice de mesure déterministe proposée. 64
3.17 Les 8 premières colonnes de la matrice $\mathbf{\Psi}_{\text{IDCT}}$ (en bleu) et celles de $\mathbf{A} = \mathbf{\Phi}_{\text{DBBD}}\mathbf{\Psi}_{\text{IDCT}}$ (en rouge) pour $\mathcal{M} = 16$ et $\mathcal{N} = 64$, respectivement notées par $\boldsymbol{\psi}_{\text{i}\in\{1,\ldots,8\}}$ et $\mathbf{a}_{\text{i}\in\{1,\ldots,8\}}$. 65
3.18 Taux de succès pour $\mathcal{M} = 25$. 67
3.19 Taux de succès pour $\mathcal{M} = 35$. 67
3.20 Taux de succès pour $\mathcal{M} = 45$. 68
3.21 Nombre de mesures $\mathcal{M}$ nécessaires pour reconstruire la transformée DCT avec une erreur $\boldsymbol{\varepsilon} \leq 10^{-6}$. Les régressions linéaires ont respectivement les équations suivantes : $\text{RL}_{\text{Proposé}} : \mathcal{M} = \mathcal{K}$, $\text{RL}_{\text{OMP}} : \mathcal{M} = \mathcal{K} + 0.8$ et $\text{RL}_{\text{StOMP}} : \mathcal{M} = 1.2\mathcal{K} + 0.067$. 68
3.22 SNR en fonction du taux de compression pour les signaux ECG. 69
3.23 PRD en fonction du taux de compression pour les signaux ECG. 69
3.24 PRD en fonction du taux de compression pour le signal EMG. 70
3.25 SNR en fonction du taux de compression pour le signal EMG. 70
3.26 Temps de reconstruction. 71
3.27 Comparaison entre la méthode proposée et le codage par transformation. . . 72
3.28 PRD en fonction du taux de compression pour le signal ECG. 74
3.29 SNR en fonction du taux de compression pour le signal ECG. 74
3.30 PRD en fonction du taux de compression pour le signal EMG. 75
3.31 SNR en fonction du taux de compression pour le signal EMG. 75
3.32 Comparaison entre la méthode proposée et le codage par transformation. . . 76

4.1 Architecture du réseau. En bleu *simple_initiator_socket*, en rouge *simple_target_socket*. 81
4.2 Modèle d'un nœud sans encodeur AC. 81
4.3 Modèle d'un nœud avec encodeur AC. 81
4.4 Modèle du capteur. 82

4.5 Modèle de l'amplificateur. 82
4.6 Modèle haut niveau de l'encodeur. 83
4.7 Chronogrammes des signaux de l'encodeur. 84
4.8 Exemple de raffinement du modèle de l'encodeur. 85
4.9 Chronogramme des signaux de l'encodeur pour une tension d'entrée constante. 86
4.10 Modèle du convertisseur analogique numérique. 87
4.11 Modèle du microcontrôleur. 87
4.12 Modèle du transmetteur radiofréquence. 89
4.13 Schéma du modèle SPICE de l'encodeur analogique. 90
4.14 Tensions d'entrée (bleu) et de sortie (rouge) de l'encodeur pour une tension d'entrée constante de 0.3 V. 91
4.15 Architecture d'un nœud intégrant le modèle SPICE de l'encodeur. 92
4.16 Signal EMG original (noir) et celui reconstruit (rouge) avec un taux de compression de 75.0 %. 96
4.17 Signal ECG original (noir) et celui reconstruit (rouge) avec un taux de compression de 83.3 %. 96
4.18 Signal EEG original (noir) et celui reconstruit (rouge) avec un taux de compression de 75.5 %. 97
4.19 Consommation en courant du module NRF24L01 en mode transmission. . . 102
4.20 Format d'un paquet du module NRF24L01. 102

5.1 Architecture du réseau de capteurs utilisé pendant la validation expérimentale.106
5.2 Architecture d'un nœud utilisé pendant la validation expérimentale. 107
5.3 Nœuds capturant les signaux ECG et EMG. 109
5.4 Interface de contrôle des nœuds. 110
5.5 Signal ECG original (noir) et celui reconstruit (rouge) avec un taux de compression de 87.5 %, pour $\mathcal{M} = 16$ et $\mathcal{N} = 128$. 112
5.6 Signal EMG original (noir) et celui reconstruit (rouge) avec un taux de compression de 75 %, pour $\mathcal{M} = 16$ et $\mathcal{N} = 64$. 113
5.7 Affichage en temps réel des signaux ECG. En rouge le signal ECG provenant du nœud qui n'implémente pas d'encodeur AC. En vert celui provenant du nœud qui implémente un encodeur AC. 115
5.8 Prototype de l'encodeur analogique. 116
5.9 Fenêtre principale de l'encodeur. 117
5.10 Le sous-onglet acquisition de l'interface. 119
5.11 L'onglet visualisation de l'interface. Affichage en temps réel d'un signal ECG décodé. 120
5.12 Signal ECG original (noir) et celui reconstruit (rouge) avec un taux de compression de 87.5 %, pour $\mathcal{M} = 16$ et $\mathcal{N} = 128$. 121

Liste des tableaux

1.1 Description de quelques signaux traités dans un réseau de capteurs sans fil sur le corps humain [PB10]. 9
1.2 Plage de valeurs et bande de fréquences de quelques signaux physiologiques [KY10]. 10
1.3 Résolution du convertisseur analogique numérique et fréquence d'acquisition de quelques signaux physiologiques [LBM+11]. 11
1.4 Quelques normes utilisées pour la communication radiofréquence [PB10]. . . 13
1.5 Distribution de la consommation typique d'énergie dans un nœud [GAD+14]. 15

2.1 Qualité de reconstruction en fonction du PRD [ZCK00]. 36
2.2 Tableau récapitulatif des différents encodeurs proposés. 46

3.1 Complexité en termes de calcul et d'allocation mémoire dans le cas d'un encodeur numérique. 54
3.2 Borne supérieure de $\mathcal{K}$ avec l'OMP dans le cas d'une matrice de mesure aléatoire pour $\mathcal{N} = 500$. 58
3.3 Complexité de l'encodage. 73

4.1 Signaux de tests utilisés lors de la validation du modèle SystemC-AMS. . . . 93
4.2 Caractéristiques des nœuds utilisés pour évaluer la qualité de reconstruction. 93
4.3 PRD et SNR en fonction du taux de compression pour le signal EMG. 94
4.4 PRD et SNR en fonction du taux de compression pour le signal ECG. 95
4.5 PRD et SNR en fonction du taux de compression pour le signal EEG. 95
4.6 PRD et SNR en fonction du taux de compression pour le signal EMG. 98
4.7 Impact de la valeur de $\mathcal{M}$ sur la qualité de reconstruction et le temps de reconstruction. 99
4.8 Caractéristiques des nœuds utilisés pour évaluer l'impact de la résolution du CAN. 100
4.9 Impact de la valeur de la résolution du CAN sur la qualité de reconstruction. 101
4.10 Débit du réseau avec et sans les encodeurs AC. 103
4.11 Consommation des modules RF. 103

5.1 PRD et SNR en fonction du taux de compression pour le signal ECG. 108
5.2 PRD et SNR en fonction du taux de compression pour le signal EMG. 108
5.3 Caractéristiques des nœuds utilisés pendant la validation expérimentale. . . . 109
5.4 Occupation mémoire des microcontrôleurs. 109
5.5 Qualité de reconstruction. 111

5.6 Consommation du module RF pendant un enregistrement de 1 seconde. . . . 113
5.7 Temps d'attente Δt. 114
5.8 Retard introduit par l'encodeur AC. 114
5.9 Qualité du signal reconstruit en fonction du taux de compression. 120

Acronymes et abréviations

AC :	acquisition comprimée
AES :	advanced encryption standard
AFE :	analog front end
AIC :	analog to information converter
ASIC :	application specific integrated circuit
BP :	basis pursuit
BPDN :	basis pursuit denoising
BPIC :	basis pursuit with inequality constraints
CAN :	convertisseur analogique-numérique
CF :	compression factor
CoSaMP :	compressive sampling matching pursuit
CR :	compression ratio
CRC :	cyclic redundancy check
CS-AFE :	compressed sensing analog front-end
DCT :	discrete cosine transform
DWT :	discrete wavelet transform
ECG :	électrocardiogramme
EEG :	électroencéphalogramme
ELN :	electrical linear network
EMG :	électromyogramme
FPGA :	field-programmable gate array
GPRS :	general packet radio service
GSM :	global system for mobile communications
GSR :	galvanic skin response
HDL :	hardware description language
IDCT :	inverse discrete cosine transform
IHT :	iterative hard thresholding

IHWT : inverse Haar wavelet transform

ISM : industriel, scientifique et médical

LSF : linear signal flow

MAC : media access control

MICS : medical implant communications service

MP : matching pursuit

NUS : non-uniform sampler

OMP : orthogonal matching pursuit

PDA : personal digital assistant

PRD : percentage root-mean-square deviation

RD : random demodulator

RESP : respiration

RF : radiofréquence

RIC : restricted isometry constant

RIP : restricted isometry property

RMPI : random modulation pre-integrator

RTL : register transfer level

SD : secure digital

SNR : signal to noise ratio

StOMP : stagewise orthogonal matching pursuit

TDF : timed dataflow

TDMA : time division multiple access

TEG : thermoelectric generator

TLM : transaction level modeling

UMTS : universal mobile telecommunications system

USB : universal serial bus

VHDL : very high speed integrated circuit (VHSIC) hardware description language (HDL)

VHSIC : very high speed integrated circuit

WBAN : wireless body area network

WiMAX : worldwide interoperability for microwave access

WMTS : wireless medical telemetry services

WPAN : wireless personal area network

WSN : wireless sensor network

Introduction générale

1 Contexte et problématique

Les nouvelles technologies en microélectronique et circuits intégrés, en conception de système sur puce, en communication sans fils et en capteurs intelligents ont permis la réalisation d'une nouvelle technologie de réseau sans fil appelé « *wireless body area network (WBAN)* ». Le WBAN est un réseau de dispositifs électroniques miniatures et intelligents, appelés nœuds, placés aux alentours ou à l'intérieur du corps humain. Chaque nœud est doté d'un capteur, d'un convertisseur analogique-numérique (CAN), d'une unité de calcul (généralement un microcontrôleur), d'un module radiofréquence (RF) et d'une batterie pour alimenter l'ensemble du système. Les nœuds récupèrent les paramètres physiologiques d'une personne et les caractéristiques de l'environnement qui l'entoure. Les données sont transmises vers un centre de contrôle pour être stockées et analysées. Les données récupérées vont permettre de surveiller et d'analyser, à distance et en temps réel, l'état de la personne [UHB+12].

Le WBAN est utilisé dans de nombreux domaines, surtout dans la télémédecine et la télésurveillance. Ces applications vont aider les personnels de santé à mieux diagnostiquer les maladies. Elles vont améliorer les conditions de vie des patients en leur accordant plus de mobilité et de confort. Elles vont diminuer les frais de santé des patients soufrant de maladies chroniques qui nécessitent des traitements et des suivis de longue durée. Par ailleurs, la télésurveillance facilite l'assistance à domicile des personnes âgées ou à mobilité réduite, et apporte une solution aux problèmes engendrés par le vieillissement de la population. Le WBAN rend possible le suivi des sportifs et des professionnels travaillant dans les milieux hostiles comme par exemple les centrales nucléaires, les laboratoires de chimie, etc.

Comparé aux autres réseaux sans fils, comme par exemple le réseau personnel sans fil ou « *wireless personal area network (WPAN)* » dont la portée peut atteindre une dizaine de mètres (cas d'une communication entre un ordinateur et une imprimante locale), le WBAN est conçu pour fonctionner à proximité du corps humain, et sa portée est alors limitée à seulement quelques mètres, typiquement entre 1 et 2 m. De plus, le WBAN utilise uniquement des dispositifs portatifs alors que le WPAN peut interconnecter des appareils pas forcément portatifs. De même, si le WBAN peut être considéré comme un réseau de capteurs sans fils ou « *wireless sensor network (WSN)* » , les contraintes associées, surtout en termes de mobilité, de fiabilité et de sécurité, ne sont pas les mêmes. Latré *et al.* [LBM+11] ont fait une étude détaillée sur la différence entre le WBAN et le WSN.

Les nœuds sont surtout soumis à une contrainte énergétique importante puisque la miniaturisation a réduit les dimensions de leurs batteries. La bande passante du support de transmission limite aussi la capacité du réseau. Recharger régulièrement les batteries des nœuds peut entrainer une gêne vis-à-vis des utilisateurs. Pour certains types d'applications, en particulier les nœuds implantés dans le corps humain, le rechargement des batteries est difficile. Cette contrainte énergétique a amené les scientifiques à chercher une solution pour diminuer la consommation des nœuds. Les études faites ont montré que, parmi les modules présents dans un nœud, c'est surtout le module RF qui consomme le plus. L'énergie consommée est directement proportionnelle à la quantité de données transmises. Une solution parmi celles proposées consiste à compresser les données avant de les transmettre. La compression des données économise aussi la bande passante du support de transmission et augmente la capacité du réseau.

Les signaux issus des capteurs sont échantillonnés et numérisés par le CAN à la fréquence de Nyquist. Puis, le microcontrôleur applique un algorithme de compression sur les données numérisées. Finalement, le module RF transmet les données compressées vers le centre de contrôle. Cette approche n'est pas adaptée pour le WBAN particulièrement à cause de la puissance de calcul requise et de la consommation qui en résulterait [FW14]. Récemment, le nouveau paradigme consiste à utiliser l'acquisition comprimée (AC). C'est une technique introduite dans le traitement du signal par Donoho [Don06] et Candès *et al.* [CRT06]. L'AC s'appuie principalement sur deux principes : la parcimonie et l'incohérence.

L'AC est adaptée pour compresser et reconstruire des signaux dits compressibles, c'est-à-dire ceux qui ont des représentations parcimonieuses par rapport à certaines bases ou certains domaines. Au lieu d'échantillonner et numériser les signaux compressibles à la fréquence de Nyquist puis d'appliquer un algorithme de compression, l'AC les capture directement à une fréquence en dessous de celle de Nyquist sous forme compressée. En revanche, contrairement à la phase de mesure, la phase de reconstruction de l'AC nécessite un calcul assez intense. La complexité de la phase de mesure est déportée vers la phase de reconstruction. Cependant, le centre de contrôle est moins contraignant par rapport aux nœuds et a les ressources nécessaires pour effectuer cette phase de reconstruction.

Puisque la plupart des signaux physiologiques sont compressibles, plusieurs travaux se portaient sur l'application de l'AC dans le WBAN en général [CRV12, BRK12]. Certains travaux utilisaient seulement l'AC pour compresser et reconstruire un type de signal comme l'électrocardiogramme (ECG) [MKAV11], l'électroencéphalogramme (EEG) [ZJMR13] ou bien l'électromyogramme (EMG) [DAGA12].

Bien que l'AC soit une solution attrayante, il reste encore des points à améliorer :

- La construction de la matrice de mesure facilitant la réalisation pratique de l'encodeur qui effectue la phase de mesure de l'AC. En effet, la complexité de l'encodeur dépend du choix de la matrice de mesure. Au début, des matrices aléatoires ont été utilisées. Pour faciliter l'implémentation, au lieu d'utiliser des matrices aléatoires, des matrices

déterministes ont été proposées récemment.

- Le développement d'algorithmes de reconstruction à faible complexité et efficaces, c'est-à-dire des algorithmes pouvant reconstruire le signal seulement avec peu d'échantillons.
- La recherche d'un domaine ou d'une base permettant d'avoir un degré de parcimonie très faible. En effet, plus le signal est parcimonieux, plus il est compressible. Seulement quelques échantillons seront nécessaires pour le reconstruire correctement. Généralement, les domaines de transformée temps-fréquence sont les plus utilisés.

2 Contributions

L'objectif principal de cette thèse est d'apporter des solutions pour résoudre les problèmes énoncés par les deux premiers points ci-dessus.

Premièrement, nous avons proposé une matrice de mesure déterministe pour réduire la complexité de l'encodeur et faciliter son implémentation. Par rapport aux autres matrices, à notre connaissance, celle que nous proposons est à la fois la plus simple et la plus facile à implémenter côté matériel.

Deuxièmement, grâce à cette matrice de mesure, nous avons proposé un algorithme de reconstruction moins complexe et plus rapide par rapport à « *l'orthogonal matching pursuit (OMP)* ». L'algorithme de reconstruction proposé effectue un seuillage dans le domaine de la transformée en cosinus discrète ou « *discrete cosine transform (DCT)* » .

De plus, avant de valider expérimentalement la méthode proposée, nous avons développé un modèle exécutable au niveau système d'un WBAN implémentant la méthode proposée avec le langage SystemC-AMS. Ce modèle exécutable nous a permis de vérifier et de valider les fonctionnalités de la méthode proposée en amont de la phase de développement.

Nous avons implémenté la méthode proposée dans un WBAN développé et commercialisé par l'entreprise TEA (Technologie Ergonomie Appliquées). Ce travail collaboratif avec un industriel nous a permis de valider expérimentalement la version numérique de l'encodeur et l'algorithme de reconstruction. Nous avons aussi réalisé une version analogique de l'encodeur en utilisant des composants standards.

3 Plan du manuscrit

Le manuscrit est structuré de la manière suivante :

Le premier chapitre présentera plus en détail l'architecture d'un WBAN et les technologies mises en œuvre ainsi que les principaux domaines d'application. Les contraintes auxquelles le WBAN est soumis seront exposées et les différentes solutions proposées vont être énumérées.

Le deuxième chapitre introduira les principes de base de l'AC. Les propriétés que la matrice de mesure devrait satisfaire seront citées, ainsi que quelques algorithmes de recons-

truction couramment utilisés. Les différentes implémentations d'encodeurs analogiques et numériques existants seront présentées.

Le troisième chapitre présentera la construction de la matrice de mesure proposée et l'évaluation de ses performances par rapport aux autres matrices. Le principe de l'algorithme de reconstruction proposé sera détaillé. Puis les performances de la méthode proposée, la combinaison de la matrice de mesure et de l'algorithme de reconstruction, seront comparées avec celles d'une méthode classique de seuillage dans le domaine de la DCT.

Le quatrième chapitre détaillera la description du modèle exécutable au niveau système d'un WBAN implémentant la méthode proposée avec le langage SystemC-AMS. Ce modèle va permettre de vérifier et de valider les fonctionnalités de la méthode proposée en amont de la phase de développement.

Le cinquième et dernier chapitre présentera la validation expérimentale de la méthode proposée.

Nous terminerons ce manuscrit par une conclusion générale qui va établir un bilan sur les travaux effectués et mettre en exergue les perspectives ouvertes à l'issue de ces travaux.

Chapitre 1

Réseau de capteurs sans fil sur le corps humain

Sommaire

1.1 Introduction 6
1.2 Applications 6
1.3 Architecture d'un réseau de capteurs sans fil sur le corps humain 7
1.3.1 Les nœuds 8
1.3.2 Le nœud central ou coordinateur 16
1.4 Discussion 16
1.5 Conclusion 19

1.1 Introduction

Le WBAN est un réseau sans fil dédié à la surveillance de l'état du corps humain. Dans ce réseau, des dispositifs électroniques miniatures munis de différents types de capteurs, appelés aussi nœuds, sont disposés aux alentours ou à l'intérieur du corps humain. Les nœuds collectent des données physiologiques comme l'ECG, l'EEG, la tension artérielle, l'EMG, la respiration (RESP), etc [LBM+11]. Les nœuds peuvent être placés sur les vêtements (« *smart clothes* ») sous forme de patch dépendant de l'application et de la technologie mise en œuvre. Ils peuvent aussi être intégrés dans les assistants numériques personnels ou « *personal digital assistant (PDA)* », dans les téléphones (« *smart phones* »), ou bien dans les montres (« *smart watches* »). Dans certains cas, les nœuds peuvent être même placés dans le corps humain, comme dans le cas des implants.

La surveillance peut se faire soit en temps réel, soit en temps différé. Dans une surveillance en temps réel, les données recueillies par les nœuds sont transmises en temps réel vers un centre médical ou un centre de surveillance, où des personnels spécialisés vont faire les analyses et vont prendre des décisions en fonction de l'état du patient. Pour une surveillance en temps différé, les données collectées sont sauvegardées localement dans une mémoire, par exemple dans une carte « *secure digital (SD)* ». Puis les données peuvent être analysées ultérieurement.

Le support de communication sans fil apporte deux avantages majeurs par rapport au filaire :

a) Il améliore le confort des patients en leur donnant beaucoup plus de mobilité. En effet, le support filaire limite les mouvements des patients.

b) Il est moins coûteux.

Par contre en termes de sécurité, il y a beaucoup plus de risques d'attaque avec le support sans fil. La sécurité est un aspect important puisque les données qui circulent dans le réseau sont strictement privées et confidentielles.

La première partie de ce chapitre va énumérer les avantages que le WBAN peut apporter ainsi que ses domaines d'applications potentielles. L'architecture du réseau et les technologies mises en œuvre seront détaillées dans la deuxième partie. La dernière partie de ce chapitre évoquera les défis majeurs auxquels le WBAN est confronté. Les solutions proposées pour faire face à ces problèmes y seront aussi présentées.

1.2 Applications

Le WBAN est principalement utilisé dans le domaine de la télémédecine. Les différents nœuds déployés aux alentours ou à l'intérieur du corps humain vont permettre aux médecins de surveiller à distance et en temps réel les états de santé des patients. Les incidents qui nécessitent une intervention immédiate comme les crises cardiaques et épileptiques peuvent

être détectés et même évités par la surveillance en permanence des activités du cœur et du cerveau. Le WBAN apporte beaucoup plus de liberté et de mobilité aux patients. En effet grâce à la télésurveillance, les personnes atteintes de maladies chroniques ou nécessitant un suivi de long durée, ne sont pas obligées de séjourner à l'hôpital ou dans un centre médical. Ces personnes peuvent être domiciliées selon leurs choix et poursuivre leurs activités journalières. Les nœuds vont transmettre régulièrement leurs états de santé vers le centre médical. La télésurveillance permet ainsi de réduire les frais de santé qui ne cessent d'augmenter, en particulier le coût d'hospitalisation. Cette hausse des frais médicaux va atteindre les 20 % en 2020.

Le WBAN apporte aussi une solution face aux problèmes engendrés par le vieillissement de la population notamment dans les pays développés. Les prévisions faites aux États-Unis ont montré que d'ici 2025 le nombre de personnes âgées, de 65 à 84 ans, va doubler de 35 à 70 millions. Il n'y aura pas assez de foyers ou maisons de retraite pour accueillir ces personnes. Grâce à la télésurveillance du WBAN, les personnes âgées ne sont pas obligées d'habiter dans des foyers ou dans des maisons de retraite. Elles peuvent se loger en toute tranquillité chez elles. Les nœuds munis de modules de géolocalisation vont transmettre leurs états de santé et leurs positions vers un centre surveillance. En cas d'anomalie, une application qui tourne dans ce centre de surveillance va alerter les secours.

Le WBAN est utilisé par les sportifs de haut niveau, aussi bien que les amateurs, pour le suivi et l'amélioration de leurs performances. Des enregistrements journaliers de leurs paramètres physiologiques, comme le battement du cœur, la tension artérielle ou bien la posture, vont permettre d'adapter correctement les exercices physiques et de rendre les entraînements plus efficaces. Ces enregistrements vont aussi faciliter la détection et la prévention des blessures. Le WBAN est utilisé dans d'autres domaines à part ceux cités précédemment, par exemple dans l'armée et dans les milieux hostiles comme les centrales nucléaires, etc.

1.3 Architecture d'un réseau de capteurs sans fil sur le corps humain

L'architecture typique d'un WBAN est illustrée à la Figure 1.1 [KYBH12]. Le système est divisé en trois parties en fonction de leurs rôles, à savoir :

a) Les nœuds proprement dits.

b) Le nœud central ou le coordinateur.

c) L'ordinateur de surveillance dans le centre de contrôle.

La transmission des données mesurées par les différents capteurs intégrés dans les nœuds s'effectue en deux étapes. Tout d'abord, elles sont transmises vers le nœud central ou le coordinateur via une liaison sans fil à courte distance. Le fait de transmettre les données à

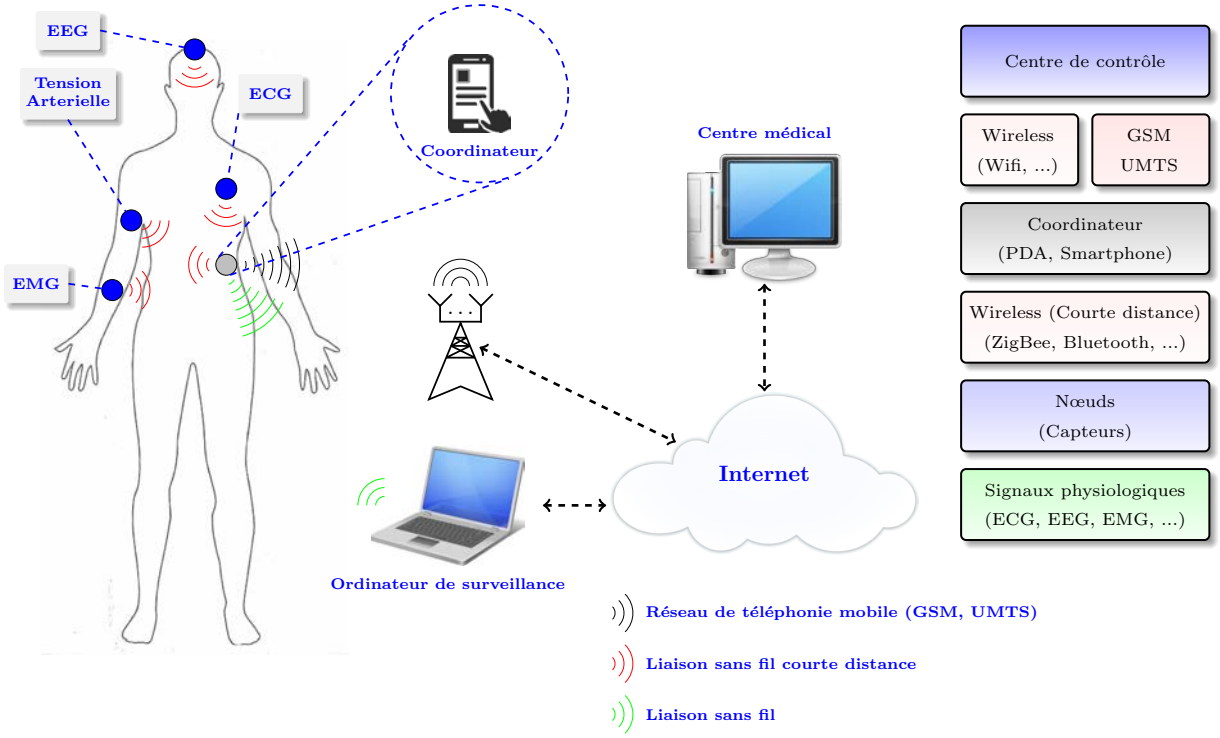


FIGURE *1.1 – Architecture typique d'un réseau de capteurs sans fil sur le corps humain.*

courte distance permet d'économiser l'énergie des nœuds. Ensuite, le nœud central retransmet les données vers le centre de contrôle. En supposant que le nœud central possède les ressources nécessaires, notamment en termes d'énergie, pour transmettre les données vers le centre de contrôle.

À partir du nœud central, l'acheminement des données peut se faire de deux manières différentes. Elles peuvent être transmises via une liaison sans fil à longue ou moyenne distance vers un ordinateur de surveillance local. Puis ce dernier les transmet vers le centre de contrôle via Internet ou une ligne privée. Ou bien, les données sont transmises directement vers le centre de contrôle via un réseau mobile comme le « *global system for mobile communications (GSM)* », le « *general packet radio service (GPRS)* », « l'*universal mobile telecommunications system (UMTS)* » ou le « *worldwide interoperability for microwave access (WiMAX)* ».

1.3.1 Les nœuds

i) Différents types de signaux

Les nœuds intègrent un ou plusieurs capteurs qui vont collecter les données utiles à la surveillance du corps humain. Les capteurs peuvent être groupés en trois catégories dépendant de la nature du signal à mesurer [HPB+09] :

a) Capteurs physiologiques : ils mesurent les données relatives aux fonctionnements des organes, des tissus et des cellules du corps humain, comme l'ECG, l'EEG, la température

TABLEAU *1.1 – Description de quelques signaux traités dans un réseau de capteurs sans fil sur le corps humain [PB10].*

Signal	Description
ECG	Activité électrique générée par le cœur (signal montrant les phases de contraction et de relaxation des cycles cardiaques)
EEG	Activité électrique du cerveau
EMG	Courants électriques qui accompagnent l'activité musculaire
Tension artérielle	Force exercée par le sang sur la paroi des artères
Glycémie	Concentration de glucose dans le sang
Battement du cœur	Fréquence du cycle cardiaque

du corps, etc. Le TABLEAU 1.1 donne la description de quelques signaux physiologiques traités dans un WBAN.

b) Capteurs biocinétiques : ils mesurent les informations en relation avec le mouvement du corps humain, comme l'accélération, l'angle de rotation des membres, etc.

c) Capteurs ambiants : ils mesurent les informations en rapport avec l'environnement qui entoure le corps humain, par exemple la lumière, l'humidité, la température, etc.

ii) Architecture d'un nœud

L'architecture des nœuds est semblable à celle d'un système d'acquisition de données classique. Ces nœuds possèdent en particulier des modules RF qui vont leur permettre de recevoir des consignes et de transmettre les données provenant de leurs capteurs. Pour être autonomes, chaque nœud est doté de sa propre batterie. Il est également de plus en plus courant d'intégrer des circuits de captation d'énergie dans ces nœuds.

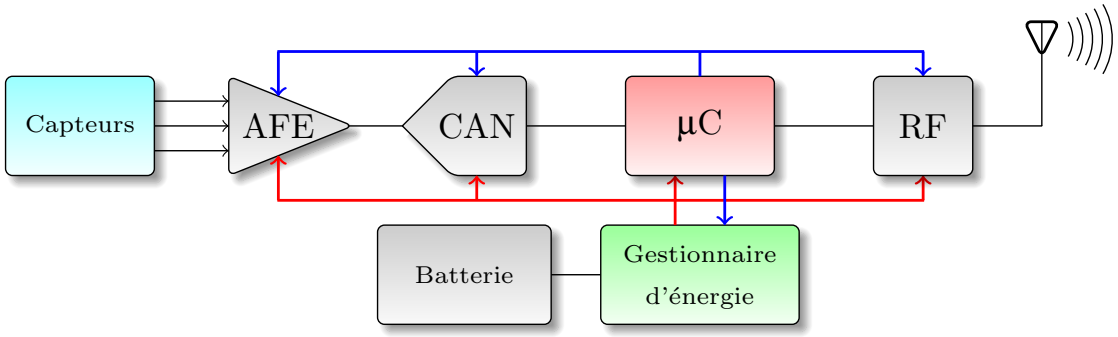


FIGURE *1.2 – Architecture typique d'un nœud du réseau.*

L'architecture d'un nœud est illustrée à la FIGURE 1.2 [KY10]. Les signaux bruts provenant des capteurs ont généralement une amplitude faible (voir TABLEAU 1.2) et sont bruités. Le module « *analog front end (AFE)* » effectue l'amplification, le filtrage des bruits et le

TABLEAU *1.2 – Plage de valeurs et bande de fréquences de quelques signaux physiologiques [KY10].*

Signal	**Plage de valeurs**	**Bande de fréquences [Hz]**
ECG	0.5 - 4 mV	0.01 - 250
EEG	3 - 300 µV	0.5 - 60
EMG	10 µV - 15 mV	10 - 5000
Conductance cutanée ou « *galvanic skin response (GSR)* »	30 µS - 3 S	0.03 - 20
Taux de respiration	2 - 50 respirations / min	0.1 - 10
Tension artérielle	10 - 400 mmHg	0 - 50
Température du corps	32 - 40 °C	0 - 0.1

multiplexage des signaux bruts issus des capteurs. Ce module joue un rôle important dans la chaîne car la qualité de l'information, en termes de rapport signal sur bruit, dépend de lui. Le CAN numérise les signaux pré-conditionnés par l'AFE. Le gestionnaire d'énergie gère l'énergie du nœud. Le microcontrôleur (µC) coordonne toutes les opérations du nœud. Il gère le gain de l'amplificateur et sélectionne les entrées du multiplexeur. C'est le microcontrôleur qui cadence la fréquence de conversion du CAN et qui configure sa résolution. Il peut aussi effectuer un pré-traitement des signaux numérisés. Il empaquette les données et les transmet vers le module RF. Pour économiser de l'énergie, le microcontrôleur peut ordonner le gestionnaire d'énergie de couper l'alimentation de certains modules pendant le mode veille.

iii) Fréquence d'acquisition et résolution du convertisseur analogique numérique

La fréquence d'acquisition et la résolution du CAN dépendent de la nature du signal à mesurer. Le TABLEAU 1.2 rapporte les paramètres de quelques signaux physiologiques traités dans un WBAN. C'est la bande de fréquence du signal qui fixe la fréquence d'acquisition. En effet, d'après le théorème de Shannon-Nyquist, la fréquence d'acquisition doit être supérieure ou égale à deux fois la fréquence maximale du signal. En ce qui concerne la résolution du CAN, c'est la largeur de la plage de valeurs du signal qui la détermine. Pour minimiser les erreurs introduites durant la phase de quantification, plus la plage de valeurs est large, plus la résolution minimale du CAN doit être grande. Le TABLEAU 1.3 donne la résolution ainsi que la fréquence d'acquisition du CAN utilisé pour numériser les signaux physiologiques présentés précédemment au TABLEAU 1.2. La fréquence d'acquisition a été fixée à la fréquence de Nyquist qui est égale à deux fois la fréquence maximale du signal. La dernière colonne du tableau donne le débit minimal brut pour transmettre les données numérisées. Il est calculé à partir du nombre de bits de la résolution du CAN et de la

TABLEAU *1.3 – Résolution du convertisseur analogique numérique et fréquence d'acquisition de quelques signaux physiologiques [LBM+11].*

Signal	**Résolution du CAN [bit]**	**Fréquence d'acquisition [Hz]**	**Débit minimal brut [kbps]**
ECG	12	500	6
EEG	12	120	1.44
EMG	16	10000	160
GSR	16	40	0.64
Taux de respiration	8	20	0.16
Tension artérielle	8	100	0.8
Température du corps	8	0.2	0.0016

fréquence d'acquisition.

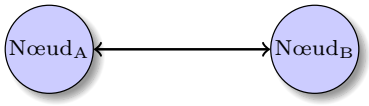
iv) Emplacement des nœuds et topologie du réseau

Dans un WBAN, pour que les données physiologiques soient correctement mesurées, les nœuds doivent être placés à des endroits spécifiques du corps humain. En effet, un mauvais emplacement des nœuds peut engendrer une dégradation de la qualité des signaux mesurés. Pour minimiser les perturbations engendrées par cet emplacement inapproprié, des mesures doivent être prises tout au début de la conception jusqu'au déploiement des nœuds. La technologie utilisée pour les mettre en boîte et le type d'électrodes ou de sondes de mesure doivent être choisis convenablement.

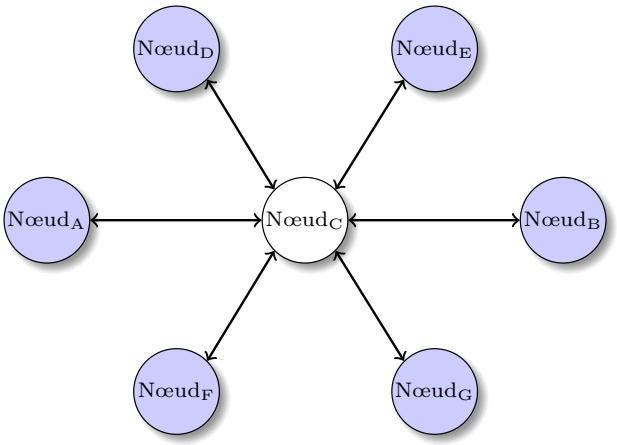
La topologie du réseau régit l'organisation logique de la communication entre les différents nœuds du réseau. Le choix de la topologie utilisée est important puisque les performances globales du réseau, notamment en termes de fiabilité et de consommation d'énergie, en dépendent [UHB+12]. Les topologies point-à-point, en étoile, maillée, en arbre et étoile-maillée sont les plus utilisées. La FIGURE 1.3 illustre l'organisation logique des différentes topologies.

La topologie point-à-point est la plus simple. Elle peut être mise en place lorsque le réseau est composé seulement de deux nœuds qui se communiquent directement, comme illustré à la FIGURE1.3(a). L'un d'entre eux joue le rôle de passerelle vers l'ordinateur de contrôle ou le centre de surveillance.

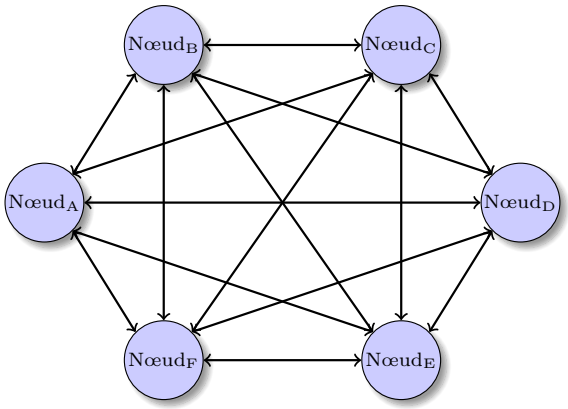
La topologie en étoile est illustrée à la FIGURE1.3(b). Dans cette topologie, tous les nœuds communiquent directement avec un nœud central ou coordinateur. Le coordinateur est la passerelle vers l'ordinateur de contrôle ou le centre de surveillance. Cette topologie simplifie le routage des paquets qui circulent dans le réseau et permet d'avoir une bande passante élevée. Cependant le fait d'avoir une seule passerelle crée un seul point de défaillance,



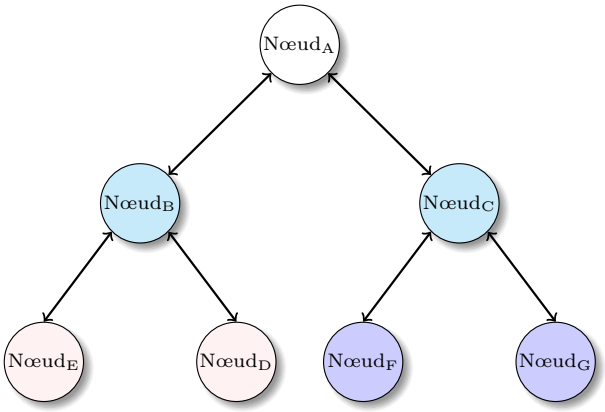
(a) Topologie point-à-point.



(b) Topologie en étoile. Le Nœud$_C$ est le nœud central.



(c) Topologie maillée.



(d) Topologie en arbre.

FIGURE *1.3 – Différentes topologies des nœuds.*

diminuant ainsi la fiabilité du réseau.

Dans une topologie maillée, tous les nœuds se communiquent pair à pair sans hiérarchie centrale. Les nœuds se présentent comme une structure en forme de filet et il peut y avoir plusieurs passerelles vers le réseau extérieur, comme illustré à la FIGURE1.3(c). En cas de défaillance, les nœuds peuvent se réorganiser entre eux, augmentant ainsi la fiabilité du réseau. Par contre dans cette topologie, les nœuds doivent implémenter un protocole de routage avancé, augmentant ainsi leur charge et leur consommation.

La FIGURE1.3(d) illustre la topologie en arbre. Elle comporte un nœud principal à la racine de l'arbre. C'est la passerelle vers le réseau extérieur, généralement vers l'ordinateur de surveillance ou le centre de contrôle. Ce nœud principal est connecté aux nœuds secondaires qui se trouvent dans le premier niveau de l'arbre. Chacun de ces nœuds secondaires est connecté avec certains nœuds du niveau suivant, et ainsi de suite jusqu'au dernier niveau de l'arbre. La transmission des messages se fait par sauts en parcourant l'arbre d'un niveau à un autre. L'inconvénient majeur de cette topologie est que si un nœud d'un niveau quelconque est défaillant, les nœuds qui en dépendent sont automatiquement déconnectés

du réseau. La topologie en étoile-maillée utilise à la fois bien évidemment la structure de la topologie en étoile et celle de la topologie maillée [BTG+11].

Les fréquences utilisées par les nœuds pour la communication radiofréquence se trouvent notamment dans les bandes suivantes [CMR+14] :

a) Industriel, scientifique et médical (ISM) : de 2454 à 2483.5 MHz, 100 mW en intérieur et 10 mW (10 dB m) en extérieur. Cette bande sans licence peut être utilisée pour des applications industrielles, scientifiques, médicales et domestiques.

b) « *Medical implant communications service (MICS)* » : de 402 à 405 MHz avec une puissance maximale de transmission de 25 µW. Cette bande est spécialement dédiée pour la communication avec les implants [SSW+05].

c) « *Wireless medical telemetry services (WMTS)* » : 608 - 614 MHz, 1.395 - 1.4 GHz et 1.427-1.432 GHz. Ces bandes de fréquences sont allouées spécifiquement aux États-Unis pour la transmission de données personnelles relatives à la santé des patients.

Tableau *1.4 – Quelques normes utilisées pour la communication radiofréquence [PB10].*

Norme	**Distance**	**Débit maximal**	**Consommation**	**Bande de fréquence**
ZigBee	10 - 75 m	20 kpbs 40 kpbs 250 kbps	30 mW	868 MHz 915 MHz 2.4 GHz
Bluetooth	10 - 100 m	1 - 3 Mpbs	2.5 - 100 mW	2.4 GHz
Wifi	200 m	54 Mpbs	1 W	2.4 GHz

Les normes radiofréquences couramment utilisées dans un WBAN sont le Bluetooth (IEEE 802.15.1), le ZigBee (IEEE 802.15.4) et le Wifi (IEEE 802.11g). Le Tableau 1.4 donne quelques caractéristiques de ces normes. La colonne consommation du tableau se réfère à la puissance maximale consommée lorsque de la transmission ou de la réception de messages. Parmi ces normes, le ZigBee a le plus faible débit. Par conséquent, il a une faible consommation d'énergie. Le Bluetooth permet d'avoir un bon rapport débit-consommation. Le Wifi consomme le plus, mais il offre un très haut débit.

Le ZigBee [Zig] permet d'avoir un réseau ayant une topologie en étoile, arbre ou bien maillée. Il est conçu pour fonctionner dans trois bandes de fréquences différentes dépendant de l'emplacement géographique :

a) 915 Mhz aux États-Unis, avec 10 canaux de 40 kbps chacun.

b) 868 Mhz en Europe, avec un seul canal de 20 kbps.

c) 2.4 GHz indépendamment de l'emplacement géographique, avec 16 canaux de 250 kbps chacun.

Pour authentifier les messages et garantir ainsi leur intégrité et leur confidentialité, le ZigBee utilise un algorithme standard de chiffrement avancé (« *advanced encryption standard (AES)* ») avec une clé de 128-bit.

Le Bluetooth [Blu] fonctionne dans la bande de fréquence de 2.4 GHz (ISM). Il utilise une technique de saut de fréquence sur 76 canaux avec un débit maximal pouvant atteindre 3 Mbps. Cette technique lui permet de lutter contre les interférences et lui donne une forte immunité au bruit. Pour assurer la sécurité du réseau et garder la confidentialité des messages transmis, le Bluetooth utilise aussi un algorithme AES avec une clé de 128-bit. Les dispositifs qui partagent le même canal forment un groupe appelé « *piconet* ». Il peut y avoir autant de « *piconets* » que de canaux. Mais un « *piconet* » peut contenir au maximum huit dispositifs avec un seul maître et sept esclaves formant une topologie en étoile [SBTL05].

Le débit maximal est un paramètre important parce qu'il limite le nombre de nœuds du réseau. Par exemple, dans le cas d'un réseau en étoile composé d'un coordinateur et de deux nœuds munis de capteurs EMG nécessitant chacun un débit minimal de 160 kbps, il faut que le réseau ait un débit minimal de 320 kbps, pour que les nœuds puissent transmettre leurs données. Même si le ZigBee permet d'économiser de l'énergie, il n'est pas adapté pour ce type de réseau parce que son débit maximal est en dessous de la valeur minimale requise. Alors, il faut utiliser soit le Bluetooth, soit le Wifi, soit une autre norme qui permet d'avoir un débit supérieur à 320 kbps.

v) Consommation d'énergie dans un nœud

Pour assurer le confort et éviter tout encombrement de la personne, les nœuds devraient avoir une dimension assez petite et un poids négligeable. Les avancées technologiques actuelles, en particulier dans le domaine de la microélectronique, ont permis de faire face à ces exigences. Mais l'inconvénient majeur de la miniaturisation est la diminution de la taille de la batterie. En effet, elle est directement proportionnelle à la dimension des nœuds [PB10].

L'autonomie ou la durée de vie d'un nœud est définie comme la durée entre l'instant où il est mis sous tension jusqu'à ce qu'il s'éteigne automatiquement sans interruption, à cause d'un manque d'énergie dû à l'épuisement de sa batterie. Les nœuds devraient être conçus pour avoir une autonomie allant de plusieurs semaines à quelques années, comme le cas des implants. Par exemple, un nœud qui surveille le taux de glycémie d'une personne nécessite une autonomie d'au moins cinq ans [LBM+11]. Le facteur principal qui limite l'autonomie des nœuds est la consommation d'énergie des différents modules qui les composent.

La consommation d'énergie dans un nœud se répartit sur les trois principales parties suivantes :

a) Mesure et instrumentation qui regroupe les capteurs et le module AFE.

b) Traitement de données incluant le CAN et le microcontrôleur.

c) Transmission radiofréquence.

TABLEAU *1.5 – Distribution de la consommation typique d'énergie dans un nœud [GAD+14].*

	Mesure et instrumentation	**Traitement de données**	**Transmission radiofréquence**
Consommation d'énergie [%]	2	25	73

Le TABLEAU 1.5 rapporte la distribution de la consommation typique d'énergie dans un nœud [GAD+14]. Celle de la partie mesure et instrumentation est presque négligeable. La partie traitement de données consomme à peu près le quart de l'énergie disponible. Ce tableau montre que c'est la partie transmission radio qui consomme le plus d'énergie dans un nœud.

Bien que le TABLEAU 1.5 donne la consommation typique d'énergie dans un nœud, en réalité elle dépend du débit des données à traiter, des composants utilisés et de la norme radiofréquence mise en œuvre. La FIGURE 1.4 rapporte la consommation d'énergie moyenne de quelques microcontrôleurs et modules radiofréquences couramment utilisés dans un WBAN en fonction du débit des données [HPB+09].

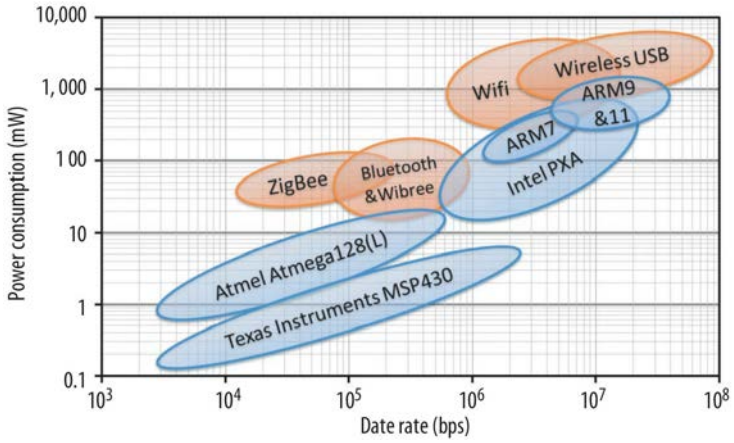


FIGURE *1.4 – Consommation d'énergie moyenne de quelques microcontrôleurs et modules radiofréquences en fonction du débit des données [HPB+09].*

Cette figure montre que, pour un débit donné, les modules radiofréquences consomment toujours en moyenne beaucoup plus d'énergie que les microcontrôleurs. Elle montre aussi que, dans tous les cas, la consommation d'énergie moyenne augmente en fonction du débit des données. Réduire le débit en compressant les données diminue la consommation moyenne du module radiofréquence mais peut éventuellement augmenter celle du microcontrôleur. En effet, implémenter un algorithme de compression dans le microcontrôleur augmente à la fois sa charge et sa consommation d'énergie. Le choix du microcontrôleur dépend de la complexité de l'algorithme de compression mis en œuvre.

1.3.2 Le nœud central ou coordinateur

Le coordinateur rassemble les données provenant des différents nœuds et les transmet vers un centre de surveillance local ou distant. En plus de son rôle de passerelle vers les infrastructures de haut niveau, le coordinateur peut aussi avoir ses propres capteurs. Du point de vue des patients, le coordinateur permet de recevoir les informations provenant du centre de surveillance concernant leurs états de santé.

Le coordinateur n'est pas soumis aux mêmes contraintes que les nœuds en termes d'énergie disponible et de puissance de calcul. En effet, le coordinateur peut avoir une dimension beaucoup plus grande par rapport aux nœuds. Il a assez d'espace pour embarquer une batterie de plus grande capacité, suffisante pour faire tourner un microcontrôleur ayant une puissance de calcul élevée. Le coordinateur peut être un dispositif dédié, un PDA ou bien un smartphone [BWTJ11, WKC+12].

1.4 Discussion

Les sections précédentes ont permis d'identifier les exigences et les conditions à remplir pour qu'un WBAN puisse être utilisé dans la télémédecine et la télésurveillance. Elles incluent principalement l'autonomie et la capacité du réseau (le nombre maximal de nœuds), la sécurité et la confidentialité des données, et la fiabilité des nœuds et du système global.

Il est essentiel d'assurer la sécurité et la confidentialité des données des patients puisqu'elles sont accessibles et partagées par plusieurs entités à savoir les personnels soignants, les chercheurs et les administrateurs du réseau. Li *et al.* [LLR10] et Al Ameen *et al.* [AALK12] ont fait une étude sur les différents problèmes rencontrés dans un WBAN en termes de sécurité et de confidentialité. Les solutions proposées pour faire face à ces problèmes ont été présentées. Les aspects pratiques des solutions ont été aussi analysés.

Remplacer ou charger régulièrement la batterie des nœuds peut entrainer une gêne vis-à-vis des utilisateurs ou des patients. Particulièrement pour les nœuds implantés dans le corps humain, le remplacement de la batterie est très difficile. Pour résoudre le problème d'autonomie du réseau, deux solutions complémentaires peuvent être utilisées.

La première solution consiste à diminuer la consommation d'énergie des nœuds.

La deuxième solution est le captage d'énergie pour recharger la batterie des nœuds. Différentes sources d'énergie peuvent être utilisées à savoir la lumière, la vibration, le son, l'onde électromagnétique et la chaleur. Xu *et al.* [XSG+14] ont fait une étude généralisée sur le captage d'énergie dans un WBAN. Les avantages et les inconvénients des différentes sources ainsi que la quantité d'énergie moyenne qu'elles peuvent fournir ont été évalués. Hoang *et al.* [HTCP09] ont utilisé un générateur thermoélectrique (« *thermoelectric generator (TEG)* ») pour transformer la chaleur du corps humain en signal électrique. La méthode proposée a été utilisée pour augmenter l'autonomie d'un nœud qui détecte les chutes à base d'accéléromètre. Leurs expérimentations ont montré que pour un gradient de température

de 15 °C, le générateur thermoélectrique a pu fournir une énergie de 520 µW. Liberale *et al.* [LDB14] ont proposé une solution à base de cellules photovoltaïques de façon à transformer la lumière en signal électrique pour alimenter les nœuds d'un WBAN. Le système proposé est totalement autonome parce qu'aucune batterie n'est nécessaire au démarrage. Il est opérationnel même dans un environnement intérieur où l'intensité lumineuse est assez faible, aux alentours de 500 lx.

Pour diminuer la consommation des nœuds, la solution la plus évidente consiste à utiliser seulement des composants électroniques optimisés pour consommer des courants très faibles. Il faut aussi bien étudier et concevoir le fonctionnement du nœud pour que les composants inactifs soient mis hors tension ou bien mis en mode veille, en particulier le module radiofréquence. En effet, la partie communication radiofréquence est la plus énergivore. À peu près 73 % de l'énergie disponible est utilisée par le nœud pour transmettre ou bien recevoir des données. La couche contrôle d'accès radio ou contrôle d'accès au support (« *media access control (MAC)* ») du protocole de communication est un facteur important à tenir en compte pour gérer la consommation du module radiofréquence. La couche MAC assure deux fonctions principales :

a) Éviter les collisions pendant l'accès au support de communication partagé.

b) Empêcher les transmissions simultanées tout en préservant un débit maximal avec un délai d'attente court et une communication fiable.

Gopalan *et al.* [GP11] ont présenté une étude généralisée sur les différents protocoles MAC à haute efficacité énergétique dans un WBAN. Les principaux critères qu'un bon protocole MAC devrait satisfaire ont été identifiés suivi d'une étude comparative des différentes solutions proposées. Récemment, Jing *et al.* [JMBW15] ont suggéré un protocole MAC à efficacité énergétique appelé « *Quasi-Sleep-Preempt-Supported (QS-PS)* ». Le protocole proposé est basé sur une technique d'accès multiple à répartition dans le temps (« *time division multiple access (TDMA)* »).

Pour augmenter la capacité du réseau, il faut compresser les données recueillies par les capteurs avant de les transmettre vers le centre de contrôle. Le fait de compresser les données sollicite moins le module radiofréquence et permet de diminuer en même temps sa consommation. La méthode de compression mise en œuvre doit être facile à implémenter puisque dans un nœud les ressources disponibles en termes de calcul et de mémoire sont très limitées.

Typiquement, les signaux provenant des capteurs sont échantillonnés et numérisés à la fréquence de Nyquist par un CAN. Puis, le microcontrôleur applique un algorithme de compression sur les données numérisées. Finalement, le module radiofréquence transmet les données compressées vers le centre de contrôle. Cette approche n'est pas adaptée pour un WBAN parce qu'elle nécessite un calcul assez intense et entraîne une augmentation de la consommation d'énergie des nœuds [FW14]. Récemment, une solution alternative consiste à utiliser la technique d'AC (« *compressed sensing* »). Elle a été introduite dans le traitement

du signal par Donoho [Don06] et Candès *et al.* [CRT06]. L'AC est adaptée pour compresser et reconstruire des signaux dits compressibles, c'est-à-dire ceux qui ont des représentations parcimonieuses par rapport à certaines bases ou domaines. Au lieu d'échantillonner et numériser les signaux compressibles à la fréquence de Nyquist puis d'appliquer un algorithme de compression, l'AC les capture directement à une fréquence en dessous de celle de Nyquist sous forme compressée. En revanche, contrairement à la phase de mesure, la phase de reconstruction de l'AC nécessite un calcul assez intense. La complexité de la phase de mesure est déportée vers la phase de reconstruction [CCS10].

L'AC a été largement utilisée pour optimiser la phase de mesure de nombreuses applications. Surtout dans des applications soumises à de fortes contraintes en termes d'énergie et de calcul, comme le cas du WBAN. En effet, la plupart des signaux physiologiques traités dans un WBAN sont compressibles. C'est-à-dire qu'ils ont des représentations parcimonieuses par rapport à certaines bases [BRK12]. La phase de mesure de l'AC va en même temps faciliter l'acquisition des données et diminuer la consommation d'énergie des nœuds. Même si la phase de reconstruction de l'AC est complexe, elle va être exécutée soit sur le coordinateur, soit sur l'ordinateur de surveillance. C'est avantageux puisque, par rapport aux nœuds, ils sont soumis à des contraintes moins fortes en termes d'énergie et de calcul.

Dans la compression de signaux ECG, Mamaghanian *et al.* [MKAV11] ont comparé les performances de l'AC en termes de compression et de consommation d'énergie par rapport à une méthode à base d'ondelettes. Leurs résultats ont montré que, malgré le fait que l'AC est moins performante en termes de qualité de reconstruction, elle est plus facile à mettre en œuvre et est beaucoup plus efficace du point de vue de la consommation d'énergie. En effet, l'AC a augmenté l'autonomie des nœuds de 37.1 % par rapport à la méthode à base d'ondelettes.

De plus, la simplicité de l'AC leur a permis de faire une acquisition en temps réel des signaux ECG. De même, une étude faite par Fauvel *et al.* [FW14] dans la compression de signaux EEG a montré que, pour un même facteur de compression, l'AC a diminué de cinq à huit fois la consommation d'énergie des nœuds par rapport à une méthode classique à base d'ondelettes (JPEG2000). Par contre, en termes de qualité de reconstruction, la méthode à base d'ondelettes est plus efficace. En particulier, lorsque le facteur de compression est élevé. Pour mettre en évidence la simplicité de l'AC et son efficacité énergétique, Liu *et al.* [LZX+14] ont implémenté dans un « *field-programmable gate array (FPGA)* » une méthode à base de l'AC et une autre à base d'ondelettes. L'utilisation des ressources du FPGA a montré que la méthode à base de l'AC est plus simple. De plus, elle consomme seulement 23.7 % de l'énergie totale consommée par la méthode à base d'ondelettes.

1.5 Conclusion

Le suivi en temps réel des paramètres physiologiques d'une personne par le biais du WBAN a facilité la mise en place de la télémédecine. Le WBAN va aussi permettre l'assistance à domicile des personnes âgées ou à mobilité réduite, grâce à la télésurveillance, et apporte une solution aux problèmes engendrés par le vieillissement de la population. Les applications du WBAN ne se limitent pas seulement à la télémédecine et la télésurveillance. Il peut apporter des solutions pour promouvoir le bon fonctionnement de nombreux domaines, à savoir la défense, le sport, etc.

Ce chapitre a présenté l'architecture et le principe de fonctionnement du WBAN, ainsi que les technologies mises en œuvre. Malgré le fait que le WBAN est une solution prometteuse, des problèmes restent encore à résoudre, principalement l'autonomie des nœuds. Trois solutions complémentaires ont été proposées pour faire face à ce problème : (i) le captage d'énergie, (ii) l'utilisation de protocole radio basse consommation et (iii) la compression de données. En effet, c'est la partie transmission radio qui consomme le plus d'énergie. Les études faites ont montré que les méthodes classiques de compression ne sont pas adaptées pour le WBAN parce qu'elles nécessitent un calcul assez complexe entraînant une augmentation de la consommation d'énergie des nœuds. Récemment, une solution alternative consiste à utiliser l'acquisition comprimée (AC). L'AC facilite l'acquisition et la compression de données au niveau des nœuds en déportant la complexité vers la phase de reconstruction. Contrairement aux nœuds, le centre de contrôle est moins contraignant et a les ressources nécessaires pour effectuer cette phase de reconstruction.

Le chapitre suivant va présenter le principe de base de l'AC. Les phases d'acquisition et de reconstruction d'un signal avec l'AC seront détaillées.

Chapitre 2

L'Acquisition comprimée

Sommaire

2.1 Introduction 22
2.2 Parcimonie 22
2.3 Acquisition et reconstruction d'un signal avec l'acquisition comprimée 23
2.3.1 Phase d'acquisition 24
2.3.2 Phase de reconstruction 24
2.3.3 Discussion 25
2.4 Matrices de mesure 25
2.4.1 Propriétés 25
2.4.2 Exemples de matrices de mesure 28
2.5 Algorithmes de reconstruction 28
2.5.1 Relaxation convexe 28
2.5.2 Poursuite gloutonne 29
2.5.3 Relaxation convexe vs. poursuite gloutonne 29
2.5.4 Quelques algorithmes gloutons 29
2.6 Critères de solvabilité 31
2.6.1 Critère de solvabilité en fonction de la cohérence 31
2.6.2 Critère de solvabilité en fonction du restricted isomtry property . 32
2.6.3 Critère de solvabilité en fonction de la cohérence mutuelle 32
2.7 Domaine de parcimonie 32
2.7.1 Transformée en cosinus discrète 33
2.7.2 Transformée en ondelettes 34
2.7.3 Seuillage 35
2.8 Métriques d'évaluation de la qualité de reconstruction 35
2.9 Chaîne d'acquisition comprimée 36
2.9.1 Encodeur analogique 37
2.9.2 Encodeur numérique 43
2.10 Conclusion 45

2.1 Introduction

Pour numériser un signal analogique à bande limitée, l'approche conventionnelle consiste à utiliser le théorème de Shannon-Nyquist : pour éviter toute perte d'information, le signal devrait être échantillonné à une fréquence supérieure ou égale au double de la fréquence maximale présente dans le signal. Si cette fréquence maximale est égale à f_{max}, alors la fréquence d'échantillonnage f_e devrait être supérieure ou égale à 2 f_{max}, c'est-à-dire $f_e \geq 2\ f_{max}$. Cette fréquence minimale d'échantillonnage est appelée fréquence de Nyquist (f_N), $f_N = 2\ f_{max}$. Ce théorème est utilisé dans les systèmes d'acquisition numérique comme les appareils photos et les caméscopes.

Pour certaines applications, comme le radar et les communications à large bande, l'application de ce théorème aboutit à des fréquences d'échantillonnage qui sont presque au-delà de la limite des capacités physiques des convertisseurs analogique-numérique [LKD+07]. Pour d'autres applications, comme le cas du WBAN, même si elles ne sont pas directement confrontées à ce problème puisque la fréquence de Nyquist des signaux à mesurer n'est pas aussi élevée, les données numérisées devraient être compressées pour économiser l'énergie, la bande passante du support de transmission et la mémoire de stockage.

Pour contourner ce problème, Donoho [Don06] et Candès *et al.* [CRT06] ont introduit l'AC dans le traitement du signal. En plus de l'hypothèse que le signal est à bande limitée, dans la plupart des cas, le signal à mesurer peut avoir une représentation parcimonieuse dans un domaine particulier. L'AC exploite cette information additionnelle pour capturer le signal à une fréquence en dessous de celle de Nyquist directement sous forme compressée et sans perte d'information. Pour rendre cela possible, l'AC s'appuie sur deux principes : la parcimonie et l'incohérence [CW08].

La suite de ce chapitre est organisée en deux parties. La première présentera l'aspect théorique de l'AC où le principe de base de l'AC sera détaillé. La seconde partie se focalisera sur le côté pratique. La chaîne d'acquisition de données avec l'AC sera étudiée et les différents encodeurs proposés dans la littérature seront présentés.

2.2 Parcimonie

Un signal est dit parcimonieux ou creux s'il contient seulement quelques éléments significatifs ou non nuls. Dans le cas d'un signal représenté par un vecteur à dimension finie et à valeurs discrètes $\mathbf{x} \in \mathbb{R}^{\mathcal{N}}$, il est dit parcimonieux si sa norme l_0 (1) est égale à $\mathcal{K}$, $\|\mathbf{x}\|_0 = \mathcal{K}$, et $\mathcal{K} \ll \mathcal{N}$.

Un signal qui n'est pas parcimonieux dans le domaine temporel peut avoir une représentation parcimonieuse par rapport à un autre domaine $\mathbf{\Psi} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$. Un signal $\mathbf{x}$ a une

(1). La norme l_0 d'un vecteur est définie par : $\|\mathbf{x}\|_0 = \#\{i : x_i \neq 0\}$. Le symbole $\#$ représente le cardinal d'un ensemble quelconque. Ce n'est pas vraiment une norme puisque $\|k\ \mathbf{x}\|_0 \neq k\ \|\mathbf{x}\|_0$, pour tout $k \in \mathbb{R}$.

représentation parcimonieuse par rapport à un domaine $\Psi \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ si sa transformée $\alpha \in \mathbb{R}^{\mathcal{N}} = \Psi\mathbf{x}$ est parcimonieuse, c'est-à-dire $\|\alpha\|_0 = \mathcal{K}$ et $\mathcal{K} \ll \mathcal{N}$. Le degré de parcimonie ρ d'un signal $\mathbf{x}$ par est défini comme suit :

$$\rho = \frac{\mathcal{K}}{\mathcal{N}} \tag{2.1}$$

où $\mathcal{K}$ et $\mathcal{N}$ représentent respectivement le nombre d'éléments non nuls et la dimension de $\mathbf{x}$. Il permet de mesurer le degré de compressibilité : plus ρ est faible, plus le signal est compressible. Dépendant de la nature du signal, plusieurs domaines peuvent être utilisés comme le Fourier, la transformée en cosinus discrète (*DCT)*, ou bien les transformées en ondelettes discrètes (*discrete wavelet transform (DWT)*).

La parcimonie n'est pas seulement une propriété exploitée par l'AC, elle est utilisée par plusieurs algorithmes de compression notamment par les codages par transformation [BDE09].

2.3 Acquisition et reconstruction d'un signal avec l'acquisition comprimée

La FIGURE 2.1 illustre les phases d'acquisition et de reconstruction de l'AC. Le signal d'entrée $\mathbf{x} \in \mathbb{R}^{\mathcal{N}}$ est supposé parcimonieux dans un domaine ou une base Ψ, c'est-à-dire, $\mathbf{x} = \Psi\alpha$, $\|\alpha\|_0 = \mathcal{K}$ et $\mathcal{K} \ll \mathcal{N}$. Le vecteur $\mathbf{y} \in \mathbb{R}^{\mathcal{M}}$ représente le résultat de la phase d'acquisition, avec $\mathcal{K} < \mathcal{M} < \mathcal{N}$. La première étape de la phase de reconstruction donne le vecteur $\tilde{\alpha} \in \mathbb{R}^{\mathcal{N}}$. C'est une approximation de la transformée du signal dans le domaine Ψ. Le vecteur $\tilde{\mathbf{x}} \in \mathbb{R}^{\mathcal{N}}$, résultant de la deuxième étape de la phase de reconstruction, représente une approximation du signal $\mathbf{x}$. En particulier, lorsque le signal est parcimonieux dans le domaine temporel où $\Psi = \mathbf{I}$, la première étape de la phase de reconstruction donne directement une approximation $\tilde{\mathbf{x}}$ du signal parcimonieux $\mathbf{x}$.

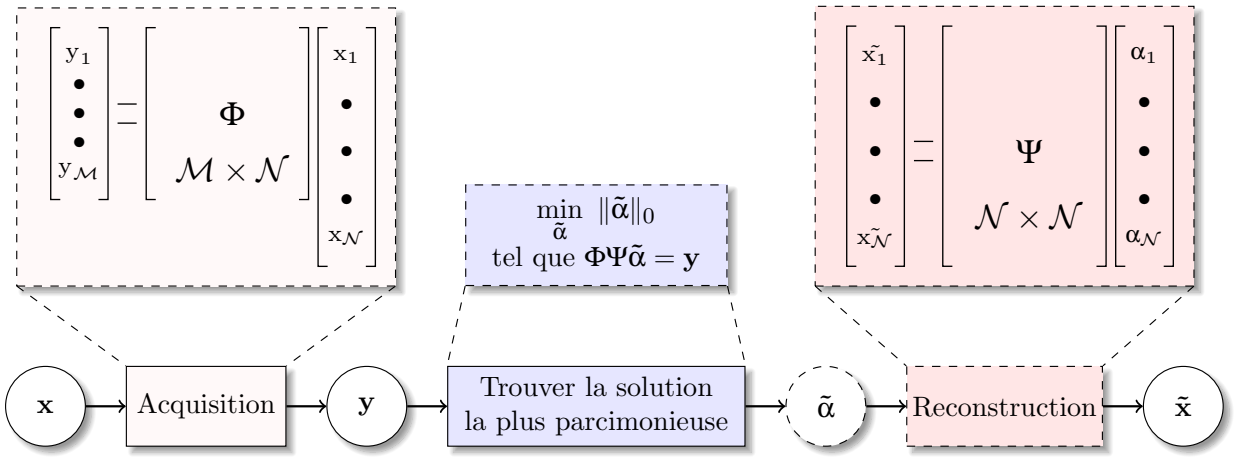


FIGURE *2.1 – Acquisition et reconstruction d'un signal avec l'acquisition comprimée.*

2.3.1 Phase d'acquisition

Le signal parcimonieux $\mathbf{x}$ est directement capturé sous un format compressé à une fréquence en dessous de celle de Nyquist, durant une phase appelée phase de mesure ou d'acquisition. Le signal est multiplié par une matrice rectangulaire $\mathbf{\Phi} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}}$, avec $\mathcal{M} < \mathcal{N}$:

$$\mathbf{y} = \mathbf{\Phi x} \tag{2.2}$$

Le vecteur résultant $\mathbf{y}$ est appelé vecteur de mesure. La matrice $\mathbf{\Phi} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}}$ est appelée matrice de mesure. Puisque le signal a une représentation parcimonieuse dans un domaine ou une base $\mathbf{\Psi} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$, l'équation (2.2) devient :

$$\mathbf{y} = \mathbf{A}\boldsymbol{\alpha} \tag{2.3}$$

où $\mathbf{A} = \mathbf{\Phi\Psi} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}}$ et $\boldsymbol{\alpha} \in \mathbb{R}^{\mathcal{N}}$ est la transformée de $\mathbf{x}$ dans le domaine $\mathbf{\Psi}$, avec $\|\boldsymbol{\alpha}\|_0 = \mathcal{K}$ et $\mathcal{K} \ll \mathcal{N}$.

Tout au long de ce manuscrit, le signal sera supposé non parcimonieux dans le domaine temporel mais a une représentation parcimonieuse dans un domaine $\mathbf{\Psi}$ particulier. Dans le cas contraire, $\mathbf{\Psi}$ est une matrice identité, $\mathbf{\Psi} = \mathbf{I}$.

En tenant compte de la présence de bruit additif pendant la phase d'acquisition, l'équation (2.3) devient :

$$\mathbf{y} = \mathbf{A}\boldsymbol{\alpha} + \boldsymbol{\varepsilon} \tag{2.4}$$

où $\boldsymbol{\varepsilon} \in \mathbb{R}^{\mathcal{M}}$ est un vecteur représentant le bruit. Dans la plupart des cas, il est considéré comme étant un bruit blanc gaussien avec une moyenne nulle et une variance $\boldsymbol{\sigma}^2$, $\mathcal{N}(0, \boldsymbol{\sigma}^2)$, et que son amplitude est bornée $\|\boldsymbol{\varepsilon}\|_2 \leq \mathrm{b}$ [2] [CW11].

2.3.2 Phase de reconstruction

Le signal original est reconstruit à partir du vecteur de mesure $\mathbf{y}$, la matrice de mesure $\mathbf{\Phi}$ et celle du domaine $\mathbf{\Psi}$ durant une phase appelée phase de reconstruction. Elle se fait en deux étapes :

a) La première étape consiste à trouver le vecteur $\tilde{\boldsymbol{\alpha}}$ correspondant à la solution de l'équation (2.3) ou (2.4).

b) Une fois $\tilde{\boldsymbol{\alpha}}$ obtenu, la dernière étape reconstruit le signal $\tilde{\mathbf{x}} = \mathbf{\Psi}\tilde{\boldsymbol{\alpha}}$.

Puisque la matrice $\mathbf{A} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}}$ est rectangulaire, $\mathcal{M} < \mathcal{N}$, il existe une infinité de vecteurs $\boldsymbol{\alpha}$ satisfaisant l'équation (2.3) ou (2.4). Mais en tenant compte de l'hypothèse que

(2). La norme l_2 d'un vecteur $\mathbf{x} \in \mathbb{R}^{\mathcal{N}}$ est définie par : $\|\mathbf{x}\|_2 = \sqrt{\sum_{\mathrm{i}=1}^{\mathcal{N}} \mathrm{x_i^2}}$

$\boldsymbol{\alpha}$ est parcimonieux dans le domaine $\boldsymbol{\Psi}$, le problème consiste à résoudre la minimisation suivante :

$$\min_{\tilde{\boldsymbol{\alpha}}} \|\tilde{\boldsymbol{\alpha}}\|_0 \text{ tel que } \mathbf{A}\tilde{\boldsymbol{\alpha}} = \mathbf{y} \tag{2.5}$$

En prenant compte du bruit additif, l'équation (2.5) devient :

$$\min_{\tilde{\boldsymbol{\alpha}}} \|\tilde{\boldsymbol{\alpha}}\|_0 \text{ tel que } \|\mathbf{y} - \mathbf{A}\tilde{\boldsymbol{\alpha}}\|_2 \leq \mathrm{b} \tag{2.6}$$

où $\mathrm{b} = \|\boldsymbol{\varepsilon}\|_2$ est l'amplitude du bruit.

2.3.3 Discussion

Deux questions fondamentales se posent ainsi avec l'AC :

a) Est-ce que n'importe quelle matrice peut être utilisée pendant la phase de mesure ? Autrement dit, quelles sont les critères que la matrice de mesure $\boldsymbol{\Phi}$ devrait satisfaire ?

b) Comment reconstruire efficacement le signal original à partir de $\mathbf{y}$, $\boldsymbol{\Phi}$ et $\boldsymbol{\Psi}$? Quel est le nombre minimal de mesures $\mathcal{M}$ nécessaires pour reconstruire correctement le signal original ?

2.4 Matrices de mesure

La matrice de mesure doit être conçue pour que la phase d'acquisition de l'AC n'altère pas l'information contenue dans le signal [Bar07]. D'une part, elle doit permettre d'identifier l'unique signal $\mathbf{x}$ à partir du vecteur de mesure $\mathbf{y}$ [DE11]. D'autre part, elle doit être conçue pour assurer la reconstruction du signal parcimonieux. Pour répondre à ces exigences, la matrice de mesure doit remplir certaines conditions [MMT14]. Les paragraphes suivantes présenteront les critères couramment utilisés dans la littérature.

2.4.1 Propriétés

i) Condition d'unicité

Pour garantir que deux signaux distincts $\boldsymbol{\alpha}_1$ et $\boldsymbol{\alpha}_2$ ($\boldsymbol{\alpha}_1 \neq \boldsymbol{\alpha}_2$) engendrent deux vecteurs de mesure différents $\mathbf{y}_1 = \mathbf{A}\boldsymbol{\alpha}_1$ et $\mathbf{y}_2 = \mathbf{A}\boldsymbol{\alpha}_2$ ($\mathbf{y}_1 \neq \mathbf{y}_2$), Donoho *et al.* [DE03] ont établi le théorème suivant :

Théorème 1 (Condition d'unicité [DE03])**.** *Lorsque*

$$\mathcal{K} < \frac{1}{2} spark(\mathbf{A}) \tag{2.7}$$

, alors pour chaque vecteur de mesure $\mathbf{y} \in \mathbb{R}^{\mathcal{N}}$ *il existe au plus un signal parcimonieux* α*, avec* $\|\alpha\|_0 = \mathcal{K}$*, tel que* $\mathbf{y} = \mathbf{A}\alpha$*.*

Par définition, le *spark*($\mathbf{A}$) d'une matrice $\mathbf{A}$ est égal au plus petit nombre de colonnes de $\mathbf{A}$ qui sont linéairement dépendantes. Puisque la valeur de *spark*($\mathbf{A}$) varie entre 2 et $(\mathcal{M}+1)$, alors il faut que $\mathcal{M} > 2\mathcal{K}$. Bien que ce théorème soit important puisqu'il garantit l'unicité de la représentation d'un vecteur parcimonieux, il est difficile à évaluer pour une matrice $\mathbf{A}$ donnée.

ii) Restricted isometry property

Une matrice $\mathbf{A}$ satisfait le « *restricted isometry property (RIP)* » d'ordre $\mathcal{K}$ si la plus petite constante $\delta_{\mathcal{K}}$, $0 < \delta_{\mathcal{K}} < 1$ appelée « *restricted isometry constant (RIC)* » , vérifiant la condition suivante existe [CT05] :

$$(1-\delta_{\mathcal{K}})\|\alpha\|_2^2 \leq \|\mathbf{A}\alpha\|_2^2 \leq (1+\delta_{\mathcal{K}})\|\alpha\|_2^2 \tag{2.8}$$

pour tout vecteur parcimonieux α, tel que $\|\alpha\|_0 = \mathcal{K}$. Le RIP permet de vérifier si la matrice de mesure $\mathbf{A}$ est proche d'une isométrie, c'est-à-dire si elle préserve la distance entre deux vecteurs de mesure. Autrement dit, si la matrice de mesure satisfait le RIP, alors la distance entre deux vecteurs de mesure $\mathbf{y}_1 = \mathbf{A}\alpha_1$ et $\mathbf{y}_2 = \mathbf{A}\alpha_2$ est proportionnelle à la distance entre α_1 et α_2. Le RIP est une propriété importante qui permet de garantir la reconstruction du signal. Cependant, il est difficile de vérifier si une matrice satisfait ou non le RIP [BDMS13].

iii) Cohérence

Une autre propriété utilisée dans la littérature est la cohérence. Elle est plus facile à évaluer par rapport aux deux propriétés présentées précédemment. C'est l'avantage majeur de la cohérence. La cohérence $\mu(\mathbf{A})$ d'une matrice $\mathbf{A}$ est égale à la plus grande valeur absolue du produit scalaire entre deux vecteurs colonnes distincts de $\mathbf{A}$ [BHEE10] :

$$\mu(\mathbf{A}) = \max_{i \neq j} \frac{\left|\langle a_i, a_j \rangle\right|}{\|a_i\|_2 \|a_j\|_2} \tag{2.9}$$

où a_i et a_j sont les colonnes de la matrice $\mathbf{A}$ [(3)]. La cohérence permet de mesurer la corrélation maximale entre les différents vecteurs colonnes d'une matrice. Welch a démontré que pour une matrice $\mathbf{A}$ donnée [Wel74, LZB13] :

$$\sqrt{\frac{\mathcal{N}-\mathcal{M}}{\mathcal{M}(\mathcal{N}-1)}} \leq \mu(\mathbf{A}) \leq 1 \tag{2.10}$$

(3). Le produit scalaire de deux vecteurs $\mathbf{x}_1 \in \mathbb{R}^{\mathcal{N}}$ et $\mathbf{x}_2 \in \mathbb{R}^{\mathcal{N}}$ est noté par $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle$: $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = \sum_{i=1}^{\mathcal{N}} x_{1_i} x_{2_i}$

La borne inférieure de μ est appelée la limite de Welch. Lorsque $\mathcal{M} \ll \mathcal{N}$, cette borne inférieure tend vers $\frac{1}{\sqrt{\mathcal{M}}}$. Une matrice $\mathbf{A}$ est dite incohérente lorsque la valeur de $\mu(\mathbf{A})$ tend vers cette limite de Welch.

Condition d'unicité en fonction de la cohérence

La borne inférieure du $spark(\mathbf{A})$ d'une matrice $\mathbf{A}$ peut être exprimée en fonction de $\mu(\mathbf{A})$ [Tro04] :

$$spark(\mathbf{A}) \geq 1 + \frac{1}{\mu(\mathbf{A})} \tag{2.11}$$

La condition d'unicité peut être exprimée en fonction de la cohérence. En combinant les équations (2.7) et (2.11), lorsque la relation suivante est satisfaite :

$$\mathcal{K} < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{A})}\right) \tag{2.12}$$

alors pour un vecteur de mesure $\mathbf{y} \in \mathbb{R}^{\mathcal{N}}$, il existe au plus un signal α tel que $\mathbf{y} = \mathbf{A}\alpha$ [DE11].

Ces équations montrent que la matrice de mesure $\mathbf{A}$ doit avoir une faible cohérence. En effet, une faible valeur de la cohérence μ augmente la borne supérieure de $\mathcal{K}$. Pour minimiser la valeur de μ, il faut que les vecteurs colonnes de $\mathbf{A}$ soient les plus orthogonaux possibles [ZMRE11].

Restricted isometry property en fonction de la cohérence

Cai *et al.* [CXZ09] ont démontré que la borne supérieure du RIP peut être exprimée en fonction de la cohérence. Si une matrice $\mathbf{A}$ a une cohérence $\mu(\mathbf{A})$ alors elle satisfait le RIP avec une constante $\delta_{\mathcal{K}}$ telle que :

$$\delta_{\mathcal{K}} \leq (\mathcal{K} - 1)\mu(\mathbf{A}) \tag{2.13}$$

Cette équation montre que lorsqu'une matrice a une faible cohérence, cela implique qu'elle satisfait le RIP [CW11]. Puisque le RIP est difficile à évaluer, dans la pratique il est remplacé par la cohérence [MKAV11].

La cohérence mutuelle

La notion de cohérence peut être étendue pour une paire de bases. La cohérence mutuelle entre Φ et Ψ est définie comme suit [CR07] :

$$\mu(\Phi, \Psi) = \sqrt{\mathcal{N}} \max_{i,j} \frac{\left|\left\langle \phi_i, \psi_j \right\rangle\right|}{\|\phi_i\|_2 \|\psi_j\|_2} \tag{2.14}$$

où ϕ_i et ψ_j représentent respectivement les vecteurs lignes de $\mathbf{\Phi}$ et les vecteurs colonnes de $\mathbf{\Psi}$. Elle mesure la corrélation maximale entre les vecteurs lignes de $\mathbf{\Phi}$ et les vecteurs colonnes de $\mathbf{\Psi}$. La plage de valeurs de la cohérence mutuelle est [CW08] :

$$1 \leq \mu(\mathbf{\Phi}, \mathbf{\Psi}) \leq \sqrt{\mathcal{N}} \tag{2.15}$$

2.4.2 Exemples de matrices de mesure

Au début, les études faites ont démontré que lorsque les matrices de mesure sont construites aléatoirement, ces conditions peuvent être remplies avec une forte probabilité [CT06]. En particulier, lorsque les éléments de la matrice de mesure $\mathbf{A} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}}$ sont générés à partir :

a) d'un processus gaussien identique et indépendamment distribué (i.i.d) avec une moyenne nulle et une variance $1/\mathcal{M}$: $\mathcal{N}(0, 1/\mathcal{M})$;

b) d'un processus aléatoire équiprobable prenant des valeurs $\pm 1/\sqrt{\mathcal{M}}$;

Récemment, des matrices déterministes ont été proposées pour faciliter l'implémentation. Elles ont été conçues spécialement pour avoir une faible cohérence [LZB13, MMT14, LG14].

2.5 Algorithmes de reconstruction

La résolution des équations (2.5) et (2.6) nécessite une recherche exhaustive de la solution la plus parcimonieuse $\tilde{\alpha}$ et est très complexe à mettre en œuvre [CT05]. Plusieurs méthodes ont été proposées dans la littérature pour contourner ce problème. Principalement, elles peuvent être catégorisées en trois groupes [SW12] :

a) Relaxation convexe, par exemple la poursuite de base ou « *basis pursuit (BP)* » [CDS98].

b) Poursuite gloutonne, par exemple la poursuite adaptative ou « *matching pursuit (MP)* » [MZ93] et son extension appelée poursuite adaptative orthogonale ou OMP [PRK93].

c) Inférence bayésienne (« *bayesian inference* »), comme le « *sparse bayesian learning* » [WR04].

2.5.1 Relaxation convexe

Pour contourner la complexité liée à la norme l_0, les méthodes basées sur la relaxation convexe, comme le BP, relaxent le problème en remplaçant la norme l_0 par une norme l_1 (4) et utilisent des solveurs convexes pour le résoudre [CDS98] :

$$\min_{\tilde{\alpha}} \|\tilde{\alpha}\|_1 \text{ tel que } \mathbf{A}\tilde{\alpha} = \mathbf{y} \tag{2.16}$$

(4). La norme l_1 d'un vecteur $\mathbf{x} \in \mathbb{R}^{\mathcal{N}}$ est définie par : $\|\mathbf{x}\|_1 = \sum_{i=1}^{\mathcal{N}} |x_i|$

Lorsque la matrice $\mathbf{A}$ remplit certains critères, BP peut trouver l'unique solution la plus parcimonieuse $\tilde{\alpha}$ avec une forte probabilité. En tenant compte du bruit additif, le problème formulé par l'équation (2.16) devient [DET06] :

$$\min_{\tilde{\alpha}} \|\tilde{\alpha}\|_1 \text{ tel que } \|\mathbf{y} - \mathbf{A}\tilde{\alpha}\|_2 \leq b \tag{2.17}$$

Dans la littérature, cette méthode est appelée « *basis pursuit with inequality constraints (BPIC)* ». Une autre variante de cette méthode, appelée « *basis pursuit denoising (BPDN)* », consiste à reformuler le problème comme suit [BHEE10] :

$$\min_{\tilde{\alpha}} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\tilde{\alpha}\|_2^2 + \lambda\|\tilde{\alpha}\|_1 \tag{2.18}$$

2.5.2 Poursuite gloutonne

Ce sont des méthodes itératives et généralement faciles à mettre en œuvre. A chaque itération, elles sélectionnent une ou plusieurs colonnes de la matrice $\mathbf{A}$ en fonction de sa corrélation avec le vecteur de mesure $\mathbf{y}$. Puis, elles calculent une approximation du signal et mettent à jour le résiduel qui sera utilisé dans la prochaine itération.

Les algorithmes gloutons se distinguent notamment par la méthode de sélection des colonnes de la matrice $\mathbf{A}$ et sur la façon dont le résiduel est mis à jour [SC12, BD07, DB08].

2.5.3 Relaxation convexe vs. poursuite gloutonne

D'après la littérature, les algorithmes gloutons sont faciles à implémenter et potentiellement rapides par rapport à ceux basés sur la relaxation convexe [KR08, TG07, CC13]. Par contre, ces derniers nécessitent moins d'échantillons pour pouvoir reconstruire le signal original [DGNT08].

2.5.4 Quelques algorithmes gloutons

Lorsque la matrice de mesure $\mathbf{A}$ est une base orthogonale, il est possible de reconstruire une approximation du signal en sélectionnant une par une les colonnes de $\mathbf{A}$ ayant une corrélation maximale avec le résiduel. La poursuite adaptative (MP) [MZ93] est la version la plus simple des algorithmes gloutons.

i) Poursuite adaptative

La MP commence par initialiser le résiduel $\mathbf{r}$ avec le vecteur de mesure $\mathbf{y}$. Elle initialise aussi l'approximation du signal $\tilde{\alpha}$ par un vecteur nul, comme suit [Tro04] :

$$\mathbf{r}_0 \leftarrow \mathbf{y} \text{ et } \tilde{\alpha} \leftarrow 0 \tag{2.19}$$

A chaque itération k, la MP sélectionne une colonne de la matrice $\mathbf{A}$ ayant une corrélation maximale avec le résiduel :

$$\lambda_k = \arg\max |\langle \mathbf{r}_{k-1}, \mathbf{a}_i \rangle| \tag{2.20}$$

où :

- λ_k est l'indice de la colonne sélectionnée.
- $\mathbf{r}_{k-1}$ est le résiduel de l'itération précédente.
- $\mathbf{a}_{i \in \{1,\mathcal{N}\}}$ représente les colonnes de la matrice $\mathbf{A}$.

Ensuite, la MP calcule une nouvelle approximation du signal et met à jour le résiduel :

$$\tilde{\alpha}_k = \tilde{\alpha}_{k-1} + \left\langle \mathbf{r}_{k-1}, \mathbf{a}_{\lambda_k} \right\rangle \mathbf{a}_{\lambda_k} \tag{2.21}$$

$$\mathbf{r}_k = \mathbf{r}_{k-1} - \left\langle \mathbf{r}_{k-1}, \mathbf{a}_{\lambda_k} \right\rangle \mathbf{a}_{\lambda_k} \tag{2.22}$$

où :

- $\tilde{\alpha}_{k-1}$ représente l'approximation du signal obtenue durant l'itération précédente.
- $\mathbf{a}_{\lambda_k}$ est la colonne de la matrice $\mathbf{A}$ sélectionnée.

Le résiduel peut aussi être exprimé en fonction de l'approximation du signal $\tilde{\alpha}_k$ et du vecteur de mesure $\mathbf{y}$:

$$\mathbf{r}_k = \mathbf{y} - \tilde{\alpha}_k \tag{2.23}$$

L'itération s'arrête lorsqu'une certaine condition est remplie. Dans la littérature, plusieurs conditions d'arrêt ont été proposées, notamment [TW10] :

a) Arrêt après un nombre fini d'itérations.

b) Arrêt lorsque l'amplitude du résiduel est inférieure à un seuil prédéfini : $\|\mathbf{r}\|_2 \leq b$.

L'inconvénient de cette méthode est que la matrice de mesure $\mathbf{A}$ n'est pas toujours orthogonale. Dans ce cas, une colonne de la matrice $\mathbf{A}$ peut être sélectionnée à plusieurs reprises pendant la phase de sélection. La MP converge exponentiellement [Tro04].

ii) Poursuite adaptative orthogonale

Pour améliorer la convergence, l'OMP rend toujours le résiduel orthogonal avec les colonnes de la matrice $\mathbf{A}$ qui sont déjà sélectionnées [PRK93]. Ainsi, une colonne de la matrice $\mathbf{A}$ est choisie au maximum une seule fois pendant la phase de sélection. L'OMP exécute au maximum $\mathcal{M}$ itérations puisque le vecteur de mesure $\mathbf{y}$ a $\mathcal{M}$ éléments [GV06].

La description de l'OMP est illustrée à l'Algorithme (1). Les phases d'initialisation et de sélection de l'OMP sont les mêmes que celles de la MP.

Algorithme 1 Orthogonal Matching Pursuit.

Prérequis: Matrice $\mathbf{A}$, vecteur de mesure $\mathbf{y}$, niveaux de parcimonie $\mathcal{K}$

1: $\Omega_0 \leftarrow \emptyset$
2: $\mathbf{r}_0 \leftarrow \mathbf{y}$
3: $\tilde{\alpha} \leftarrow 0$
4: $t \leftarrow 1$
5: **tant que** $t = \mathcal{M}$ ou $t = \mathcal{K}$ ou $\|\mathbf{r}\|_2 \leq b$ **fait**
6: $\quad \lambda_t \leftarrow \arg\max |\langle \mathbf{r}_{t-1}, \mathbf{a}_i \rangle|$
7: $\quad \Omega_t \leftarrow \Omega_{t-1} \cup \{\lambda_t\}$
8: $\quad \tilde{\alpha}_{\Omega_t} \leftarrow \mathbf{A}_{\Omega_t}^{\dagger}\ \mathbf{y}$ où $\mathbf{A}_{\Omega_t}^{\dagger} = (\mathbf{A}_{\Omega_t}^{T}\ \mathbf{A}_{\Omega_t})^{-1}\ \mathbf{A}_{\Omega_t}^{T}$
9: $\quad \mathbf{r}_t \leftarrow \mathbf{P}_t^{\perp}\ \mathbf{y}$ où $\mathbf{P}_t^{\perp} = \mathbf{I} - \mathbf{A}_{\Omega_t}\ \mathbf{A}_{\Omega_t}^{\dagger}$
10: $\quad t \leftarrow t + 1$
11: **fin tant que**
12: **Retourne** $\tilde{\alpha}$

En termes de complexité, c'est la phase de sélection (décrite à l'étape 6 de l'algorithme) qui nécessite le plus de calculs [TG07]. Bien que la phase d'estimation ne soit pas couteuse par rapport à la section, sa complexité peut être réduite en utilisant des techniques de décomposition de matrice, comme le QR ou le Cholesky. Sturm *et al.* [SC12] ont fait une étude comparative sur la complexité des différentes implémentations de l'algorithme OMP utilisant ces techniques.

iii) Autres algorithmes

D'autres algorithmes gloutons ont été proposés dans la littérature. Ils visent notamment à apporter des améliorations sur la performance de l'algorithme de base sous certaines conditions, par exemple le « *stagewise orthogonal matching pursuit (StOMP)* » [DTDS12], le « *compressive sampling matching pursuit (CoSaMP)* » [NT09] ou bien le « *iterative hard thresholding (IHT)* » [BD09].

2.6 Critères de solvabilité

Comme énoncé précédemment, la matrice $\mathbf{A}$ doit être aussi conçue pour garantir la reconstruction du signal. Les sous-sections suivantes vont énumérer quelques critères de solvabilité.

2.6.1 Critère de solvabilité en fonction de la cohérence

En l'absence de bruit, Tropp [Tro04] a démontré que l'OMP et le BP reconstruisent le signal lorsque la condition décrite par l'équation (2.12) est respectée. C'est-à dire lorsque [FWL+12] :

$$\mu(\mathbf{A}) < \frac{1}{2\mathcal{K}-1} \tag{2.24}$$

2.6.2 Critère de solvabilité en fonction du restricted isomtry property

En l'absence de bruit, Davenport *et al.* [DW10] ont fait l'étude de la garantie de la reconstruction d'un signal parcimonieux avec l'algorithme OMP en fonction du RIP. Ils ont démontré que lorsque la matrice $\mathbf{A}$ satisfait le RIP d'ordre $\mathcal{K}+1$ avec une RIC $\delta < \frac{1}{3\sqrt{\mathcal{K}}}$, alors un algorithme OMP peut reconstruire avec exactitude un signal parcimonieux α tel que $\|\alpha\|_0 = \mathcal{K}$ après $\mathcal{K}$ itérations.

Théorème 2 (Uniform uncertainty principle [CRT05])**.** *Soit $\mathcal{K}$ un nombre positif tel que $\delta_{3\mathcal{K}} + 3\delta_{4\mathcal{K}} < 2$, alors pour tout signal parcimonieux α, avec $\|\alpha\|_0 \leq \mathcal{K}$, la solution $\tilde{\alpha}$ du problème formulé par l'équation (2.17) vérifie la condition suivante :*

$$\|\tilde{\alpha} - \alpha\|_2 \leq C_{\mathcal{K}} b \tag{2.25}$$

où la constante $C_{\mathcal{K}}$ peut dépendre seulement de $\delta_{4\mathcal{K}}$. Par exemple, $C_{\mathcal{K}} \approx 8.82$ lorsque $\delta_{4\mathcal{K}} = 1/5$ et $C_{\mathcal{K}} \approx 10.47$ lorsque $\delta_{4\mathcal{K}} = 1/4$.

2.6.3 Critère de solvabilité en fonction de la cohérence mutuelle

Lorsque le nombre de mesures $\mathcal{M}$ vérifie la condition suivante :

$$\mathcal{M} > \mathcal{C}\mu^2(\Phi, \Psi)\mathcal{K}\log(\mathcal{N}) \tag{2.26}$$

où $\mathcal{C}$ est une constante positive arbitraire et $\mathcal{K} = \|\alpha\|_0$, alors BP reconstruit avec exactitude le signal avec une forte probabilité [CR07].

Cette condition montre que lorsque la cohérence mutuelle $\mu(\Phi, \Psi)$ est faible, seulement quelques échantillons seront nécessaires pour reconstruire α. En particulier, lorsque $\mu(\Phi, \Psi) = 1$, seulement $\mathcal{K}\log(\mathcal{N})$ échantillons seront nécessaires pour reconstruire α au lieu de $\mathcal{N}$.

2.7 Domaine de parcimonie

Le degré de parcimonie $\rho = \mathcal{K}/\mathcal{N}$ du signal à mesurer est un paramètre important. Plus ρ est faible, plus le signal est compressible. Dans ce cas, l'algorithme de reconstruction a besoin de moins d'échantillons pour reconstruire le signal.

Cependant, le signal à mesurer n'est pas toujours directement compressible. Dans la plupart des cas, un domaine de parcimonie est utilisé pour le rendre compressible. Le domaine

de parcimonie Ψ doit être choisi convenablement pour réduire ρ aussi faible que possible. L'approche conventionnelle consiste à utiliser des domaines de transformée temps-fréquence tels que les ondelettes, le Gabor ou bien la transformée en cosinus discrète. Une approche alternative consiste à générer la matrice Ψ à partir d'un algorithme d'apprentissage qui s'adapte en fonction des caractéristiques du signal à mesurer, comme celui proposé par Zhang *et al.* [ZSM+14].

2.7.1 Transformée en cosinus discrète

La transformée en cosinus discrète ou DCT est largement utilisée dans la compression d'images ou de vidéos [SSB98]. Elle est aussi utilisée pour compresser les signaux physiologiques, comme l'EEG [AT97] ou bien l'ECG [LB99]. Il existe quelques versions de la DCT, par exemple, DCT-I, DCT-II, DCT-III ou DCT-IV. Cependant, les versions II et IV sont les plus utilisées dans la compression de données [Rez13]. La DCT peut être aussi exprimée en plusieurs dimensions : une dimension (1-D), deux dimensions (2-D) et trois dimensions (3-D). La matrice DCT-II 1-D est construite comme suit :

$$\text{DCT-II } (\mathcal{N} \times \mathcal{N}) = \left[\psi_{ij} = \mathcal{C} \cos\left(i(1+2j)\frac{\pi}{2\mathcal{N}}\right)\right] \tag{2.27}$$

où i et j représentent respectivement les numéros de ligne et de colonne de la matrice. Ils varient entre 0 et $\mathcal{N}-1$. $\mathcal{C}$ est une constante définie comme suit :

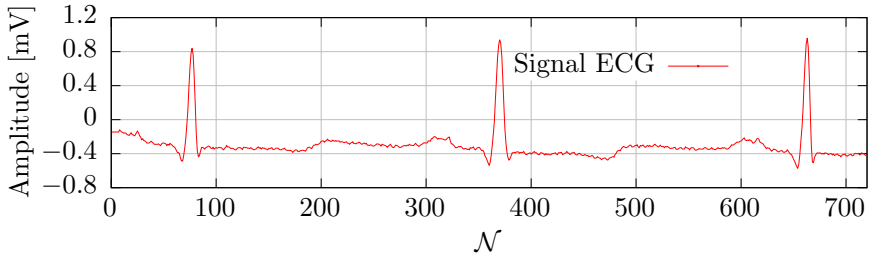
$$\mathcal{C} = \begin{cases} \sqrt{\frac{1}{\mathcal{N}}} & \text{lorsque } i = 0 \\ \sqrt{\frac{2}{\mathcal{N}}} & \text{lorsque } i \neq 0 \end{cases} \tag{2.28}$$

Puisque la matrice DCT-II est orthogonale, sa matrice inverse ou « *inverse discrete cosine transform (IDCT)* » est obtenue en transposant seulement la matrice DCT-II, $\text{IDCT-II} = \text{DCT-II}^{-1} = \text{DCT-II}^{T}$.

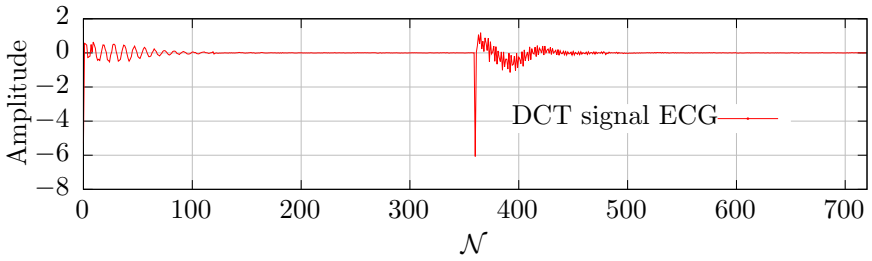
La version II de la DCT est utilisée tout au long de ce manuscrit. La DCT rend un signal parcimonieux en concentrant la plupart de ces informations dans quelques composantes basses fréquences. Les composantes hautes fréquences restantes tendent à avoir de faibles valeurs et deviennent moins importantes. Elles peuvent être ainsi supprimées sans pertes visuelles.

La FIGURE 2.2(a) illustre une partie d'un signal ECG provenant de la base de données *MIT-BIH Arrhythmia* du site Physionet [Phy], l'enregistrement numéro 100. Cette figure montre que le signal ECG n'est pas parcimonieux dans le domaine temporel. Cependant, comme illustré à la FIGURE 2.2(b), la DCT l'a rendu parcimonieux en concentrant son énergie dans seulement quelques composantes basses fréquences.

La FIGURE 2.3(a) illustre une partie d'un signal EMG provenant de la base de données *Electromyograms* du site Physionet [Phy]. La DCT a concentré l'énergie du signal EMG dans la plupart des composantes basses fréquences comme illustré à la FIGURE 2.3(b).

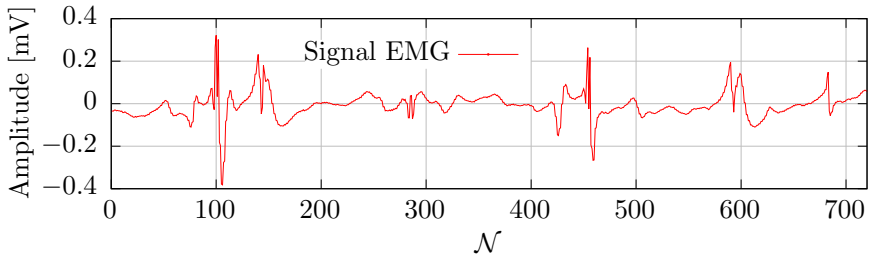


(a) Signal ECG (enregistrement numéro 100).

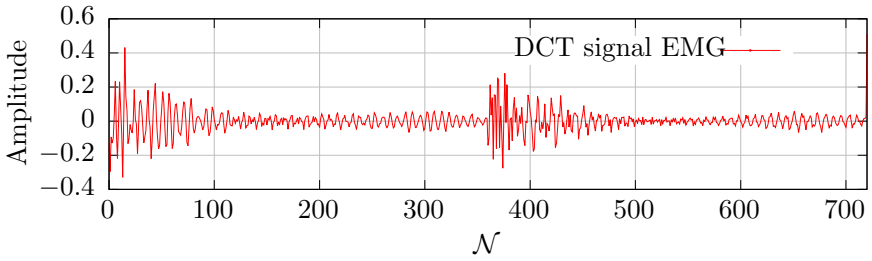


(b) DCT du signal ECG.

FIGURE *2.2 – Signal ECG et sa transformée DCT.*



(a) Signal EMG (enregistrement emg _ healthy).



(b) DCT du signal EMG.

FIGURE *2.3 – Signal EMG et sa transformée DCT.*

2.7.2 Transformée en ondelettes

La transformée en ondelettes utilise des décompositions à plusieurs niveaux. Comme montré à la FIGURE 2.4, les coefficients sont générés hiérarchiquement à travers une série de filtres passe-bas **h** et passe-haut **g**, suivis de sous-échantillonnages. Les coefficients résultants sont respectivement appelés approximations et détails [RR11]. Les réponses impulsionnelles des filtres varient en fonction du type d'ondelettes, par exemple Haar (HWT) ou bien Daubechies (DWT).

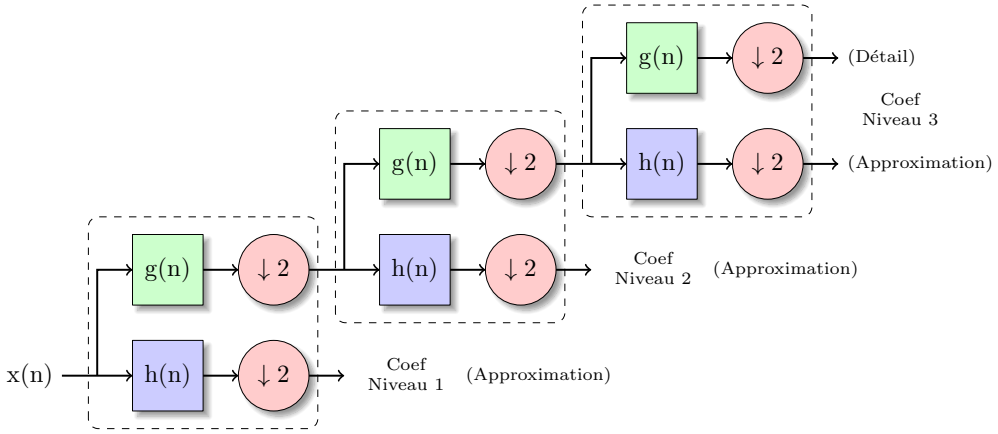


FIGURE *2.4 – Transformée en ondelettes.*

2.7.3 Seuillage

Pour diminuer le degré de parcimonie ρ du signal à mesurer dans le domaine temporel, où $\mathbf{\Psi} = \mathbf{I}^{\mathcal{N}\times\mathcal{N}}$, une technique de seuillage peut être appliquée avant de le compresser. Cela permet de diminuer le nombre d'échantillons $\mathcal{M}$ nécessaires pour reconstruire le signal. Le seuillage diminue le nombre d'éléments non nuls $\mathcal{K}$ en mettant à zéro ceux ayant des amplitudes inférieures à un seuil prédéfini.

Pour contrôler la parcimonie des signaux ECG et EMG, Dixon *et al.* [DAGA12] ont proposé un seuillage dynamique avant d'entamer la phase d'acquisition de l'AC.

Bien que le seuillage temporel soit efficace, il introduit un traitement additionnel dans la phase d'acquisition/compression de l'AC. De la même manière, le seuillage peut être appliqué dans le domaine fréquentiel. Pour le cas de la DCT, le seuillage peut se faire en gardant seulement quelques composantes basses fréquences de la transformée. Lorsque la plupart des informations sont contenues dans ces composantes basses fréquences sélectionnées, le signal est reconstruit avec une faible distorsion. La différence par rapport au signal original est difficilement remarquable. La sélection est faite pendant la phase de reconstruction de l'AC. Contrairement au seuillage dans le domaine temporel, il ne nécessite aucun pré-traitement pendant la phase d'acquisition de l'AC.

2.8 Métriques d'évaluation de la qualité de reconstruction

La qualité du signal reconstruit est évaluée en mesurant la distorsion entre le signal original $\mathbf{x}$ et celui reconstruit $\tilde{\mathbf{x}}$. Nous utilisons le « *percentage root-mean-square deviation (PRD)* » et le « *signal to noise ratio (SNR)* » :

$$\text{PRD}[\%] = \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \times 100 \tag{2.29}$$

$$\text{SNR[dB]} = 20\log_{10}\frac{\|\mathbf{x}\|_2}{\|\mathbf{x}-\tilde{\mathbf{x}}\|_2} = -20\log_{10}(0.01\times\text{PRD}) \tag{2.30}$$

où $\mathbf{x}$ et $\tilde{\mathbf{x}}$ représentent le signal original et celui reconstruit.

TABLEAU *2.1 – Qualité de reconstruction en fonction du PRD [ZCK00].*

PRD [%]	**Qualité**
0 - 2	Très bonne
2 - 9	Bonne
9 - 19	Assez bonne
19 - 60	Mauvaise

La valeur du PRD permet de définir la qualité du signal reconstruit. Le TABLEAU 2.1 rapporte la qualité du signal reconstruit en fonction du PRD [ZCK00].

Nous définissons le taux et compression (« *compression ratio (CR)* ») et le facteur de compression (« *compression factor (CF)* ») comme suit :

$$\text{CR}[\%] = \frac{\mathcal{N}-\mathcal{M}}{\mathcal{N}}\times 100 \tag{2.31}$$

$$\text{CF} = \frac{\mathcal{N}}{\mathcal{M}} \tag{2.32}$$

où, $\mathcal{M}$ et $\mathcal{N}$ sont respectivement les nombres de lignes et de colonnes de la matrice de mesure $\mathbf{\Phi}$.

2.9 Chaîne d'acquisition comprimée

La structure d'une chaîne d'acquisition de données classique est rapportée à la FIGURE 2.5. A l'entrée de la chaîne se trouve la grandeur physique à mesurer comme la température, la lumière, le son, etc. Le capteur convertit les variations de la grandeur physique d'entrée en signal électrique qui sera ensuite pré-conditionné par le module AFE. Généralement, le signal provenant du capteur a une amplitude faible et est bruité. La phase de pré-conditionnement amplifie le signal électrique et filtre les bruits éventuels. Ensuite, le signal pré-conditionné est numérisé par le CAN à la fréquence de Nyquist. Finalement, dépendant de l'application les échantillons numérisés seront traités par le microprocesseur, stockés dans une mémoire locale ou bien transmis vers un site distant.

L'architecture et la complexité de l'encodeur dépendent de la matrice de mesure $\mathbf{\Phi}$. Plus $\mathbf{\Phi}$ est complexe, plus il est difficile de réaliser l'encodeur correspondant. L'encodeur peut être implémenté soit dans le domaine analogique, soit dans le domaine numérique, dépendant de sa place par rapport au CAN [CCS12].

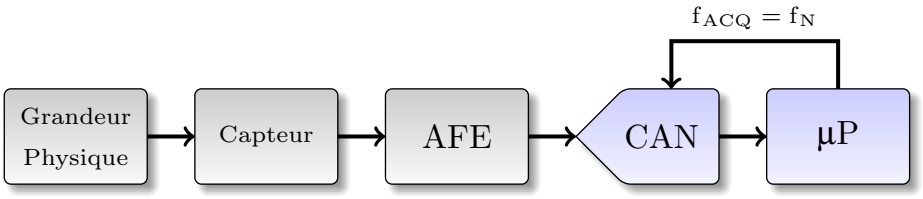


FIGURE *2.5 – Chaîne d'acquisition de données classique.* μP *: microprocesseur [Ere05].* f_{ACQ} *: fréquence d'acquisition.* f_N *: fréquence de Nyquist.*

2.9.1 Encodeur analogique

Lorsque l'encodeur est placé en amont du CAN dans une chaîne d'acquisition de données, il est appelé d'encodeur analogique. En effet, l'encodeur effectue un pré-traitement du signal lorsque ce dernier est encore dans le domaine analogique. La FIGURE 2.6 montre la place d'un encodeur analogique dans une chaîne d'acquisition de données.

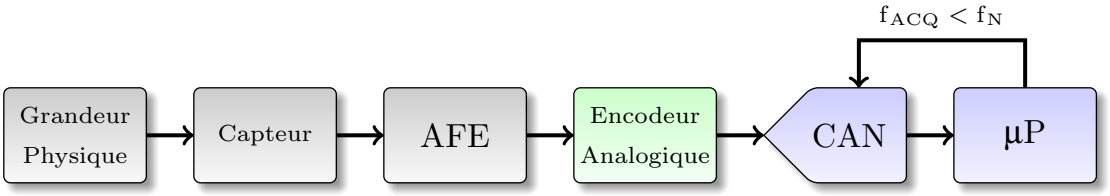


FIGURE *2.6 – Place d'un encodeur analogique dans une chaîne d'acquisition de données.*

Grâce à ce pré-traitement effectué par l'encodeur analogique, le CAN peut numériser le signal en dessous de la fréquence de Nyquist. Le CAN numérise seulement les informations utiles à la reconstruction. C'est pourquoi dans la littérature l'ensemble {encodeur analogique, CAN} est aussi appelé « *analog to information converter (AIC)* ». En effet, l'AIC récupère directement les informations utiles du signal. Le nombre d'échantillons que l'AIC doit capturer dépend de la quantité d'information contenue dans le signal. Plus le signal est parcimonieux ou a une représentation parcimonieuse dans un domaine Ψ particulier, plus l'AIC a besoin de moins d'échantillons.

i) Random demodulator

Le « *random demodulator (RD)* » fut initialement proposé par Kirolos *et al.* [KLW+06] en 2006. C'est un encodeur analogique conçu pour capturer des signaux ayant des représentations parcimonieuses dans le domaine fréquentiel (Fourier).

La FIGURE 2.7 illustre l'architecture du RD. Un mélangeur démodule le signal d'entrée $x(t)$ en le multipliant par une séquence pseudo aléatoire $p_c(t)$ prenant des valeurs dans l'ensemble $\{-1, +1\}$. La séquence pseudo aléatoire est cadencé à une fréquence supérieure ou égale à celle de Nyquist. L'objectif de cette phase de démodulation est d'étaler le spectre du signal d'entrée pour que celui-ci ne soit pas endommagé par l'étage suivant de la chaîne, un filtre passe bas avec une réponse impulsionnelle $h(t)$. Un convertisseur analogique numérique classique numérise la sortie du filtre à une fréquence $f_{\mathcal{M}}$ inférieure à la celle de Nyquist.

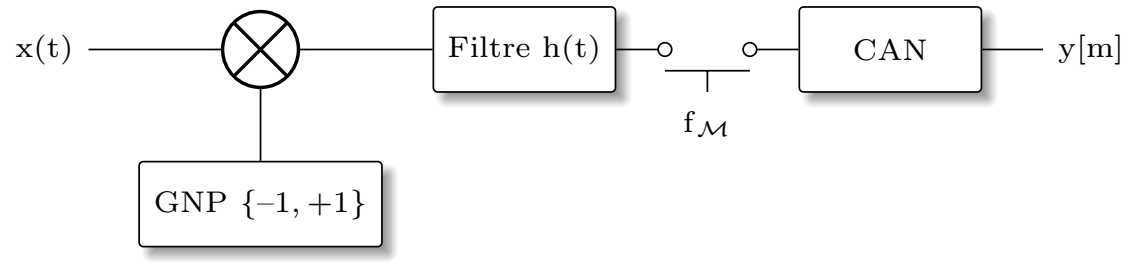


FIGURE 2.7 – *Architecture du random demodulator. GNP : générateur de nombre pseudo aléatoire.*

En 2007, une implémentation au niveau transistor d'un prototype du RD a été proposée par Laska *et al.* [LKD+07]. La séquence pseudo aléatoire était générée à partir d'un registre à décalage à rétroaction linéaire. Pour le filtre passe bas, un intégrateur RC à entrées et sorties différentielles était utilisé. Le prototype a été validé en capturant un signal modulé en amplitude. Leurs résultats ont montré qu'avec une fréquence d'acquisition 6 fois inférieure à celle de Nyquist, le signal modulé en amplitude était reconstruit avec une faible distorsion.

Ragheb *et al.* [RLN+08] ont proposé un autre prototype du RD. Un intégrateur différentiel à base de Gm-C a été utilisé comme filtre passe bas. Le prototype a été validé en capturant un signal modulé en amplitude. Leurs résultats ont montré qu'avec une fréquence d'acquisition 8 fois inférieure à celle de Nyquist, le signal modulé en amplitude était reconstruit avec une faible distorsion.

La matrice de mesure $\mathbf{\Phi}_{\mathrm{RD}} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}}$ correspondante a une structure diagonale par bloc. Chaque bloc est un vecteur de taille fixe égale à $\mathcal{N}/\mathcal{M}$. Ses éléments sont formés à partir de la séquence pseudo aléatoire $\{-1, +1\}$ [TLD+10] :

$$\mathbf{\Phi}_{\mathrm{RD}} = \begin{bmatrix} [-1 \ldots +1] & & & \\ & [+1 \ldots +1] & & \\ & & \ddots & \\ & & & [-1 \ldots -1] \end{bmatrix} \tag{2.33}$$

ii) Random modulation pre-integrator

Le « *random modulation pre-integrator (RMPI)* » est une variante du RD comportant plusieurs canaux en parallèle partageant le même signal d'entrée [YBM+12]. La FIGURE 2.8 illustre un exemple d'architecture du RMPI comportant 4 canaux.

Chaque canal est commandé par un générateur de séquence pseudo aléatoire $\{-1, +1\}$ indépendant. Un mélangeur module le signal d'entrée avec la séquence puis un filtre passe bas intègre la sortie du mélangeur pendant une durée $\mathrm{T_{int}}$. Un convertisseur numérise la sortie du filtre à une fréquence $\mathrm{f_{CAN}} = 1/\mathrm{T_{int}}$ largement inférieure à celle de Nyquist $\mathrm{f_{CAN}} \ll \mathrm{f_{Nyq}}$. Dans leur implémentation, la durée $\mathrm{T_{int}}$ est égale à 52 fois la période de Nyquist $\mathrm{T_{Nyq}}$, $\mathrm{T_{int}} = 52\ \mathrm{T_{Nyq}}$ avec $\mathrm{f_{Nyq}} = 1/\mathrm{T_{Nyq}} = 5$ GHz. Ce qui donne une fréquence

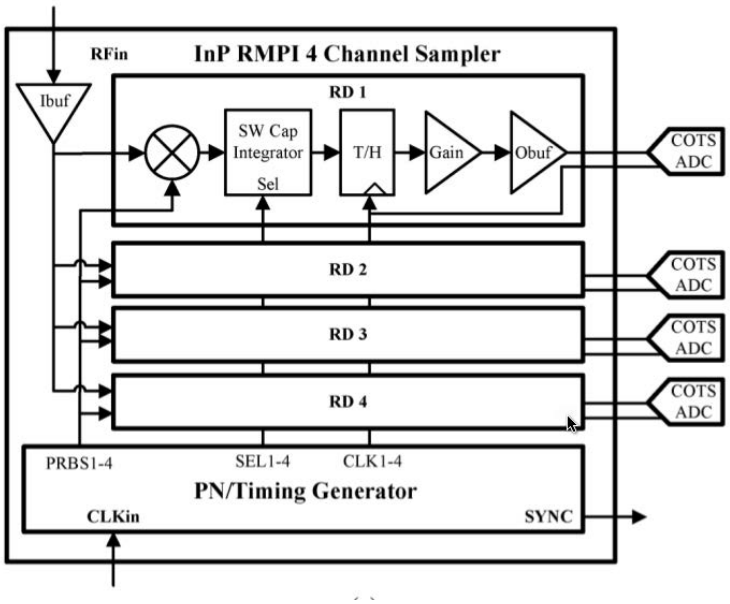


FIGURE 2.8 – *Architecture du random modulation pre-integrator comportant 4 canaux [YBM+12].*

de numérisation égale à $f_{CAN} = 1/(52 \times T_{Nyq}) = 96.154$ MHz. Puisque l'encodeur a 4 canaux en parallèle, la fréquence d'acquisition totale f_S est égale à $f_S = f_{CAN} \times 4 = 384.616$ MHz. Le facteur de compression correspondant est égal à :

$$FC = \frac{f_{Nyq}}{f_S} = 13 \tag{2.34}$$

Le nombre de lignes de la matrice de mesure Φ_{RMPI} correspondante est 13 fois inférieur au nombre de colonnes, autrement dit $\mathcal{N} = 13 \times \mathcal{M}$. C'est une matrice diagonale par bloc. Chaque bloc est une sous-matrice $\mathbf{B} \in \{-1, +1\}^{4\times52}$ formée par une partie des éléments des 4 séquences aléatoires :

a) Les 4 lignes correspondent aux 4 canaux en parallèle.

b) Les 52 colonnes correspondent aux 52 échantillons consécutifs traités par les intégrateurs.

Par exemple, pour une valeur de $\mathcal{N} = 1040$, la matrice de mesure Φ_{RMPI} a $\mathcal{M} = 80$ lignes et 20 blocs de 4×52 :

$$\Phi_{RMPI} = \begin{bmatrix} \begin{bmatrix} \mathbf{B} \\ 4 \times 52 \end{bmatrix} & & & \\ & \begin{bmatrix} \mathbf{B} \\ 4 \times 52 \end{bmatrix} & & \\ & & \ddots & \\ & & & \begin{bmatrix} \mathbf{B} \\ 4 \times 52 \end{bmatrix} \end{bmatrix}_{80\times1040} \tag{2.35}$$

Un prototype du RMPI a été développé sur une puce de 4×4.4 mm^2 en utilisant un procédé propriétaire « *Northrop Grumman (NG) 450 nm InP HBT bipolar* ».

iii) Échantillonneur non uniforme

Un CAN classique numérise un signal quelconque $\mathbf{x}$ à la fréquence de Nyquist f_N, c'est-à-dire à des intervalles régulièrement espacés dans le temps $t_1, t_2, \ldots$ où $t_{i+1} - t_i = \Delta t = 1/f_N$. Dans une fenêtre de temps $T = \mathcal{N}\Delta t$ pouvant contenir $\mathcal{N}$ échantillons, le signal $\mathbf{x}$ peut être considéré comme un vecteur $\mathbf{x} \in \mathbb{R}^{\mathcal{N}}$. Idéalement, l'opération effectuée par le CAN peut être exprimée comme suit :

$$\mathbf{y} = \mathbf{I}\,\mathbf{x} \tag{2.36}$$

où $\mathbf{y} \in \mathbb{R}^{\mathcal{N}}$ et $\mathbf{I} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ est une matrice identité :

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix} \tag{2.37}$$

Conceptuellement, l'échantillonnage non uniforme consiste à numériser le signal d'entrée à des intervalles irréguliers dans le temps [WBN+12]. Il prend seulement $\mathcal{M}$ échantillons parmi les $\mathcal{N}$, avec $\mathcal{M} < \mathcal{N}$. Le choix de ces $\mathcal{M}$ échantillons se fait aléatoirement. La FIGURE 2.9 illustre un exemple du principe de l'échantillonneur non uniforme ou « *non-uniform sampler (NUS)* ». Dans la première ligne, le CAN numérise le signal $\mathbf{x}$ à des intervalles réguliers et génère $\mathcal{N} = 10$ échantillons numérisés $\mathbf{x}_{i\in\{1,10\}}$. Le NUS numérise le signal à des intervalles irréguliers et choisit aléatoirement $\mathcal{M} = 4$ échantillons parmi les 10, dans cet exemple le NUS a choisi aléatoirement les échantillons $\{\mathbf{x}_2, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_{10}\}$.

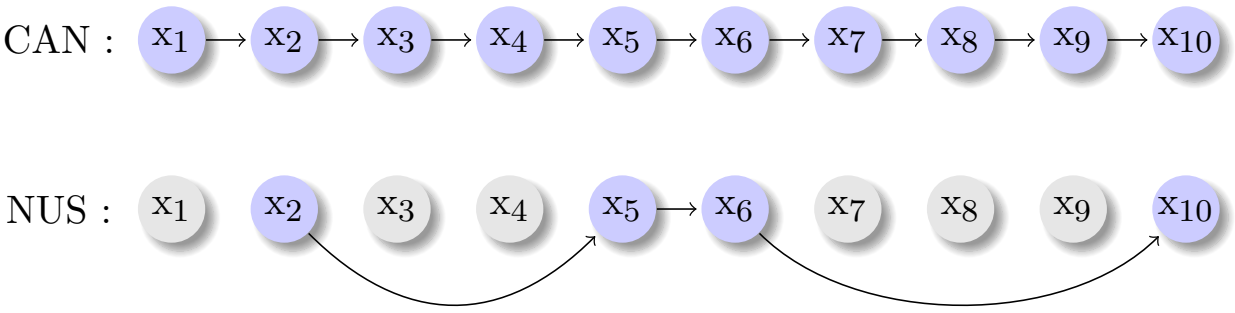


FIGURE *2.9 – Principe de l'échantillonneur non uniforme.*

La matrice de mesure $\mathbf{\Phi} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}}$ du NUS est construite en choisissant aléatoirement $\mathcal{M}$ lignes de la matrice identité $\mathbf{I}$. L'opération effectuée par le NUS est exprimé comme suit :

$$\mathbf{y} = \mathbf{\Phi}_{\text{NUS}}\ \mathbf{x} \tag{2.38}$$

où $\mathbf{y} \in \mathbb{R}^{\mathcal{M}}$ et $\mathbf{\Phi}_{\text{NUS}} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}}$ est construite comme suit :

$$\mathrm{I} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

$$\downarrow$$

$$\mathcal{M} \text{ lignes aléatoires}$$

$$\downarrow$$

$$\mathbf{\Phi}_{\text{NUS}} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ & & \vdots & & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

Chaque ligne de la matrice $\mathbf{\Phi}_{\text{NUS}}$ contient exactement un seul « 1 ». Cela facilite l'implémentation en éliminant à la fois les opérations d'addition et de multiplication. Wang *et al.* [WBB10] ont appliqué cette matrice de mesure dans la compression d'image et ont proposé une implémentation sur FPGA. Le PSNR (« *peak signal to noise ratio* ») de l'image compressée avec la matrice de mesure $\mathbf{\Phi}_{\text{NUS}}$ est inférieur par rapport à une matrice de mesure gaussienne. La différence maximale est de 2.64 dB. Cependant, cette matrice de mesure est plus facile à implémenter qu'une gaussienne.

iv) Compressed sensing analog front-end

Gangopadhyay *et al.* [GAD+14] ont proposé un encodeur analogique appelé « *compressed sensing analog front-end (CS-AFE)* ». Un prototype du circuit a été développé en utilisant un procédé CMOS 0.13 µm, dans une surface de 2×3 mm^2. Le CS-AFE effectue une multiplication de matrice à la fréquence de Nyquist et une numérisation en dessous de la fréquence de Nyquist. Il peut générer une matrice de mesure $\mathbf{\Phi}$ construite à partir d'un processus indépendant et identiquement distribué gaussien ou bernoullien $\{0, 1\}$.

L'architecture du CS-AFE est illustrée à la FIGURE 2.10. Le signal $\mathbf{x}$ est d'abord amplifié par un amplificateur à faible bruit, Bio-LNA (« *low noise amplifier* »). Le CS-AFE fonctionne en mode pipeline et comporte $\mathcal{M}$ canaux pour calculer les éléments $\mathrm{y}_{\mathrm{i}\in\{1,\mathcal{M}\}}$. Chaque canal utilise un convertisseur numérique-analogique multiplicateur/intégrateur ou « *multiplying digital-to-analog converter/integrator (MDAC/I)* », suivi d'un CAN pour calculer un élément $\mathrm{y_i}$. Le MDAC/I est implémenté avec une technique de capacité commutée.

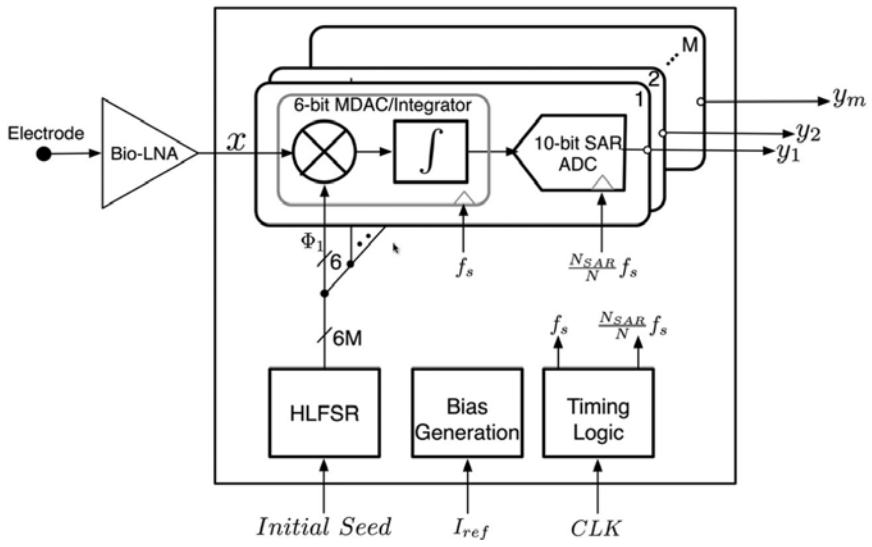


FIGURE *2.10 – Architecture du Compressed Sensing Analog Front-End [GAD+14].*

Chaque MDAC/I fonctionne à la fréquence de Nyquist ($f_s = f_{\mathcal{N}}$) et chaque CAN numérise en dessous de la fréquence de Nyquist. La matrice de mesure $\mathbf{\Phi}$ est générée à partir d'un registre à décalage à rétroaction linéaire hybride (« *hybrid linear feedback shift register* ») (HLFSR).

Le circuit fonctionne comme suit :

- Le signal d'entrée $\mathbf{x}$ est divisé en trames. Chaque trame est formée par $\mathcal{N}$ échantillons successifs.
- À l'instant $t_1 = T_s = 1/f_s$, le MDAC multiplie le premier échantillon $X_1 = \mathbf{x}(t_1)$ par la première colonne de la matrice de mesure $\mathbf{\Phi}_{i1}$, résultant un premier produit partiel $P_1 = \{\mathbf{\Phi}_{11}X_1, \mathbf{\Phi}_{21}X_1, ..., \mathbf{\Phi}_{M1}X_1\}$.
- À l'instant $t_2 = 2T_s$, le MDAC multiplie le second échantillon $X_2 = \mathbf{x}(t_2)$ par la deuxième colonne de la matrice de mesure $\mathbf{\Phi}_{i2}$, résultant un deuxième produit partiel $P_2 = \{\mathbf{\Phi}_{12}X_2, \mathbf{\Phi}_{22}X_2, ..., \mathbf{\Phi}_{M2}X_2\}$.
- Et ainsi de suite, jusqu'à l'instant $t_N = NT_s$ où le MDAC multiplie le dernier échantillon $X_N = \mathbf{x}(t_N)$ par la dernière colonne de la matrice de mesure $\mathbf{\Phi}_{iN}$, résultant le dernier produit partiel $P_N = \{\mathbf{\Phi}_{1N}X_N, \mathbf{\Phi}_{2N}X_N, ..., \mathbf{\Phi}_{MN}X_N\}$.
- Entre temps, dans chacun des $\mathcal{M}$ canaux, les produits partiels sont accumulés par chaque intégrateur.
- À la fin d'une trame, les CANs numérisent les sorties des MDAC/I :

$$Y = P_1 + P_2 + ... + P_N = \begin{cases} Y_1 = \mathbf{\Phi}_{11}X_1 + \mathbf{\Phi}_{12}X_2 + ... + \mathbf{\Phi}_{1N}X_N \\ Y_2 = \mathbf{\Phi}_{21}X_1 + \mathbf{\Phi}_{22}X_2 + ... + \mathbf{\Phi}_{2N}X_N \\ \dots \\ Y_M = \mathbf{\Phi}_{M1}X_1 + \mathbf{\Phi}_{M2}X_2 + ... + \mathbf{\Phi}_{MN}X_N \end{cases} \tag{2.39}$$

• Avant de traiter la prochaine trame, les MDAC/I sont réinitialisés.

2.9.2 Encodeur numérique

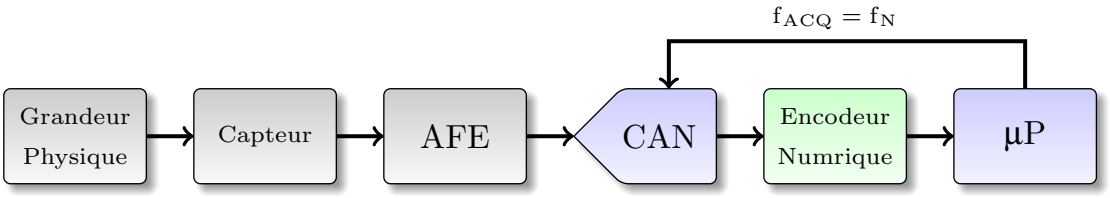


FIGURE *2.11 – Place d'un encodeur numérique dans une chaîne d'acquisition de données.*

L'encodeur numérique, quant-à lui, est placé en aval du CAN. La FIGURE 2.11 montre la place d'un encodeur numérique dans une chaîne d'acquisition de données. Le signal est toujours numérisé à la fréquence de Nyquist. La phase d'acquisition de l'AC est appliquée sur les échantillons numérisés. L'encodeur numérique effectue généralement la multiplication de la matrice de mesure $\mathbf{\Phi}$ avec le signal. Cette multiplication peut se faire aussi bien au niveau matériel que logiciel par le microprocesseur. D'ailleurs, la plupart des systèmes d'acquisition embarquent des microprocesseurs. Bien que cette approche soit plus facile à mettre en œuvre, elle exploite seulement l'aspect compression de l'AC vue que le signal est numérisé à la fréquence de Nyquist.

Mamaghanian *et al.* [MKAV11] ont implémenté un encodeur numérique sur le microcontrôleur MSP430F1611 d'une carte Shimmer. Pour faciliter l'implémentation et accélérer la phase de mesure, une matrice de mesure creuse aléatoire $\mathbf{\Phi} = [\Phi_1|\Phi_2|\Phi_3|\dots|\Phi_{\mathcal{N}}]$ a été proposée. Chaque vecteur colonne $\Phi_{i\in\{1,\dots,\mathcal{N}\}}$ de la matrice contient exactement $d = 12$ éléments non nuls placés aléatoirement. La valeur de chaque élément non nul est égale à $1/\sqrt{d} = 1/\sqrt{12}$. Des signaux ECG provenant de la base de données Physiobank ont été utilisés pour valider l'encodeur. Leurs expérimentations ont montré que la performance en termes de qualité de reconstruction de la matrice de mesure creuse $\Phi_{[0,\ 1/\sqrt{12}]}$ est comparable à celle d'une gaussienne.

Imtiaz *et al.* [ICRV14] ont aussi implémenté un encodeur numérique sur un microcontrôleur MSP430. Pour faciliter davantage l'implémentation de l'encodeur et minimiser la consommation d'énergie, une matrice de mesure aléatoire dont les éléments sont générés à partir d'un processus de Bernoulli a été proposée. Les éléments prennent des valeurs 0 ou 1 avec une probabilité de 0.4 et 0.6. Les éléments de la matrice de mesure sont générés à l'avance sous MATLAB avec la fonction *randn()* puis sauvegardés dans la mémoire FLASH du microcontrôleur. Des signaux EEG ont été utilisés pour valider l'encodeur.

Dans un système de télésurveillance, pour diminuer la consommation des nœuds et augmenter leur autonomie, Liu *et al.* [LZX+14] ont proposé un algorithme de reconstruction rapide basé sur le « *block sparse Bayesian learning* ». Des signaux ECG et EEG ont été utilisés pour tester la méthode proposée. Dans la partie acquisition un encodeur numérique,

implémentant une matrice de mesure creuse binaire $\Phi = [\Phi_1|\Phi_2|\Phi_3|\ldots|\Phi_{\mathcal{N}}]$, a été utilisé. Les éléments de la matrice prennent aléatoirement des valeurs 0 ou 1. Chaque colonne Φ_i de la matrice de mesure a exactement deux éléments non nuls disposés aléatoirement. La méthode proposée a été comparée avec une autre à base d'une transformée en ondelettes. Les deux méthodes ont été implémentées sur un FPGA. La consommation d'énergie et l'utilisation de ressources du FPGA ont été évaluées pour montrer la simplicité de la méthode à base de l'AC par rapport à une autre à base d'une transformée en ondelettes.

Chen *et al.* [CCS12] ont proposé un encodeur numérique dont l'architecture est illustrée à la FIGURE 2.12. L'encodeur effectue un produit matriciel entre la matrice de mesure Bernoulli prenant des valeurs $\{-1, +1\}$ $\Phi = [\Phi_1|\Phi_2|\Phi_3|\ldots|\Phi_{\mathcal{N}}]$ et le vecteur représentant le signal d'entrée :

$$\mathbf{y} = \Phi_1\ \mathrm{x}_1 + \Phi_2\ \mathrm{x}_2 + \Phi_3\ \mathrm{x}_3 + \ldots + \Phi_{\mathcal{N}}\ \mathrm{x}_{\mathcal{N}} \tag{2.40}$$

où $\Phi_{i\in\{1,\ldots,\mathcal{N}\}}$ $(\mathcal{M} \times 1)$ est le $i^{ème}$ vecteur colonne de la matrice de mesure et x_i est le $i^{ème}$ élément du signal. Le signal d'entrée est d'abord amplifié puis numérisé par un convertisseur à la fréquence de Nyquist. La sortie du convertisseur passe par $\mathcal{M}$ accumulateurs en parallèles. En fonction de la valeur courante de $\Phi_i[n]$, l'accumulateur accumule (lorsque $\Phi_i[n] = +1$) ou décumule (lorsque $\Phi_i[n] = -1$) l'échantillon. Le produit matriciel est terminé au bout de $\mathcal{N}$ échantillons consécutifs.

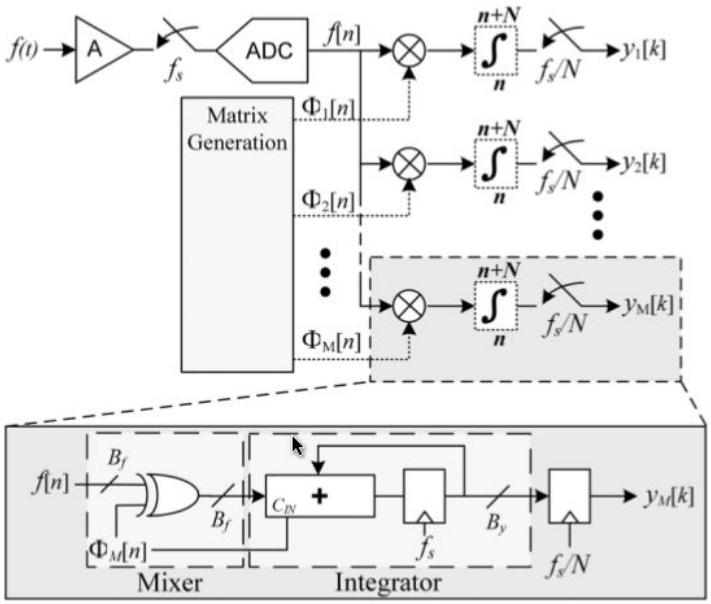


FIGURE *2.12 – Bloc diagramme et implémentation de l'encodeur numérique proposé par Chen et al. [CCS12].*

Un prototype de l'encodeur a été fabriqué sur une puce de 1×2 mm^2 en utilisant un procédé CMOS 90 nm. Le prototype a été validé en capturant un signal EEG. Leurs résultats ont montré qu'avec un facteur de compression de 10, le signal EEG capturé avec l'encodeur est reconstruit avec une faible distorsion.

Le TABLEAU 2.2 récapitule les différents encodeurs proposés dans la littérature. Ce tableau renseigne la matrice de mesure Φ que l'encodeur implémente, le type, la complexité d'encodage, ainsi que le domaine d'application. Le champ type définit la façon dont l'encodeur est implémenté, analogique ou bien numérique. Ce champ informe aussi la technologie mise en œuvre durant l'implémentation : circuit intégré spécialisé (« *application specific integrated circuit (ASIC)* »), circuit logique programmable (FPGA) ou bien microcontrôleur. La complexité d'encodage donne le nombre d'opérations arithmétiques et la fréquence d'acquisition.

2.10 Conclusion

Ce chapitre a détaillé le principe de base de l'AC. Les propriétés que la matrice de mesure devrait satisfaire ont été citées, ainsi que quelques algorithmes de reconstruction couramment utilisés. Les différentes implémentations d'encodeurs analogiques et numériques existants ont été présentées.

Bien que l'AC soit une solution efficace, il reste encore des points à améliorer :

- La construction de la matrice de mesure Φ facilitant la réalisation pratique de l'encodeur. En effet, la complexité de l'encodeur dépend du choix de la matrice de mesure. Pour faciliter l'implémentation, au lieu d'utiliser des matrices aléatoires, des matrices déterministes ont été proposées récemment.
- Le développement d'algorithmes de reconstruction à faible complexité et efficaces, c'est-à-dire des algorithmes pouvant reconstruire le signal seulement avec peu d'échantillons. Par rapport au point cité précédemment, ce deuxième a été bien étudié puisque plusieurs algorithmes de reconstruction ont été proposés.
- La recherche de domaine ou de base Ψ permettant d'avoir un degré de parcimonie $\rho = \mathcal{K}/\mathcal{N}$ très faible. Effectivement, plus ρ est faible, plus le signal est compressible. Seulement quelques échantillons seront nécessaires pour le reconstruire correctement. Généralement, les domaines de transformée temps-fréquence sont les plus utilisés. Récemment, une solution alternative consiste à adapter la matrice Ψ en fonction des caractéristiques du signal à mesurer.

Notre objectif principal est de simplifier davantage l'implémentation de l'encodeur en utilisant une matrice de mesure déterministe. Le chapitre suivant va présenter la construction de la matrice de mesure déterministe proposée. Ses performances seront évaluées et les résultats obtenus vont être comparés avec ceux d'autres matrices de mesure. Nous verrons aussi que la reconstruction du signal sera rendue simple grâce à la matrice de mesure déterministe proposée.

TABLEAU *2.2 – Tableau récapitulatif des différents encodeurs proposés.*

Encodeur	**Matrice de mesure Φ**	**Type**	**Complexité d'encodage**	**Application**
Random demodulator [LKD+07, RLN+08]	Matrice ternaire diagonale par bloc. Chaque bloc formant le diagonal est un vecteur prenant aléatoirement des valeurs ±1.	Analogique (ASIC)	- Multiplications : 0 - Additions : $\mathcal{N} - \mathcal{M}$ - Fréquence de numérisation : inférieure à f_N	Signaux modulés en amplitude
Échantillonneur non uniforme [WBB10]	Matrice binaire construite en sélectionnant aléatoirement $\mathcal{M}$ lignes d'une matrice identité $\mathbf{I} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ avec $\mathcal{M} \ll \mathcal{N}$.	Numérique (FPGA)	- Multiplications : 0 - Additions : 0 - Fréquence de numérisation : f_N	Images
Encodeur proposé par Mamaghanian *et al.* [MKAV11]	Matrice binaire dont les éléments prennent des valeurs 0 ou $1/\sqrt{12}$. Chaque vecteur colonne de la matrice contient exactement 12 valeurs non nulles placées aléatoirement.	Numérique (Microcontrôleur MSP430)	- Multiplications : 0 - Additions : $12 \times (\mathcal{N} - 1)$ - Fréquence de numérisation : f_N	ECG
Random modulation pre-integrator [YBM+12]	Matrice ternaire diagonale par bloc. Chaque bloc formant le diagonal est une sous-matrice prenant aléatoirement des valeurs ±1.	Analogique (ASIC)	- Multiplications : 0 - Additions : $\mathcal{C} \times (\mathcal{N} - \mathcal{M})$ où $\mathcal{C}$ est le nombre de canaux - Fréquence de numérisation : f_N	Signaux modulés en amplitude

Encodeur proposé par Chen *et al.* [CCS12]	Matrice binaire dont les éléments prennent aléatoirement des valeurs ± 1 à partir d'un processus bernoullien.	Numérique (ASIC)	- Multiplications : 0 - Additions : $\mathcal{M} \times (\mathcal{N} - 1)$ - Fréquence de numérisation : f_N	EEG
Compressed sensing analog front-end [GAD+14]	- Matrice pleine aléatoire dont les éléments sont générés à partir d'un processus indépendant et identiquement distribué gaussien. - Matrice binaire aléatoire dont les éléments sont générés à partir d'un processus bernoullien.	Analogique (ASIC)	Matrice gaussienne : - Multiplications : $\mathcal{M} \times \mathcal{N}$ - Additions : $\mathcal{M} \times (\mathcal{N} - 1)$ - Fréquence de numérisation : inférieure à f_N Matrice bernoullienne : - Multiplications : 0 - Additions : $\mathcal{M} \times (\mathcal{N} - 1)$ - Fréquence de numérisation : inférieure à f_N	ECG
Encodeur proposé par Imtiaz *et al.* [ICRV14]	Matrice binaire dont les éléments prennent des valeurs 0 ou 1 avec une probabilité de 0.6.	Numérique (Microcontrôleur MSP430)	- Multiplications : 0 - Additions : $\mathcal{C} \times (\mathcal{N} - 1)$, où $\mathcal{C}$ est le nombre d'éléments non nuls par colonnes - Fréquence de numérisation : f_N	EEG

Encodeur proposé par Liu *et al.* [LZX+14]	Matrice binaire dont les éléments prennent des valeurs 0 ou 1. Chaque vecteur colonne de la matrice contient deux éléments non nuls disposés aléatoirement	Numérique (FPGA)	- Multiplications : 0 - Additions : $2 \times (\mathcal{N} - 1)$ - Fréquence de numérisation : f_N	ECG et EEG

Chapitre 3

Approche déterministe de l'acquisition comprimée

Sommaire

3.1 Introduction 50

3.2 Matrice de mesure proposée 51

3.2.1 Construction de la matrice de mesure proposée 51

3.2.2 Complexité en termes de calcul et allocation mémoire 54

3.2.3 Performances de la matrice de mesure 55

3.3 Algorithme de reconstruction proposé 64

3.3.1 Description de l'algorithme 64

3.3.2 Performances de l'algorithme 66

3.4 Comparaison entre la méthode proposée et le codage par transformation 72

3.4.1 Complexité de l'encodage 73

3.4.2 Qualité de reconstruction 74

3.5 Conclusion 76

3.1 Introduction

Les matrices pleines aléatoires, où les éléments sont générés à partir d'un processus indépendant et uniformément distribué gaussien ou bernoullien $\{\pm 1\}$, sont couramment utilisées parce qu'elles satisfont le RIP et la faible cohérence avec une forte probabilité [Bar07]. Cependant, elles présentent quelques limitations en pratique.

Premièrement, les matrices pleines aléatoires ralentissent la phase de mesure puisque l'encodeur doit traiter tous les éléments non nuls. De plus, la plupart des microcontrôleurs utilisés dans le WBAN ne possèdent pas suffisamment de puissance de calcul en raison de la nécessité de réduire la consommation d'énergie. Deuxièmement, l'encodeur effectuant la phase de mesure de l'AC a besoin d'un générateur de nombres aléatoires ou bien d'une mémoire assez large pour pouvoir sauvegarder localement les éléments.

Pour surmonter ces limitations, les matrices creuses ou parcimonieuses aléatoires ont été utilisées [BT13]. Contrairement aux matrices pleines, la plupart de leurs éléments sont nuls. En particulier, la matrice creuse aléatoire proposée par Mamaghanian *et al.* [MKAV11]. Leurs résultats ont montré que la matrice creuse aléatoire proposée a une performance comparable à celle d'une matrice pleine en termes de qualité de reconstruction. Cependant, elle est moins complexe et plus facile à mettre en œuvre en pratique. En plus, la matrice creuse aléatoire a accéléré la phase de mesure de l'AC. Grâce à cette matrice, l'acquisition en temps réel d'un signal ECG était faisable.

Plusieurs travaux se portaient aussi sur l'étude de la performance des matrices ayant une structure particulière comme Toeplitz, circulaire ou bien diagonale par bloc. Wang *et al.* [WTF+12] ont étudié la performance des matrices aléatoires ayant une structure généralisée diagonale par bloc. Une étude comparative sur les performances des matrices aléatoires diagonales par bloc et pleines a été faite. Leurs résultats ont montré que les matrices diagonales par bloc nécessitent un peu plus d'échantillons pour reconstruire correctement les signaux parcimonieux. Par contre, elles sont plus faciles à réaliser en pratique que les matrices pleines aléatoires. En particulier, He *et al.* [HOH10] ont proposé une matrice diagonale par bloc où leurs colonnes ont été permutées aléatoirement.

Récemment, les recherches se sont focalisées sur la construction de matrices de mesure déterministes [AM11, LLXJ12, YL13, LG14]. En effet, elles sont beaucoup plus faciles à réaliser et à mettre en œuvre.

Pour faciliter l'implémentation de l'encodeur, nous proposons une matrice de mesure binaire déterministe. La première partie de ce chapitre présentera la construction de cette matrice. Ses performances seront évaluées et comparées avec celles d'autres matrices de mesure. Nous découvrirons dans la deuxième partie que cette matrice de mesure va aussi faciliter la reconstruction du signal. Le principe de l'algorithme de reconstruction proposé sera détaillé. Dans la troisième et dernière partie de ce chapitre, les performances de la méthode proposée, la combinaison de la matrice de mesure et de l'algorithme de reconstruction,

seront comparées avec celles d'un codage par transformation.

3.2 Matrice de mesure proposée

Les matrices de mesure peuvent être catégorisées selon leurs propriétés ou structures internes, à savoir :

- Aléatoire : les éléments de la matrice sont générés à partir d'un processus aléatoire, par exemple gaussien ou de Bernoulli.
- Déterministe : les éléments de la matrice sont générés à partir d'un processus déterministe.
- Ternaire : les éléments de la matrice prennent seulement trois valeurs possibles, par exemple des valeurs dans l'ensemble $\{-1, 0, +1\}$.
- Binaire : les éléments de la matrice prennent seulement deux valeurs possibles, par exemple des valeurs dans l'ensemble $\{-1, +1\}$ ou $\{0, 1\}$.
- Toeplitz : les éléments de la matrice sur une diagonale descendant de gauche à droite sont les mêmes.
- Bloc-diagonale : une matrice possédant des blocs sur la diagonale principale, tels que les blocs non-diagonaux soient des matrices nulles.

Les matrices binaires $\{0, 1\}$ sont extrêmement étudiées parce qu'elles sont moins complexes et faciles à réaliser en circuits électroniques. Grâce à cette propriété binaire de la matrice de mesure, l'encodeur correspondant effectue seulement une opération d'addition.

Des études ont été aussi faites sur l'impact de la structure de la matrice de mesure sur les performances de l'encodeur en termes de surface et de consommation. Les résultats obtenus ont montré que les matrices de mesure ayant une structure Toeplitz présentent des avantages majeurs [ACD+10]. Rappelons qu'une matrice Toeplitz est une matrice dont les éléments sur une diagonale descendante de gauche à droite sont les mêmes.

$$\mathbf{\Phi}_{\mathrm{T}} = \begin{bmatrix} \phi_{\mathrm{n}} & \phi_{\mathrm{n-1}} & \cdots & \phi_{1} \\ \phi_{\mathrm{n+1}} & \phi_{\mathrm{n}} & \cdots & \phi_{2} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{\mathrm{2n-1}} & \phi_{\mathrm{2n-2}} & \cdots & \phi_{\mathrm{n}} \end{bmatrix} \tag{3.1}$$

L'équation (3.1) donne un exemple de matrice ayant une structure Toeplitz où les éléments de la diagonale sont tous ϕ_{n}.

3.2.1 Construction de la matrice de mesure proposée

Pour faciliter l'implémentation de l'encodeur, nous proposons une matrice de mesure déterministe binaire bloc-diagonale (DBBD), que nous notons $\mathbf{\Phi}_{\mathrm{DBBD}}$, construite comme suit :

$$\mathbf{\Phi}_{\text{DBBD}} = \begin{bmatrix} \mathbf{B} & 0 & \dots & 0 \\ 0 & \mathbf{B} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \mathbf{B} \end{bmatrix} \tag{3.2}$$

$\mathbf{\Phi}_{\text{DBBD}}$ est binaire puisque ses éléments prennent seulement des valeurs $\{0, 1\}$. Elle a les mêmes propriétés qu'une Toeplitz puisque les blocs $\mathbf{B}$ formant sa diagonale sont identiques. $\mathbf{\Phi}_{\text{DBBD}}$ est déterministe puisque $\mathbf{B}$ est un vecteur ne contenant que des « 1 » et a une longueur fixe $\text{m} = \frac{\mathcal{N}}{\mathcal{M}}$. Rappelons que $\mathcal{M}$ et $\mathcal{N}$ représentent respectivement le nombre de lignes et le nombre de colonnes de la matrice. Par exemple pour $\text{m} = 2$, alors $\mathbf{B} = [1\ 1]$; pareillement pour $\text{m} = 3$, alors $\mathbf{B} = [1\ 1\ 1]$. D'après la description de la matrice de mesure, l'encodeur correspondant accumule seulement « $\text{m} = \frac{\mathcal{N}}{\mathcal{M}}$ » échantillons successifs du signal pour le compresser.

He *et al.* [HOH10] ont proposé une matrice de mesure binaire bloc-diaginale permutée (BBDP) que nous notons $\mathbf{\Phi}_{\text{BBDP}}$. Cette matrice peut être construite à partir de celle que nous proposons en permutant aléatoirement ses colonnes (voir l'équation 3.3).

$$\mathbf{\Phi}_{\text{DBBD}} \mapsto \text{ permutation aléatoire des colonnes } \mapsto \mathbf{\Phi}_{\text{BBDP}} \tag{3.3}$$

Ils ont utilisé cette matrice pour compresser des images. Ils ont démontré que la matrice $\mathbf{\Phi}_{\text{BBDP}}$ est incohérente avec les domaines de parcimonie couramment utilisés comme les ondelettes ou la DCT.

La matrice de mesure $\mathbf{\Phi}_{\text{RD}}$ du RD est ternaire bloc-diagonale. Ses éléments prennent des valeurs dans $\{-1, 0, +1\}$. $\mathbf{\Phi}_{\text{RD}}$ et $\mathbf{\Phi}_{\text{DBBD}}$ ont la même structure [TLD+10] :

$$\mathbf{\Phi}_{\text{RD}} = \begin{bmatrix} \mathbf{D} & 0 & \dots & 0 \\ 0 & \mathbf{D} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \mathbf{D} \end{bmatrix} \tag{3.4}$$

Les blocs $\mathbf{D}$ formant la diagonale de $\mathbf{\Phi}_{\text{RD}}$ ont une longueur fixe $\text{m} = \frac{\mathcal{N}}{\mathcal{M}}$. À la différence de $\mathbf{\Phi}_{\text{DBBD}}$, les éléments de $\mathbf{D}$ prennent aléatoirement des valeurs dans $\{-1, +1\}$.

La FIGURE 3.1 rapporte la cohérence entre la matrice $\mathbf{\Psi}_{\text{IDCT}}$ et quelques matrices de mesure à savoir :

a) $\mathbf{\Phi}_{\text{G}}$: matrice dont les éléments sont générés à partir d'un processus aléatoire gaussienne avec une moyenne nulle et une variance $1/\mathcal{M}$.

b) $\mathbf{\Phi}_{\text{B}[0,1]}$: matrice dont les éléments sont générés à partir d'un processus aléatoire bernoullien et prennent des valeurs dans $\{0, 1\}$.

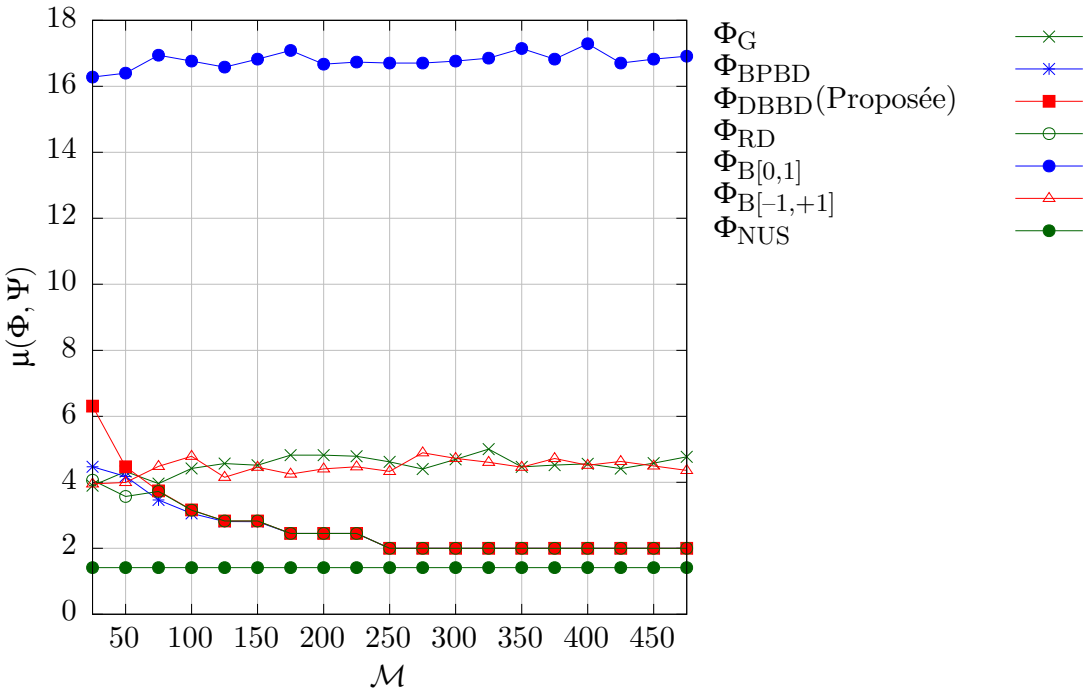


FIGURE 3.1 – *Cohérence* μ *entre la matrice* Ψ_{IDCT} *et quelques matrices de mesure, pour* $\mathcal{N} = 500$ *et* $\mathcal{M}$ *variant de 25 à 475.*

c) $\Phi_{B[-1,+1]}$: matrice dont les éléments sont générés à partir d'un processus aléatoire bernoullien et prennent des valeurs dans $\{-1, +1\}$.

d) Φ_{BPBD} : matrice de mesure proposée par He *et al.* [HOH10].

e) Φ_{RD} : matrice de mesure du *random demodulator* [TLD+10].

f) Φ_{NUS} : matrice de mesure du *non uniform sampler*.

g) Φ_{DBBD} : la matrice de mesure proposée.

Ces résultats ont été obtenus pour $\mathcal{N} = 500$ et $\mathcal{M}$ variant de 25 à 475. Les cohérences entre la matrice Ψ_{IDCT} et les matrices aléatoires gaussienne et bernoullienne restent quasiment constantes indépendamment de la valeur de $\mathcal{M}$. Elles varient aux alentours de 4.5. Les cohérences entre la matrice Ψ_{IDCT} et les matrices Φ_{BBDP}, Φ_{RD} et Φ_{DBBD} décroissent et tendent vers 2 lorsque de la valeur de $\mathcal{M}$ augmente. Rappelons que la borne inférieure de la cohérence est 1. Les trois courbes convergent à partir de $\mathcal{M} = 100$. Ces résultats montrent que le fait d'éliminer :

(i) la permutation aléatoire des colonnes.

(ii) la génération des valeurs aléatoires des blocs formant la diagonale.

n'affecte pas la cohérence entre la matrice de mesure et Ψ_{IDCT}.

De même, la FIGURE 3.2 rapporte la cohérence entre la matrice inverse de la transformée en ondelette de Haar ou « *inverse Haar wavelet transform (IHWT)* » Ψ_{IHWT} et les matrices de mesure utilisées précédemment avec la matrice Ψ_{IDCT}. Ces résultats ont été obtenus pour $\mathcal{N} = 500$ et $\mathcal{M}$ variant de 25 à 475. Cette figure montre que les matrices Φ_G et $\Phi_{B[-1,+1]}$

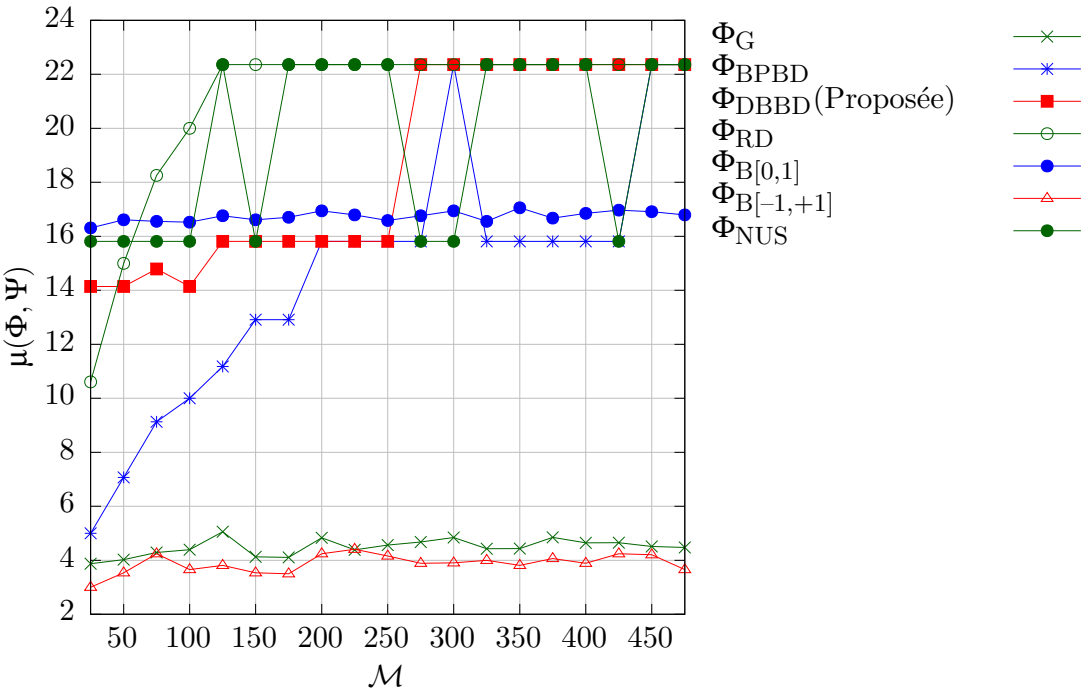


FIGURE 3.2 – *Cohérence* μ *entre la matrice* Ψ_{IHWT} *et quelques matrices de mesure, pour* $\mathcal{N} = 500$ *et* $\mathcal{M}$ *variant de 25 à 475.*

sont incohérentes avec la matrice Ψ_{IHWT}. Par contre les matrices Φ_{BBDP}, Φ_{RD} et Φ_{DBBD} sont cohérentes avec Ψ_{IHWT} puisque les valeurs de μ sont élevées. Elles se rapprochent de la borne supérieure, $\sqrt{\mathcal{N}}$, lorsque $\mathcal{M}$ augmente.

3.2.2 Complexité en termes de calcul et allocation mémoire

TABLEAU 3.1 – *Complexité en termes de calcul et d'allocation mémoire dans le cas d'un encodeur numérique.*

Φ	**Multiplication**	**Addition**	**Allocation mémoire**
Φ_G	$\mathcal{M} \times \mathcal{N}$	$\mathcal{M} \times (\mathcal{N} - 1)$	$\mathcal{M} \times \mathcal{N}$
$\Phi_{B[-1,+1]}$	0	$\mathcal{M} \times (\mathcal{N} - 1)$	$\mathcal{M} \times \mathcal{N}$
Φ_{BBDP}	0	$\mathcal{M} \times (m - 1) = \mathcal{N} - \mathcal{M}$	$\mathcal{M} \times \mathcal{N}$
Φ_{RD}	0	$\mathcal{M} \times (m - 1) = \mathcal{N} - \mathcal{M}$	$\mathcal{N}$
Φ_{DBBD}	0	$\mathcal{M} \times (m - 1) = \mathcal{N} - \mathcal{M}$	0

Le TABLEAU 3.1 rapporte la complexité en termes de calcul et d'allocation mémoire de quelques matrices de mesure dans le cas d'un encodeur numérique. La matrice gaussienne Φ_G est la plus complexe puisque l'encodeur correspondant doit effectuer à la fois des opérations d'addition et de multiplication sur des nombres réels. De plus, tous les éléments

de la matrice doivent être sauvegardés en mémoire. Le nombre de bits pour coder les éléments de la matrice dépend de l'implémentation. Pour réduire la complexité de l'encodeur, Gangopadhyay *et al.* [GAD+14] ont proposé une implémentation où chaque élément de la matrice est codé avec seulement 6 bits.

La matrice bernoullienne $\{\pm 1\}$ $\mathbf{\Phi}_{\mathrm{B}[-1,+1]}$ simplifie davantage l'implémentation de l'encodeur en éliminant les opérations de multiplication. De plus, les éléments de la matrice peuvent être codés chacun sur 1 bit. Cependant ils doivent être aussi tous sauvegardés dans la mémoire interne de l'encodeur.

Les matrices $\mathbf{\Phi}_{\mathrm{BBDP}}$, $\mathbf{\Phi}_{\mathrm{RD}}$ et $\mathbf{\Phi}_{\mathrm{DBBD}}$ diminuent le nombre d'additions à $(\mathcal{N}-\mathcal{M})$ au lieu de $\mathcal{M} \times (\mathcal{N}-1)$. Cependant la matrice que nous proposons $\mathbf{\Phi}_{\mathrm{DBBD}}$ facilite l'implémentation parce qu'elle n'a pas besoin d'emplacement mémoire. En effet pour $\mathbf{\Phi}_{\mathrm{BBDP}}$ et $\mathbf{\Phi}_{\mathrm{RD}}$, les éléments de la matrice doivent être sauvegardés en mémoires. Dans le cas d'un encodeur analogique, les éléments pseudo-aléatoires de la matrice peuvent être générés à partir de registres à décalage à rétroaction linéaire ou LFSR, au lieu de les sauvegarder dans une mémoire interne.

3.2.3 Performances de la matrice de mesure

L'objectif de ce paragraphe est de comparer les performances de la matrice de mesure proposée $\mathbf{\Phi}_{\mathrm{DBBD}}$ en termes de qualité de reconstruction par rapport aux matrices suivantes :

a) Une matrice gaussienne $\mathbf{\Phi}_{\mathrm{G}}$: $\phi_{\mathrm{i,j}} \in \mathcal{N}(0, 1/\mathcal{M})$.

b) La matrice $\mathbf{\Phi}_{\mathrm{BPBD}}$ proposée par [HOH10].

c) La matrice de mesure du *random demodulator* $\mathbf{\Phi}_{\mathrm{RD}}$ [TLD+10].

d) Une matrice binaire Bernoulli $\mathbf{\Phi}_{\mathrm{B}[0,1]}$.

e) Une matrice Bernoulli $\mathbf{\Phi}_{\mathrm{B}[-1,+1]}$.

f) La matrice de mesure de l'échantillonneur non uniforme $\mathbf{\Phi}_{\mathrm{NUS}}$.

Les simulations ont été faites sous MATLAB. Des algorithmes de reconstruction de l'outil SparseLab [Spa] ont été utilisés. Pour une valeur de $\mathcal{K}$ donnée, les $\mathcal{K}$ premiers coefficients de la transformée DCT $\boldsymbol{\alpha}$ ont été générés aléatoirement à partir d'un processus uniformément distribué dans l'intervalle $[-10, 10]$. Les coefficients restants ont été mis à zéro.

Nous avons effectué une série de tests durant lesquels nous avons appliqué la phase de mesure de l'AC en multipliant la transformée DCT $\boldsymbol{\alpha}$ par les matrices de mesure suivantes :

a) $\mathbf{A}_1 = \mathbf{\Phi}_{\mathrm{G}}\mathbf{\Psi}_{\mathrm{IDCT}}$

b) $\mathbf{A}_2 = \mathbf{\Phi}_{\mathrm{BPBD}}\mathbf{\Psi}_{\mathrm{IDCT}}$

c) $\mathbf{A}_3 = \mathbf{\Phi}_{\mathrm{DBBD}}\mathbf{\Psi}_{\mathrm{IDCT}}$

d) $\mathbf{A}_4 = \mathbf{\Phi}_{\mathrm{RD}}\mathbf{\Psi}_{\mathrm{IDCT}}$

e) $\mathbf{A}_5 = \mathbf{\Phi}_{\mathrm{B}[0,1]}\mathbf{\Psi}_{\mathrm{IDCT}}$

f) $\mathbf{A}_6 = \mathbf{\Phi}_{\mathrm{B}[-1,+1]}\mathbf{\Psi}_{\mathrm{IDCT}}$

g) $\mathbf{A}_7 = \mathbf{\Phi}_{\mathrm{NUS}}\mathbf{\Psi}_{\mathrm{IDCT}}$

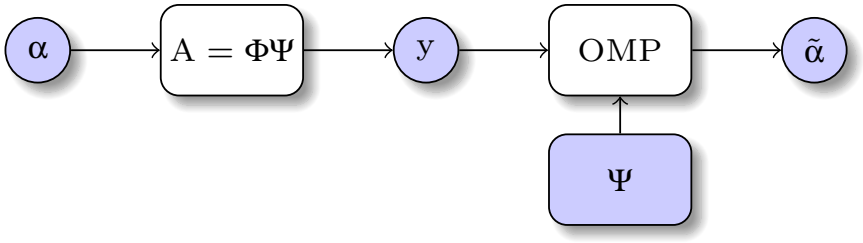


FIGURE *3.3 – Compression et reconstruction de la transformée DCT* α *générée aléatoirement.*

Comme illustré à la FIGURE 3.3, nous avons reconstruit la transformée DCT $\tilde{\alpha}$ avec l'algorithme OMP. Nous avons fixé la valeur de $\mathcal{N}$ à 500 et nous avons pris trois valeurs de $\mathcal{M} \in \{25, 35, 45\}$.

Nous avons répété les tests 100 fois pour chaque $\mathcal{K} \in \{1, \ldots, 50\}$, puis nous avons évalué le taux de succès. Nous avons supposé que la transformée DCT était reconstruite avec succès lorsque l'erreur $\varepsilon = \|\alpha - \tilde{\alpha}\|_2$ est inférieure ou égale à 10^{-6} :

$$\text{succès} \Leftrightarrow \varepsilon = \|\alpha - \tilde{\alpha}\|_2 \leq 10^{-6} \tag{3.5}$$

Nous avons défini le taux de succès comme suit :

$$\text{taux de succès} = \frac{\text{Nombre de tests avec succès}}{\text{Nombre total de tests}} \tag{3.6}$$

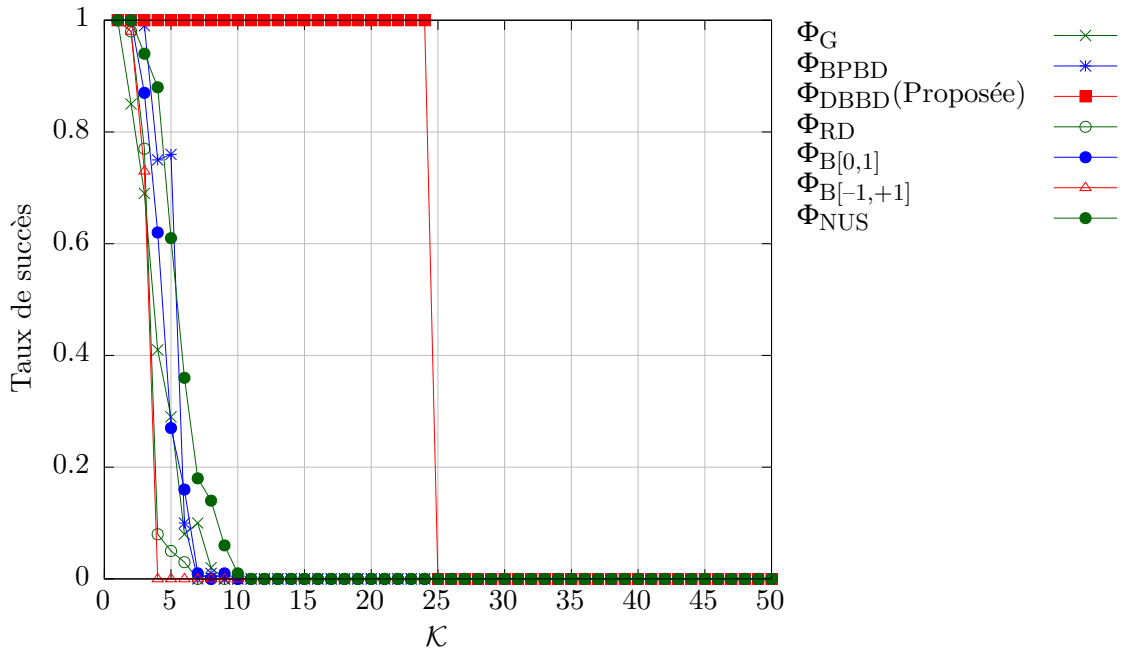


FIGURE *3.4 – Taux de succès pour* $\mathcal{M} = 25$.

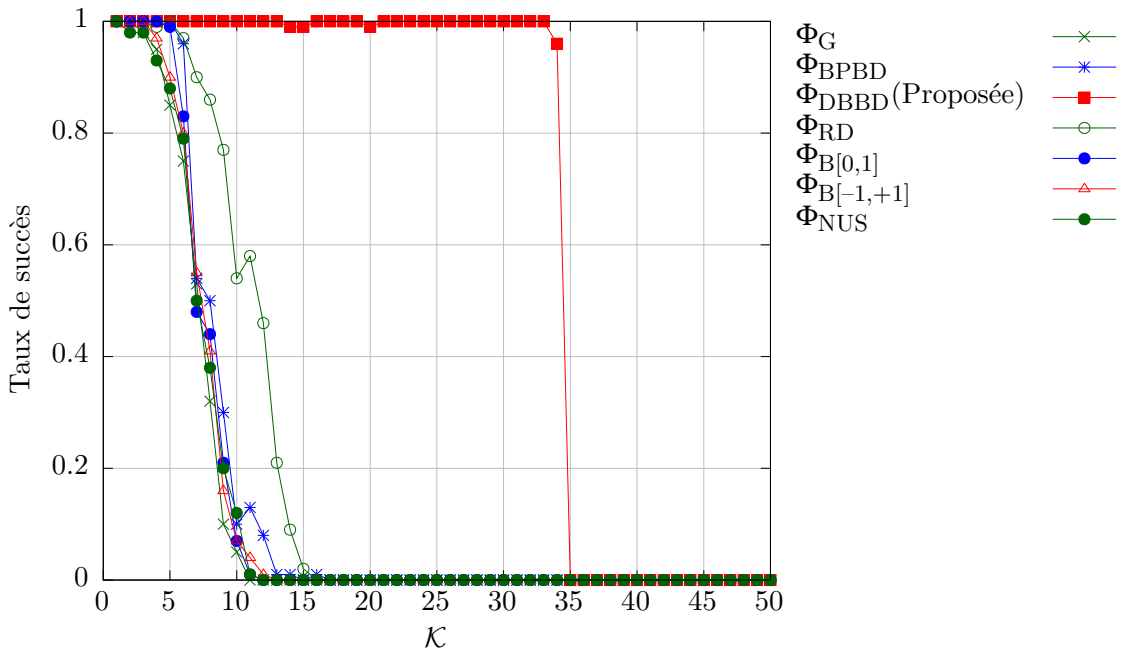


FIGURE *3.5 – Taux de succès pour* $\mathcal{M} = 35$.

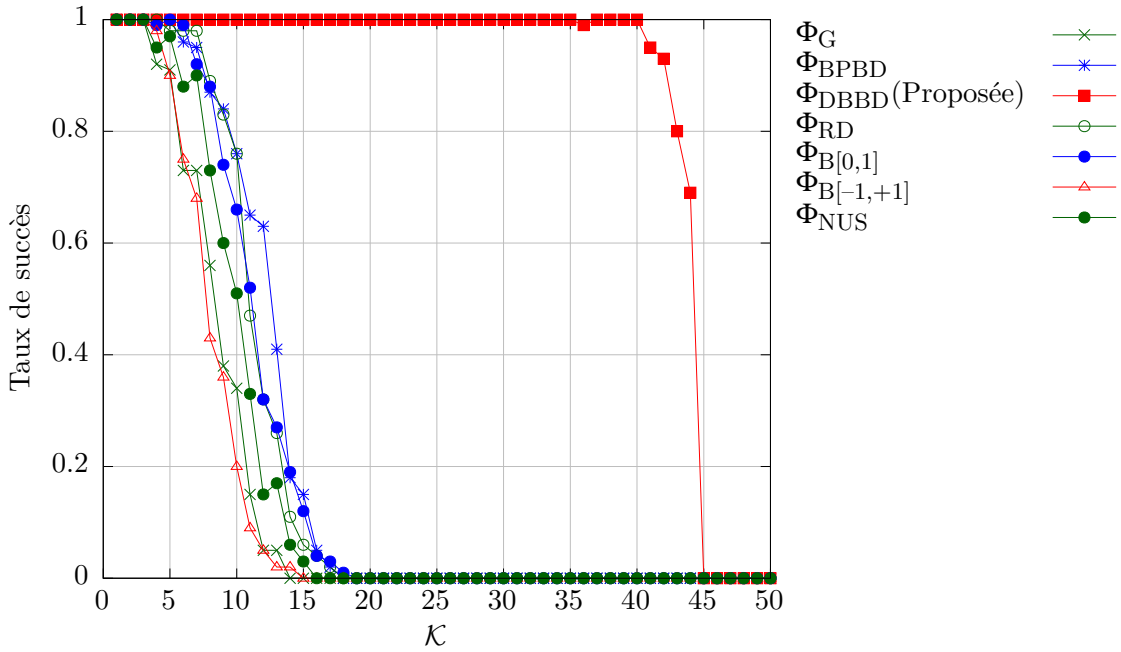


FIGURE *3.6 – Taux de succès pour* $\mathcal{M} = 45$.

Les FIGURE 3.4, FIGURE 3.5 et FIGURE 3.6 illustrent les résultats obtenus pour les trois valeurs de $\mathcal{M}$ égales à 25, 35 et 45. La valeur de $\mathcal{N}$ est fixée à 500. Indépendamment de la valeur de $\mathcal{M}$ nous avons obtenu presque les mêmes taux de succès avec les matrices aléatoires. En effet, les études faites par Tropp *et al.* [TG07] ont montré que lorsque la matrice de mesure est générée à partir d'un processus aléatoire, l'algorithme OMP reconstruit un signal parcimonieux avec une forte probabilité lorsque la condition suivante est satisfaite :

$$\mathcal{K} \leq \frac{\mathcal{M}}{1.5 \ln \mathcal{N}} \tag{3.7}$$

où, $\mathcal{K}$ est le nombre d'éléments non nuls du signal, $\mathcal{M}$ est le nombre de mesures prises lors de la phase de mesure de l'AC et $\mathcal{N}$ est la dimension du signal. En appliquant l'équation (3.7), le TABLEAU 3.2 donne la borne supérieure de $\mathcal{K}$ pour $\mathcal{N} = 500$ et $\mathcal{M} \in \{25, 35, 45\}$.

TABLEAU *3.2 – Borne supérieure de $\mathcal{K}$ avec l'OMP dans le cas d'une matrice de mesure aléatoire pour $\mathcal{N} = 500$.*

$\mathcal{N} = 500$	
$\mathcal{M}$	$\mathcal{K}$
25	$2.6 \approx 2$
35	$3.7 \approx 3$
45	$4.8 \approx 4$

Ces figures montrent que les résultats obtenus lors de l'expérimentation se concordent avec le TABLEAU 3.2 pour les matrices aléatoires. Par contre, l'algorithme OMP est plus efficace en termes de taux de succès avec la matrice déterministe proposée par rapport aux matrices aléatoires. En effet, d'après ces figures, l'OMP peut reconstruire la transformée DCT avec un taux de succès supérieur à 0.6 lorsque $\mathcal{K}$ strictement inférieur à $\mathcal{M}$:

$$\mathcal{K} < \mathcal{M} \tag{3.8}$$

L'OMP devient plus performant avec la matrice déterministe proposée puisque avec elle la borne supérieure de $\mathcal{K}$ est multipliée d'environ $1.5 \ln \mathcal{N}$ fois.

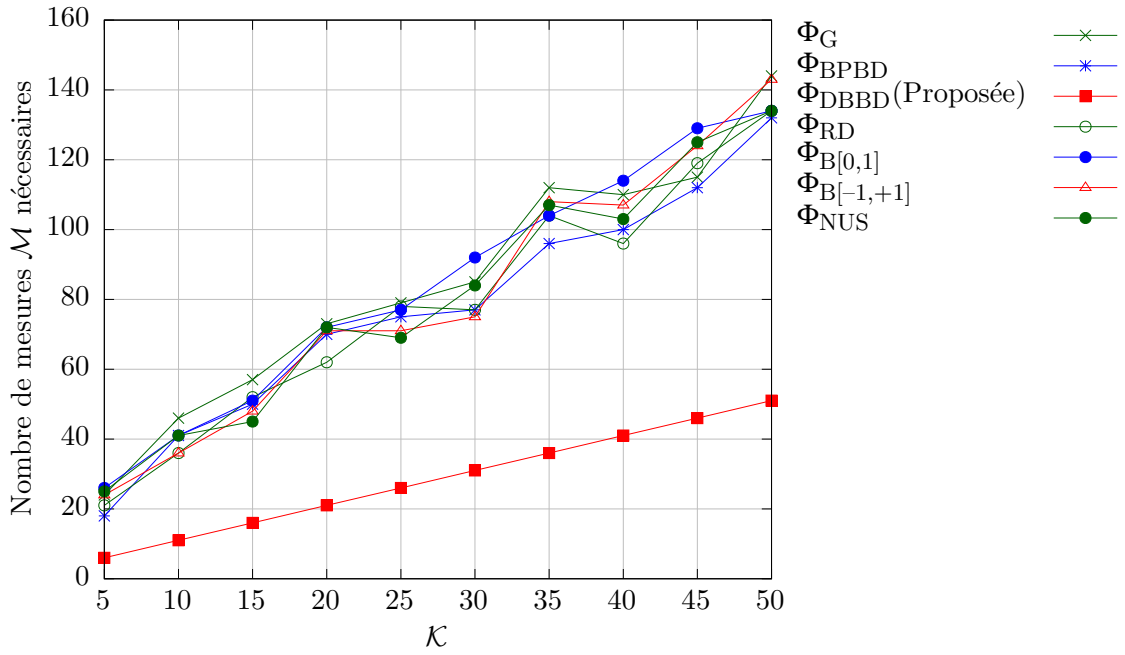


FIGURE *3.7 – Nombre de mesures $\mathcal{M}$ nécessaires pour reconstruire la transformée DCT avec une erreur $\varepsilon \leq 10^{-6}$, pour une valeur constante de $\mathcal{N}$ égale à 500.*

Nous avons aussi évalué le nombre de mesures $\mathcal{M}$ nécessaires pour reconstruire la transformée DCT avec une erreur $\boldsymbol{\varepsilon} < 10^{-6}$ pour chaque $\mathcal{K} \in \{1, \ldots, 50\}$, $\mathcal{N}$ étant fixé à 500. La FIGURE 3.7 rapporte les résultats obtenus. Ils confirment les résultats obtenus lors de l'expérimentation précédente. L'OMP a une meilleure performance avec la matrice déterministe proposée par rapport aux matrices aléatoires. La pente de la droite $\mathcal{M} = \mathrm{f}(\mathcal{K})$ est largement réduite avec la matrice déterministe proposée comparée aux matrices aléatoires. Pour une même valeur de $\mathcal{K}$, avec elle l'OMP a toujours besoin de moins de mesures pour reconstruire la transformée DCT avec une erreur très faible.

Pour évaluer davantage les performances de la matrice déterministe proposée sur des signaux réels, nous avons pris des signaux ECG et EMG. En pratique, nous appliquons la phase de mesure de l'AC sur le signal $\mathbf{x}$ mais pas sur sa transformée DCT $\boldsymbol{\alpha}$. Nous utilisons directement la matrice $\boldsymbol{\Phi}$ à la place de $\mathbf{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$. Par rapport aux expérimentations précédentes, cela facilite la phase de mesure de l'AC mais ajoute un traitement supplémentaire lors de la reconstruction.

i) Application à des signaux électrocardiogrammes

Les signaux ECG proviennent de la base de données *MIT-BIH Arrhythmia* (mitdb) du site Physionet [Phy]. Les 19 premiers enregistrements ont été choisis.

Comme illustré à la FIGURE 3.8, nous avons divisé le signal ECG en blocs contenant chacun $\mathcal{N}$ échantillons consécutifs. Chaque bloc étant considéré comme un vecteur $\mathbf{x} \in \mathbb{R}^{\mathcal{N}}$. Nous avons compressé chaque bloc du signal avec les matrices de mesure citées précédemment. Nous avons utilisé l'algorithme OMP pour reconstruire les blocs. Lorsque le signal ECG est entièrement reconstruit, nous avons évalué leur PRD et leur SNR.

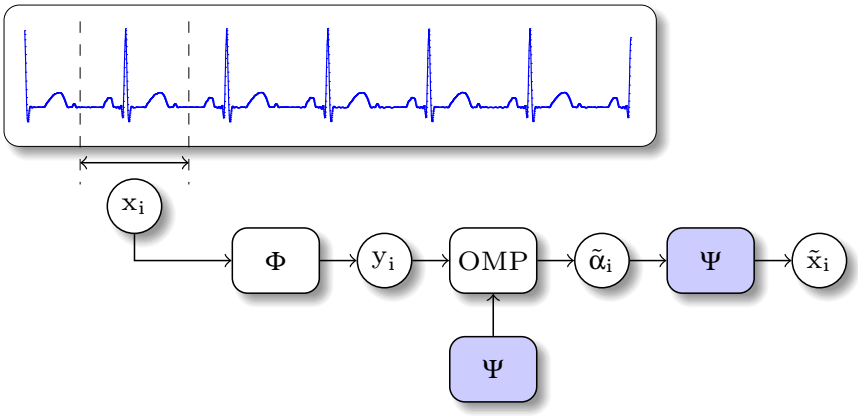


FIGURE *3.8 – Compression et reconstruction du signal ECG provenant de la base de donnée MIT-BIH Arrhythmia du site Physionet [Phy].*

Pour chaque enregistrement, nous avons effectué une série de tests durant lesquels nous avons fixé $\mathcal{N}$ à 500 et nous avons varié la valeur du taux de compression CR. Puis, nous avons calculé la valeur moyenne des PRD et celle des SNR.

Les FIGURE 3.9 et FIGURE 3.10 rapportent les résultats obtenus. Comme attendu, la

qualité des signaux ECG reconstruits décroit avec CR. Les résultats obtenus sont conformes aux expérimentations précédentes. En effet, quelque soit la valeur de CR, les signaux ECG reconstruits avec la matrice de mesure déterministe proposée ont une meilleure qualité en termes de PRD et de SNR. Nous avons obtenu presque les mêmes PRD et SNR avec les matrices aléatoires.

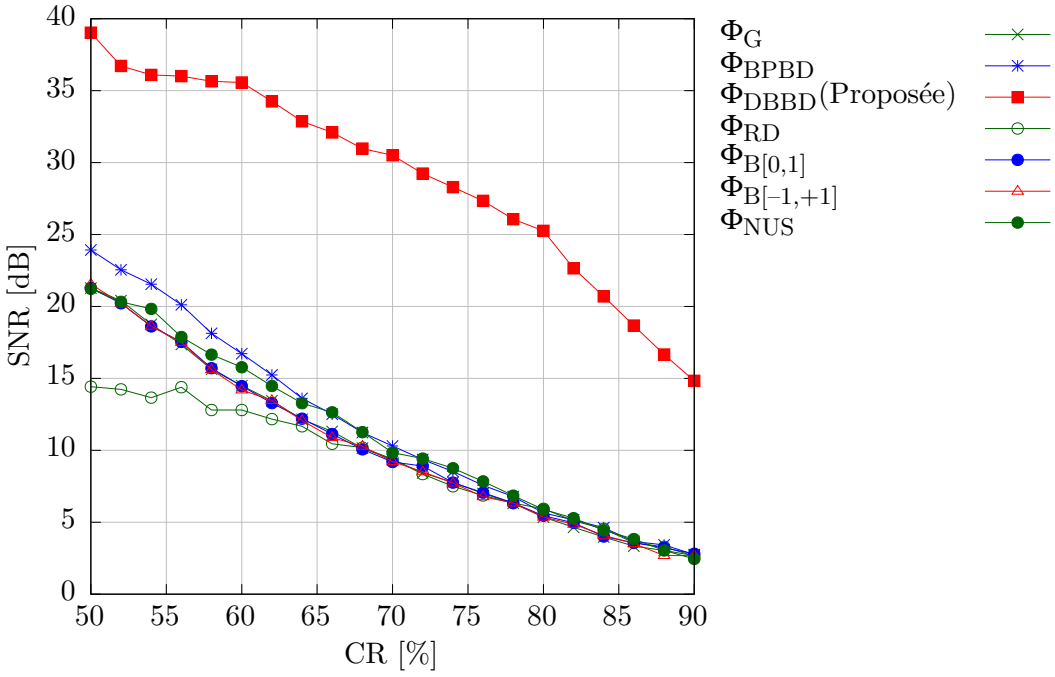


FIGURE *3.9 – SNR en fonction du taux de compression.*

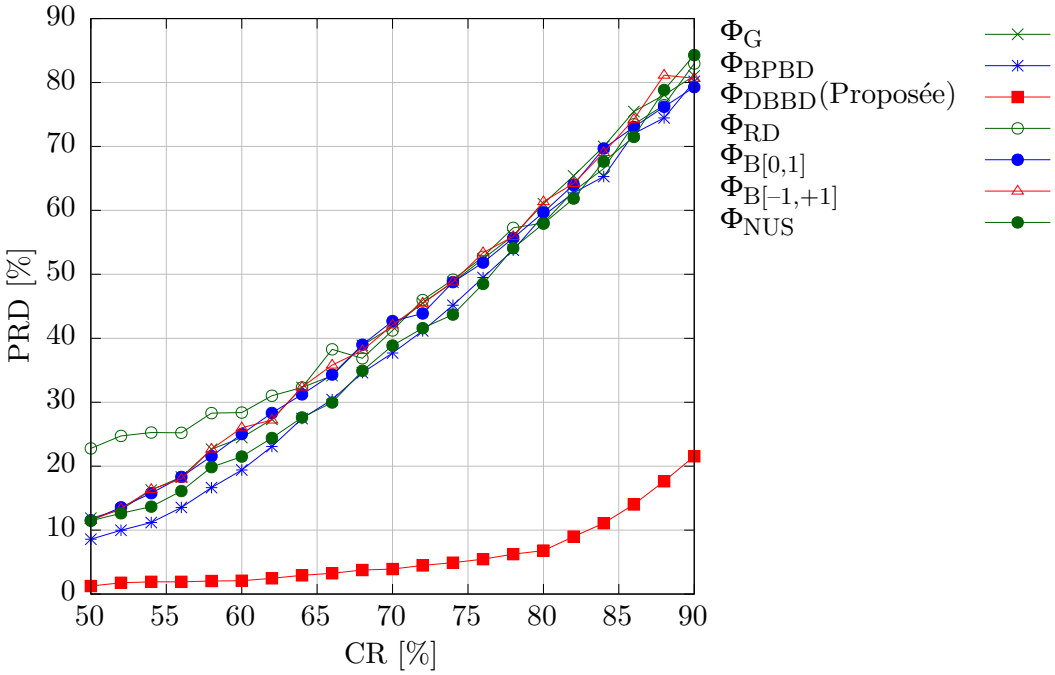
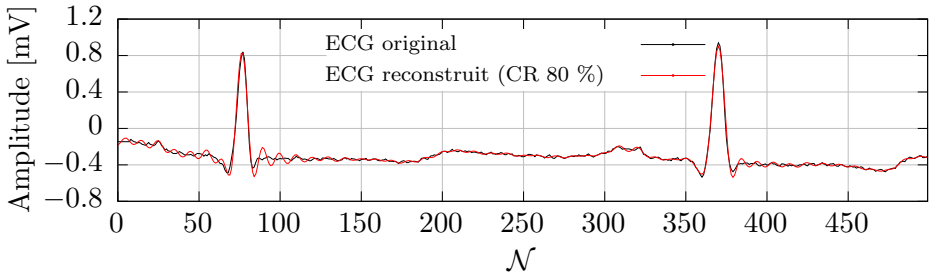


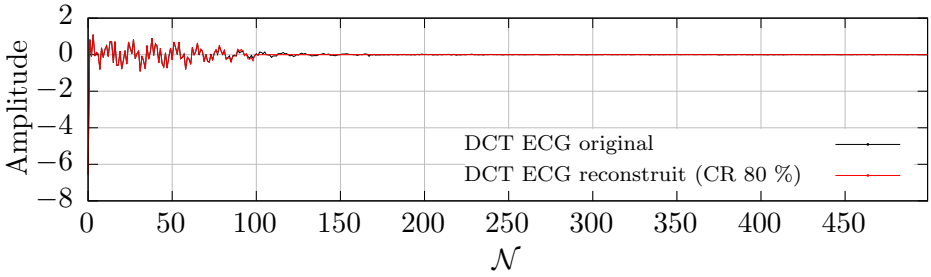
FIGURE *3.10 – PRD en fonction du taux de compression.*

Ces résultats confirment que la matrice de mesure déterministe proposée a une meilleure performance en termes de qualité de reconstruction que les matrices aléatoires. Dépendant

de la valeur du taux de compression CR, la matrice déterministe améliore le SNR des signaux ECG reconstruits de 6 à 20 dB et le PRD de 11 à 30 %.

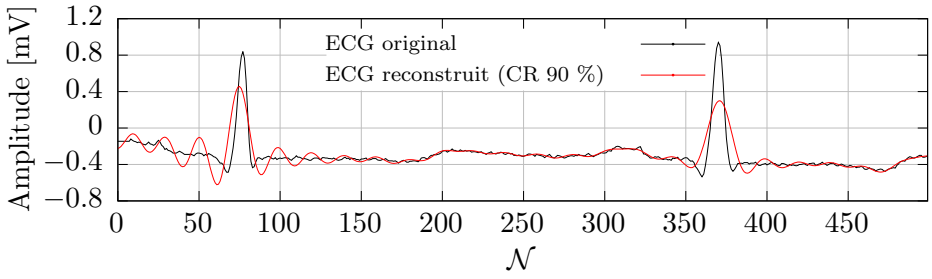


(a) Signaux ECG (enregistrement numéro 100).

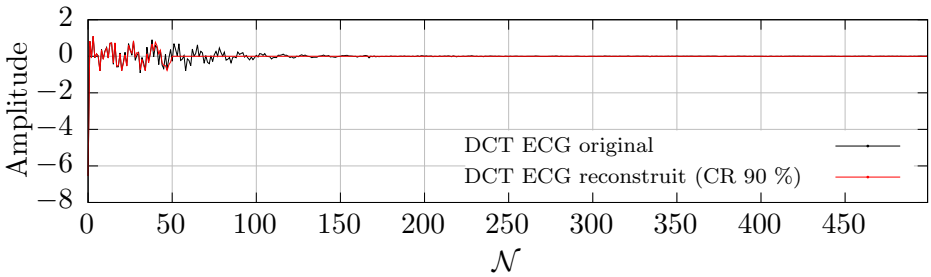


(b) DCT des signaux ECG.

FIGURE *3.11 – Signal ECG original (noir) et celui reconstruit (rouge) avec un taux de compression de 80 % ($\mathcal{M} = 100$ et $\mathcal{N} = 500$). Le signal était compressé avec la matrice de mesure déterministe proposée.*



(a) Signaux ECG (enregistrement numéro 100).



(b) DCT des signaux ECG.

FIGURE *3.12 – Signal ECG original (noir) et celui reconstruit (rouge) avec un taux de compression de 90 % ($\mathcal{M} = 50$ et $\mathcal{N} = 500$). Le signal était compressé avec la matrice de mesure déterministe proposée.*

Les FIGURE 3.11 et FIGURE 3.12 illustrent une partie des signaux ECG originaux et ceux reconstruits avec des taux de compression de 80 % et de 90 %, pour l'enregistrement numéro

100. Dans les deux cas, le signal ECG était compressé avec la matrice de mesure déterministe proposée. La valeur de $\mathcal{N}$ était fixée à 500. Avec le CR de 80 % (lorsque $\mathcal{M} = 100$), le signal ECG est reconstruit avec une faible distorsion. La différence entre le signal original et celui reconstruit est difficilement perceptible. Par contre, pour un très fort taux de compression, CR de 90 %, le signal reconstruit présente de faibles ondulations qui pourraient nuire à l'interprétation médicale du signal. Le signal ECG reconstruit est distordu. Cependant, les piques R sont toujours visibles et l'intervalle R-R est encore respecté.

Les FIGURE 3.11(b) et FIGURE 3.12(b) montrent que le signal ECG est parcimonieux dans le domaine de la DCT. L'énergie du signal est concentrée dans les basses fréquences, à peu près dans les $\mathcal{K} = 150$ premières composantes. Pour le CR de 80 % (lorsque $\mathcal{M} = 100$), l'OMP a pu récupérer les 99 composantes basses fréquences de la DCT du signal. Le signal ECG est reconstruit avec une faible distorsion puisque les composantes restantes ont des valeurs presque nulles. Pour le CR de 90 % (lorsque $\mathcal{M} = 50$), l'OMP a pu récupérer les 49 composantes basses fréquences de la DCT du signal. Le signal ECG reconstruit est distordu puisque la plupart des composantes restantes ont des valeurs non négligeables.

ii) Application à des signaux électromyogrammes

Parmi les signaux physiologiques traités dans un WBSN, le signal EMG est principalement celui qui occupe plus de mémoire et nécessite une bande passante élevée. En effet, un signal EMG est typiquement échantillonné à une fréquence comprise entre 1 kHz et 4 kHz, et est numérisé avec 2 octets par échantillon. Le signal EMG nécessite un débit brut de l'ordre de 8 [Ko/s]. De ce fait, il est bénéfique de compresser le signal EMG pour économiser à la fois la mémoire et la bande passante.

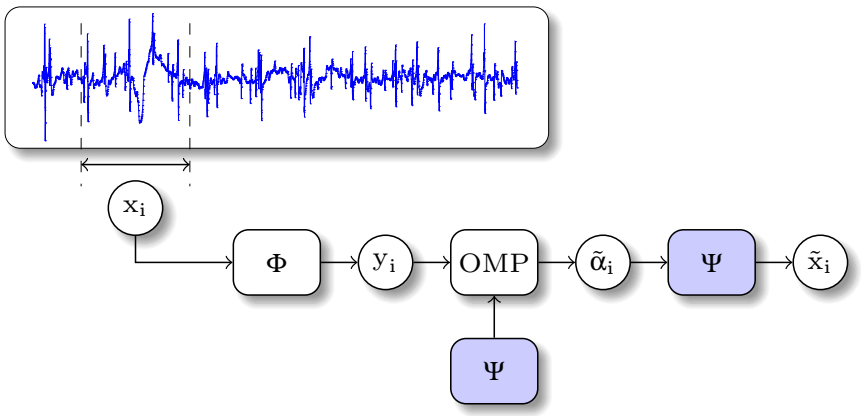


FIGURE *3.13 – Compression et reconstruction du signal EMG provenant de la base de donnée Physionet [Phy].*

Le signal EMG provient de la base de données *Electromyograms* (emgdb) du site Physionet [Phy]. L'enregistrement emg_healthy a été choisi.

La FIGURE 3.13 illustre les phases de compression et de reconstruction du signal EMG. Nous avons divisé le signal EMG en blocs de $\mathcal{N}$ échantillons consécutifs. Nous avons com-

pressé chaque bloc du signal avec les matrices de mesure citées précédemment. Nous avons utilisé l'algorithme OMP pour reconstruire les blocs. Lorsque le signal EMG est entièrement reconstruit, nous avons évalué leur PRD et leur SNR.

Nous avons effectué une série de tests durant lesquels nous avons fixé $\mathcal{N}$ à 500 et nous avons varié la valeur du taux de compression CR. Les FIGURE 3.14 et FIGURE 3.15 rapportent les résultats obtenus.

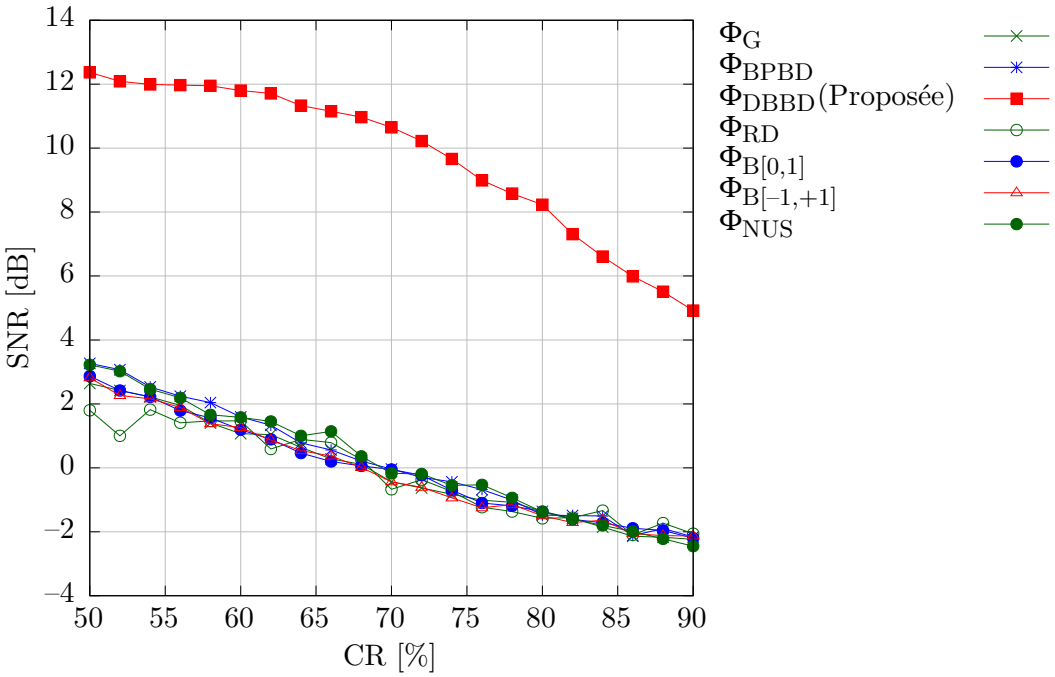


FIGURE *3.14 – SNR en fonction du taux de compression.*

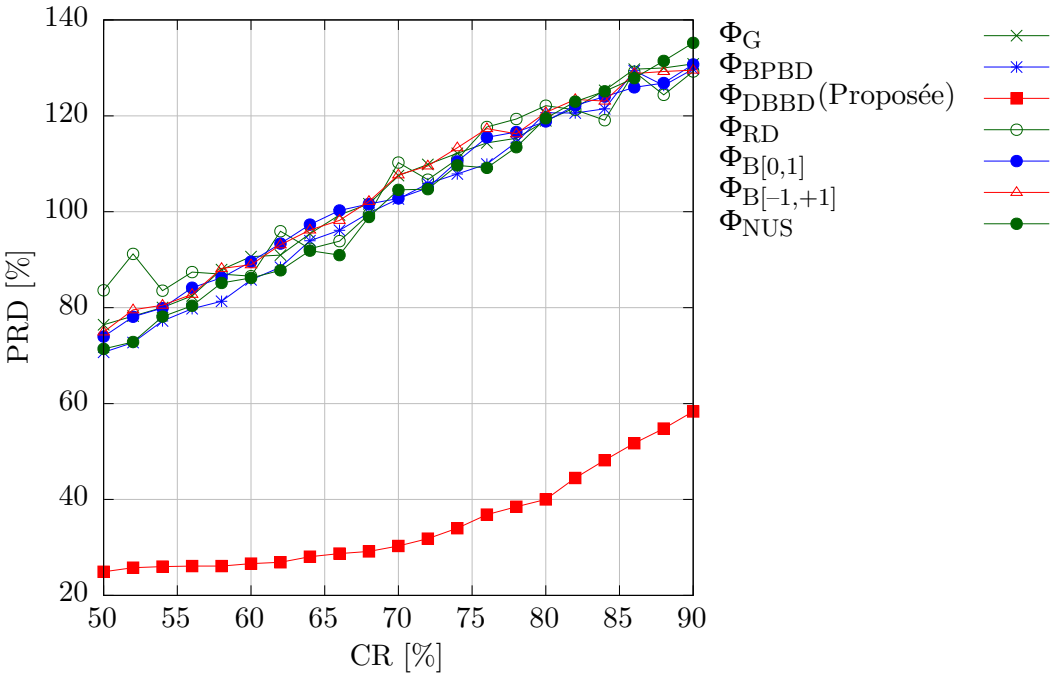
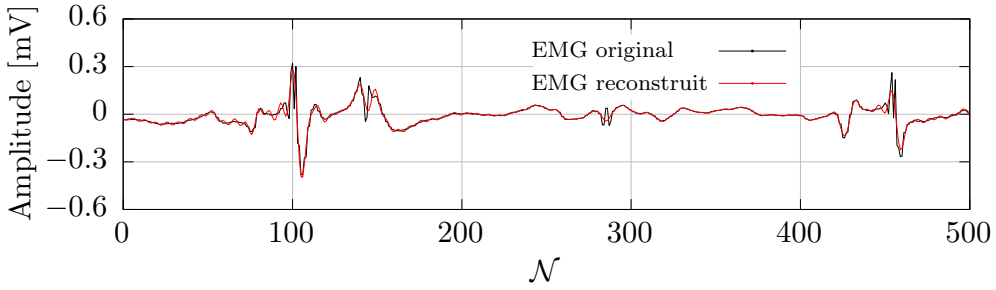
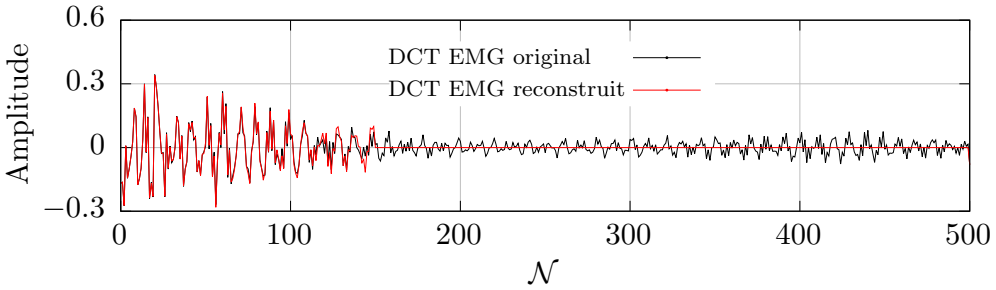


FIGURE *3.15 – PRD en fonction du taux de compression.*

La qualité du signal EMG reconstruit décroit avec CR. Le signal EMG est reconstruit avec une meilleure qualité avec la matrice de mesure déterministe proposée. Dépendant de la valeur de CR, le SNR est amélioré de 10 dB et le PRD de 50 %. En comparant ces résultats avec ceux obtenus précédemment avec le signal ECG, l'EMG est moins compressible parce qu'il est moins parcimonieux dans le domaine de la DCT.



(a) Signaux EMG (emg_healthy).



(b) DCT des signaux EMG.

FIGURE *3.16 – Signal EMG original (noir) et celui reconstruit (rouge) avec un taux de compression de 70 % ($\mathcal{M} = 150$ et $\mathcal{N} = 500$). Le signal était compressé avec la matrice de mesure déterministe proposée.*

La FIGURE 3.16(a) illustre une partie du signal EMG original (en noir) et celui reconstruit (en rouge) avec un taux de compression de 70 %, pour $\mathcal{M} = 150$ et $\mathcal{N} = 500$. Le signal EMG était compressé avec la matrice de mesure déterministe proposée. Le signal EMG reconstruit est un peu distordu par rapport à l'original (SNR = 10 dB et PRD = 30 %). Cette distorsion est due au fait que seulement les 149 premiers coefficients de sa transformée DCT ont été retenus. Alors que d'après la FIGURE 3.16(b), les coefficients restants ne sont pas négligeables.

3.3 Algorithme de reconstruction proposé

3.3.1 Description de l'algorithme

Les algorithmes gloutons, comme l'OMP, se fient à l'incohérence entre la matrice de mesure $\mathbf{\Phi}$ et celle du domaine de parcimonie $\mathbf{\Psi}$. L'OMP est un algorithme itératif et facile à mettre en œuvre. Il commence par initialiser le résiduel, l'ensemble de colonnes sélectionnées et l'approximation de la transformée comme suit : $\mathbf{r} = \mathbf{y}$, $\mathbf{\Omega}_0 = \emptyset$ et $\tilde{\boldsymbol{\alpha}} = 0$.

À chaque itération, l'OMP sélectionne une colonne de la matrice $\mathbf{A} = \mathbf{\Phi\Psi}$ qui a une corrélation maximale avec le résiduel courant. Puis, il met à jour le résiduel et calcule une nouvelle approximation de la transformée. Cette phase de sélection de l'OMP est importante car les étapes suivantes ainsi que le résultat final en dépendent. Lorsque les matrices $\mathbf{\Phi}$ et $\mathbf{\Psi}$ sont incohérentes, l'OMP n'a pas d'ambiguïté pour trouver la bonne colonne à chaque itération.

Plusieurs algorithmes à base de l'OMP qui fonctionnent mieux en pratique ont été proposés. Ces algorithmes apportent des améliorations en sélectionnant plusieurs colonnes par itération, en réduisant l'ensemble des colonnes actives à chaque itération, ou bien en résolvant le problème de manière itérative. Par exemple, le *stagewise* OMP (StOMP) sélectionne plusieurs colonnes à chaque itération pour accélérer la phase de reconstruction [TG07].

Le fait de multiplier la matrice déterministe $\mathbf{\Phi}_{\text{DBBD}}$ par la matrice $\mathbf{\Psi}_{\text{IDCT}}$ accumule $\text{m} = \frac{\mathcal{N}}{\mathcal{M}}$ lignes consécutives de $\mathbf{\Psi}_{\text{IDCT}}$. La matrice $\mathbf{A}$ résultante a la même structure que la matrice $\mathbf{\Psi}_{\text{IDCT}}$. La première colonne $\mathbf{a}_1$ de $\mathbf{A}$ correspond à la composante DC. La deuxième colonne $\mathbf{a}_2$ correspond à la première composante AC ayant la plus basse fréquence. Les colonnes restantes $\mathbf{a}_{\text{i}\in\{3,\ldots,\text{N}\}}$ correspondent aux composantes AC ayant des fréquences progressivement élevées. La FIGURE 3.17 illustre les 8 premières colonnes de la matrice $\mathbf{\Psi}_{\text{IDCT}}$ et celles de $\mathbf{A} = \mathbf{\Phi}_{\text{DBBD}}\mathbf{\Psi}_{\text{IDCT}}$ pour $\mathcal{M} = 16$ et $\mathcal{N} = 64$, respectivement notées par $\boldsymbol{\psi}_{\text{i}\in\{1,\ldots,8\}}$ et $\mathbf{a}_{\text{i}\in\{1,\ldots,8\}}$. Elle montre bien que les deux matrices ont la même structure.

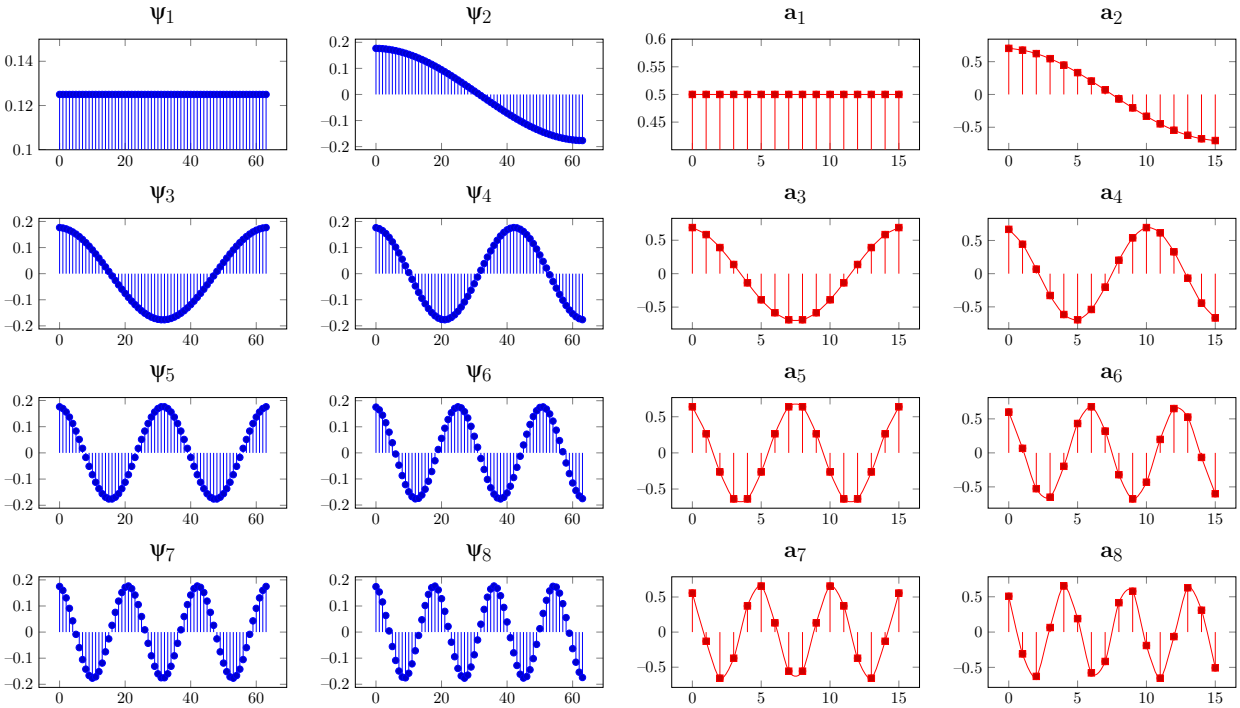


FIGURE *3.17 – Les 8 premières colonnes de la matrice* $\mathbf{\Psi}_{\text{IDCT}}$ *(en bleu) et celles de* $\mathbf{A} = \mathbf{\Phi}_{\text{DBBD}}\mathbf{\Psi}_{\text{IDCT}}$ *(en rouge) pour* $\mathcal{M} = 16$ *et* $\mathcal{N} = 64$*, respectivement notées par* $\boldsymbol{\psi}_{\text{i}\in\{1,\ldots,8\}}$ *et* $\mathbf{a}_{\text{i}\in\{1,\ldots,8\}}$.

Grâce à la matrice déterministe proposée, nous n'avons pas besoin de faire la phase de sélection de l'OMP puisque nous connaissons à l'avance l'emplacement des colonnes

significatives de $\mathbf{A}$. Nous proposons une méthode simple et rapide.

Au lieu de choisir une colonne de $\mathbf{A}$ qui a une corrélation maximale avec le résiduel courant à chaque itération, l'algorithme que nous proposons sélectionne directement les $\mathcal{M}$ premières colonnes de $\mathbf{A}$. Les colonnes sélectionnées forment une sous matrice de $\mathbf{A}$ notée $\mathbf{A}_{\Omega_\mathcal{M}} \in \mathbb{R}^{\mathcal{M}\times\mathcal{M}} = [\mathbf{a}_1|\mathbf{a}_2|\dots|\mathbf{a}_\mathcal{M}]$. Le fait de sélectionner les $\mathcal{M}$ premières colonnes de $\mathbf{A}$ revient à effectuer un seuillage dans le domaine de la DCT en gardant uniquement les $\mathcal{M}$ composantes basses fréquences de la transformée. C'est un gain de temps puisque la phase de sélection domine le temps d'exécution de l'OMP [TG07].

L'algorithme que nous proposons n'est pas itératif. Il s'exécute une seule fois et calcule directement une approximation de la transformée DCT $\tilde{\alpha}$. Au début, $\tilde{\alpha}$ est initialisée à 0. Ensuite, ses $\mathcal{M}$ premiers éléments, notés $\tilde{\alpha}_{\Omega_\mathcal{M}}$, sont mis à jour en résolvant l'équation :

$$\mathbf{y} = \mathbf{A}_{\Omega_\mathcal{M}}\tilde{\alpha}_{\Omega_\mathcal{M}} \tag{3.9}$$

Algorithme 2 Algorithme de reconstruction proposé.

Prérequis: $\mathbf{y} \in \mathbb{R}^\mathcal{M}$ {Vecteur de mesure.}

Prérequis: $\mathbf{A} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}} = \mathbf{\Phi}_{\text{DBBD}}\mathbf{\Psi}_{\text{IDCT}}$

$\tilde{\alpha} \in \mathbb{R}^\mathcal{N} \leftarrow 0$ {Initialiser la transformée du signal reconstruit.}

$\mathbf{A}_{\Omega_\mathcal{M}} \in \mathbb{R}^{\mathcal{M}\times\mathcal{M}} \leftarrow [\mathbf{a}_1|\mathbf{a}_2|\dots|\mathbf{a}_\mathcal{M}]$ {Sélectionner les $\mathcal{M}$ premières colonnes de $\mathbf{A}$.}

$\mathbf{y} = \mathbf{A}_{\Omega_\mathcal{M}}\tilde{\alpha}_{\Omega_\mathcal{M}}$ {Résoudre l'équation.}

Retourne $\tilde{\alpha} \leftarrow [\tilde{\alpha}_{\Omega_\mathcal{M}}|0\dots0]$

L'Algorithme (2) résume les étapes de l'algorithme proposé. Une fois que la transformée $\tilde{\alpha}$ est obtenue, le signal reconstruit est généré comme suit :

$$\tilde{\mathbf{x}} = \mathbf{\Psi}\tilde{\alpha} \tag{3.10}$$

3.3.2 Performances de l'algorithme

L'objectif de l'expérimentation qui suit est de comparer les performances de l'algorithme proposé avec celles de l'OMP et du StOMP. Nous avons utilisé la matrice de mesure déterministe proposée pour compresser la transformée DCT α tout au long de l'expérimentation. Pour le StOMP, nous avons fixé le nombre d'itérations à 10. Pour évaluer leur taux de succès, nous avons effectué 100 tests pour chaque valeur de $\mathcal{K} \in \{1,\dots,50\}$. Nous avons estimé que la transformée DCT est reconstruite avec succès si l'erreur $\varepsilon = \|\alpha - \tilde{\alpha}\|_2$ était inférieure ou égale à 10^{-6} ($\varepsilon \leq 10^{-6}$).

Les Figure 3.18, Figure 3.19 et Figure 3.20 rapportent les taux de succès des trois algorithmes en fonction de $\mathcal{K}$ pour $\mathcal{M} \in \{25, 35, 45\}$ et $\mathcal{N}$ fixé à 500. Indépendamment de la

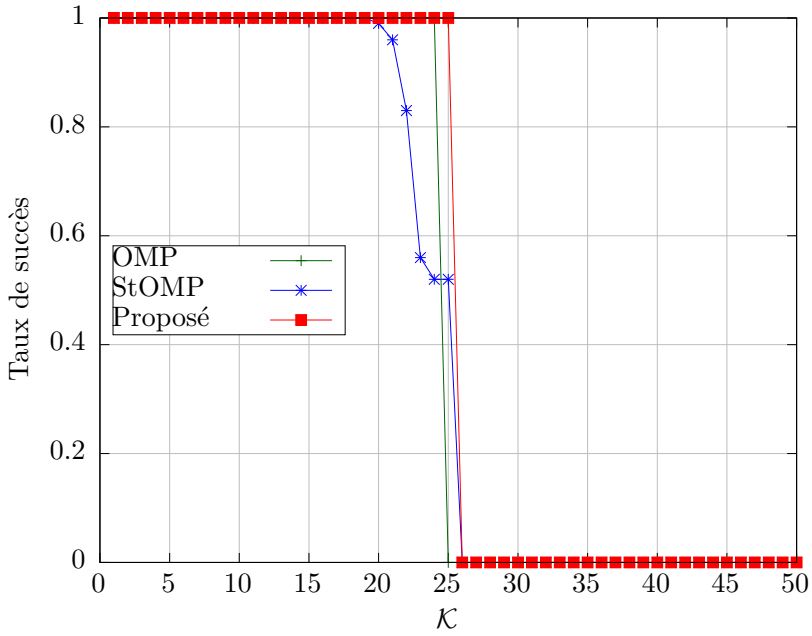


FIGURE *3.18 – Taux de succès pour* $\mathcal{M} = 25$.

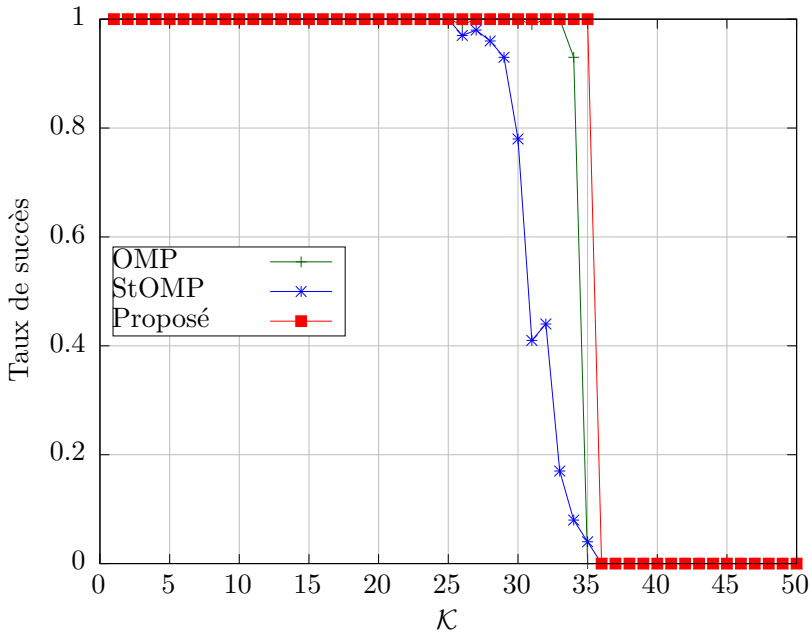


FIGURE *3.19 – Taux de succès pour* $\mathcal{M} = 35$.

valeur de $\mathcal{M}$, ces résultats montrent que l'algorithme proposé a une performance meilleure par rapport à l'OMP en termes de taux de succès. L'algorithme proposé peut reconstruire la transformée DCT avec succès lorsque le nombre d'éléments non nuls $\mathcal{K}$ est inférieur ou égal au nombre de mesures $\mathcal{M}$ ($\mathcal{K} \leq \mathcal{M}$). Par contre l'OMP reconstruit la transformée DCT avec un taux de succès supérieur à 0.6 seulement lorsque $\mathcal{K}$ est strictement inférieur à $\mathcal{M}$ ($\mathcal{K} < \mathcal{M}$). Comparé à l'OMP et l'algorithme proposé, le StOMP est moins efficace en termes de taux de succès .

La FIGURE 3.21 confirme les résultats obtenus. Elle illustre le nombre de mesures nécessaires $\mathcal{M}$ pour reconstruire la transformée DCT avec une erreur $\varepsilon \leq 10^{-6}$. Pour le StOMP, la

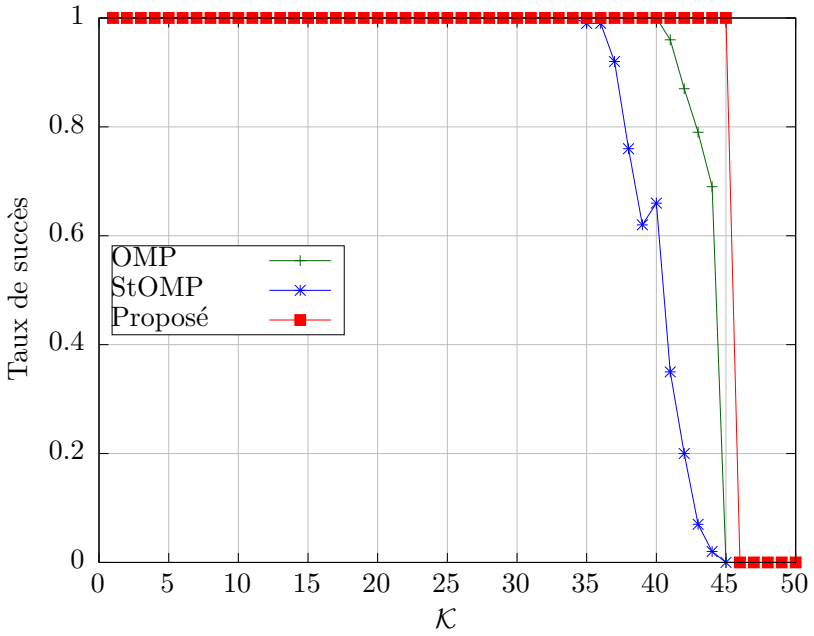


FIGURE *3.20 – Taux de succès pour* $\mathcal{M} = 45$.

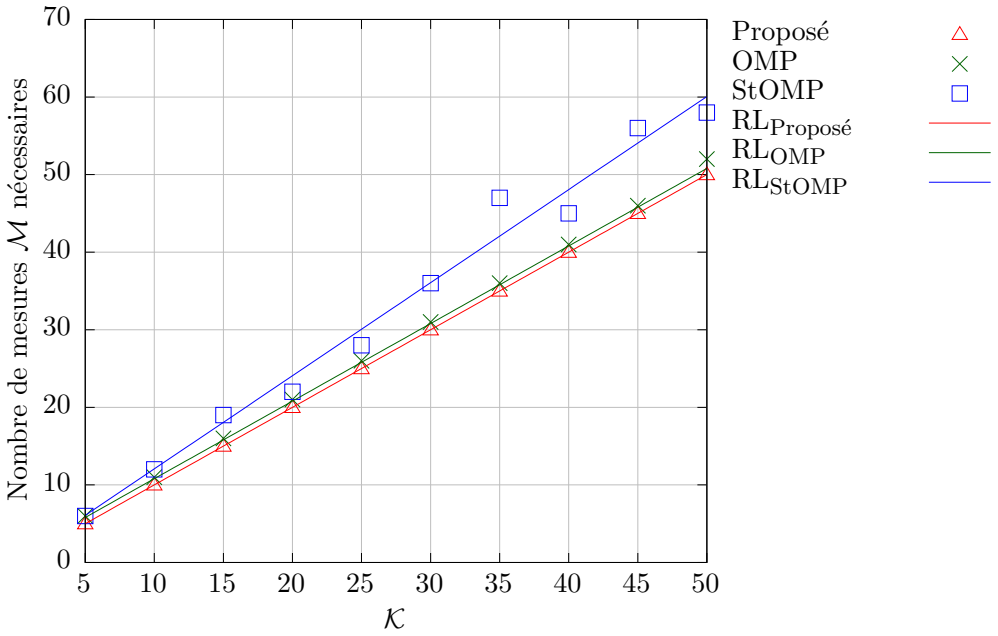


FIGURE *3.21 – Nombre de mesures* $\mathcal{M}$ *nécessaires pour reconstruire la transformée DCT avec une erreur* $\varepsilon \leq 10^{-6}$. *Les régressions linéaires ont respectivement les équations suivantes :* $\mathrm{RL}_{Proposé}$: $\mathcal{M} = \mathcal{K}$, $\mathrm{RL}_{\mathrm{OMP}} : \mathcal{M} = \mathcal{K} + 0.8$ *et* $\mathrm{RL}_{\mathrm{StOMP}} : \mathcal{M} = 1.2\mathcal{K} + 0.067$.

pente de régression linéaire est légèrement supérieure à celle de l'OMP et celle l'algorithme proposé.

i) Application à des signaux électrocardiogrammes et électromyogrammes

Comme exemples concrets, nous avons aussi pris des signaux ECG et EMG provenant de la base de données Physionet. Nous avons divisé les signaux en blocs de $\mathcal{N}$ échantillons consécutifs. Nous avons compressé chaque bloc avec la matrice de mesure déterministe

proposée, puis nous avons reconstruit respectivement chaque bloc avec les trois algorithmes de reconstruction. Lorsque les signaux sont entièrement reconstruits, nous avons évalué leur PRD et leur SNR.

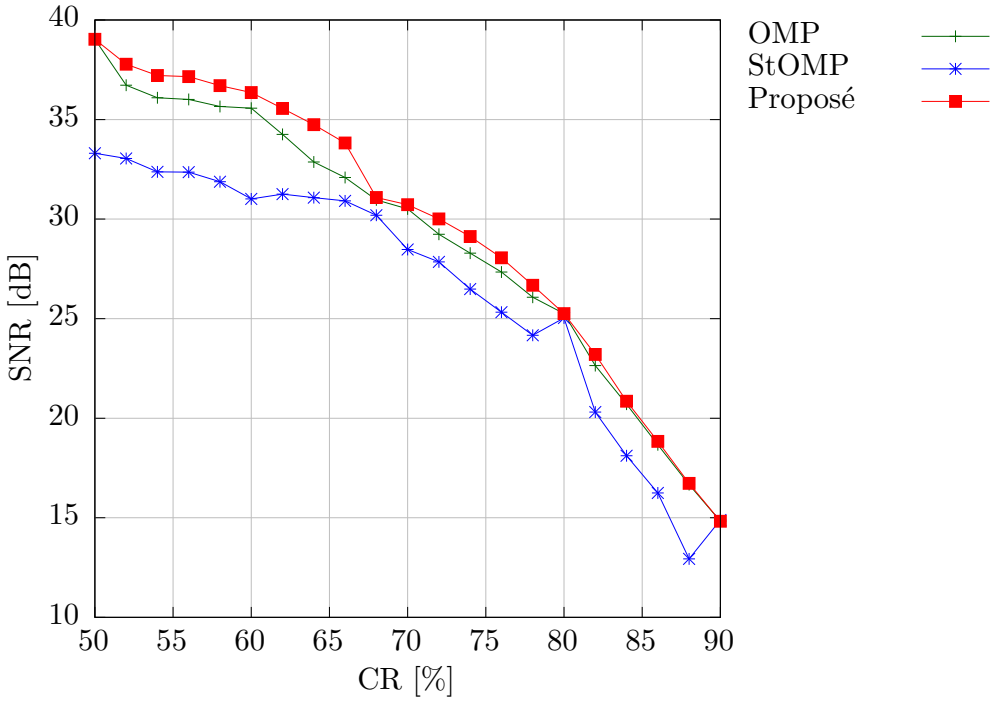


FIGURE 3.22 – *SNR en fonction du taux de compression pour les signaux ECG.*

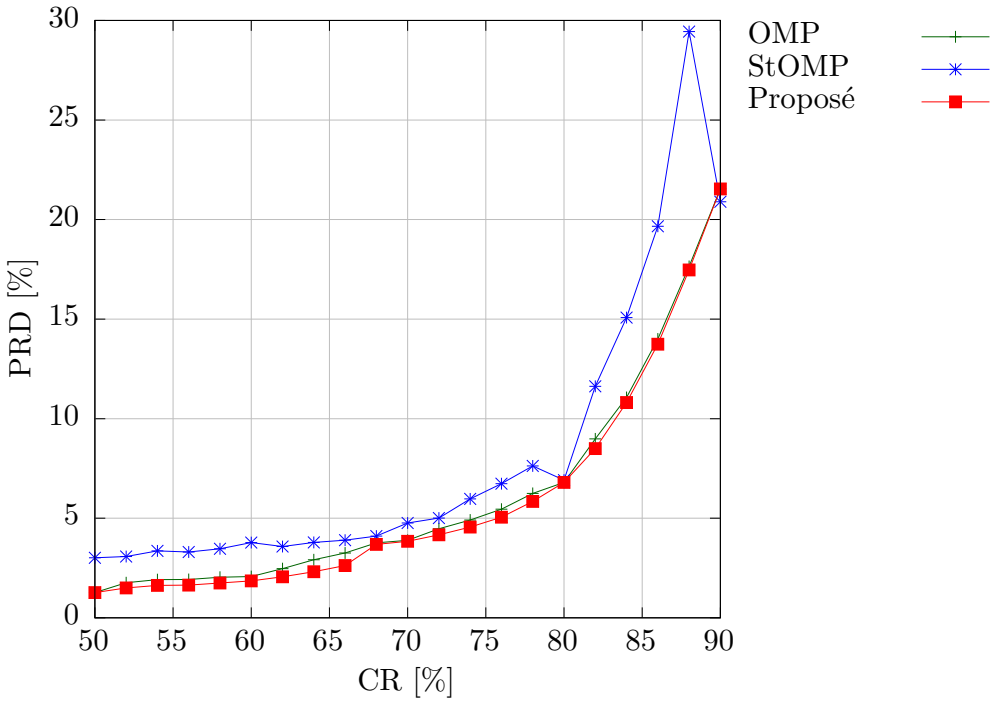


FIGURE 3.23 – *PRD en fonction du taux de compression pour les signaux ECG.*

Nous avons effectué une série de tests durant lesquels nous avons fixé la valeur de $\mathcal{N}$ à 500 et nous avons varié la valeur du taux de compression. Les FIGURE 3.22, FIGURE 3.23,

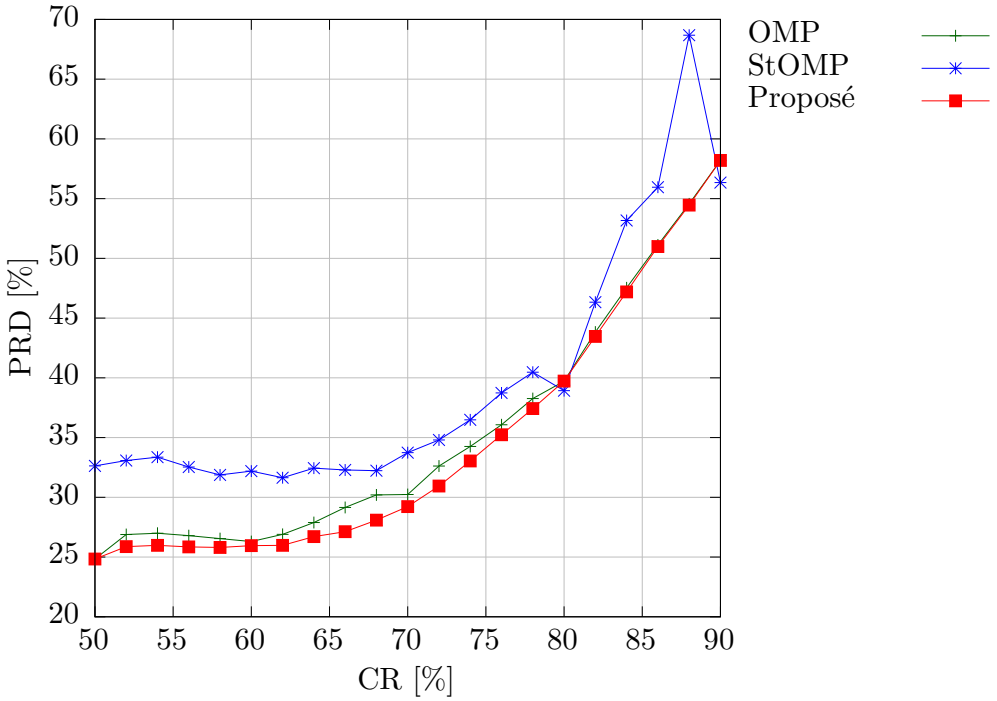


FIGURE *3.24 – PRD en fonction du taux de compression pour le signal EMG.*

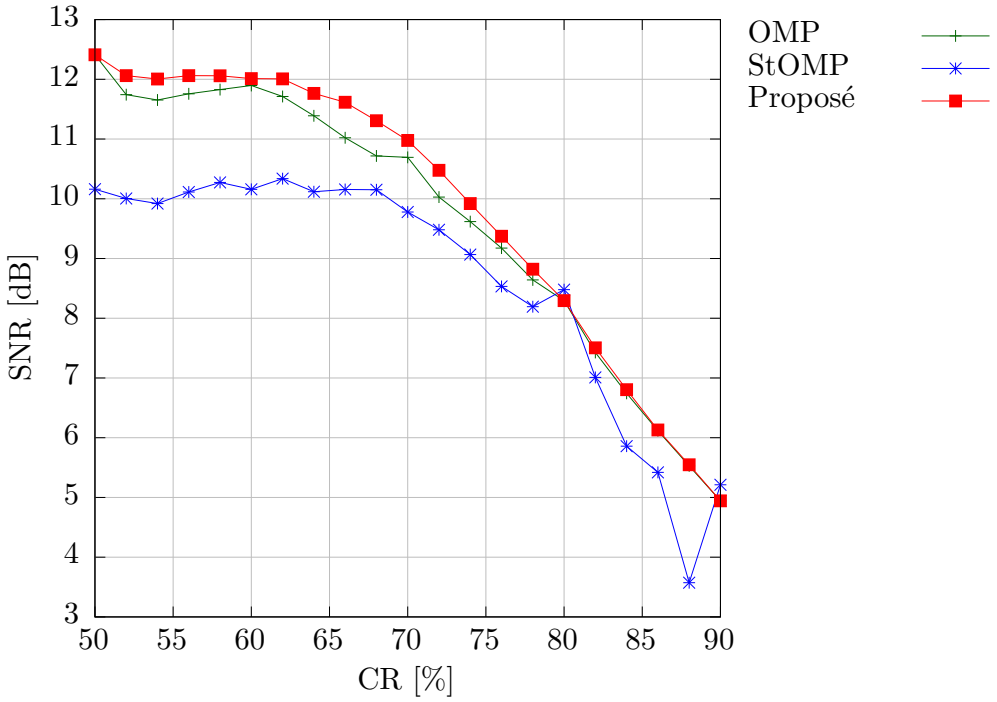


FIGURE *3.25 – SNR en fonction du taux de compression pour le signal EMG.*

FIGURE 3.24 et FIGURE 3.25 rapportent le PRD et le SNR en fonction du taux de compression. Plus le taux de compression augmente, plus la qualité du signal reconstruit se dégrade. Le signal ECG est plus compressible que l'EMG. En effet, avec une même valeur du taux de compression, nous avons obtenu un SNR plus élevé et un PRD plus faible. Cela est dû au fait que le signal ECG est plus parcimonieux dans le domaine de la DCT. Les résultats obtenus confirment ceux obtenus lors des expérimentations précédentes. Par rapport

à l'OMP et le StOMP, nous avons obtenu un SNR plus élevé lorsque les blocs compressés étaient reconstruits avec l'algorithme proposé.

Pour qu'une application puisse faire l'acquisition et la reconstruction de signaux en temps réel, les durées des temps d'encodage T_E et de reconstruction T_R devraient être assez courtes. Elles dépendent respectivement de la matrice de mesure $\mathbf{\Phi}$ et de la complexité de l'algorithme de reconstruction.

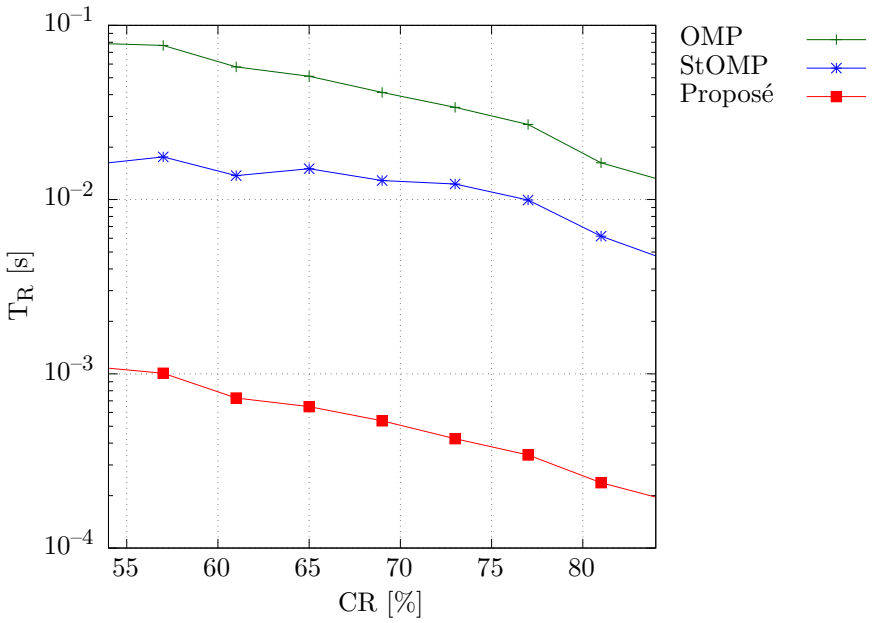


FIGURE *3.26 – Temps de reconstruction.*

La FIGURE 3.26 rapporte la durée du temps de reconstruction T_R des trois algorithmes en fonction du taux de compression. Nous pouvons observer que plus CR augmente, plus T_R diminue. En plus de la complexité de l'algorithme, elle varie aussi en fonction de la dimension $[\mathcal{M} \times \mathcal{N}]$ de la matrice $\mathbf{A}$. Puisque $\mathcal{N}$ était fixé à 500, le fait d'augmenter CR entrainait la diminution de la valeur de $\mathcal{M}$, raccourcissant ainsi la durée de T_R. Comme énoncé précédemment, la phase de sélection de l'OMP domine son temps de reconstruction. En sélectionnant plusieurs colonnes à chaque itération, le StOMP a réduit la durée de T_R d'environ 5 fois par rapport à l'OMP. Par contre, l'algorithme proposé est le plus efficace en termes de durée de reconstruction puisqu'il est 23 fois plus rapide que l'OMP. Cette efficacité est due au fait que l'algorithme proposé supprime totalement la phase de sélection de l'OMP.

La méthode proposée est efficace pour les signaux ayant des représentations parcimonieuses dans le domaine de la DCT, où les $\mathcal{K}$ coefficients non nuls sont concentrés dans les basses fréquences. Si les $\mathcal{K}$ coefficients sont répartis sur tout le domaine, alors la méthode proposée n'est pas adaptée et est inefficace. Dans ce cas, nous devons utiliser l'OMP ou bien d'autres algorithmes de reconstruction pour les localiser.

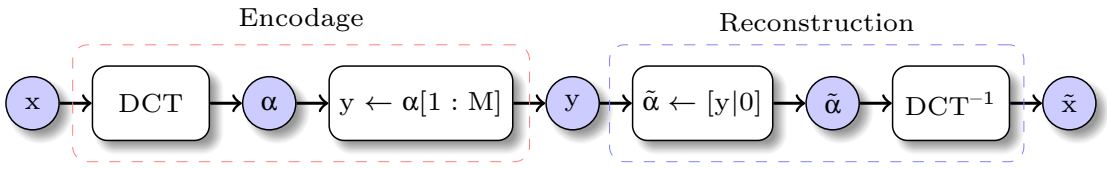
Cependant, la DCT concentre l'énergie de la plupart des signaux dans les basses fréquences, en particulier les signaux physiologiques comme l'ECG [AMH75] et l'EMG [GM97].

Les composantes hautes fréquences restantes ont tendance à avoir de faibles valeurs. Elles peuvent être ignorées sans perte visuelle dans le domaine temporel. Pour ces types de signaux, la méthode proposée est avantageuse parce qu'elle réduit à la fois la complexité de la phase de mesure et celle de la reconstruction. Premièrement, l'encodeur AC a besoin d'effectuer seulement $(\mathcal{N}-\mathcal{M})$ opérations d'addition sans utiliser d'espace mémoire puisque la matrice est déterministe. Deuxièmement, l'algorithme de reconstruction proposé élimine la phase de sélection de l'OMP qui prend le plus de temps.

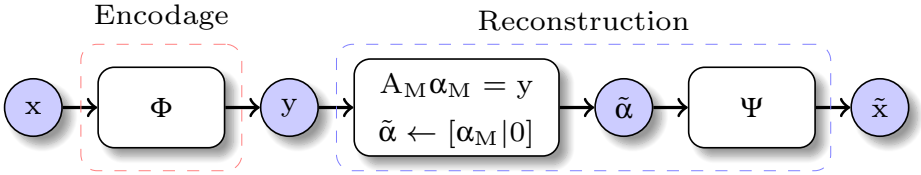
Les résultats obtenus ont montré que l'OMP est plus efficace en termes de taux de succès avec la matrice de mesure déterministe proposée $\mathbf{\Phi}_{\text{DBBD}}$. En effet, l'OMP reconstruit un signal parcimonieux, où les $\mathcal{K}$ composantes non nulles sont concentrées dans les basses fréquences, lorsque $\mathcal{K} < \mathcal{M}$. Alors qu'avec les matrices aléatoires, l'OMP reconstruit le même signal parcimonieux seulement lorsque $\mathcal{K} \leq \frac{\mathcal{M}}{1.5 \ln \mathcal{N}}$.

Si nous utilisons des matrices aléatoires, les colonnes significatives de la matrice $\mathbf{A}$ sont réparties sur tout le domaine, même si la DCT concentre l'énergie du signal dans les basses fréquences. Pour les localiser, nous devons utiliser l'OMP ou d'autres algorithmes. Cependant grâce à la matrice de mesure proposée $\mathbf{\Phi}_{\text{DBBD}}$, les colonnes significatives de $\mathbf{A}$ restent concentrées dans les basses fréquences. Puisque nous connaissons à l'avance leurs positions, nous pouvons réduire la complexité de l'OMP avec l'algorithme proposé en éliminant la phase de sélection qui prend le plus de temps. De plus, les résultats obtenus ont montré que le SNR des signaux reconstruits est légèrement supérieur avec l'algorithme proposé. C'est une version simplifiée de l'OMP qui peut être utilisé seulement avec la matrice de mesure proposée $\mathbf{\Phi}_{\text{DBBD}}$.

3.4 Comparaison entre la méthode proposée et le codage par transformation



(a) Codage par transformation.



(b) Méthode proposée.

FIGURE *3.27 – Comparaison entre la méthode proposée et le codage par transformation.*

La méthode que nous proposons, la combinaison de la matrice de mesure et de l'algo-

rithme de reconstruction, est comparable à un codage par transformation dans le domaine de la DCT où seulement les $\mathcal{M}$ coefficients basses fréquences de la transformée DCT sont conservés. Comme indiqué à la FIGURE 3.27(a), l'encodage du signal avec le codage par transformation consiste à :

1. Calculer la DCT du signal d'entrée $\mathbf{x}$. Le vecteur $\boldsymbol{\alpha}$ résultant contient $\mathcal{N}$ éléments.
2. Puisque la DCT concentre la plupart de l'information dans les basses fréquences, garder seulement les $\mathcal{M}$ premiers coefficients de $\boldsymbol{\alpha}$, où $\mathcal{M} < \mathcal{N}$. Le vecteur encodé $\mathbf{y}$ contient $\mathcal{M}$ éléments.

La reconstruction du signal s'effectue aussi en deux étapes :

1. Restituer la transformée DCT $\tilde{\boldsymbol{\alpha}}$ à partir du vecteur $\mathbf{y}$ en remplissant de zéros les coefficients manquants :
$$\tilde{\boldsymbol{\alpha}} \leftarrow [\mathbf{y}|0] \tag{3.11}$$
2. Reconstruire le signal en calculant la DCT inverse de $\tilde{\boldsymbol{\alpha}}$.

La méthode que nous proposons à base de l'AC simplifie l'encodage du signal. En effet, comme illustré à la FIGURE 3.27(b), le vecteur encodé $\mathbf{y}$ est obtenu directement en multipliant $\mathbf{x}$ par la matrice de mesure $\boldsymbol{\Phi}$ qui est une matrice binaire $\mathcal{M} \times \mathcal{N}$. Elle a déplacé la complexité vers la partie reconstruction. En effet, il faut résoudre le système d'équations $\mathbf{A}_{\mathrm{M}}\boldsymbol{\alpha}_{\mathrm{M}} = \mathbf{y}$ pour pouvoir reconstruire la transformée DCT $\tilde{\boldsymbol{\alpha}}$ du signal, où $\mathbf{A}_{\mathrm{M}}$ est une matrice $\mathcal{M} \times \mathcal{M}$ formée par les premières colonnes de $\mathbf{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$.

3.4.1 Complexité de l'encodage

Nous nous intéressons particulièrement à la complexité de la partie encodage puisque la capacité en termes de calcul des nœuds dans un réseau de capteurs est très limitée. Pour ce genre d'application, la phase de reconstruction est moins contraignante puisqu'elle est généralement faite sur une plateforme ayant les ressources nécessaires en termes de calcul et d'énergie.

Nous avons évalué la complexité en énumérant les opérations arithmétiques mises en œuvre pour pouvoir encoder le signal. Le TABLEAU 3.3 donne la complexité de l'encodage avec les deux méthodes.

TABLEAU *3.3 – Complexité de l'encodage.*

	Multiplication	**Addition**
Codage par transformation	$\mathcal{M} \times \mathcal{N}$	$\mathcal{M} \times (\mathcal{N} - 1)$
Méthode proposée	0	$\mathcal{M} \times (\mathrm{m} - 1) = \mathcal{N} - \mathcal{M}$

Ce tableau montre la simplicité de l'encodage avec la méthode proposée puisqu'elle nécessite seulement $(\mathcal{N} - \mathcal{M})$ additions au lieu de $(\mathcal{M} \times (\mathcal{N} - 1))$. En plus de cela, l'encodage

avec la méthode proposée n'a pas besoin de multiplication.

3.4.2 Qualité de reconstruction

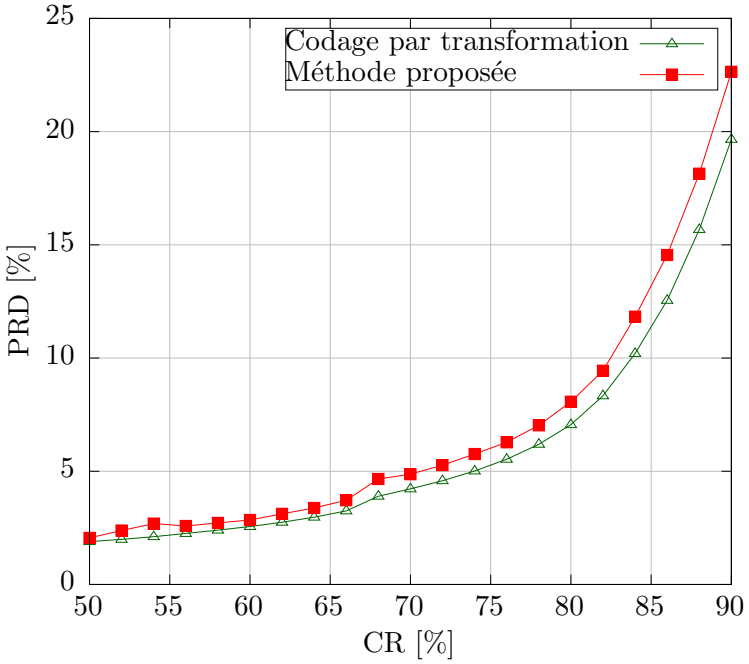


FIGURE *3.28 – PRD en fonction du taux de compression pour le signal ECG.*

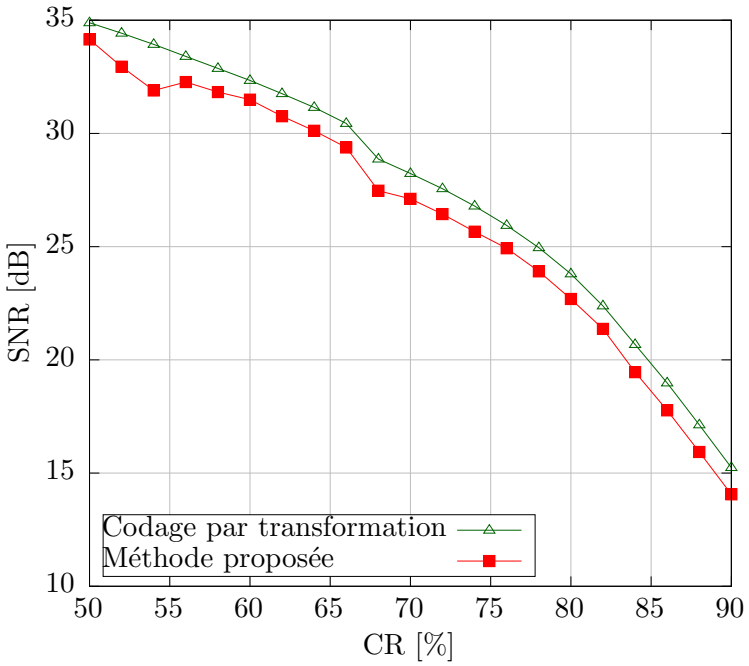


FIGURE *3.29 – SNR en fonction du taux de compression pour le signal ECG.*

Nous avons aussi évalué la qualité du signal reconstruit avec les deux méthodes. Pour cela, nous avons pris des signaux ECG et EMG provenant de la base de données Physionet. Nous avons effectué une série de tests sous Matlab durant lesquels nous avons fixé la valeur

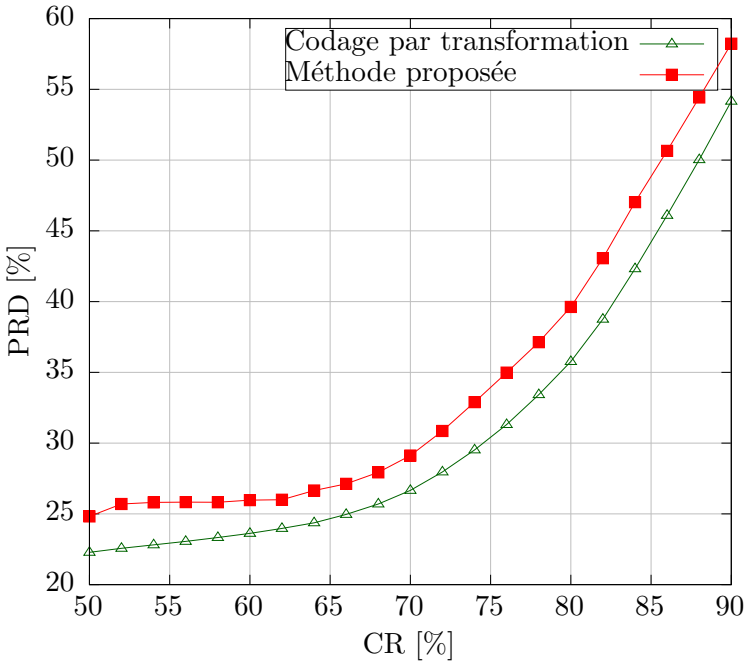


FIGURE *3.30 – PRD en fonction du taux de compression pour le signal EMG.*

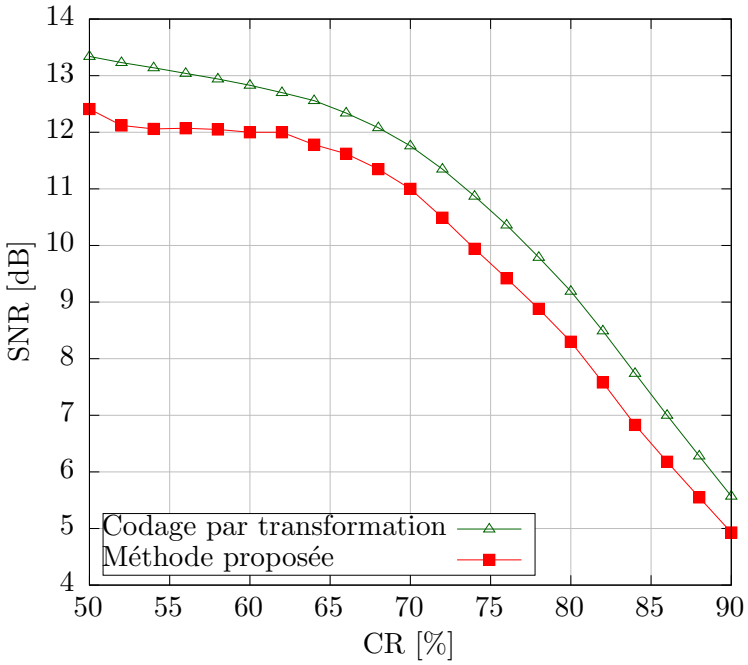
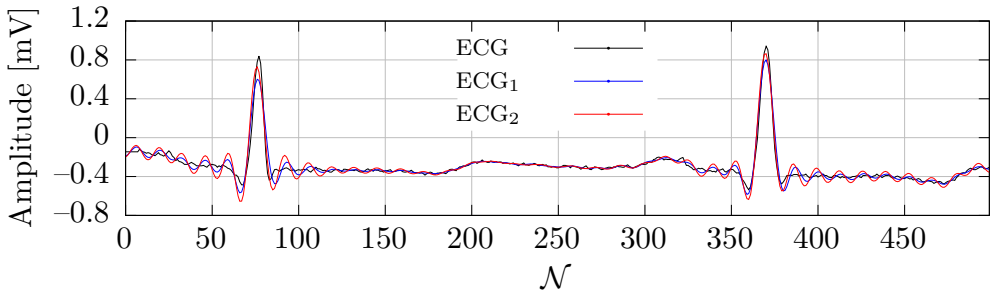


FIGURE *3.31 – SNR en fonction du taux de compression pour le signal EMG.*

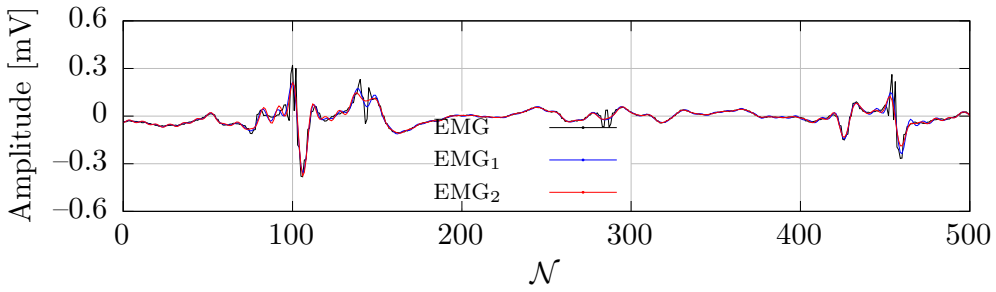
de $\mathcal{N}$ à 500 et nous avons varié la valeur du taux de compression. À la fin de chaque test, nous avons évalué les PRD et les SNR des signaux reconstruits.

Les FIGURE 3.28, FIGURE 3.29, FIGURE 3.30 et FIGURE 3.31 rapportent les PRD et les SNR des signaux ECG et EMG reconstruits avec les deux méthodes. Ces résultats montrent que le codage par transformation est plus efficace que la méthode proposée en termes de qualité de reconstruction. Avec le même taux de compression, les PRD des signaux reconstruits sont plus faibles et leurs SNR sont plus élevés.

Cette différence est due au fait que les coefficients de la transformée DCT $\tilde{\alpha}$ reconstruite avec la méthode proposée sont différents de ceux obtenus avec le codage par transformation. Cependant cette erreur est acceptable puisque les différences maximales en termes de PRD et de SNR sont respectivement 4.7 % et 2.2 dB pour les signaux ECG et EMG.



(a) Signal ECG original (en noir) et ceux reconstruits avec le codage par transformation (en bleu) et la méthode proposée (en rouge), pour un taux de compression de 85 % (enregistrement numéro 100).



(b) Signal EMG original (en noir) et ceux reconstruits avec le codage par transformation (en bleu) et la méthode proposée (en rouge), pour un taux de compression de 75 % (emg_helthy).

FIGURE *3.32 – Comparaison entre la méthode proposée et le codage par transformation.*

Les FIGURE 3.32(a) et FIGURE 3.32(b) illustrent une partie des signaux originaux et reconstruits. Pour le signal ECG, le taux de compression est configuré à 85 %, pour $\mathcal{M} = 75$ et $\mathcal{N} = 500$. La différence entre le signal ECG reconstruit avec le codage par transformation (en bleu) et celui reconstruit avec la méthode proposée (en rouge) n'est pas remarquable. Pour le signal EMG, le taux de compression est configuré à 75 %, pour $\mathcal{M} = 125$ et $\mathcal{N} = 500$. De même, la différence entre le signal EMG reconstruit avec le codage par transformation et celui reconstruit avec la méthode proposée est difficilement perceptible.

3.5 Conclusion

La matrice de mesure déterministe proposée est adaptée pour compresser des signaux ayant des représentations parcimonieuses dans le domaine de la DCT puisqu'elle a une faible cohérence avec la matrice IDCT. Par rapport aux autres matrices de mesure, elle est plus facile à implémenter côté matériel parce que, premièrement, elle nécessite seulement $(\mathcal{N}–\mathcal{M})$ opérations d'addition pour encoder un bloc de $\mathcal{N}$ échantillons. Deuxièmement, elle n'a pas

besoin de générateur de nombres aléatoires, ou bien d'espace mémoire pour sauvegarder ses éléments, puisqu'elle est déterministe. Les expérimentations faites au niveau algorithmique sur des signaux ECG et EMG ont montré que la matrice de mesure proposée surpasse les matrices aléatoires en termes de qualité de reconstruction.

La matrice de mesure déterministe proposée a simplifié la reconstruction du signal en éliminant la phase de sélection de l'OMP. Ce qui a permis de réduire en même temps sa complexité et son temps d'exécution. L'algorithme de reconstruction proposé effectue un seuillage en sélectionnant seulement les $\mathcal{M}$ composantes basses fréquences de la transformée DCT. Une étude comparative entre l'OMP et l'algorithme de reconstruction proposé a montré que ce dernier a une meilleure performance en termes de qualité de reconstruction.

La méthode proposée, la combinaison de la matrice de mesure et de l'algorithme, est similaire à une technique de seuillage classique dans le domaine de la DCT. Une étude comparative en termes de qualité de reconstruction et de complexité d'encodage entre les deux méthodes a été faite. Même si la méthode classique est plus efficace en termes de qualité de reconstruction, l'encodage avec celle proposée est moins complexe et est plus adapté pour le WBAN.

Le chapitre suivant va présenter une évaluation au niveau système cette méthode proposée dans le cadre d'un WBAN. Un exemple d'architecture de l'encodeur implémentant la matrice de mesure proposée sera étudié et évalué.

Chapitre 4

Évaluation système - Réseau de capteurs à base de l'acquisition comprimée

Sommaire

4.1 Introduction 80

4.2 Modélisation du réseau de capteurs utilisant l'acquisition comprimée 80

4.2.1 Modèle du nœud 81

4.2.2 Modèle SPICE de l'encodeur 90

4.2.3 Routeur 91

4.2.4 Décodeur 92

4.3 Résultats de simulation 93

4.3.1 Évaluation de la qualité de reconstruction 93

4.3.2 Validation du modèle SPICE de l'encodeur 97

4.3.3 Influence de la valeur de $\mathcal{M}$ sur la qualité et le temps de reconstruction 98

4.3.4 Impact de la résolution du convertisseur analogique numérique sur la qualité de reconstruction 100

4.3.5 Évaluation de la bande passante et de la consommation 101

4.4 Conclusion 104

4.1 Introduction

Les phases d'acquisition et de reconstruction de signaux avec l'AC sont fortement liées. Un modèle exécutable au niveau système d'une chaîne d'acquisition complète est nécessaire. Ce modèle exécutable va inclure à la fois les parties logicielles et matérielles de la chaîne et va permettre de vérifier ses fonctionnalités en amont de la phase de développement. Par rapport aux autres langages de description de matériel ou « *HDLs* » , par exemple le « *VHSIC HDL (VHDL)* » ou le Verilog, le SystemC est plus adapté pour décrire et simuler des modèles au niveau système puisqu'il permet de modéliser à la fois parties logicielles et matérielles [BDM+05].

Le SystemC [SC] est à la fois une bibliothèque de classes C++ et un HDL destiné à la modélisation et à la simulation de systèmes numériques. Il ajoute des fonctionnalités au-dessus du C++ standard en introduisant la notion du temps (séquencement d'événements) et de la concurrence, et en ajoutant de nouveaux types de données permettant de décrire une architecture matérielle. Le SystemC permet de décrire des modèles d'algorithmes logiciels et d'architectures matérielles sur plusieurs niveaux d'abstraction, depuis le niveau système jusqu'au transfert entre registres ou « *register transfer level (RTL)* ». De plus, le modèle transactionnel ou « *transaction level modeling (TLM)* » [TLM] lui permet de dissimuler, à un niveau d'abstraction élevé, les détails de communication entre les différents modules qui composent le système.

Le noyau natif du SystemC ne permet pas de décrire et de simuler le modèle d'un système analogique à temps continu. C'est pourquoi l'extension SystemC-AMS [SCA] a été introduite pour permettre de modéliser des systèmes comportant à la fois des parties analogiques, numériques et mixtes. L'architecture du langage SystemC-AMS 1.0 comprend trois modèles de calcul normalisés, notamment, le flot de donnés temporalisé ou « *timed dataflow (TDF)* », le flot de signal linéaire ou « *linear signal flow (LSF)* » et le réseau électrique linéaire ou « *electrical linear network (ELN)* » [Bar10, BEG+11].

Ce chapitre est structuré en deux parties. La première partie va décrire au niveau système le modèle d'un WBAN utilisant l'AC. La matrice de mesure et l'algorithme de reconstruction que nous avons proposés dans le chapitre précédent seront utilisés pour compresser et reconstruire les signaux mesurés. Un exemple d'encodeur implémentant la matrice de mesure sera proposé. La seconde et dernière partie présentera les résultats de simulation.

4.2 Modélisation du réseau de capteurs utilisant l'acquisition comprimée

La FIGURE 4.1 illustre l'architecture du réseau. Il peut comporter plusieurs nœuds et décodeurs. Le routeur permet de router les données provenant des nœuds vers un décodeur spécifique. Des travaux ont été déjà faits sur la modélisation au niveau système et la simula-

tion des réseaux de capteurs sans fils ou WSN avec le langage SystemC, par exemple IDEA1 [DMNC11]. Cependant pour valider le concept, nous avons utilisé le TLM version 2.0 pour modéliser la communication entre les nœuds et les décodeurs. Les sections suivantes vont décrire les modèles des différents éléments qui composent le réseau.

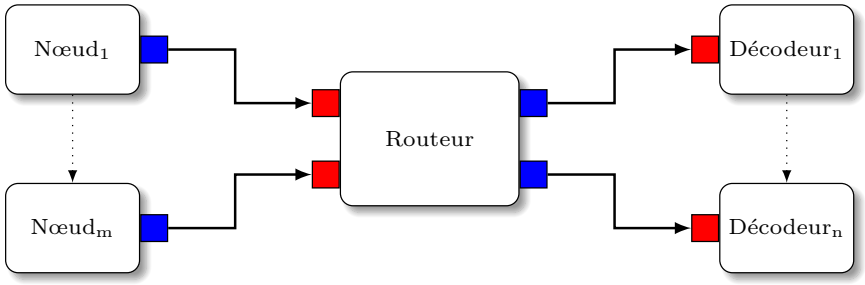


FIGURE *4.1 – Architecture du réseau. En bleu simple_initiator_socket, en rouge simple_target_socket.*

4.2.1 Modèle du nœud

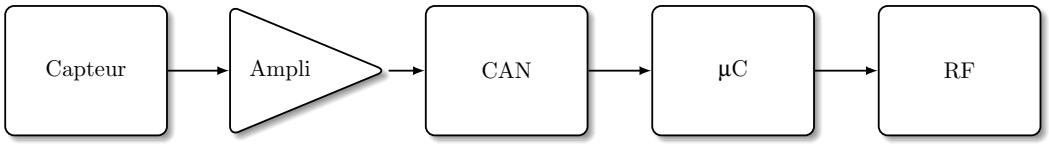


FIGURE *4.2 – Modèle d'un nœud sans encodeur AC.*

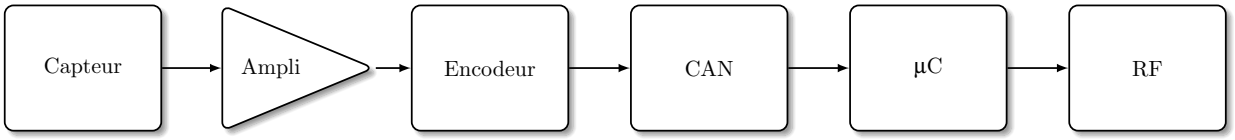


FIGURE *4.3 – Modèle d'un nœud avec encodeur AC.*

Les FIGURE 4.2 et FIGURE 4.3 illustrent respectivement les modèles d'un nœud avec et sans encodeur AC. Un nœud comporte un capteur, un amplificateur à faible bruit, éventuellement un encodeur AC, un CAN, un microcontrôleur et un transmetteur RF. Le modèle du nœud est décrit de façon structurelle. C'est un module SystemC qui instancie les différents sous-modules qui le composent et les relie avec des signaux internes. Les sous-sections suivantes vont présenter plus en détail les différents sous-modules qui forment le nœud.

i) Capteur

Le modèle du capteur est illustré à la FIGURE 4.4. Le capteur utilise des enregistrements sauvegardés dans des fichiers locaux pour générer les signaux physiologiques. En plus de son nom, le constructeur du module prend trois paramètres supplémentaires, le nom du fichier contenant les enregistrements, la fréquence à laquelle les données enregistrées étaient

capturées et l'unité en V. Le format du fichier est simple, chaque ligne doit contenir un échantillon.

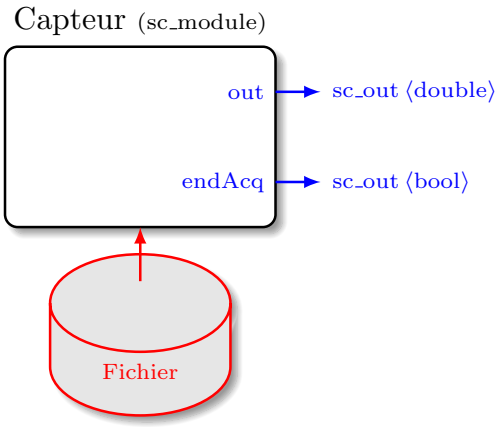


FIGURE *4.4 – Modèle du capteur.*

Le constructeur ouvre un flux de données à partir du fichier passé en paramètre. Puis, il démarre une horloge interne cadencée à la fréquence d'acquisition des données enregistrées. Le module a un *processus* de type SC_METHOD qui est sensible au front montant de l'horloge interne. Ce *processus* lit une donnée à partir du flux et l'écrit sur le port de sortie *out*. Lorsque la fin du fichier est atteinte, le *processus* arrête l'horloge interne et met la sortie *endAcq* à l'état haut pour informer le microcontrôleur que l'enregistrement est terminé. Le destructeur du module ferme le flux de données ouvert par le constructeur.

ii) Amplificateur

Le modèle de l'amplificateur est illustré à la FIGURE 4.5. C'est un module TDF comportant un port d'entrée de type sca_de :: sca_in qui va lui permettre de lire des données provenant d'un module SystemC à évènements discrets, plus précisément du capteur.

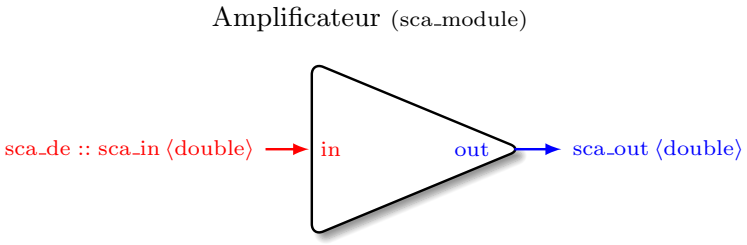


FIGURE *4.5 – Modèle de l'amplificateur.*

Le constructeur du module prend en paramètres le gain et l'offset. Le module redéfinit la méthode *processing()* de sa classe de base sca_module. Dans cette méthode, le module lit la valeur de son port d'entrée *in*. Puis, il multiplie la valeur lue par le gain et ajoute l'offset. Il écrit le résultat sur son port de sortie *out*.

iii) Encodeur

Modèle haut niveau

D'après la description de la matrice de mesure Φ_{BDDB} que nous proposons, l'encodeur doit accumuler « m = $\frac{\mathcal{N}}{\mathcal{M}}$ » échantillons successifs. Pour simplifier la mise en œuvre et valider le concept, un modèle haut niveau de l'encodeur est illustré à la FIGURE 4.6.

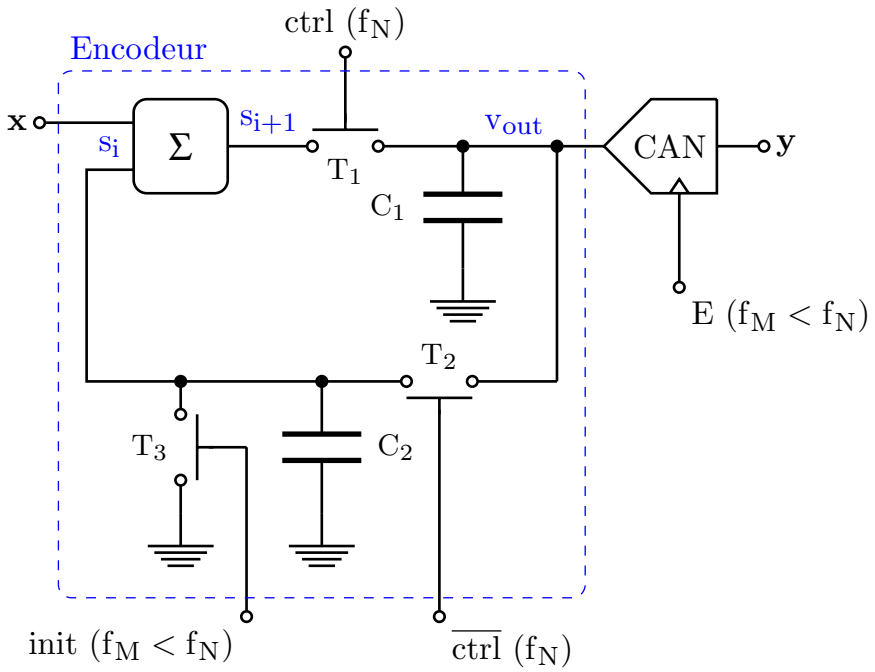


FIGURE *4.6 – Modèle haut niveau de l'encodeur.*

L'encodeur comporte deux échantillonneurs bloqueurs (EBs). Le premier, noté par EB_1, est formé par l'interrupteur T_1 commandé par le signal *ctrl* et le condensateur C_1. Le second, noté par EB_2, est formé par le deuxième interrupteur T_2 commandé par le signal $\overline{ctrl}$ et le condensateur C_2. Les signaux de commandes *ctrl* et $\overline{ctrl}$ sont cadencés à la fréquence de Nyquist f_N. Alors que le signal *init* et celui qui commande la conversion du CAN *E* sont cadencés à une fréquence f_M en dessous de celle de Nyquist ($f_M < f_N$). L'encodeur accumule le signal à la fréquence de Nyquist mais diminue celle de la conversion numérique en dessous de la fréquence de Nyquist. C'est avantageux puisque l'accumulation est plus simple que la numérisation.

Lorsque le signal *ctrl* est au niveau haut, EB_1 est dans la phase d'échantillonnage. La tension s_{i+1} est restituée à la sortie v_{out} de l'encodeur. v_{out} varie instantanément avec la tension d'entrée x puisque s_i est constante. En effet pendant cette période, EB_2 est dans la phase de maintien.

Lorsque le signal *ctrl* est au niveau bas, EB_1 est dans la phase de maintien, la tension échantillonnée est sauvegardée dans le condensateur C_1 et la tension de sortie v_{out} est maintenue à ce niveau. Pendent ce temps, EB_2 est dans la phase d'échantillonnage. La tension de sortie v_{out} est sauvegardée dans le condensateur C_2. La tension s_i est mise à

jour à ce niveau ($s_i = v_{out}$).

Lorsque le signal *init* est au niveau haut, l'interrupteur T_3 est fermé. La tension s_i est mise à zéro.

La FIGURE 4.7 illustre les chronogrammes des signaux de l'encodeur. Les échantillons sont accumulés par paquets de $m = \frac{\mathcal{N}}{\mathcal{M}}$. Pour démarrer l'encodage, le signal *init* est mis à l'état haut pour initialiser la tension s_i à zéro. À chaque phase d'échantillonnage de EB_1, les échantillons x_i de la tension d'entrée sont accumulés sur la sortie v_{out} de l'encodeur. Au bout de $m = \frac{\mathcal{N}}{\mathcal{M}}$ échantillons consécutifs, la tension de sortie v_{out} est égale à $(x_1 + \ldots + x_m)$. Le signal E est activé pour démarrer la conversion numérique de v_{out}, résultant le premier élément y_1 du vecteur de mesure $\mathbf{y}$. Le signal *init* est de nouveau activé pour démarrer l'encodage des prochains échantillons, et ainsi de suite.

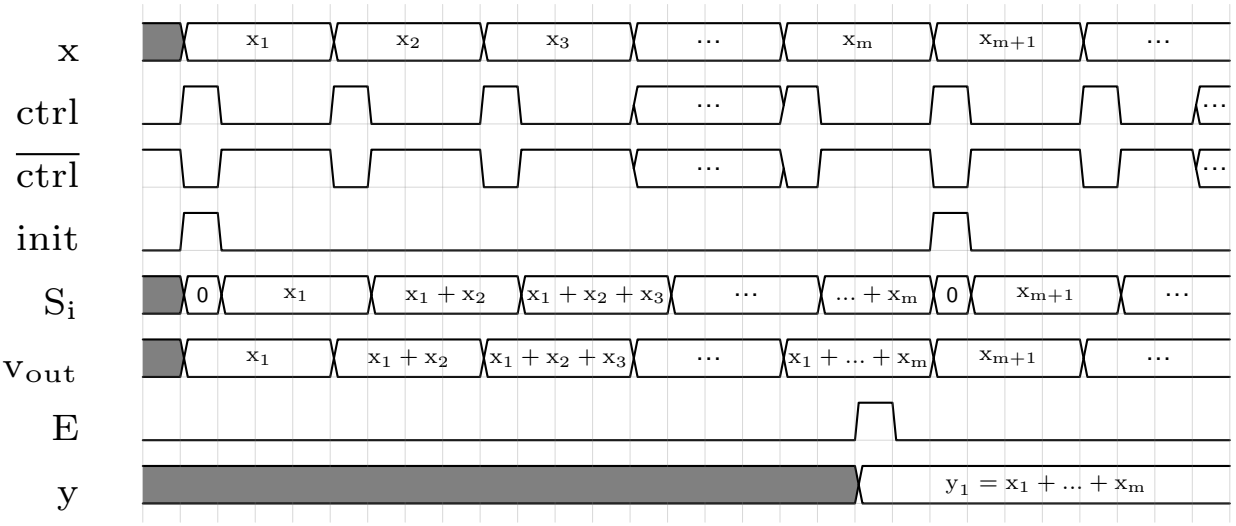


FIGURE *4.7 – Chronogrammes des signaux de l'encodeur.*

La précision de l'encodeur dépend de la durée T_S de la phase d'échantillonnage de EB_1 et du courant de fuite des condensateurs.

Puisque la tension de sortie v_{out} varie instantanément avec l'entrée pendant la phase d'échantillonnage de EB_1, T_S doit être la plus courte possible. Cependant, elle doit être assez longue pour pouvoir charger correctement le condensateur C_1. Le temps de charge de C_1 dépend de sa valeur et celle de la résistance ON de T_1. Pour diminuer le temps de charge de C_1 et minimiser ainsi T_S, il serait préférable de choisir un condensateur de faible valeur.

Le courant de fuite décharge le condensateur et corrompe la valeur de la tension mémorisée. Plus la valeur du condensateur est grande, plus le courant de fuite est faible. Il faut trouver un compromis sur le choix de la valeur de C_1 et la durée T_S.

Modèle au niveau circuit

La FIGURE 4.8 illustre le raffinement du modèle de l'encodeur au niveau circuit.

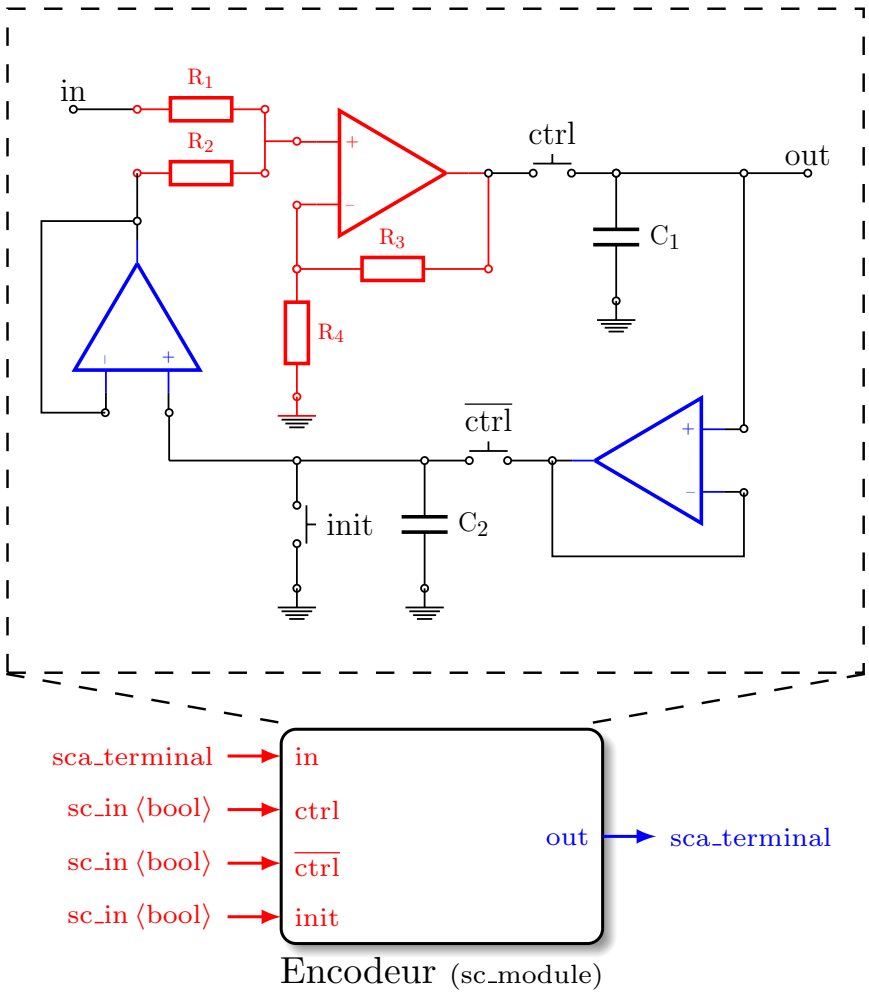


FIGURE 4.8 – *Exemple de raffinement du modèle de l'encodeur.*

Le modèle de l'encodeur est décrit en instanciant des modules ELN prédéfinis et en les interconnectant par des nœuds pour former un réseau électrique. Les résistances R_i sont modélisées par des modules ELN sca_eln :: sca_r. Les condensateurs C_i sont modélisés par des modules ELN sca_eln :: sca_c. Les interrupteurs sont modélisés par des modules ELN sca_enl :: sca_de_rswitch, des interrupteurs commandés par des signaux à évènements discrets. Les amplificateurs opérationnels (AOP) sont modélisés par des modules ELN sca_eln :: sca_nullor, des AOP idéaux.

Les AOPs colorés en bleu fonctionnent en suiveur tandis que celui coloré en rouge fonctionne en sommateur non inverseur. Les résistances R_i ont la même valeur pour avoir un gain de « 1 » à la sortie.

Pour tester le fonctionnement de l'encodeur, nous avons branché une source de tension constante de 0.3 V de type *sca_vsource* sur son entrée *in*. Nous avons généré les signaux de commandes *ctrl* et $\overline{ctrl}$ à une fréquence de 4 kHz. Nous avons configuré le signal *init* pour que la valeur de $m = \frac{\mathcal{N}}{\mathcal{M}}$ soit égale à 4. Nous avons fixé la valeur des résistances R_i à 10 kΩ et celle des condensateurs C_i à 0.1 µF. Nous avons configuré les résistances ON et OFF des interrupteurs à 10 Ω et 10 MΩ respectivement. Nous avons configuré la durée

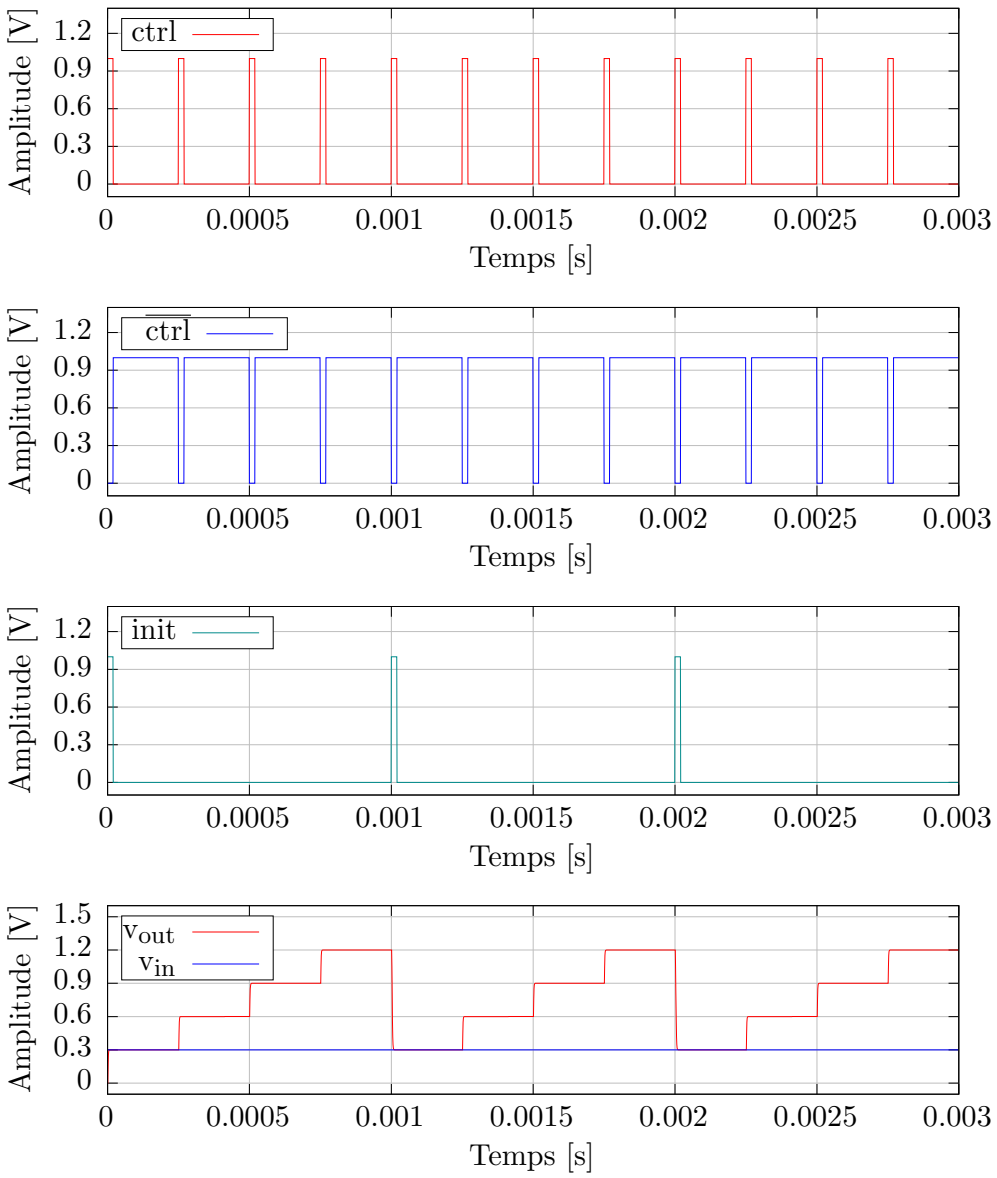


FIGURE *4.9 – Chronogramme des signaux de l'encodeur pour une tension d'entrée constante.*

T_S de la phase d'échantillonnage de EB_1 à 20 µs. La FIGURE 4.9 illustre le chronogramme des signaux de commandes ainsi que les tensions d'entrée et de sortie de l'encodeur. Elle montre que l'encodeur accumule la tension d'entrée *in* sur sa sortie *out*.

iv) Convertisseur analogique numérique

La FIGURE 4.10 illustre le modèle du CAN. Les entrées *vn* et *vp* correspondent aux tensions de références positive et négative. La résolution du CAN est configurable. Elle peut prendre une des valeurs suivantes : 8-bit, 10-bit, 12-bit ou 16-bit. Le module sauvegarde les résultats de la conversion dans une mémoire FIFO (« *first in first out* ») sous un format sc_lv ⟨16⟩.

Le module implémente un *processus* de type SC_METHOD sensible au front montant du signal *convert* et qui effectue la conversion numérique de la tension d'entrée *in*. Le

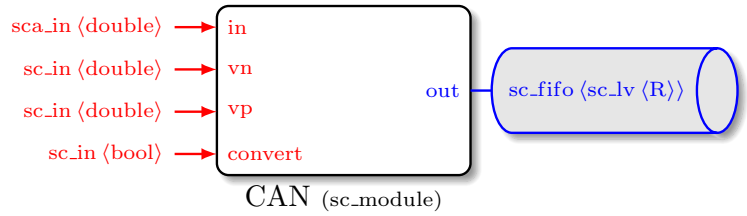


FIGURE *4.10 – Modèle du convertisseur analogique numérique.*

processus effectue une approximation successive de la tension d'entrée.

v) Microcontrôleur

Le modèle du microcontrôleur est illustré à la FIGURE 4.11. Il prend en entrées la mémoire FIFO du CAN et le signal *endAcq* provenant du module capteur marquant la fin de l'enregistrement.

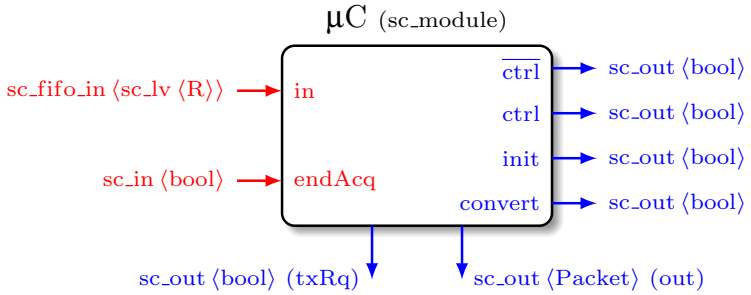


FIGURE *4.11 – Modèle du microcontrôleur.*

C'est le module qui gère le fonctionnement du nœud. Il assure les trois fonctions principales suivantes :

Génération les signaux de commandes

Le microcontrôleur génère les signaux de commandes *init*, *ctrl* et $\overline{ctrl}$ de l'encodeur, ainsi que le signal *convert* permettant de démarrer la conversion du CAN.

Pour cela, le constructeur du module prend en paramètres la fréquence du signal *ctrl*, la valeur de $m = \frac{\mathcal{N}}{\mathcal{M}}$ et la durée T_S. Rappelons que la fréquence du signal *ctrl* est égale à la celle de Nyquist. Le module génère une horloge interne cadencée à la fréquence de *ctrl*. Il génère ces quatre signaux dans un *processus* de type SC_THREAD sensible au front montant de cette horloge interne.

Lorsque le nœud n'a pas d'encodeur analogique, le microcontrôleur ne génère pas les signaux de commandes *init*, *ctrl* et $\overline{ctrl}$ de l'encodeur. Le *processus* responsable de la gestion des signaux de commandes génère seulement le signal *convert* pour démarrer la conversion numérique des échantillons à la fréquence de Nyquist.

Empaquetage des échantillons numérisés

Le microcontrôleur met en paquets les échantillons numérisés avant de les transmettre vers le module transmetteur RF.

Nous avons créé un nouveau type de données pour représenter un paquet RF que nous avons appelé *Packet*. C'est une structure ayant les quatre champs suivants :

a) *num* : entier représentant le numéro du paquet. Il doit être incrémenté automatiquement par le nœud pour permettre au décodeur de détecter une éventuelle perte de paquet.

b) *id* : entier unique représentant le nœud qui a transmis le paquet. Il va permettre au décodeur d'identifier un nœud particulier lors du décodage des données.

c) *type* : énumération représentant le type de paquet. Dans la version actuelle, il peut prendre l'une des deux valeurs suivantes : DATA, STOP.

d) *payload* : un tableau de 32 éléments de type sc_lv $\langle 8 \rangle$ représentant les données utiles.

Pour gérer l'empaquetage des données, le module a un *processus* de type SC_METHOD qui est sensible à la fois à l'évènement *data_written* de la FIFO et au signal d'entrée *endAcq*.

Lorsqu'un échantillon numérisé est écrit dans la FIFO par le CAN, ce *processus* l'ajoute progressivement dans le *payload* d'un paquet de type DATA. Lorsque le *payload* est plein, le *processus* écrit le paquet sur le port de sortie *out* du module. Puis il notifie le transmetteur RF qu'un paquet est prêt à être transmis vers le décodeur en basculant le niveau du port de sortie *txRq*.

Lorsque le module reçoit un signal de fin d'enregistrement provenant du capteur, le *processus* informe le décodeur en envoyant un paquet de type STOP.

Encodage des échantillons numérisés

Dans le cas d'un encodeur numérique, c'est le microcontrôleur qui encode les échantillons numérisés provenant du CAN. Au lieu de les sauvegarder directement dans le *payload* du paquet, le microcontrôleur encode les $m = \frac{\mathcal{N}}{\mathcal{M}}$ échantillons successifs en les accumulant dans une variable locale initialement mise à zéro. Après avoir encodé les m échantillons, le microcontrôleur sauvegarde le résultat de l'encodage dans le *payload*. Puis, il remet la variable locale à zéro avant d'entamer l'encodage des prochains échantillons, et ainsi de suite.

Par rapport à l'encodeur analogique, le numérique est beaucoup plus facile à mettre en œuvre. Par contre, le signal est toujours numérisé à la fréquence de Nyquist.

vi) Transmetteur radiofréquence

La FIGURE 4.12 illustre le modèle du transmetteur RF. Il prend en entrées le paquet et le signal *txRq* provenant du module microcontrôleur. Il a un socket de type *simple_initiator_socket* pour pouvoir démarrer une nouvelle transaction.

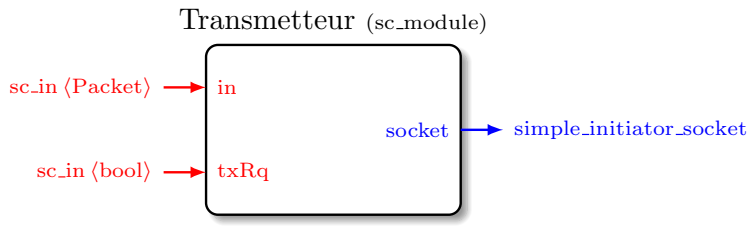


FIGURE *4.12 – Modèle du transmetteur radiofréquence.*

Pour transmettre les paquets provenant du microcontrôleur vers le décodeur, le module a un *precessus* de type SC_THREAD qui est sensible au signal d'entrée *txRq*. Lorsqu'un paquet est disponible, le *precessus* le lit et le sauvegarde dans une variable locale. Puis, il crée une transaction de type *tlm_generic_payload*. Ensuite, il initialise la transaction en mettant à jour les attributs suivants :

a) *cmd* : TLM_WRITE_COMMAND.

b) *data_ptr* : pointeur vers la variable locale stockant le paquet.

c) *data_length* : la taille d'un paquet.

d) *response_status* : TLM_INCOMPLETE_RESPONSE.

Après, il transmet la transaction au socket en appelant sa méthode *b_transport()*. Et enfin, il vérifie la réponse et envoie un message en cas d'erreur.

vii) Configuration des nœuds

Nous avons créé une structure permettant de regrouper les paramètres de configuration d'un nœud que nous avons appelé *NodeCfg*. Le constructeur du module prend en paramètre cette structure qui sera utilisée pour initialiser les différents sous-modules qui le composent. Cette structure a les champs suivants :

a) *id* : entier représentant l'identité unique du nœud.

b) *db_file* : le fichier contenant les données que le capteur va utiliser.

c) *acq_f* : la fréquence à laquelle les données du fichier étaient capturées. Nous avons fixé la valeur de $\mathcal{N}$ à cette fréquence.

d) *unit* : l'unité des données en [V].

e) *gain* : le gain de l'amplificateur.

f) *offset* : l'offset de l'amplificateur en [V].

g) *adc_res* : résolution du CAN : 10-bit, 12-bit ou bien 16-bit.

h) *v_ref_p* : tension de référence positive du CAN en [V].

i) *v_ref_n* : tension de référence négative du CAN en [V].

j) *type_enc* : une énumération permettant de connaitre le type d'encodeur utilisé. Elle peut prendre une des valeurs suivantes : ANALOG (encodeur analogique), DIGITAL (encodeur numérique) ou bien NONE (sans encodeur).

k) m : le rapport $\frac{\mathcal{N}}{\mathcal{M}}$ de l'encodeur.

l) t_s : durée de la phase d'échantillonnage de EB_1 en [s].

m) C_1 : la valeur du condensateur C_1 en [F] (encodeur analogique seulement).

n) C_2 : la valeur du condensateur C_2 en [F] (encodeur analogique seulement).

o) R_{ON} : la valeur de la résistance ON des interrupteurs en [Ω] (encodeur analogique seulement).

p) R_{OFF} : la valeur de la résistance OFF des interrupteurs en [Ω] (encodeur analogique seulement).

4.2.2 Modèle SPICE de l'encodeur

i) Description du modèle

Nous avons créé un modèle SPICE de l'encodeur analogique avec l'outil LTSpice [LTS]. C'est un simulateur SPICE basé sur SPICE III qui possède un module schématique pour éditer des schémas électroniques ainsi qu'un module permettant de visualiser les résultats de simulation. LTSpice contient des modèles d'amplificateurs opérationnels, de transistors, de portes logiques etc. LTSpice ne permet pas de modéliser un nœud complet. Nous l'avons utilisé seulement pour modéliser l'encodeur analogique.

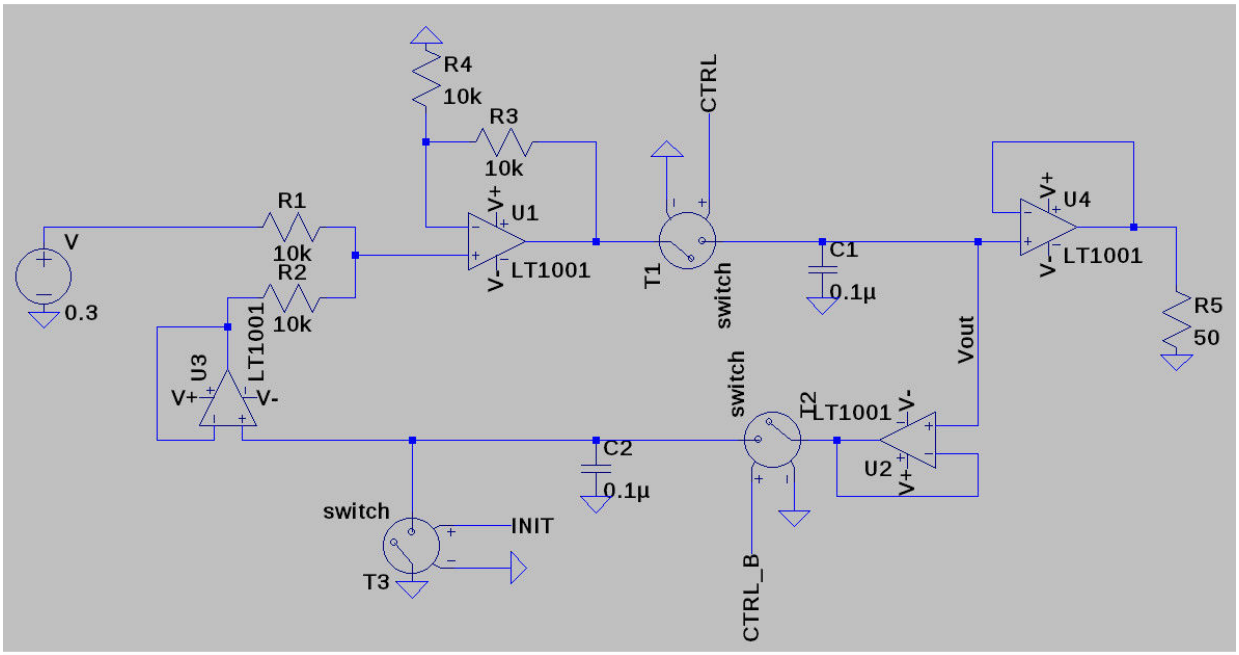


FIGURE *4.13 – Schéma du modèle SPICE de l'encodeur analogique.*

Le schéma du modèle SPICE de l'encodeur analogique est illustré à la FIGURE 4.13. Ce schéma est identique à celui du SystemC-AMS (voir FIGURE 4.8). Mais au lieu d'utiliser des composants ELN de la bibliothèque SystemC-AMS, nous avons utilisé ceux de LTSpice. Nous avons utilisé des amplificateurs opérationnels de type LT1001. Nous avons configuré

les composants du modèle SPICE avec les mêmes paramètres que ceux du modèle SystemC-AMS :

a) Valeur des résistances R1 à R4 : 10 kΩ.

b) Valeur des condensateurs C1 et C2 : 0.1 µF.

c) Valeurs des résistances ON et OFF des interrupteurs T1 et T2 : 10 Ω et 1 MΩ.

Pour valider le fonctionnement de l'encodeur, nous avons appliqué une tension constante de 0.3 V sur son entrée. Nous avons configuré la valeur de m = $\frac{\mathcal{N}}{\mathcal{M}}$ à 4 et la fréquence des signaux ctrl et $\overline{\text{ctrl}}$ à 4 kHz, ce qui correspond à une période de 0.25 ms. Nous avons fixé la durée de la phase d'échantillonnage de EB1, formé par T1 et C1, à 20 µs.

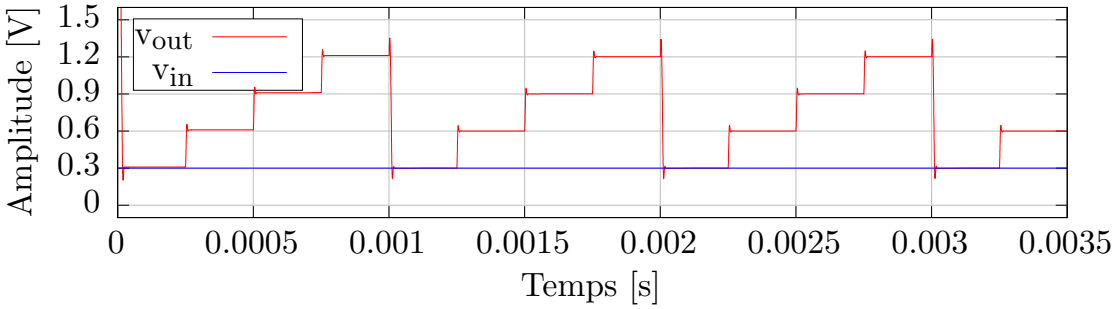


FIGURE *4.14 – Tensions d'entrée (bleu) et de sortie (rouge) de l'encodeur pour une tension d'entrée constante de 0.3* V.

La FIGURE 4.14 illustre le chronogramme des tensions d'entrée et de sortie de l'encodeur. Elle montre que l'encodeur accumule successivement les échantillons de la tension d'entrée. Au bout de m = 4 périodes, la tension de sortie est égale à 1.2 V ($0.3 \times 4 = 1.2$). Les FIGURE 4.14 et FIGURE 4.9 montrent que nous avons obtenu les mêmes résultats avec les modèles SystemC-AMS et SPICE.

ii) Intégration du modèle SPICE de l'encodeur dans le réseau

Pour pouvoir intégrer le modèle SPICE de l'encodeur dans le réseau, nous avons créé un nœud qui comporte les éléments suivants : une interface LTSpice, un CAN, un micro-contrôleur et un module RF. Nous avons sauvegardé les résultats du modèle SPICE dans un fichier. L'interface LTSpice est un module SystemC-AMS qui va lire le fichier exporté et va générer un signal TDF à partir des données lues. La FIGURE 4.15 illustre l'architecture du nœud intégrant le modèle SPICE de l'encodeur.

4.2.3 Routeur

Le routeur est un module SystemC qui comporte un tableau dynamique de sockets (*simple_target_socket*) pour recevoir les transactions provenant des nœuds et un autre tableau dynamique de sockets (*simple_initiator_socket*) pour transférer les transactions vers les décodeurs. Pour router correctement les paquets vers les décodeurs, il utilise une

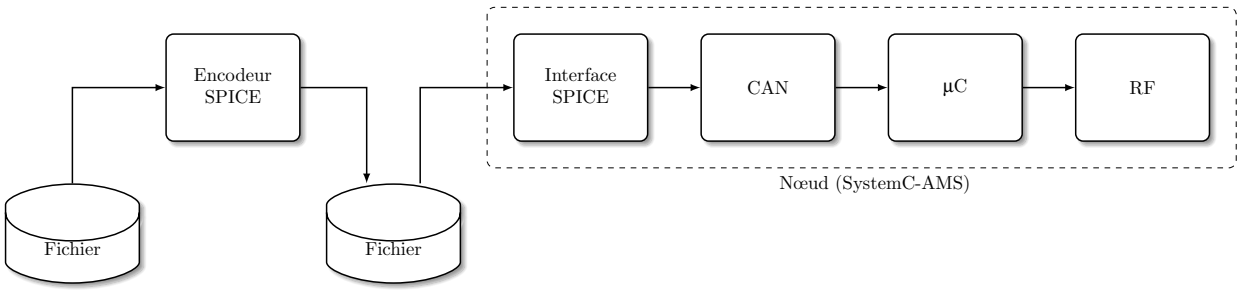


FIGURE *4.15 – Architecture d'un nœud intégrant le modèle SPICE de l'encodeur.*

table de routage comportant deux colonnes. La première colonne de la table indique le champ *Id* du paquet et la deuxième colonne spécifie le numéro du *simple_initiator_socket* correspondant.

4.2.4 Décodeur

Le décodeur a un socket de type *simple_target_socket* pour recevoir les transactions provenant des nœuds. Le constructeur du module prend aussi en paramètres une liste des configurations des nœuds. Le module se sert de ces informations pour décoder et sauvegarder les données provenant des nœuds.

Pour traiter les transactions provenant des nœuds, le décodeur implémente la méthode *b_transport()*. Dans cette méthode, le décodeur récupère le paquet à partir de la transaction *tlm_generic_payload* passée en paramètre. Puis, il crée un *processus* dynamique qui va traiter le paquet avec la méthode *sc_spawn()*. Et enfin, il met à jour l'attribut *response_status* de la transaction en TLM_OK_RESPONSE.

Le *processus* créé dynamiquement par la méthode *b_transport()* pour traiter le paquet reçu le désassemble. Le champ *id* du paquet lui permet d'identifier le nœud et de récupérer les configurations correspondantes à partir de la liste de *NodeCfg* passée en paramètre au constructeur. Le champ *type* lui permet de savoir s'il s'agit d'un paquet transportant des données dans le *payload* ou bien d'un paquet informant la fin de l'enregistrement.

a) Dans le cas d'un paquet de données, si le nœud correspondant n'a pas d'encodeur, le *payload* contient des données brutes. Le *processus* extrait les données du *payload*, puis il les sauvegarde directement dans un fichier. Si le nœud qui a transmis le paquet implémente un encodeur numérique ou analogique, alors le *payload* contient des données encodées. Le *processus* les sauvegarde localement jusqu'à ce qu'il ait reçu $\mathcal{M}$ échantillons encodés. Après avoir reçu les $\mathcal{M}$ échantillons encodés, le *processus* reconstruit le signal original avec l'algorithme de reconstruction proposé, puis il sauvegarde le résultat dans un fichier.

b) Dans le cas d'un paquet de type STOP, le *processus* supprime la configuration du nœud de la liste des *NodeCfg*. Une fois la liste vide, le *processus* arrête la simulation en appelant la méthode *sc_stop()*.

4.3 Résultats de simulation

4.3.1 Évaluation de la qualité de reconstruction

i) Description du banc de test

Pour valider le modèle, nous avons pris des signaux EMG, ECG et EEG provenant de la base de données Physionet [Phy]. Le Tableau 4.1 résume les caractéristiques des signaux où f_S est la fréquence d'acquisition.

Tableau *4.1 – Signaux de tests utilisés lors de la validation du modèle SystemC-AMS.*

Signal	**Base de données**	f_S
EMG	Electromyogram (emgdb)	4 kHz
ECG	MIT-BIH Arrhythmia (mitdb)	360 Hz
EEG	EEG Motor Movement/Imagery Dataset (eegmmidb)	160 Hz

Nous avons configuré la résolution du CAN à 16-bit. Les tensions de références du CAN ont été fixées à 3.3 et 0 V. Le pas de quantification correspondant est :

$$\Delta V = \frac{V_p - V_n}{2^{16} - 1} = 50.355\ \mu V \tag{4.1}$$

Tableau *4.2 – Caractéristiques des nœuds utilisés pour évaluer la qualité de reconstruction.*

Nœud	**Type d'encodeur**	$m = \frac{\mathcal{N}}{\mathcal{M}}$
$Nœud_1$	Aucun	-
$Nœud_2$	Numérique	2
$Nœud_3$	Numérique	4
$Nœud_4$	Numérique	6
$Nœud_5$	Numérique	8
$Nœud_6$	Numérique	10
$Nœud_7$	Numérique	12
$Nœud_8$	Analogique (SystemC-AMS)	2
$Nœud_9$	Analogique (SystemC-AMS)	4
$Nœud_{10}$	Analogique (SystemC-AMS)	6
$Nœud_{11}$	Analogique (SystemC-AMS)	8
$Nœud_{12}$	Analogique (SystemC-AMS)	10
$Nœud_{13}$	Analogique (SystemC-AMS)	12

Puisque les amplitudes des signaux sont de l'ordre de millivolts et microvolts, nous avons configuré le gain de l'amplificateur à 2×10^2. Nous avons ajouté un offset de 0.2 V pour que la dynamique du signal amplifié soit comprise entre 0 et 3.3 V. Pour les encodeurs analogiques, nous avons fixé la valeur des résistances R_i à 10 kΩ et celle des condensateurs C_i à 0.1 µF. Les résistances ON et OFF des interrupteurs ont été configurées à 10 Ω et 10 MΩ respectivement. Nous avons fixé la durée T_S de la phase d'échantillonnage de EB_1 à 20 µs.

Nous avons créé un réseau composé de 13 nœuds et d'un décodeur. Le Tableau 4.2 donne les caractéristiques des différents nœuds utilisés pendant la simulation. Le premier nœud n'est pas muni d'encodeur. Le signal capturé par ce nœud est utilisé comme référence pour évaluer la qualité des signaux capturés avec les nœuds restants qui implémentent soit des encodeurs numériques, soit des encodeurs analogiques. Le facteur de compression $m = \frac{\mathcal{N}}{\mathcal{M}}$ des nœuds implémentant des encodeurs augmente progressivement de 2 à 12 avec un pas de 2. Nous avons fixé la valeur de $\mathcal{M}$ à 32, c'est-à-dire que le décodeur démarre la phase de reconstruction après avoir reçu 32 échantillons encodés et génère ($\mathcal{N} = 32 \times m$) échantillons.

Tableau *4.3 – PRD et SNR en fonction du taux de compression pour le signal EMG.*

CR [%]	**MATLAB**		**SystemC-AMS**			
			Numérique		**Analogique**	
	PRD [%]	**SNR [dB]**	**PRD [%]**	**SNR [dB]**	**PRD [%]**	**SNR [dB]**
50.00	20.57	16.13	20.69	16.11	20.70	16.10
75.00	32.86	10.83	33.16	10.87	33.17	10.86
83.33	46.30	7.31	47.32	7.14	47.32	7.14
87.50	56.27	5.23	55.93	5.34	55.93	5.34
90.00	61.16	4.49	60.93	4.53	60.93	4.53
91.67	63.54	4.10	63.85	4.07	63.85	4.07

ii) Résultats

Les Tableau 4.3, Tableau 4.4 et Tableau 4.5 rapportent les PRD et SNR des signaux reconstruits en fonction du taux de compression CR pour les trois signaux. Ces tableaux rapportent aussi les résultats obtenus pendant la simulation MATLAB. Ces résultats montrent que :

a) L'encodeur numérique est plus performant que l'analogique même s'il n'y a pas de grande différence en termes de PRD et de SNR. Les différences maximales sont respectivement 2.7 % et 0.4 dB. L'avantage de l'encodeur analogique est que ce dernier numérise le signal en dessous de la fréquence de Nyquist.

TABLEAU 4.4 – *PRD et SNR en fonction du taux de compression pour le signal ECG.*

CR [%]	MATLAB		SystemC-AMS			
			Numérique		Analogique	
	PRD [%]	SNR [dB]	PRD [%]	SNR [dB]	PRD [%]	SNR [dB]
50.00	1.27	38.00	1.30	37.83	1.34	37.60
75.00	4.84	27.58	4.86	27.70	4.89	27.62
83.33	12.41	20.10	12.15	20.23	12.18	20.18
87.50	24.22	12.37	24.17	12.38	24.20	12.37
90.00	30.74	10.29	31.32	10.13	31.35	10.12
91.67	35.40	9.12	34.85	9.23	34.87	9.23

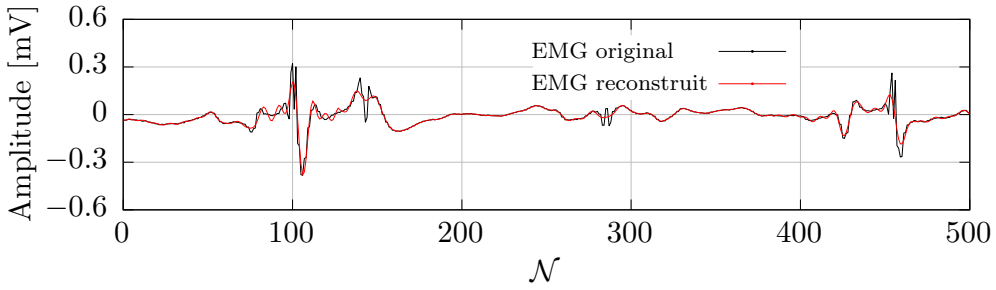
TABLEAU 4.5 – *PRD et SNR en fonction du taux de compression pour le signal EEG.*

CR [%]	MATLAB		SystemC-AMS			
			Numérique		Analogique	
	PRD [%]	SNR [dB]	PRD [%]	SNR [dB]	PRD [%]	SNR [dB]
50.00	22.52	13.22	23.15	12.96	25.01	12.26
75.00	34.00	9.47	33.38	9.61	36.07	8.92
83.33	43.31	7.37	43.77	7.28	46.28	6.78
87.50	50.04	6.07	50.01	6.06	52.43	5.65
90.00	54.62	5.26	54.84	5.23	57.12	4.87
91.67	57.93	4.77	58.71	4.66	60.92	4.33

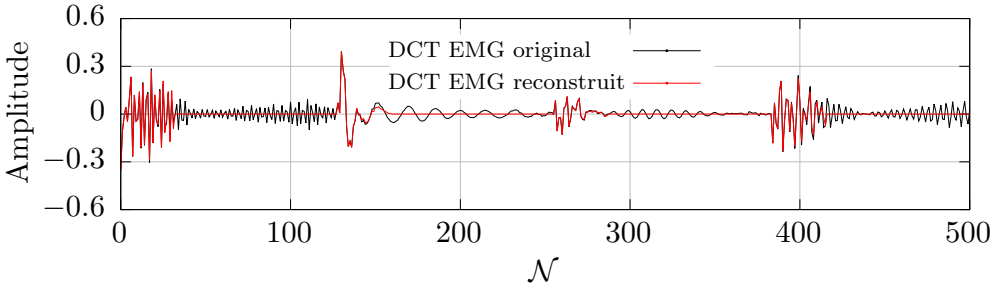
b) Par rapport à l'EMG et l'EEG, l'ECG est plus compressible parce qu'il a une représentation plus parcimonieuse dans le domaine de la DCT. En effet, avec les même taux de compression, nous avons obtenu des meilleurs résultats en termes de PRD et de SNR avec le signal ECG.

La FIGURE 4.16(a) illustre une partie du signal EMG original et celui reconstruit avec l'encodeur analogique pour un taux de compression de 75.0 %, pour des valeurs de $\mathcal{M}$ et de $\mathcal{N}$ égales à 32 et 128. Elle montre que le signal EMG reconstruit (en rouge) est un peu distordu par rapport à l'original (en noir). Les PRD et SNR sont respectivement 36.07 % et 8.92 dB. En se référant à la FIGURE 4.16(b), les coefficients significatifs de la DCT du signal EMG original sont presque répartis sur tout le domaine. Alors que l'algorithme de reconstruction récupère seulement les 32 premiers coefficients de la DCT. Le fait d'ignorer les 96 restants génère une distorsion dans le domaine temporel.

La FIGURE 4.17(a) illustre une partie du signal ECG original et celui reconstruit avec l'encodeur analogique pour un taux de compression de 83.3 %, pour des valeurs de $\mathcal{M}$ et de $\mathcal{N}$ égales à 32 et 192. La différence entre le signal ECG original (en noir) et celui reconstruit

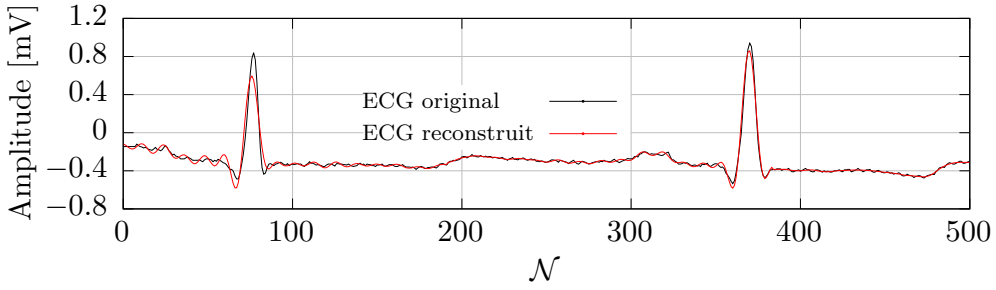


(a) Signaux EMG (emg_healthy).

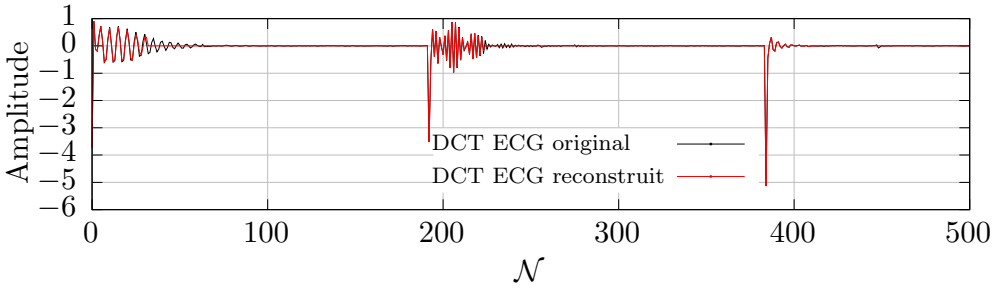


(b) DCT des signaux EMG.

FIGURE *4.16 – Signal EMG original (noir) et celui reconstruit (rouge) avec un taux de compression de 75.0 %.*



(a) Signaux ECG (enregistrement numéro 100).

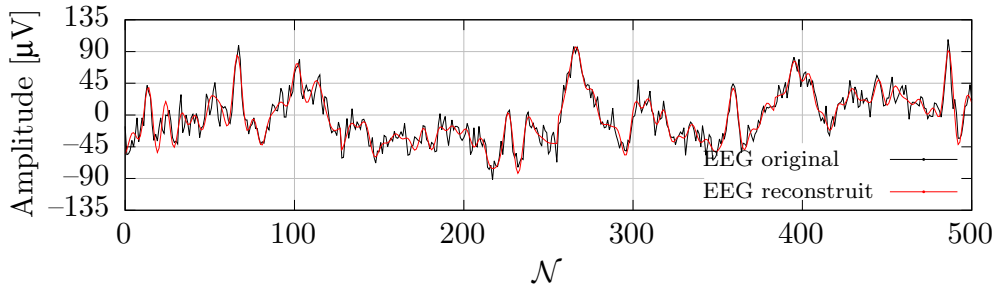


(b) DCT des signaux ECG.

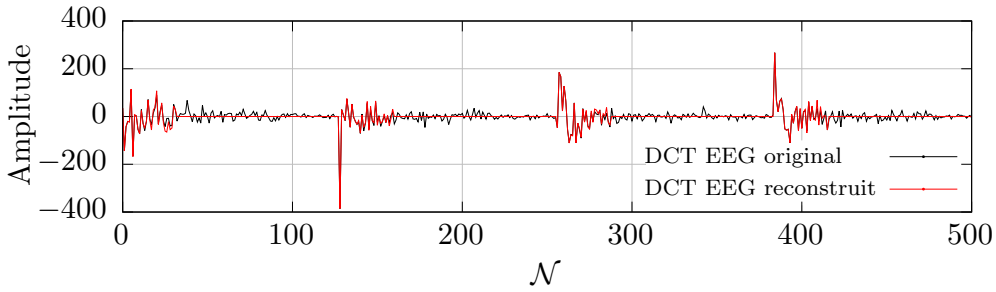
FIGURE *4.17 – Signal ECG original (noir) et celui reconstruit (rouge) avec un taux de compression de 83.3 %.*

(en rouge) est difficilement perceptible. Les PRD et SNR sont respectivement 12.18 % et 20.18 dB. La FIGURE 4.17(b) montre que les coefficients significatifs de la DCT du signal ECG sont concentrés dans les 70 premiers éléments. L'algorithme de reconstruction récupère seulement les 32 premiers coefficients de la DCT et ignore les 160 restants. Le fait de les ignorer n'affecte pas la qualité du signal ECG reconstruit puisque leurs valeurs sont presque

nulles.



(a) Signaux EEG.



(b) DCT des signaux EEG.

FIGURE *4.18 – Signal EEG original (noir) et celui reconstruit (rouge) avec un taux de compression de 75.5 %.*

La FIGURE 4.18(a) illustre une partie du signal EEG original et celui reconstruit avec l'encodeur analogique pour un taux de compression de 75.0 %, pour des valeurs de $\mathcal{M}$ et $\mathcal{N}$ égales à 32 et 128. Le signal EEG reconstruit (en rouge) est distordu par rapport au signal original (en noir). Cette distorsion est due au fait que les coefficients de la DCT du signal EEG sont répartis sur presque tout le domaine (voir FIGURE 4.18(b)).

4.3.2 Validation du modèle SPICE de l'encodeur

i) Description du banc de test

Pour évaluer les performances du modèle SPICE de l'encodeur en termes de qualité de reconstruction, nous avons créé un réseau composé de 13 nœuds et un décodeur :

a) Un nœud qui n'implémente pas d'encodeur. Le signal capturé par ce nœud est utilisé comme référence pour évaluer les qualités des signaux capturés avec ceux qui implémentent des encodeurs.

b) 6 nœuds qui comportent des interfaces SPICE.

c) 6 nœuds qui implémentent des encodeurs analogiques.

Nous avons utilisé un signal EMG provenant de base de données Physionet. Nous avons varié de 2 à 12 avec un pas de 2 les facteurs de compression m = $\mathcal{N}/\mathcal{M}$ des nœuds implémentant des encodeurs. Nous avons configuré la résolution des CAN à 16-bit et nous avons fixé les tensions de référence positive et négative à 0 et 3 V. Le décodeur reconstruit

les signaux après avoir reçu $\mathcal{M} = 32$ échantillons encodés. À la fin de l'enregistrement, nous avons évalué les PRD et SNR des signaux reconstruits.

TABLEAU *4.6 – PRD et SNR en fonction du taux de compression pour le signal EMG.*

CR [%]	**LTSpice**		**SystemC-AMS**	
	PRD [%]	**SNR [dB]**	**PRD [%]**	**SNR [dB]**
50.00	20.54	16.33	20.70	16.10
75.00	34.02	10.56	33.17	10.86
83.33	46.13	7.33	47.32	7.14
87.50	56.27	5.21	55.93	5.34
90.00	56.66	5.33	60.93	4.53
91.67	64.12	4.01	63.85	4.07

ii) Résultats

Le TABLEAU 4.6 rapporte les PRD et SNR des signaux reconstruits en fonction du taux de compression. Nous avons obtenu presque les mêmes PRD et SNR avec les deux modèles. Les différences maximales sont respectivement 1.26 % et 0.19 dB.

L'avantage du modèle SPICE par rapport au modèle SystemC-AMS est qu'il est plus proche du circuit réel. Cependant, le modèle SystemC-AMS est beaucoup plus rapide. En effet, une simulation de 10 secondes avec le modèle SPICE durent en moyenne 4 minutes. Alors qu'avec le modèle SystemC-AMS, la simulation dure moins d'une minute.

4.3.3 Influence de la valeur de $\mathcal{M}$ sur la qualité et le temps de reconstruction

Durant les expérimentations précédentes, nous avons fixé la valeur de $\mathcal{M}$ à 32. C'est-à-dire que le décodeur démarre la phase de reconstruction après avoir reçu 32 échantillons encodés. Il est intéressant d'évaluer la qualité du signal reconstruit en fonction de la valeur de $\mathcal{M}$. Sachant que le temps de reconstruction T_R varie en fonction de $\mathcal{M}$. Plus la valeur de $\mathcal{M}$ est grande, plus le temps de reconstruction T_R est long.

Nous avons créé un réseau composé de 2 nœuds et 4 décodeurs. L'un des nœuds implémente un encodeur numérique et l'autre n'implémente pas. Nous avons configuré les décodeurs avec 4 différentes valeurs de $\mathcal{M}$: 16, 32, 64 et 128. Nous avons configuré la table de routage du routeur pour que les paquets provenant du nœud implémentant l'encodeur soient transmis vers les 4 décodeurs en même temps. Le signal provenant du nœud qui n'implémente pas d'encodeur est utilisé comme référence pour évaluer la qualité des signaux reconstruits par les décodeurs.

Nous avons configuré les deux nœuds pour qu'ils capturent un signal ECG provenant de base de données Physionet. Nous avons effectué une série des tests où nous avons varié le taux de compression du nœud.

TABLEAU *4.7 – Impact de la valeur de $\mathcal{M}$ sur la qualité de reconstruction et le temps de reconstruction.*

CR [%]	$\mathcal{M}$ ($\mathcal{N}$)	**PRD [%]**	**SNR [dB]**	T_R [ms]
50.00	16 (32)	1.34	37.75	0.04
	32 (64)	1.30	37.83	0.24
	64 (128)	1.27	37.96	1.06
	128 (256)	1.27	37.96	7.63
75.00	16 (64)	3.93	29.62	0.04
	32 (128)	4.86	27.70	0.24
	64 (256)	5.61	25.42	1.06
	128 (512)	5.64	25.14	7.63
83.33	16 (96)	9.67	25.27	0.04
	32 (192)	12.15	20.23	0.24
	64 (384)	13.80	17.20	1.06
	128 (768)	14.19	16.96	7.63
87.50	16 (128)	18.60	20.37	0.04
	32 (256)	24.17	12.38	0.24
	64 (512)	24.36	12.29	1.06
	128 (1024)	24.29	12.29	7.63
90.00	16 (160)	25.59	15.69	0.04
	32 (320)	31.32	10.13	0.24
	64 (640)	28.88	10.79	1.06
	128 (1280)	28.56	10.68	7.63
91.67	16 (192)	30.96	13.93	0.04
	32 (384)	34.85	9.23	0.24
	64 (768)	35.92	8.89	1.06
	128 (1536)	36.04	8.41	7.63

Le TABLEAU 4.7 rapporte les résultats obtenus au cours de 6 tests où le taux de compression prend une des valeurs suivantes : 50 %, 75 %, 83.33 %, 87.5 %, 90 % et 91.67 %. Ces résultats montrent que, pour un CR donné, à l'exception de CR = 50 %, la qualité du signal reconstruit se dégrade lorsque la valeur de $\mathcal{M}$ augmente. Le signal reconstruit

se dégrade dans le sens où le PRD augmente et le SNR diminue. La dernière colonne du tableau rapporte la durée maximale du temps de reconstruction T_R. Ces résultats montrent que T_R augmente avec $\mathcal{M}$. La durée de T_R dépend seulement de la valeur de $\mathcal{M}$ mais pas celle de $\mathcal{N}$ puisque l'algorithme de reconstruction proposé cherche la solution du système d'équations :

$$\mathbf{A}_{\Omega_{\mathcal{M}}}\alpha_{\Omega_{\mathcal{M}}} = \mathbf{y} \tag{4.2}$$

où $\mathbf{A}_{\Omega_{\mathcal{M}}}$ est une matrice $\mathcal{M} \times \mathcal{M}$ formée par les $\mathcal{M}$ colonnes de $\mathbf{A} \in \mathbb{R}^{\mathcal{M}\times\mathcal{N}}$. Il serait avantageux de prendre une faible valeur de $\mathcal{M}$ pour avoir une meilleure qualité de reconstruction et un court temps de reconstruction.

4.3.4 Impact de la résolution du convertisseur analogique numérique sur la qualité de reconstruction

i) Description du banc de test

Durant les expérimentations précédentes, nous avons fixé la résolution du CAN à 16-bit avec des tensions de références positive et négative de 0 et 3.3 V. Dans cette expérimentation, nous avons évalué l'impact de la résolution du CAN sur la qualité de reconstruction. Nous avons créé un réseau composé de 5 nœuds et un décodeur. Nous avons configuré les résolutions des CAN à 16-bit, 12-bit, 10-bit et 8-bit. Les tensions de références positive et négative ont été fixées à 0 et 3.3V. Le Tableau 4.8 rapporte les caractéristiques des nœuds utilisés.

Tableau *4.8 – Caractéristiques des nœuds utilisés pour évaluer l'impact de la résolution du CAN.*

Nœud	Encodeur	Résolution du CAN [bit]
Nœud_1	Aucun	16
Nœud_2	Analogique	16
Nœud_3	Analogique	12
Nœud_4	Analogique	10
Nœud_5	Analogique	8

Les nœuds capturent un signal EMG provenant de la base de données Physionet. Nous avons configuré le décodeur pour qu'il démarre la phase de reconstruction après avoir reçu $\mathcal{M} = 16$ échantillons encodés. Nous avons effectué une série de tests où nous avons varié le taux de compression des nœuds implémentant des encodeurs. Le signal EMG provenant Nœud_1 est utilisé comme référence pour évaluer la qualité de reconstruction de ceux provenant des nœuds restants.

TABLEAU *4.9 – Impact de la valeur de la résolution du CAN sur la qualité de reconstruction.*

CR [%]	**R [bit]**	**PRD [%]**	**SNR [dB]**	**SNR [dB] / R[bit]**
50.00	16	17.28	18.02	1.12
	12	17.87	17.47	**1.45**
	10	24.95	13.48	1.34
	8	62.06	5.77	0.72
75.00	16	28.93	13.32	0.83
	12	28.95	13.30	1.10
	10	30.05	12.48	**1.24**
	8	40.39	8.81	1.10
83.33	16	43.90	8.63	0.53
	12	43.88	8.63	0.71
	10	44.09	8.51	0.85
	8	47.45	7.32	**0.91**
87.50	16	51.96	6.54	0.40
	12	51.97	6.54	0.54
	10	52.06	6.51	0.65
	8	53.23	6.21	**0.77**

ii) Résultats

Le TABLEAU 4.9 rapporte les résultats obtenus pour 4 valeurs du taux de compression CR. La dernière colonne du tableau calcule le rapport (SNR / R). Elle permet de trouver le bon compromis entre la résolution du CAN et la qualité du signal reconstruit. Lorsque la valeur de CR est assez élevée (83.33 % et 87.50 %), la résolution du CAN n'a pas beaucoup d'effet sur la qualité du signal reconstruit. Dans ces cas, il est avantageux d'utiliser un CAN 8-bit. Pour un taux de compression de 75 %, la résolution du CAN commence à affecter la qualité du signal reconstruit. Cependant la dernière colonne du tableau montre qu'il est intéressant d'utiliser un CAN 10-bit. De même pour un taux de compression de 50 %, la dernière colonne du tableau montre qu'il est bénéfique d'utiliser un CAN 12-bit.

4.3.5 Évaluation de la bande passante et de la consommation

La version actuelle du modèle ne permet pas de mesurer la consommation globale d'un nœud mais seulement celle du module radiofréquence. Nous avons pris le module RF NRF24L01 comme exemple. La FIGURE 4.19 illustre sa consommation en courant pendant la transmission d'un paquet d'après son datasheet [RF2]. Lorsque le module est en mode veille (Standby), il consomme un courant très faible de l'ordre de 22 µA pour une tension

d'alimentation V_{DD} égale à 3 V. Avant de passer en mode transmission, le module passe par une étape intermédiaire de configuration (PLL lock) pendant une durée T_S où il consomme un courant I_S de l'ordre de 8 mA pour une V_{DD} de 3 V. Après l'étape intermédiaire, il passe en mode transmission (TX) pendant une durée T_{OA} où il consomme un courant de 11.3 mA pour une V_{DD} de 3 V. Une fois le paquet transmis, le module retourne en mode veille pour économiser de l'énergie.

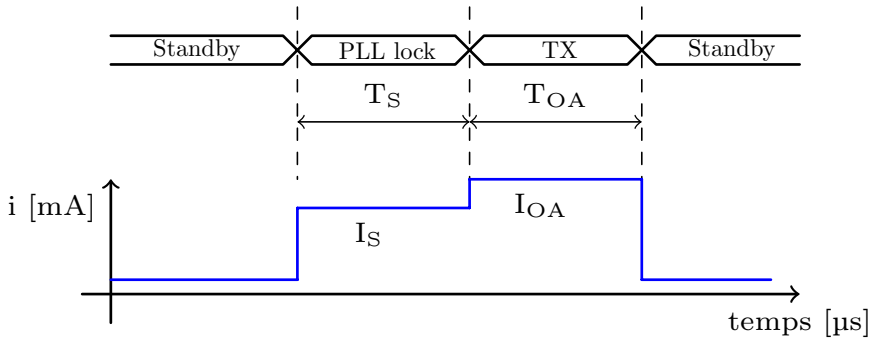


FIGURE *4.19 – Consommation en courant du module NRF24L01 en mode transmission.*

La durée de T_{OA} dépendent à la fois de la taille du paquet et du débit de transmission.

$$T_{OA} = \frac{\text{taille d'un paquet [bit]}}{\text{débit de transmission [bit / s]}} \tag{4.3}$$

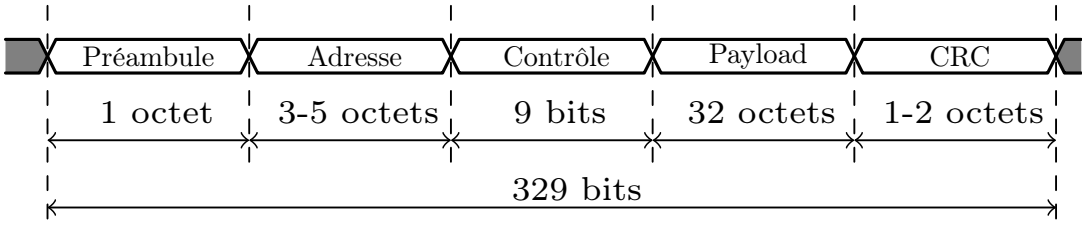


FIGURE *4.20 – Format d'un paquet du module NRF24L01.*

La FIGURE 4.20 illustre le format d'un paquet du module NRF24L01. La taille de l'adresse est configurable sur 3 ou 5 octets, de même celle du contrôle de redondance cyclique ou « *cyclic redundancy check (CRC)* » est configurable sur 1 ou 2 octets. La taille maximale d'un paquet est égale à 329 bits.

Pour un débit de 1 Mbps et un paquet de 329 bits, T_{OA} dure 329 µs. La durée T_S de la phase de configuration ne dépend pas de la taille du paquet et a une valeur fixe égale à 130 µs.

L'énergie consommée pour transmettre un paquet est exprimée comme suit :

$$E_p = V_{DD}(I_S\ T_S + I_{OA}\ T_{OA}) \tag{4.4}$$

où V_{DD} est la tension d'alimentation. Pour les valeurs suivantes :

a) $V_{DD} = 3$ V

b) $I_S = 8$ mA

c) $I_{OA} = 11.3$ mA

d) $T_S = 130$ µs

e) $T_{OA} = 329$ µs

Le module radiofréquence consomme une énergie E_p de 14 µJ pour transmettre un paquet. Nous avons utilisé cette valeur dans le modèle.

Pour évaluer le débit de la voie montante du réseau (nœuds vers décodeur), le routeur compte le nombre de paquets reçus en provenance des nœuds. À chaque fois que le routeur en reçoit un, il incrémente une variable locale. Un processus de type SC_THREAD, configuré pour se déclencher à chaque seconde, calcule le débit du réseau en kilo octet par seconde [Ko/s] à partir du nombre de paquets reçus, en tenant compte du fait que la taille d'un paquet est égale à 329 bits. Après avoir calculé le débit, le processus sauvegarde le résultat dans une variable locale et réinitialise à zéro le nombre de paquets reçus. Le routeur implémente une méthode permettant de récupérer la valeur du débit. Cette méthode peut être appelée après avoir lancé la simulation, c'est-à-dire après la méthode sc_core :: sc_start().

i) Description du banc de test

Nous avons construit un réseau composé de 2 nœuds et d'un décodeur : un nœud muni d'un capteur ECG et un autre avec un EMG. Dans un premier temps, nous avons évalué la consommation des modules RF et le débit du réseau avec des nœuds sans encodeur AC. Puis nous avons configuré les nœuds avec des encodeurs numériques. Nous avons fixé les facteurs de compression à 8 pour l'ECG et à 4 pour l'EMG. La résolution des CAN a été configurée à 16-bit.

ii) Résultats

Tableau *4.10 – Débit du réseau avec et sans les encodeurs AC.*

	Débit [Kops]
Sans encodeur AC	10.9238
Avec encodeur AC	2.61047

Tableau *4.11 – Consommation des modules RF.*

Nœuds	**ECG**	**EMG**
Consommation [µJ] (sans encodeur AC)	952	10486
Consommation [µJ] (avec encodeur AC)	126	2618

Les Tableau 4.10 et Tableau 4.11 rapportent les résultats de simulation. Le nœud EMG a monopolisé le réseau puisqu'il a une fréquence d'échantillonnage plus élevée (4 kHz) que le nœud ECG (360 Hz). Les encodeurs AC ont baissé de 4.2 fois le débit du réseau. La consommation des modules RF est propositionnelle au facteur de compression. Pour le nœud ECG, sans l'encodeur AC, il a fallu 68 paquets pour transmettre les échantillons numérisés. L'encodeur AC a diminué le nombre de paquets à 9. Pour le nœud EMG, le fait d'implémenter l'encodeur AC a réduit le nombre de paquets à 187 au lieu de 749.

4.4 Conclusion

Un modèle exécutable au niveau système d'un WBAN implémentant la méthode proposée avec SystemC-AMS a été présenté. Ce modèle a permis de vérifier et de valider les fonctionnalités de la méthode proposée en amont de la phase de développement. Pour simplifier la mise en œuvre et valider le concept, un modèle haut niveau de l'encodeur implémentant la matrice de mesure a été proposé. Un raffinement de ce modèle au niveau circuit en utilisant des composants ELN a été validé. Un modèle SPICE de la partie analogique de l'encodeur a été aussi développé avec l'outil LTSpice. Des signaux ECG, EEG et EMG provenant de la base de données Physionet ont été utilisés pour alimenter le modèle.

La version actuelle de ce modèle nous a principalement permis d'évaluer la qualité du signal reconstruit en fonction du taux de compression. Il nous a aussi donné l'opportunité d'explorer et de mesurer d'autres paramètres à savoir, (i) l'impact de la valeur de $\mathcal{M}$ (respectivement de $\mathcal{N}$) sur la qualité et le temps de reconstruction, (ii) l'influence de la résolution du CAN sur la qualité de reconstruction, et (iii) l'évaluation de la bande passante et la consommation du module RF.

Ce modèle peut être enrichi ultérieurement pour pouvoir étudier d'autres paramètres, notamment la consommation totale d'un nœud et l'influence de la perte de paquets sur la qualité du signal reconstruit.

Dans le chapitre suivant, nous utiliserons ce modèle pour trouver le bon compromis entre le taux de compression et la qualité du signal reconstruit avant de valider expérimentalement la méthode proposée dans un WBAN. Des signaux préalablement capturés par les nœuds seront utilisés pour alimenter le modèle.

Chapitre 5

Validation expérimentale

Sommaire

5.1 Introduction 106

5.2 Encodeur numérique 106

5.2.1 Implémentation de l'encodeur 107

5.2.2 Implémentation de l'algorithme proposé 110

5.2.3 Résultats 111

5.3 Encodeur analogique 115

5.3.1 Implémentation de l'encodeur 116

5.3.2 Signaux de contrôle 116

5.3.3 Convertisseur analogique numérique 117

5.3.4 Reconstruction des signaux 117

5.3.5 Résultats 118

5.4 Conclusion 121

5.1 Introduction

Le modèle exécutable au niveau système présenté dans le chapitre précédent permet de valider la méthode proposée en amont de la phase de développement. Dans ce chapitre, nous allons évaluer expérimentalement cette méthode. Une collaboration avec l'entreprise TEA nous a donné l'opportunité de tester et mesurer les performances de la méthode proposée sur un système opérationnel. TEA [TEA] est spécialisée dans le développement, la vente et la mise en œuvre de capteurs et solutions de mesure et d'analyse complètes pour l'activité physique, le mouvement et le suivi du regard.

Ce chapitre est structuré en deux parties. Dans la première, nous allons valider l'encodeur numérique en l'implémentant dans un WBAN développé et commercialisé par l'entreprise TEA. Dans la deuxième et dernière partie, un prototype de l'encodeur analogique construit à partir de composants standards sera étudié et évalué.

5.2 Encodeur numérique

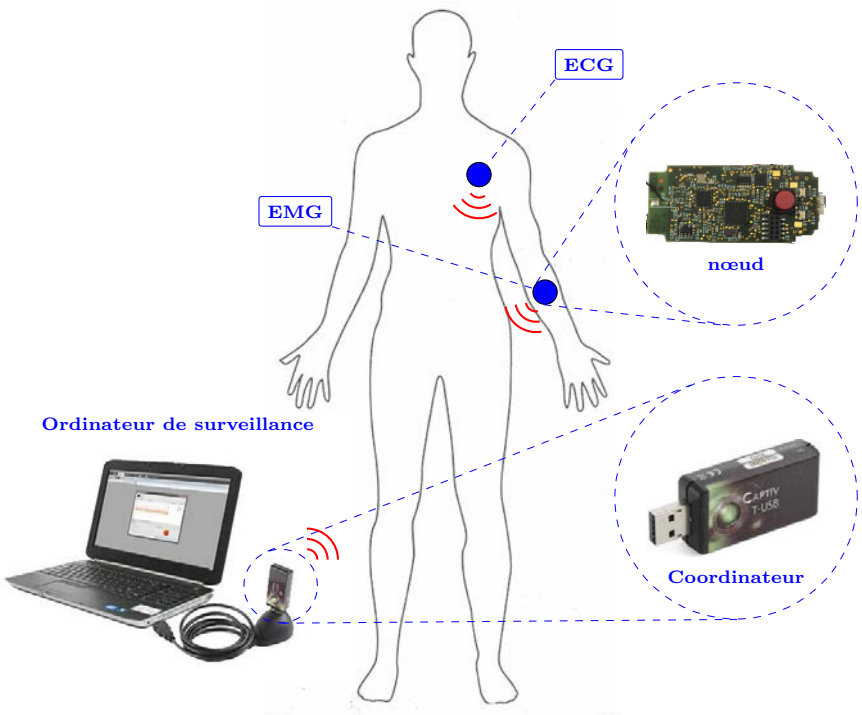


FIGURE *5.1 – Architecture du réseau de capteurs utilisé pendant la validation expérimentale.*

Nous avons appliqué la méthode proposée, la matrice de mesure et l'algorithme de reconstruction, dans un réseau de capteurs sans fils sur le corps humain développés et commercialisés par l'entreprise TEA. Comme illustré à la FIGURE 5.1, le réseau est composé de nœuds comportant chacun différents types de capteurs permettant de mesurer des signaux physiologiques comme l'ECG, l'EMG, la respiration, etc. Un nœud central (coordinateur)

coordonne et récupère les données provenant des différents nœuds via une liaison radio-fréquence. Le coordinateur et les nœuds forment un réseau en étoile. Le coordinateur est connecté sur un ordinateur de surveillance via une liaison « *universal serial bus (USB)* ». Un logiciel qui tourne sur l'ordinateur de surveillance permet de visualiser en temps réel et de sauvegarder les données provenant des différents nœuds.

L'encodeur numérique est adapté pour ce type de réseau puisque les nœuds comportent déjà des microcontrôleurs qui peuvent l'implémenter sous forme d'algorithme. La méthode proposée va permettre d'économiser la bande passante du réseau et de diminuer la consommation d'énergie des nœuds, augmentant ainsi l'autonomie de leurs batteries. Le gain en autonomie dépend de la répartition de la consommation d'énergie des différents modules présents dans les nœuds. La compression des données permet d'augmenter ce gain seulement si le module RF consomme le plus d'énergie pendant la transmission des données.

5.2.1 Implémentation de l'encodeur

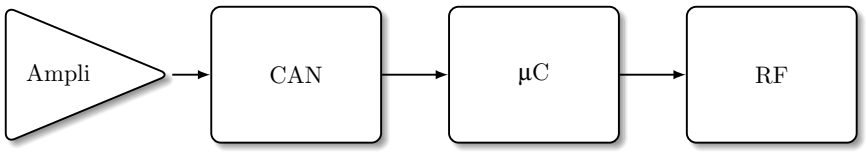


FIGURE *5.2 – Architecture d'un nœud utilisé pendant la validation expérimentale.*

L'architecture d'un nœud du réseau est illustrée à la FIGURE 5.2. Le signal est d'abord amplifié par un amplificateur à faible bruit. Puis, il est numérisé par un CAN 16-bit. Un microcontrôleur PIC24H de Microchip coordonne le fonctionnement du nœud. Les données sont transmises vers le coordinateur par un composant NRF24L01 [RF2]. Il est conçu pour opérer sur la bande de fréquence ISM, 2.400 - 2.4835 GHz, avec un débit de transfert configurable jusqu'à 2 Mb/s. Le nœud est alimenté par une batterie. C'est le microcontrôleur qui fixe la fréquence d'acquisition du CAN et empaquète les échantillons numérisés avant de les transmettre vers le module RF.

Nous avons implémenté l'encodeur numérique dans des nœuds munis de capteurs ECG et EMG. Mais avant de passer à l'implémentation, nous avons utilisé le modèle SystemC-AMS décrit dans le chapitre précédent pour trouver le bon compromis entre le taux de compression (CR) et la qualité du signal reconstruit (PRD et SNR). Des signaux ECG et EMG préalablement capturés par les nœuds ont été utilisés comme signaux de test pour alimenter le modèle. Nous avons créé un réseau composé de sept nœuds (un nœud sans encodeur AC et six autres avec encodeur AC) et d'un décodeur. Nous avons configuré le modèle avec les paramètres réels du réseau :

a) Gain de l'amplificateur : 200.

b) Résolution du CAN : 16-bit.

c) Tensions de références positive et négative du CAN : 0 et 3.3 V.

d) Fréquence d'acquisition : 512 Hz pour l'ECG et 2048 Hz pour l'EMG.

e) Type d'encodeur : numérique.

f) Facteurs de compression des six nœuds : 2, 4, 6, 8, 10, 12.

g) Valeur de $\mathcal{M}$ pour le décodeur : 16, c'est-à-dire qu'il va démarrer la phase de reconstruction après avoir reçu 16 échantillons encodés.

TABLEAU *5.1 – PRD et SNR en fonction du taux de compression pour le signal ECG.*

CR [%]	**PRD [%]**	**SNR [dB]**
50.0	25.0	17.0
75.0	27.5	14.6
83.3	30.2	13.1
87.5	33.9	11.3
90.0	42.1	9.4
91.6	52.4	7.6

TABLEAU *5.2 – PRD et SNR en fonction du taux de compression pour le signal EMG.*

CR [%]	**PRD [%]**	**SNR [dB]**
50.0	13.3	17.7
75.0	27.1	11.6
83.3	36.1	9.2
87.5	44.1	7.6
90.0	50.9	6.5
91.6	56.1	5.8

Les TABLEAU 5.1 et TABLEAU 5.2 rapportent les PRD et SNR des signaux reconstruits en fonction du taux de compression. Ces résultats montrent que les signaux ECG et EMG sont reconstruits avec de faible distorsion respectivement lorsque CR $\leq$ 87.5 % et CR $\leq$ 75 %. Le signal ECG est plus compressible que l'EMG parce qu'il est plus parcimonieux dans le domaine de la DCT. Ces valeurs de CR ont été retenues pendant l'implémentation de l'encodeur numérique dans les nœuds. Le TABLEAU 5.3 résume les caractéristiques des nœuds utilisés pendant la validation expérimentale, où f_S représente la fréquence d'acquisition des signaux en Hz.

La FIGURE 5.3 montre comment les deux nœuds ont été placés sur le corps humain. Les électrodes du nœud qui capture le signal ECG ont été placées sur la poitrine. Le nœud EMG mesurait les activités électriques générés par le muscle de l'avant-bras.

TABLEAU 5.3 – *Caractéristiques des nœuds utilisés pendant la validation expérimentale.*

Nœud	**Type**	f_S **[Hz]**	**CR [%] (m)**
Nœud_1	ECG	512	87.5 (8)
Nœud_2	EMG	2048	75 (4)

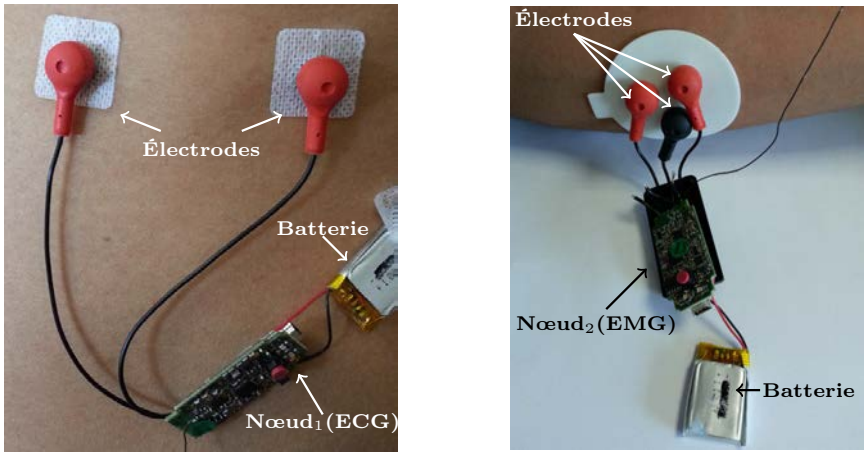


FIGURE 5.3 – *Nœuds capturant les signaux ECG et EMG.*

D'après la description de la matrice de mesure proposée, le microcontrôleur accumule $m = \frac{\mathcal{N}}{\mathcal{M}}$ échantillons consécutifs, 8 pour le Nœud_1 (ECG) et 4 pour le Nœud_2 (EMG). Pour pouvoir évaluer la qualité des signaux reconstruits, nous avons transmis en même temps les échantillons numérisés et les données encodées vers l'ordinateur de surveillance.

Nous avons vérifié l'occupation mémoire des microcontrôleurs avant et après avoir implémenté l'encodeur numérique. Le TABLEAU 5.4 rapporte l'occupation mémoire des microcontrôleurs. Ce tableau montre la simplicité de l'encodeur numérique puisqu'il a augmenté la mémoire programme du microcontrôleur seulement de 66 octets pour le nœud ECG et de 48 octets pour le nœud EMG. Ces augmentations représentent moins de 1 % de la mémoire programme totale disponible. L'encodeur numérique a augmenté de 4 octets (moins de 1 %) la mémoire de données du nœud EMG et n'a pas modifié celle du nœud ECG.

TABLEAU 5.4 – *Occupation mémoire des microcontrôleurs.*

Nœud	**Programme [octet]**	**Données [octet]**
Nœud_1 (ECG) sans l'encodeur AC	24645 ≈ 53 %	796 ≈ 9 %
Nœud_1 (ECG) avec l'encodeur AC	24711 ≈ 53 %	796 ≈ 9 %
Nœud_2 (EMG) sans l'encodeur AC	26193 ≈ 56 %	670 ≈ 8 %
Nœud_2 (EMG) avec l'encodeur AC	26241 ≈ 56 %	674 ≈ 8 %

5.2.2 Implémentation de l'algorithme proposé

L'entreprise TEA a déjà conçu une application pour pouvoir afficher en temps réel, sauvegarder et analyser les données provenant des nœuds. L'application a été développée avec le « *Framework DOTNET* » de Microsoft. Nous avons mis à jour l'application pour qu'elle puisse décoder et afficher en temps réel les données encodées issues des nœuds implémentant des encodeurs AC.

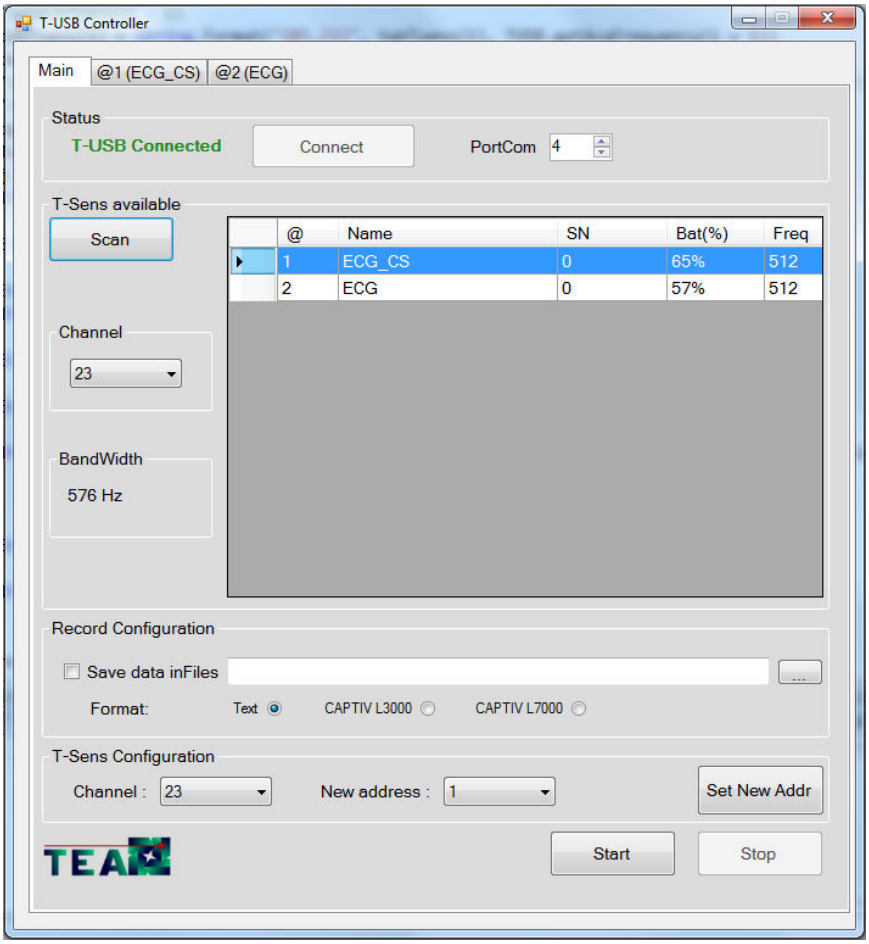


FIGURE *5.4 – Interface de contrôle des nœuds.*

La FIGURE 5.4 illustre la fenêtre principale de l'application. La procédure d'enregistrement commence toujours par une phase de détection qui va scruter tous les nœuds se trouvant dans le même canal que le coordinateur connecté sur l'ordinateur de contrôle. L'application donne la possibilité de changer le canal du coordinateur (à partir du menu déroulant « *Channel* »). Le module NRF24L01 partage la bande de fréquence ISM en 125 canaux dont la largeur varie en fonction du débit de transmission [RF2]. Le coordinateur et les nœuds peuvent dialoguer seulement s'ils se trouvent dans le même canal de transmission. Les informations sur les nœuds détectés lors de la phase de scrutation sont affichées dans le tableau central. Ces informations incluent le nom, le niveau de la batterie et la fréquence d'acquisition de chaque nœud. Dans cet exemple, deux nœuds ont été détectés sur le canal 23 :

a) Un nœud ECG qui implémente un encodeur AC (ECG_CS) avec une fréquence d'acquisition de 512 Hz.

b) Un nœud ECG classique avec une fréquence d'acquisition de 512 Hz.

Une fois les nœuds détectés, l'utilisateur peut entamer l'enregistrement. Les données provenant des nœuds sont affichées en temps réel et sauvegardées dans des fichiers.

Nous avons implémenté en langage C# l'algorithme de reconstruction proposé. Au lieu d'afficher directement les données encodées provenant des nœuds implémentant l'AC, nous les avons sauvegardées dans une mémoire temporaire. Au bout de $\mathcal{M}$ échantillons encodés, nous avons reconstruit les signaux originaux avec l'algorithme proposé et avons affiché en temps réel les résultats sur l'interface.

Pour économiser l'espace disque, nous avons sauvegardé les données provenant des nœuds implémentant l'encodeur AC dans des fichiers sous forme compressée. Nous avons créé une application indépendante permettant de décoder ultérieurement les fichiers encodés et d'évaluer la qualité de reconstruction.

5.2.3 Résultats

i) Qualité de reconstruction

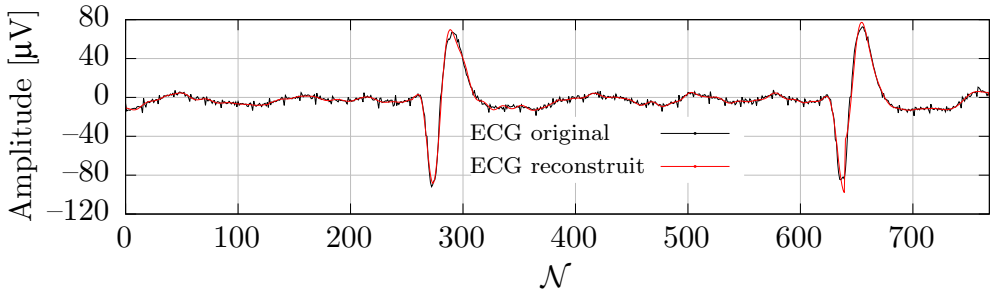
Nous avons configuré la valeur de $\mathcal{M}$ à 16 pendant la phase de reconstruction. Pour le nœud ECG, le facteur de compression m = $\frac{\mathcal{N}}{\mathcal{M}}$ était fixé à 8. La phase de reconstruction produit $\mathcal{N} = 128$ échantillons à partir des $\mathcal{M} = 16$ échantillons encodés. Alors que pour le nœud EMG, le facteur de compression était configuré à 4, la phase de reconstruction génère $\mathcal{N} = 64$ échantillons à partir des $\mathcal{M} = 16$. Nous avons effectué plusieurs tests durant lesquels nous avons calculé les PRD et SNR moyens des signaux ECG et EMG reconstruits. Le Tableau 5.5 rapporte les résultats obtenus pendant l'expérimentation en comparaison avec ceux de la simulation.

Tableau *5.5 – Qualité de reconstruction.*

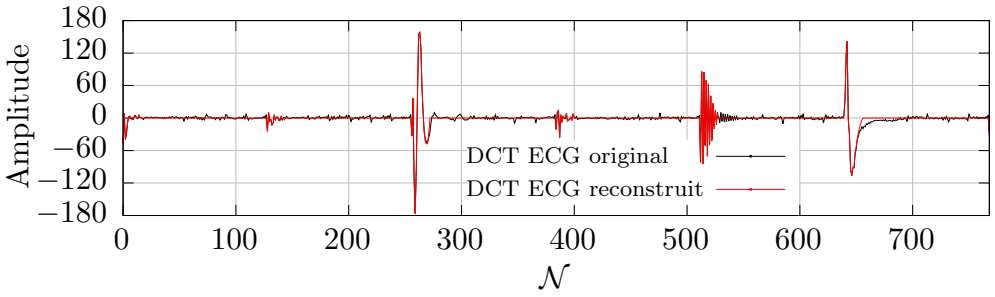
Nœud	**Expérimentation**		**Simulation**	
	PRD [%]	**SNR [dB]**	**PRD [%]**	**SNR [dB]**
Nœud_1 (ECG)	30.15	11.98	33.9	11.3
Nœud_2 (EMG)	29.77	11.12	27.1	11.6

Ce tableau montre que les résultats obtenus pendant la simulation et la validation expérimentale sont presque les mêmes. La différence est due au fait que les signaux utilisés dans les deux cas sont différents.

La Figure 5.5(a) illustre une partie du signal ECG original et celui reconstruit pour $\mathcal{M} = 16$ et $\mathcal{N} = 128$. Elle montre qu'il n'y a pas de grande différence entre les deux signaux. La Figure 5.5(b) illustre leurs transformées DCT. Le signal ECG est parcimonieux



(a) Signaux ECG.



(b) DCT des signaux ECG.

FIGURE *5.5 – Signal ECG original (noir) et celui reconstruit (rouge) avec un taux de compression de 87.5 %, pour* $\mathcal{M} = 16$ *et* $\mathcal{N} = 128$.

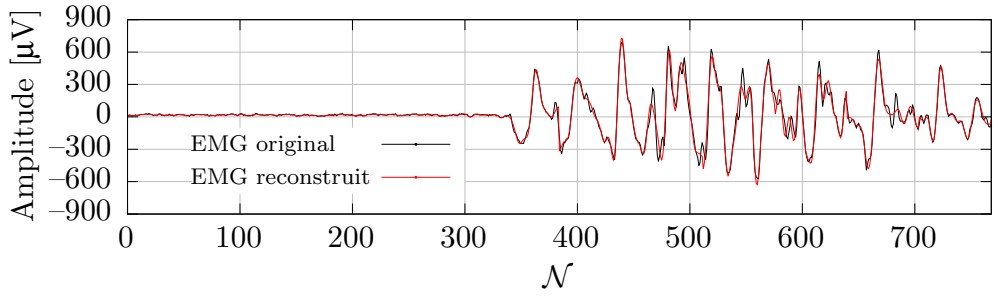
dans le domaine de la DCT. Elle a concentré l'énergie du signal dans les coefficients basses fréquences. La méthode proposée garde seulement les $\mathcal{M} = 16$ coefficients basses fréquences parmi les $\mathcal{N} = 128$. Le fait de supprimer les composantes hautes fréquences n'a pas beaucoup d'impact sur l'apparence visuelle du signal dans le domaine temporel. La méthode proposée a éliminé le bruit haute fréquence et a lissé le signal ECG reconstruit.

La FIGURE 5.6(a) illustre une partie du signal EMG original et celui reconstruit pour $\mathcal{M} = 16$ et $\mathcal{N} = 64$. Elle montre qu'il y a une petite différence entre les deux signaux. Cette distorsion est due au fait que le signal EMG n'est pas tout à fait parcimonieux dans le domaine de la DCT. Comme illustré à la FIGURE 5.6(b), la DCT a concentré l'énergie du signal EMG sur la moitié du domaine, dans les 32 premiers coefficients. La méthode proposée garde seulement les $\mathcal{M} = 16$ coefficients basses fréquences parmi les $\mathcal{N} = 64$. Le fait de supprimer les coefficients restants a introduit cette faible distorsion.

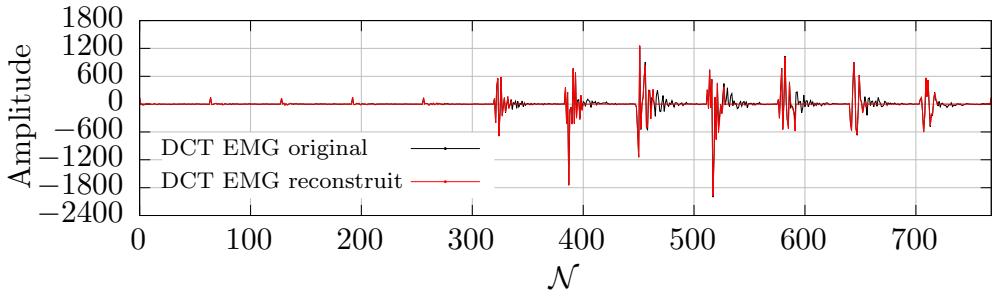
ii) Consommation et bande passante

Puisqu'un paquet peut contenir un payload de 32 octets et que chaque échantillon est numérisé sur 16-bit (2 octets), le module RF peut transmettre au maximum 16 échantillons numérisés par paquet. Le TABLEAU 5.6 rapporte l'énergie consommée par les modules radiofréquences pendant un enregistrement de 1 seconde, en considérant qu'ils consomment 14 µJ par paquet (voir section 4.3.5 du chapitre précédent).

Pour transmettre les 512 échantillons, lorsque le Nœud$_1$ (ECG) n'implémente pas d'encodeur AC, le module radiofréquence a besoin de 32 paquets puisqu'un paquet peut contenir



(a) Signaux EMG.



(b) DCT des signaux EMG.

FIGURE *5.6 – Signal EMG original (noir) et celui reconstruit (rouge) avec un taux de compression de 75 %, pour* $\mathcal{M} = 16$ *et* $\mathcal{N} = 64$.

TABLEAU *5.6 – Consommation du module RF pendant un enregistrement de 1 seconde.*

Parameter	**Sans l'encodeur AC**		**Avec l'encodeur AC**	
	Nœud$_1$ (ECG)	**Nœud$_2$ (EMG)**	**Nœud$_1$ (ECG)**	**Nœud$_2$ (EMG)**
Échantillons à transmettre	512	2048	64	512
Nombre de paquets	32	128	4	32
Énergie [µJ]	448	1792	56	448

au maximum 16 échantillons. Pendant l'enregistrement de 1 seconde, le module radiofréquence consomme une énergie de 448 µJ. Pareillement, pour le Nœud$_2$ (EMG) , le module radiofréquence consomme une énergie de 1792 µJ.

Le fait d'implémenter un encodeur AC a diminué le nombre d'échantillons à transmettre, 64 au lieu de 512 pour le Nœud$_1$ (ECG) et 512 au lien de 2048 pour le Nœud$_2$ (EMG). La consommation d'énergie du module radiofréquence est directement proportionnelle au taux de compression. L'encodeur AC a baissé la consommation du module radiofréquence de 87.5 % pour le Nœud$_1$ (ECG) et de 75 % pour le Nœud$_2$ (EMG).

Comme énoncé précédemment, le fait de compresser les données avant de les transmettre diminue la consommation globale d'un nœud surtout lorsque c'est le module radiofréquence qui consomme le plus. Ce n'est pas le cas avec les nœuds utilisés puisque la consommation d'énergie est répartie dans tous les composants présents. Le fait d'implémenter l'encodeur AC a augmenté l'autonomie des nœuds de 7 % pour le Nœud$_1$ (ECG) et de 5 % pour le

Nœud_2 (EMG).

Par contre en termes de bande passante, le gain est directement proportionnel au taux de compression. Le fait d'implémenter l'encodeur AC a économisé 87.5 % de la bande pour le Nœud_1 (ECG) et 75 % pour le Nœud_2 (EMG).

iii) Retard introduit par l'encodeur AC

TABLEAU *5.7 – Temps d'attente* Δt.

Nœud	f_S **[Hz]**	**m**	Δt **[ms]**
Nœud_1 (ECG)	512	8	250
Nœud_2 (EMG)	2048	4	31.25

Le décodeur qui tourne sur l'ordinateur de surveillance ne peut pas tout de suite démarrer la phase de reconstruction sans avoir reçu tous les $\mathcal{M}$ échantillons encodés. Ce temps d'attente Δt varie en fonction de la valeur de $\mathcal{M}$ et celle de la fréquence d'acquisition du signal. Puisque l'encodeur AC génère les $\mathcal{M}$ échantillons encodés à partir de $\mathcal{N} = m \times \mathcal{M}$ échantillons, le temps d'attente Δt est exprimé comme suit :

$$\Delta t = \mathcal{N} \times \frac{1}{f_S} = (m \times \mathcal{M}) \times \frac{1}{f_S} \tag{5.1}$$

où f_S est la fréquence d'acquisition du signal. Puisque f_S est fixée pour un signal donnée, nous pouvons agir seulement sur la valeur de $\mathcal{M}$ pour diminuer ce temps d'attente Δt. Le TABLEAU 5.7 rapporte la valeur de Δt pour le deux nœuds. Plus la fréquence d'acquisition est élevée, plus la valeur de Δt est faible. C'est pourquoi la valeur de Δt est plus faible pour le Nœud_2 (EMG) par rapport au Nœud_1 (ECG).

En plus de Δt, l'algorithme de reconstruction ajoute aussi une attente T_R qui est égal au temps nécessaire pour reconstruire les $\mathcal{N}$ échantillons à partir des $\mathcal{M}$ encodés. Les résultats du modèle SystemC-AMS ont montré que T_R augmente en fonction de la valeur de $\mathcal{M}$. Plus $\mathcal{M}$ est grand, plus T_R est long. Pour reconstruire $\mathcal{N}$ échantillons du signal à partir des $\mathcal{M} = 16$ encodés, l'ordinateur de surveillance a pris en moyenne un temps T_R égal à 110 ms. Le TABLEAU 5.8 rapporte le retard introduit par l'encodeur AC pour les deux nœuds.

TABLEAU *5.8 – Retard introduit par l'encodeur AC.*

Paramètre	**Nœud_1 (ECG)**	**Nœud_2 (EMG)**
Δt [ms]	250	31.25
T_R [ms]	110	110
Retard total [ms]	360	141.25

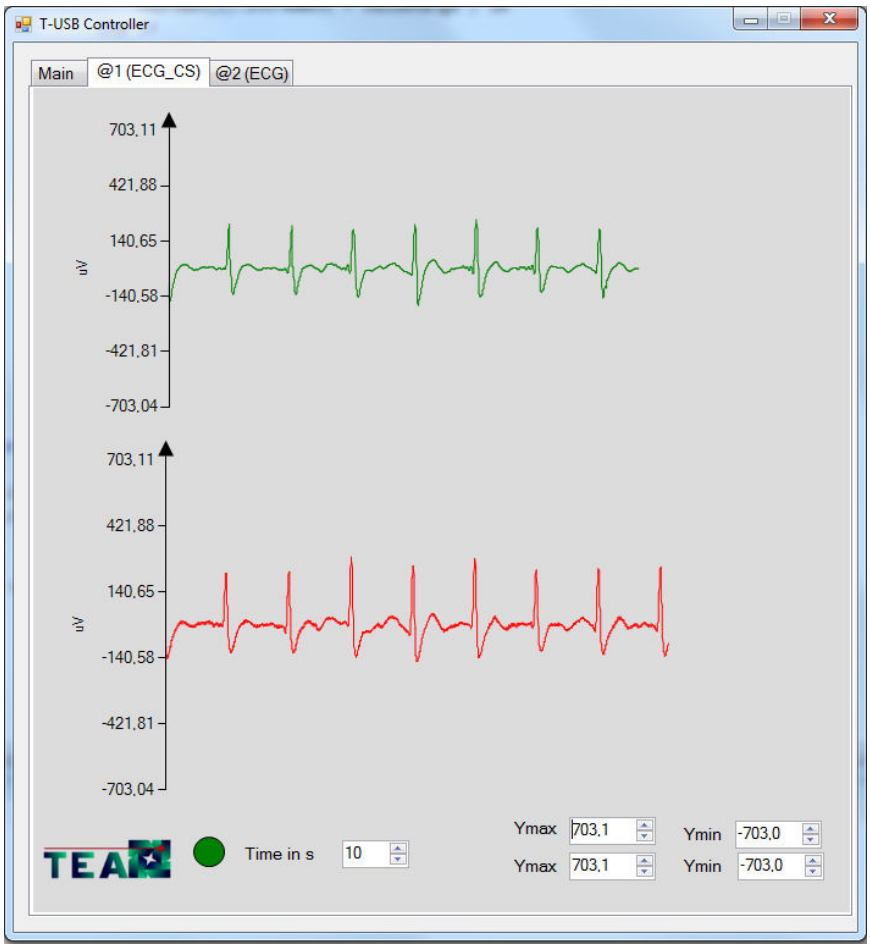


FIGURE 5.7 – *Affichage en temps réel des signaux ECG. En rouge le signal ECG provenant du nœud qui n'implémente pas d'encodeur AC. En vert celui provenant du nœud qui implémente un encodeur AC.*

Pour mettre en évidence le retard introduit par l'encodeur AC, nous avons relié les électrodes de deux nœuds ECG pour qu'ils puissent capter le même signal. L'un des deux nœuds implémente un encodeur AC et l'autre non. La FIGURE 5.7 illustre une partie de l'affichage des signaux. En rouge le signal ECG provenant du nœud qui n'implémente pas d'encodeur AC. En vert celui provenant du nœud qui implémente un encodeur AC. Cette figure montre que le signal provenant du nœud implémentant un encodeur AC en vert est légèrement en retard par rapport au rouge.

5.3 Encodeur analogique

Nous avons réalisé un prototype de l'encodeur analogique avec des composants standards. La FIGURE 5.8 illustre le prototype de l'encodeur analogique. Nous avons utilisé une carte MBED LPC1768 [MBE] pour commander l'encodeur analogique. Elle comporte un microcontrôleur NXP LPC1768 avec un cœur ARM Cortex-M3 fonctionnant à une fréquence de 96 MHz. Le microcontrôleur a une mémoire FLASH de 512 Ko et une RAM de 32 Ko. Le microcontrôleur comporte plusieurs périphériques tels qu'une interface série

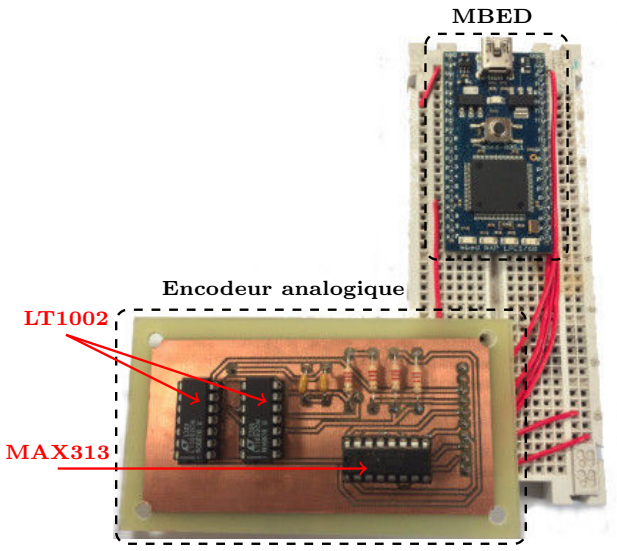


FIGURE 5.8 – *Prototype de l'encodeur analogique.*

UART (« *universal asynchronous receiver/transmitter* »), des contrôleurs SPI et I2C, des convertisseurs analogique/numérique et numérique/analogique, etc.

5.3.1 Implémentation de l'encodeur

Le schéma de l'encodeur analogique est le même que celui du modèle SPICE (voir FIGURE 4.13). Nous avons utilisé le composant LT1002 [LT1] qui comporte deux amplificateurs opérationnels (AOP) de précision de type LT1001. Pour les commutateurs, nous avons utilisé le composant MAX313 [MAX] qui comporte quatre commutateurs analogiques CMOS. La valeur maximale de leur résistance ON est égale à 10 Ω. Les temps de commutation T_{ON} et T_{OFF} sont respectivement 70 ns et 65 ns. Nous avons fixé la valeur des résistances R_i à 10 kΩ et nous avons choisi des condensateurs de 0.1 µF pour C_1 et C_2.

5.3.2 Signaux de contrôle

Nous avons utilisé les ports d'entrée/sortie du microcontrôleur pour commander les signaux de contrôle qui vont piloter l'encodeur analogique :

a) ctrl : nous avons utilisé le pin *p21* de la carte MBED qui est connecté au port *P2[5]* du microcontrôleur.

b) $\overline{\text{ctrl}}$: nous avons utilisé le pin *p23* de la carte MBED qui est connecté au port *P2[3]* du microcontrôleur.

c) init : nous avons utilisé le pin *p25* de la carte MBED qui est connecté au port *P2[1]* du microcontrôleur.

Nous avons implémenté le code du modèle SystemC-AMS dans la routine d'interruption d'une temporisation pour générer les signaux de contrôle. C'est cette routine d'interruption qui commande aussi la conversion de la sortie de l'encodeur analogique.

5.3.3 Convertisseur analogique numérique

Nous avons utilisé le convertisseur analogique numérique interne du microcontrôleur pour numériser la sortie de l'encodeur analogique qui est reliée au pin *p15* de la carte MBED. Ce pin est connecté au port *P0[23]* du microcontrôleur configuré en entrée analogique. C'est un convertisseur 12-bit à approximation successive pouvant atteindre une fréquence de conversion de 200 kHz. Cette fréquence est largement suffisante pour numériser les signaux physiologiques. Les tensions de références positive V_p et négative V_n sont respectivement 3.3 V et 0 V. Le pas de quantification est égal à :

$$\Delta V = \frac{V_p - V_n}{2^{12} - 1} = 0.805 \text{ mV} \tag{5.2}$$

5.3.4 Reconstruction des signaux

Nous avons utilisé l'interface série UART du microcontrôleur pour transmettre les données encodées vers un ordinateur. Nous avons créé une interface graphique avec le framework Qt5. La FIGURE 5.9 illustre la fenêtre principale de l'interface graphique.

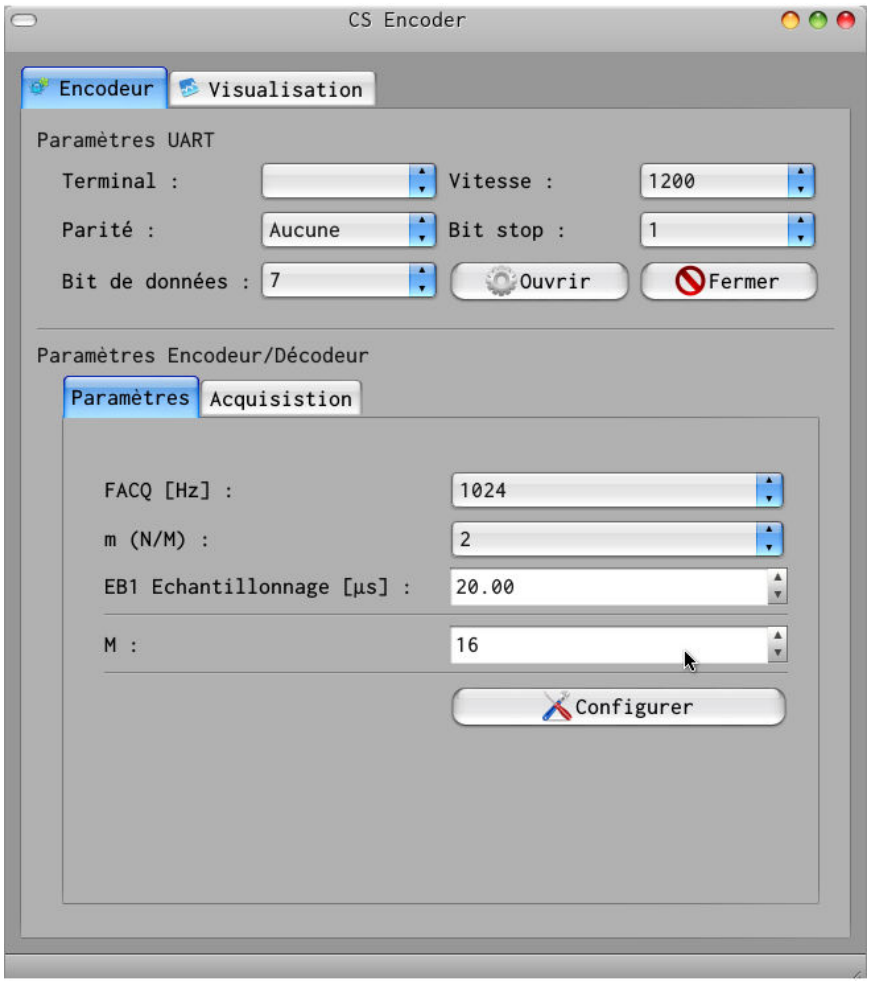


FIGURE *5.9 – Fenêtre principale de l'encodeur.*

La fenêtre principale comporte deux onglets principaux : « Encodeur » et « Visualisation » .

i) Onglet Encodeur

Le premier onglet « Encodeur » permet de configurer les paramètres de l'encodeur. Principalement, la communication UART entre le microcontrôleur et l'ordinateur :

a) Le terminal UART de l'ordinateur connecté au microcontrôleur.

b) Les paramètres de la communication à savoir la vitesse de transmission (baud rate), le type de parité (aucune, paire ou impaire), le nombre de bit de donnée (7 ou 8) et le nombre de bit stop (1, 1.5 ou 2).

Cet onglet offre aussi la possibilité à l'utilisateur de configurer les paramètres de l'encodeur et du décodeur :

a) La fréquence FACQ à laquelle le microcontrôleur synchronise les signaux de contrôle ctrl et $\overline{\text{ctrl}}$. Sa valeur est égale à la fréquence de Nyquist.

b) Le facteur de compression m qui est égal à $\mathcal{N}/\mathcal{M}$. Le microcontrôleur synchronise le CAN pour que ce dernier numérise le signal encodé à une fréquence égale à FACQ / m. Le signal d'entrée de l'encodeur est accumulé à la fréquence de Nyquist. La sortie de l'encodeur est numérisé en dessous de la fréquence de Nyquist. C'est avantageux parce que l'accumulation est plus facile à mettre en œuvre que la numérisation.

c) La durée de phase d'échantillonnage de EB_1 en µs. Le microcontrôleur garde le signal ctrl à l'état haut pendant cette période.

d) La valeur de $\mathcal{M}$. L'interface démarre la phase de reconstruction après avoir reçu $\mathcal{M}$ échantillons encodés provenant du microcontrôleur. La phase de reconstruction utilise l'algorithme proposé pour décoder les données. Nous avons repris le code du modèle SystemC-AMS présenté dans le chapitre précédent.

Comme illustré à la FIGURE 5.10, le sous-onglet « Acquisition » permet d'envoyer des commandes au microcontrôleur pour démarrer ou arrêter l'acquisition. Il permet aussi de choisir l'emplacement du fichier décodé.

ii) Onglet Visualisation

L'onglet « Visualisation » permet de visualiser en temps réel le signal décodé. Le graphe est mis à jour au fur et à mesure où les échantillons encodés provenant du microcontrôleur sont décodés. La FIGURE 5.11 illustre l'affichage en temps réel d'un signal ECG décodé.

5.3.5 Résultats

Nous avons validé l'encodeur analogique en capturant un signal ECG généré à partir d'un générateur de fonction. Nous avons effectué une série de tests durant lesquels nous

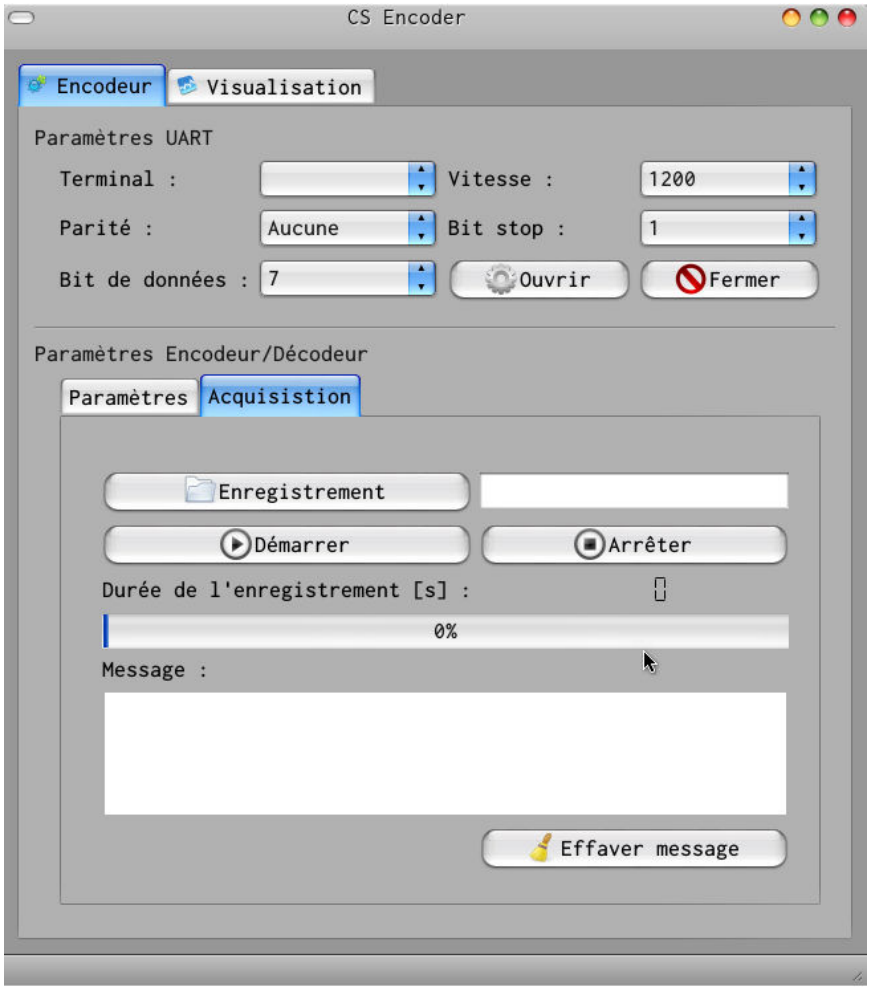


FIGURE *5.10 – Le sous-onglet acquisition de l'interface.*

avons varié la valeur du facteur de compression $m = \frac{\mathcal{N}}{\mathcal{M}}$. À la fin de chaque test, nous avons évalué la qualité du signal reconstruit. Pendant la phase de reconstruction, nous avons configuré la valeur de $\mathcal{M}$ à 16.

Nous avons comparé les résultats obtenus avec ceux du modèle SystemC-AMS. Nous avons créé un réseau composé de 6 nœuds avec différents taux de compression et un nœud central. Le premier nœud n'implémente pas d'encodeur et est utilisé comme référence pour évaluer la qualité des signaux reconstruits avec les cinq restants. Nous avons configuré les nœuds du modèle qui implémentent des encodeurs analogiques avec les paramètres réels du prototype :

a) Fréquence d'acquisition : 512 Hz.

b) Gain de l'amplificateur : 1.

c) Type d'encodeur : analogique.

d) R_{ON} et R_{OFF} des commutateurs : 10 Ω et 1 MΩ.

e) Valeur de condensateurs C_1 et C_2 : 0.1 µF.

f) Durée de la phase d'échantillonnage de EB_1 : 20 µs.

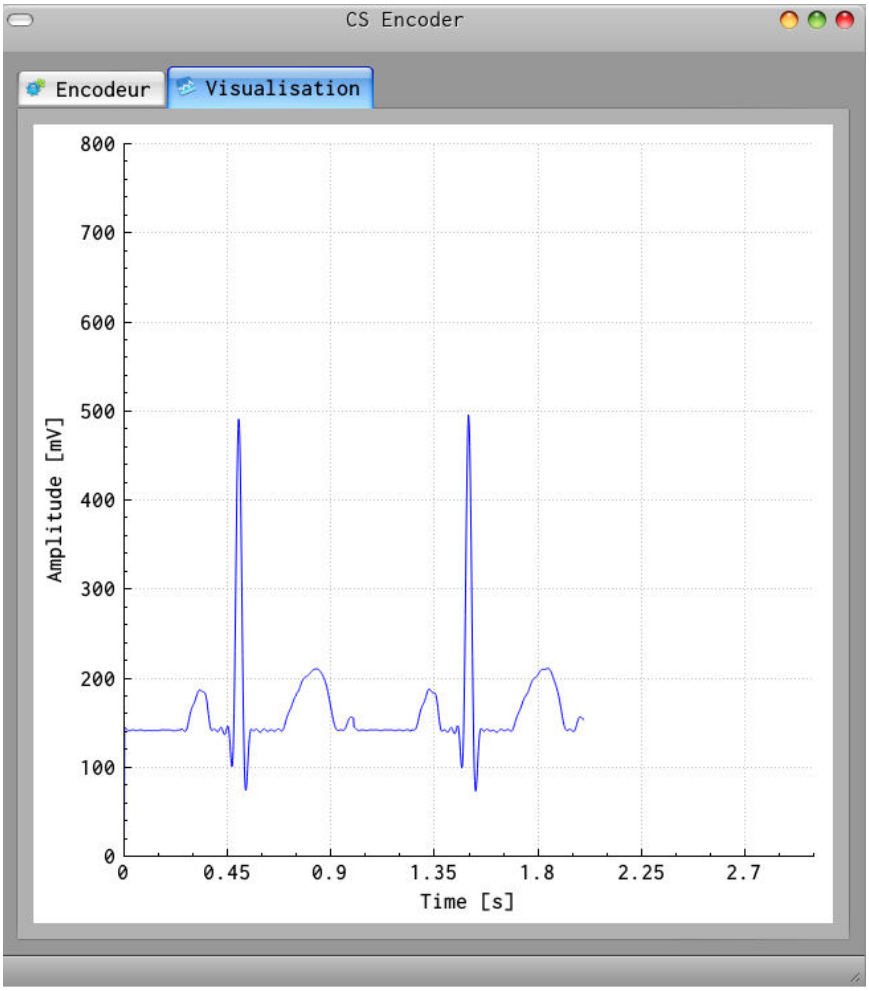


FIGURE 5.11 – *L'onglet visualisation de l'interface. Affichage en temps réel d'un signal ECG décodé.*

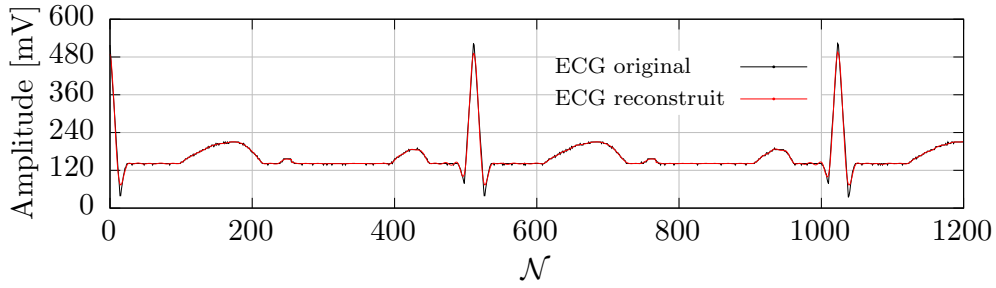
g) Résolution du CAN : 12-bit.

h) Tension de références positive et négative : 3.3 V et 0 V.

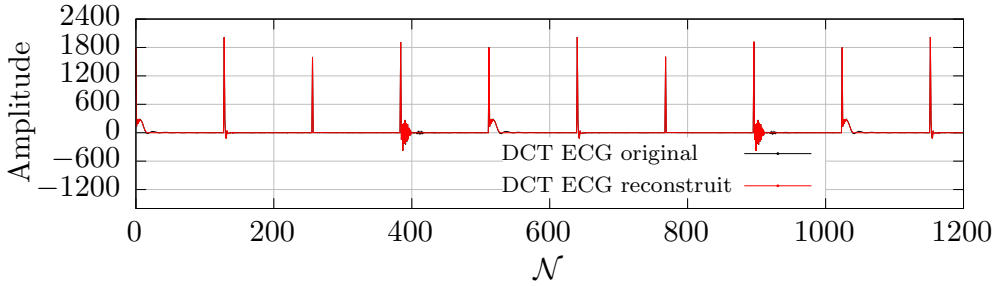
TABLEAU 5.9 – *Qualité du signal reconstruit en fonction du taux de compression.*

CR [%]	**SystemC-AMS**		**Prototype**	
	PRD [%]	**SNR [dB]**	**PRD [%]**	**SNR [dB]**
50.00	0.82	45.21	1.01	40.34
75.00	1.27	38.73	1.36	37.96
83.33	1.93	36.59	2.01	35.93
87.50	2.20	35.04	2.90	33.43
90.00	2.93	33.45	5.41	30.91

Le TABLEAU 5.9 rapporte les résultats obtenus pendant la validation expérimentale en comparaison avec ceux du modèle SystemC-AMS. Il n'y a pas de grande différence entre les résultats obtenus. Les différences maximales en termes de PRD et SNR sont respectivement 2.48 % et 4.87 dB. Le signal ECG issu du générateur de fonction est plus compressible par



(a) Signaux ECG.



(b) DCT des signaux ECG.

FIGURE *5.12 – Signal ECG original (noir) et celui reconstruit (rouge) avec un taux de compression de 87.5 %, pour* $\mathcal{M} = 16$ *et* $\mathcal{N} = 128$.

rapport aux autres signaux d'avant parce qu'il est plus parcimonieux dans le domaine de la DCT. Avec le même taux de compression, nous avons obtenu un faible PRD et un meilleur SNR.

La FIGURE 5.12(a) illustre une partie du signal ECG original provenant du générateur de fonction et celui reconstruit avec un taux de compression de 87.5 %, c'est-à-dire pour $\mathcal{M} = 16$ et $\mathcal{N} = 128$. La différence entre les deux signaux n'est pas visible. Le signal ECG est parcimonieux dans le domaine de la DCT puisque son énergie est concentrée dans seulement quelques composantes basses fréquences, comme illustré à la FIGURE 5.12(b). Le fait de supprimer les composantes hautes fréquences n'a pas d'impact sur la qualité du signal dans le domaine temporel.

5.4 Conclusion

Une validation expérimentale de la méthode proposée a été présentée. Une version numérique de l'encodeur a été implémentée sous forme d'algorithme dans des nœuds capturant des signaux ECG et EMG. L'occupation mémoire des microcontrôleurs et le temps qu'ils ont pris pour encoder les échantillons numérisés ont montré que la matrice de mesure proposée a bien facilité l'implémentation de l'encodeur. En effet, l'encodeur a augmenté moins de 1 % la taille des mémoires programmes et données des nœuds. Les résultats des tests ont montré que le fait d'implémenter l'encodeur AC a augmenté l'autonomie des nœuds de 7 % pour l'ECG et de 5 % pour l'EMG. Puisque le gain en bande passante est directement proportionnel au taux de compression, l'encodeur a économisé 87.5 % de la bande pour

l'ECG et 75 % pour l'EMG.

Un prototype de l'encodeur analogique avec des composants standards a été réalisé. Le prototype a été validé en capturant un signal ECG généré à partir d'un générateur de fonction. Les résultats expérimentaux ont montré que nous avons pu reconstruire le signal ECG sans distorsion avec un taux de compression de 90 %.

Conclusion générale et perspectives

Le « *wireless body area network (WBAN)* » est utilisé dans de nombreux domaines, surtout dans la télémédecine et la télésurveillance. Ces applications vont diminuer le coût de traitement des maladies chroniques qui nécessitent un suivi de longue durée. Elles rendent également possible l'assistance à domicile des personnes âgées ou à mobilité réduite, et apportent une solution aux problèmes engendrés par le vieillissement de la population. Le WBAN peut assurer le suivi des sportifs et des professionnels travaillant dans les milieux hostiles comme les centrales nucléaires, les laboratoires de chimie, etc. Bien que le WBAN soit une solution prometteuse, des problèmes restent encore à résoudre, concernant principalement l'autonomie des nœuds en matière d'énergie. Trois solutions complémentaires ont été proposées pour faire face à ce problème : (i) le captage d'énergie, (ii) l'utilisation de protocole radio basse consommation et (iii) la compression de données. En effet, notons que c'est la partie transmission radio qui consomme le plus d'énergie.

Les études faites ont montré que les méthodes classiques de compression ne sont pas adaptées pour le WBAN parce qu'elles nécessitent un calcul assez complexe entrainant une augmentation de la consommation d'énergie des nœuds. Récemment, une solution alternative consiste à utiliser l'acquisition comprimée (AC). Elle facilite l'acquisition et la compression de données au niveau des nœuds en déportant la complexité vers la phase de reconstruction. Contrairement aux nœuds, le centre de contrôle est moins contraignant et a les ressources nécessaires pour effectuer cette phase de reconstruction. L'AC repose principalement sur deux principes : la parcimonie et l'incohérence. Bien que l'AC demeure une solution efficace, certains points sont encore à améliorer, tels que :

- La construction de la matrice de mesure Φ facilitant la réalisation pratique de l'encodeur. En effet, la complexité de l'encodeur dépend du choix de la matrice de mesure. Pour faciliter l'implémentation, au lieu d'utiliser des matrices aléatoires, des matrices déterministes ont été proposées récemment.
- Le développement d'algorithmes de reconstruction à faible complexité et efficaces, c'est-à-dire des algorithmes pouvant reconstruire le signal seulement avec peu d'échantillons. Par rapport au point cité précédemment, ce deuxième a été mis en exergue puisque plusieurs solutions ont été proposées.
- La recherche de domaine ou de base Ψ permettant d'avoir un degré de parcimonie $\rho = \mathcal{K}/\mathcal{N}$ très faible. Effectivement, plus ρ est faible, plus le signal est compressible. Seulement quelques échantillons seront nécessaires pour le reconstruire correctement. Généralement, les domaines de transformée temps-fréquence sont les plus utilisés. Ré-

cemment, une solution alternative consiste à adapter la matrice $\mathbf{\Psi}$ en fonction des caractéristiques du signal à mesurer.

Rappelons que notre objectif principal est d'apporter une solution au premier point cité ci-dessus, c'est-à-dire simplifier davantage l'implémentation de l'encodeur. Pour cela, nous avons proposé une matrice de mesure binaire déterministe. Elle est adaptée pour compresser des signaux ayant des représentations parcimonieuses dans le domaine de la « *discrete cosine transform (DCT)* » puisqu'elle a une faible cohérence avec la matrice « *inverse discrete cosine transform (IDCT)* ». Par rapport aux autres matrices de mesure, elle est plus facile à implémenter du côté matériel parce que, premièrement, elle ne nécessite que $(\mathcal{N} - \mathcal{M})$ opérations d'addition pour encoder un bloc de $\mathcal{N}$ échantillons. Deuxièmement, elle n'a pas besoin de générateur de nombres aléatoires, ou bien d'espace mémoire pour sauvegarder ses éléments, puisqu'elle est déterministe. Les expérimentations faites au niveau algorithmique sur des signaux électrocardiogramme (ECG) et électrocardiogramme (EMG) ont montré que la matrice de mesure proposée surpasse les matrices aléatoires en termes de qualité de reconstruction.

Nous introduisons aussi une nouvelle méthode originale pour résoudre le deuxième point grâce à cette matrice de mesure déterministe. En effet, elle a simplifié la reconstruction du signal en éliminant la phase de sélection de « *l'orthogonal matching pursuit (OMP)* ». Cette stratégie nous a permis de réduire en même temps sa complexité et son temps d'exécution. L'algorithme de reconstruction proposé effectue un seuillage en sélectionnant seulement les $\mathcal{M}$ composantes basses fréquences de la transformée DCT. Une étude comparative entre l'OMP et l'algorithme de reconstruction proposé a montré que ce dernier présente une meilleure performance en termes de qualité de reconstruction.

La méthode proposée, reposant sur la combinaison de la matrice de mesure et l'algorithme de reconstruction, est similaire à une technique de seuillage classique dans le domaine de la DCT. Une étude comparative en termes de qualité de reconstruction et de complexité d'encodage entre les deux méthodes a été faite. Même si la méthode classique est plus efficace en termes de qualité de reconstruction, l'encodage avec celle proposée est moins complexe et est plus adapté pour le WBAN.

Avant de passer à la validation expérimentalement, nous avons conçu un modèle exécutable au niveau système d'un WBAN implémentant la méthode proposée avec le langage SystemC-AMS. Ce modèle nous a permis de vérifier et de valider les fonctionnalités de notre méthode en amont de la phase de développement. Pour simplifier la mise en œuvre et valider le concept, un modèle haut niveau de l'encodeur implémentant la matrice de mesure a été proposé. Un raffinement de ce modèle au niveau circuit en utilisant des composants « *electrical linear network (ELN)* » a été validé. Un modèle SPICE de la partie analogique de l'encodeur a été aussi développé avec l'outil LTSpice. Des signaux ECG, électroencéphalogramme (EEG) et EMG provenant de la base de données Physionet ont été utilisés pour alimenter le modèle.

La version actuelle de ce modèle nous a principalement permis d'évaluer la qualité du signal reconstruit en fonction du taux de compression. Il nous a aussi donné l'opportunité d'explorer et de mesurer d'autres paramètres à savoir, (i) l'impact de la valeur de $\mathcal{M}$ (respectivement de $\mathcal{N}$) sur la qualité et le temps de reconstruction, (ii) l'influence de la résolution du convertisseur analogique-numérique (CAN) sur la qualité de reconstruction, et (iii) l'évaluation de la bande passante et la consommation du module radiofréquence (RF).

Une version numérique de l'encodeur a été implémentée sous forme d'algorithme dans des nœuds capturant des signaux ECG et EMG. L'occupation mémoire des microcontrôleurs et le temps qu'ils ont pris pour encoder les échantillons numérisés ont montré que la matrice de mesure proposée a bien facilité l'implémentation de l'encodeur. En effet, l'encodeur a augmenté seulement de moins de 1 % la taille des mémoires programmes et données des nœuds. Les résultats des tests ont montré que le fait d'implémenter l'encodeur AC augmente l'autonomie des nœuds de 7 % pour l'ECG et de 5 % pour l'EMG. Puisque le gain en bande passante est directement proportionnel au taux de compression, l'encodeur a pu économiser 87.5 % de la bande pour l'ECG et 75 % pour l'EMG. Un prototype de l'encodeur analogique avec des composants standards a été aussi réalisé. Le prototype a été validé en capturant un signal ECG généré à partir d'un générateur de fonction. Les résultats expérimentaux ont montré que nous avons pu reconstruire le signal ECG sans distorsion avec un taux de compression de 90 %.

Une continuité de ces travaux consistera à traiter le dernier point cité précédemment : reconstruire les signaux capturés avec la matrice de mesure proposée en utilisant d'autres domaines de parcimonie à part la DCT, comme les ondelettes ou bien des domaines conçus pour être adaptés aux caractéristiques des signaux à mesurer. Le modèle exécutable proposé peut être aussi enrichi pour pouvoir étudier d'autres paramètres, notamment la consommation totale d'un nœud et l'influence de la perte de paquets sur la qualité du signal reconstruit. Il serait intéressant d'explorer d'autres architectures d'encodeur implémentant la matrice de mesure proposée qui vont simplifier davantage l'implémentation. Il serait aussi avantageux de comparer et vérifier expérimentalement l'apport des deux types d'encodeur, analogique et numérique, sur la diminution de la consommation d'énergie des nœuds : un nœud muni d'un encodeur analogique et d'un CAN fonctionnant en dessous de la fréquence de Nyquist. Un autre nœud muni d'un CAN fonctionnant à la fréquence de Nyquist et d'un microcontrôleur implémentant l'encodeur numérique.

Annexe A

Modèle SystemC-AMS

Nous avons utilisé la version 2.0 du langage SystemC-AMS et la version 2.3 de SystemC. Les sections suivantes donnent les descriptions des différents modules qui composent le réseau.

A.1 Modèle SystemC-AMS du capteur

Code A.1 – Description du module capteur

```
#ifndef SENSOR_H
#define SENSOR_H

#include <systemc-ams>
#include <fstream>
#include <string>
#include <Clock.h>

/**
 * @brief The Sensor class
 * @author aravelomanantsoa@gmail.cm
 */
class Sensor : sc_core::sc_module {
public:
        //!< output port
        sc_core::sc_out<double> out;
        //!< assert the end of the database file
        sc_core::sc_out<bool> endAcq;

        SC_HAS_PROCESS(Sensor);

        /**
        * @brief Sensor
        * @param name Name of the module
        * @param dbFileName Database file name
        * @param ACQ_F Frequency at which the data have been sampled
        */
        Sensor(sc_core::sc_module_name name,
               const std::string& dbFileName,
               double ACQ_F,
```

```
31                 double unit);
32
33         ~Sensor();
34
35         /**
36         * @brief processing
37         */
38         void processing();
39
40 private:
41         //!< internal clock
42         Clock _clk;
43         std::ifstream _inStream;
44         sc_core::sc_signal<bool> _clkOut;
45         double _unit;
46 };
47
48 #endif  // SENSOR_H
```

Code A.2 – Implémentation du module capteur

```
1  #include <Sensor.h>
2  #include <exception>
3
4  Sensor::Sensor(sc_core::sc_module_name name,
5                 const std::string& dbFileName,
6                 double ACQ_F,
7                 double unit)
8          : sc_core::sc_module(name), _clk("CLK", ACQ_F), _unit(unit) {
9
10         _inStream.open(dbFileName, std::ifstream::in);
11         if (_inStream.fail()) {
12                 throw std::invalid_argument("Data source file not found!"
                       );
13         }
14
15         _clk.out(_clkOut);
16
17         SC_METHOD(processing);
18         sensitive_pos << _clkOut;
19         // dont_initialize ();
20         endAcq.initialize(false);
21 }
22
23 Sensor::~Sensor() { _inStream.close(); }
24
25 void Sensor::processing() {
26         double data = 0.0;
```

```
27          // read data
28          _inStream >> data;
29          if (_inStream.fail()) {
30                  endAcq.write(true);
31                  _clk.stop();
32                  SC_REPORT_INFO("Data Source", "END OF FILE");
33          } else {
34                  out.write(data * _unit);
35          }
36  }
```

A.2 Modèle SystemC-AMS de l'amplificateur

Code A.3 – Description du module amplificateur

```
1   #ifndef AMPLIFIER_H
2   #define AMPLIFIER_H
3
4   #include <systemc-ams>
5
6   /**
7    * @brief The Amplifier class
8    * @author aravelomanantsoa@gmail.com
9    */
10  class Amplifier : public sca_tdf::sca_module {
11  public:
12          //----------- input ports
13          sca_tdf::sca_de::sca_in<double> in;
14          //__________ output ports
15          sca_tdf::sca_out<double> out;
16
17          /**
18          * @brief Amplifier
19          * @param name Name of the module
20          * @param gain Amplifier's gain
21          * @param offset Offset
22          */
23          Amplifier(sc_core::sc_module_name name, double gain, double
                offset);
24
25          void processing();
26
27          void set_attributes();
28
29  private:
30          double _gain;
31          double _offset;
```

```
};

#endif /* AMPLIFIER_H */
```

Code A.4 – Implémentation du module amplificateur

```
#include <Amplifier.h>

Amplifier::Amplifier(sc_core::sc_module_name name, double gain, double
    offset)
        : sca_tdf::sca_module(name), _gain(gain), _offset(offset) {}

void Amplifier::processing() {
        double value = in.read();
        out.write(value * _gain + _offset);
}

void Amplifier::set_attributes() {}
```

A.3 Modèle SystemC-AMS du convertisseur analogique numérique

Code A.5 – Description du module CAN

```
#ifndef ADC_H
#define ADC_H

#include <systemc-ams>
#include <Tdf2de.h>
#include <iomanip>
#include <cmath>

/**
 * @brief The ADC class
 * @author aravelomanantsoa@gmail.com
 */
class ADC : public sc_core::sc_module {
        public:
                //!< ADC resolution
                enum RESOLUTION {
                        R8 = 8,
                        R10 = 10,
                        R12 = 12,
                        R16 = 16
                };

                //!< Input port
```

```
24                sca_tdf::sca_in<double> in;
25                //!< Start ADC conversion
26                sc_core::sc_in<bool> convert;
27                //!< Positive reference voltage
28                sc_core::sc_in<double> vp;
29                //!< Negative reference voltage
30                sc_core::sc_in<double> vn;
31                //!< FIFO holding the conversion results
32                sc_core::sc_fifo<sc_dt::sc_lv<16> > fifo;
33
34                SC_HAS_PROCESS(ADC);
35
36                /**
37                * @brief ADC
38                * @param name Module's name
39                * @param res ADC resolution
40                */
41                ADC(sc_core::sc_module_name name, ADC::RESOLUTION res =
                      RESOLUTION::R8);
42
43                /**
44                * @brief SC_METHOD sensitive to the convert signal which
45                *  performs the digital convestion.
46                */
47                void processing();
48
49                /**
50                * @brief DAC
51                * @param value
52                * @return
53                */
54                double DAC(sc_dt::sc_lv<16> value);
55
56        private:
57                Tdf2de _t2d;
58                sc_core::sc_signal<double> _deOut;
59                RESOLUTION _res;
60 };
61
62 #endif  // ADC_H
```

Code A.6 – Implémentation du module CAN

```
1
2  #include <ADC.h>
3
4  ADC::ADC(sc_core::sc_module_name name, RESOLUTION res)
5          : sc_core::sc_module(name), _t2d("T2D"), _res(res) {
```

```
6         _t2d.in(in);
7         _t2d.out(_deOut);
8
9         SC_METHOD(processing);
10        sensitive << convert.pos();
11        dont_initialize();
12 }
13
14 double ADC::DAC(sc_dt::sc_lv<16> value) {
15        int intVal = value.to_uint();
16        double step = (vp.read() - vn.read()) / (std::pow(2, (int)_res) -
              1);
17        return intVal * step + vn.read();
18 }
19
20 void ADC::processing() {
21        double data = _deOut.read();
22        sc_dt::sc_lv<16> convResult(sc_dt::SC_LOGIC_0);
23
24        for (int i = 1; i <= _res; ++i) {
25                convResult[_res - i] = sc_dt::SC_LOGIC_1;
26                double value = DAC(convResult);
27                if (data < value) {
28                        convResult[_res - i] = sc_dt::SC_LOGIC_0;
29                }
30        }
31
32        // save result in FIFO
33        fifo.nb_write(convResult);
34 }
```

A.4 Modèle SystemC-AMS de l'encodeur analogique

A.4.1 Modèle du sommateur non inverseur

Code A.7 – Description du module sommateur

```
1  #ifndef ELNADDER_H
2  #define ELNADDER_H
3
4  #include <systemc-ams>
5
6  /**
7   * @brief The ElnAdder class
8   * @author aravelomanantsoa@gmail.com
9   */
10 class ElnAdder : public sc_core::sc_module {
```

```
11 public:
12         //!< ELN terminals
13         sca_eln::sca_terminal in1;
14         sca_eln::sca_terminal in2;
15         sca_eln::sca_terminal out;
16         //!< ELN reference node
17         sca_eln::sca_node_ref gnd;
18         //!< ELN nodes
19         sca_eln::sca_node n1, n2;
20 
21         sca_eln::sca_r* R1;
22         sca_eln::sca_r* R2;
23         sca_eln::sca_r* R3;
24         sca_eln::sca_r* R4;
25         sca_eln::sca_nullor* oamp;
26 
27         /**
28         * @brief ElnAdder
29         * @param name
30         */
31         ElnAdder(sc_core::sc_module_name name);
32 
33         ~ElnAdder();
34 };
35 
36 #endif // ELNADDER_H
```

Code A.8 – Implémentation du module sommateur

```
1  #include <ElnAdder.h>
2 
3  ElnAdder::ElnAdder(sc_core::sc_module_name name) : sc_core::sc_module(
       name) {
4          // port map
5          R1 = new sca_eln::sca_r("R1", 1e4);
6          R1->p(in1);
7          R1->n(n1);
8          R2 = new sca_eln::sca_r("R2", 1e4);
9          R2->p(in2);
10         R2->n(n1);
11         R3 = new sca_eln::sca_r("R3", 1e4);
12         R3->p(n2);
13         R3->n(gnd);
14         R4 = new sca_eln::sca_r("R4", 1e4);
15         R4->p(n2);
16         R4->n(out);
17         oamp = new sca_eln::sca_nullor("OAMP");
18         oamp->nip(n1);
```

```
        oamp->nin(n2);
        oamp->nop(out);
        oamp->non(gnd);
}

ElnAdder::~ElnAdder() {
        delete R1;
        delete R2;
        delete R3;
        delete R4;
        delete oamp;
}
```

A.4.2 Modèle du suiveur

Code A.9 – Description du module suiveur

```
#ifndef VFOLLOWER_H
#define VFOLLOWER_H

#include <systemc-ams>

/**
 * @brief The VFollower class
 * @author aravelomanantsoa@gmail.com
 */
class VFollower : public sc_core::sc_module {
public:
        //!< ELN terminals
        sca_eln::sca_terminal in, out;
        //!< ELN reference node
        sca_eln::sca_node_ref gnd;

        sca_eln::sca_nullor* oamp;

        /**
        * @brief VFollower
        * @param name Name of the module
        */
        VFollower(sc_core::sc_module_name name);

        ~VFollower();
};

#endif  // VFOLLOWER_H
```

Code A.10 – Implémenation du module suiveur

```
#include <VFollower.h>
```

```
2
3  VFollower::VFollower(sc_core::sc_module_name name) : sc_core::sc_module(
       name) {
4          oamp = new sca_eln::sca_nullor("OAMP");
5          oamp->nip(in);
6          oamp->nin(out);
7          oamp->nop(out);
8          oamp->non(gnd);
9  }
10
11 VFollower::~VFollower() { delete oamp; }
```

A.4.3 Modèle ELN de l'encodeur

Code A.11 – Description ENL du module encodeur

```
1  #ifndef ELNENCODER_H
2  #define ELNENCODER_H
3
4  #include <systemc-ams>
5  #include <ElnAdder.h>
6  #include <VFollower.h>
7
8  /**
9  * @brief The ElnEncoder class
10 * @author aravelomanantsoa@gmail.com
11 */
12 class ElnEncoder : public sc_core::sc_module {
13 public:
14         //!< ELN terminals
15         sca_eln::sca_terminal in, out;
16         //!< ELN reference node
17         sca_eln::sca_node_ref gnd;
18         //!< ELN nodes
19         sca_eln::sca_node n1, n2, n3, n4, n5;
20         //!< input controls
21         sc_core::sc_in<bool> ctrl;
22         sc_core::sc_in<bool> b_ctrl;
23         sc_core::sc_in<bool> init;
24
25         // internal components
26         ElnAdder* ADD;
27         VFollower* VF1;
28         VFollower* VF2;
29         VFollower* VF3;
30
31         sca_eln::sca_de_rswitch* T1;
32         sca_eln::sca_de_rswitch* T2;
```

```
33          sca_eln::sca_de_rswitch* T3;
34          sca_eln::sca_c* c1;
35          sca_eln::sca_c* c2;
36
37          ElnEncoder(sc_core::sc_module_name name,
38                     double C1,
39                     double C2,
40                     double R_ON,
41                     double R_OFF);
42
43          ~ElnEncoder();
44  };
45
46  #endif  // ELNENCODER_H
```

Code A.12 – Implémentation ELN du module encodeur

```
1   #include <ElnEncoder.h>
2
3   ElnEncoder::ElnEncoder(sc_core::sc_module_name name,
4                          double C1,
5                          double C2,
6                          double R_ON,
7                          double R_OFF)
8           : sc_core::sc_module(name) {
9           // module instantiation and port map
10          ADD = new ElnAdder("ADD");
11          ADD->in1(in);
12          ADD->in2(n5);
13          ADD->out(n1);
14
15          T1 = new sca_eln::sca_de_rswitch("T1", R_ON, R_OFF);
16          T1->p(n1);
17          T1->n(n2);
18          T1->ctrl(ctrl);
19
20          c1 = new sca_eln::sca_c("C1", C1);
21          c1->p(n2);
22          c1->n(gnd);
23
24          VF1 = new VFollower("VF1");
25          VF1->in(n2);
26          VF1->out(out);
27
28          VF2 = new VFollower("VF2");
29          VF2->in(n2);
30          VF2->out(n3);
31
```

```
32          T2 = new sca_eln::sca_de_rswitch("T2", R_ON, R_OFF);
33          T2->p(n3);
34          T2->n(n4);
35          T2->ctrl(b_ctrl);
36
37          c2 = new sca_eln::sca_c("C2", C2);
38          c2->p(n4);
39          c2->n(gnd);
40
41          T3 = new sca_eln::sca_de_rswitch("T3", R_ON, R_OFF);
42          T3->p(n4);
43          T3->n(gnd);
44          T3->ctrl(init);
45
46          VF3 = new VFollower("VF3");
47          VF3->in(n4);
48          VF3->out(n5);
49  }
50
51  ElnEncoder::~ElnEncoder() {
52          delete ADD;
53          delete VF1;
54          delete VF2;
55          delete VF3;
56          delete T1;
57          delete T2;
58          delete T3;
59          delete c1;
60          delete c2;
61  }
```

A.4.4 Modèle TDF de l'encodeur

Code A.13 – Description TDF du module encodeur

```
1  #ifndef CSENCODER_H
2  #define CSENCODER_H
3
4  #include <systemc-ams>
5  #include <ElnEncoder.h>
6
7  /**
8   * @brief The CSEncoder class
9   * @author aravelomanantsoa@gmail.com
10  */
11 class CSEncoder : sc_core::sc_module {
12 public:
13         //!< input controls
```

```
14          sca_tdf::sca_in<double> in;
15          sca_tdf::sca_out<double> out;
16
17          sc_core::sc_in<bool> ctrl;
18          sc_core::sc_in<bool> b_ctrl;
19          sc_core::sc_in<bool> init;
20
21          sca_eln::sca_tdf::sca_vsource source;
22          sca_eln::sca_tdf::sca_vsink sink;
23          //!< ELN encodeur
24          ElnEncoder* encoder;
25
26          sca_eln::sca_node_ref gnd;
27          //!< ELN node
28          sca_eln::sca_node n1, n2;
29
30          CSEncoder(sc_core::sc_module_name name,
31                    double C1,
32                    double C2,
33                    double R_ON,
34                    double R_OFF);
35
36          ~CSEncoder();
37  };
38
39  #endif  // CSENCODER_H
```

Code A.14 – Implémentation TDF du module encodeur

```
1   #include <CSEncoder.h>
2
3   CSEncoder::CSEncoder(sc_core::sc_module_name name,
4                        double C1,
5                        double C2,
6                        double R_ON,
7                        double R_OFF)
8           : sc_core::sc_module(name), source("SRC"), sink("SINK") {
9           source.inp(in);
10          source.p(n1);
11          source.n(gnd);
12
13          sink.outp(out);
14          sink.p(n2);
15          sink.n(gnd);
16
17          encoder = new ElnEncoder("ENC", C1, C2, R_ON, R_OFF);
18          encoder->in(n1);
19          encoder->out(n2);
```

```
20          encoder->ctrl(ctrl);
21          encoder->b_ctrl(b_ctrl);
22          encoder->init(init);
23  }
24
25  CSEncoder::~CSEncoder() { delete encoder; }
```

A.5 Modèle SystemC-AMS du microcontrôleur

A.5.1 Type de données représentant un paquet

Code A.15 – Descritpion d'un paquet

```
1   #ifndef PACKET_H
2   #define PACKET_H
3
4   #include <systemc-ams>
5
6   /**
7   * @brief The Packet custom data type class
8   * @author aravelomanantsoa@gmail.com
9   */
10  class Packet {
11  public:
12          //!< Type of packet
13          enum TYPE {
14                  DATA,
15                  STOP
16          };
17          //!< Packet payload length
18          static const int PAYLOAD = 32;
19
20          /**
21          * @brief No argument constructor
22          */
23          Packet();
24
25          /**
26          * @brief Constructor
27          * @param id
28          * @param type
29          */
30          Packet(int id, Packet::TYPE type);
31
32          /**
33          * @brief Copy constructor
34          * @param p
```

```
35          */
36          Packet(const Packet& p);
37
38          ~Packet();
39
40          /**
41          * @brief = operator
42          * @param p
43          */
44          Packet& operator= (const Packet& p);
45
46          /**
47          * @brief () operator
48          * @param pos
49          */
50          sc_dt::sc_lv<8>& operator() (int pos);
51
52          /**
53          * @brief == operator
54          * @param a
55          * @param b
56          */
57          friend bool operator== (const Packet& a, const Packet& b);
58
59          /**
60          * @brief << operator
61          * @param os
62          * @param p
63          */
64          friend std::ostream& operator<< (std::ostream& os, const Packet&
                p);
65
66          inline Packet::TYPE getType() { return _type; }
67
68          inline void setType(Packet::TYPE type) { _type = type; }
69
70          inline int getId() { return _id; }
71
72          inline void setId(int id) { _id = id; }
73
74          inline int getNum() { return _num; }
75
76          inline void setNum(int num) { _num = num; }
77
78          inline friend void sc_trace(sc_core::sc_trace_file* tf,
79                                      const Packet& v,
80                                      const std::string& NAME) {
```

```
81                  sc_trace(tf, v._id, NAME + ".info");
82                  sc_trace(tf, v._type, NAME + ".flag");
83          }
84
85  private:
86          int _num;
87          int _id;
88          TYPE _type;
89          sc_dt::sc_lv<8> _dataArray[PAYLOAD];
90  };
91
92  #endif  // PACKET_H
```

Code A.16 – Implémenation d'un paquet

```
1   #include <Packet.h>
2
3   const int Packet::PAYLOAD;
4
5   Packet::Packet(int id, Packet::TYPE type) : _id(id), _type(type) {}
6
7   Packet::Packet() : _id(0), _type(Packet::TYPE::DATA) {}
8
9   Packet::Packet(const Packet& p) {
10          this->_id = p._id;
11          this->_num = p._num;
12          this->_type = p._type;
13          for (int i = 0; i < PAYLOAD; ++i) {
14                  this->_dataArray[i] = p._dataArray[i];
15          }
16  }
17
18  Packet::~Packet() {}
19
20  Packet& Packet::operator=(const Packet& p) {
21          this->_id = p._id;
22          this->_num = p._num;
23          this->_type = p._type;
24          for (int i = 0; i < PAYLOAD; ++i) {
25                  this->_dataArray[i] = p._dataArray[i];
26          }
27          return *this;
28  }
29
30  bool operator== const Packet& a, const Packet& b) { return false; }
31
32  std::ostream& operator<< (std::ostream& os, const Packet& p) {
33          os << "Id: " << p._id << std::endl;
```

```
34          os << "Num: " << p._num << std::endl;
35          os << "Type: " << p._type << std::endl;
36          if (p._type == Packet::DATA) {
37                  for (int i = 0; i < Packet::PAYLOAD; i = i + 2) {
38                          os << "[" << p._dataArray[i] << p._dataArray[i +
                               1] << "]" << std::endl;
39                  }
40          }
41
42          return os;
43  }
44
45  sc_dt::sc_lv<8>& Packet::operator()(int pos) { return _dataArray[pos]; }
```

A.5.2 Modèle du générateur de signaux de contrôle

Code A.17 – Description du module générateur de signaux de contrôle

```
1   #ifndef SIGNALGENERATOR_H
2   #define SIGNALGENERATOR_H
3
4   #include <systemc-ams>
5   #include <Clock.h>
6   #include <NodeCfg.h>
7
8   /**
9   * @brief The ElnEncoder class
10  * @author aravelomanantsoa@gmail.com
11  */
12  class SignalGenerator : public sc_core::sc_module {
13  public:
14          //!< output ports
15          sc_core::sc_out<bool> ctrl;
16          sc_core::sc_out<bool> b_ctrl;
17          sc_core::sc_out<bool> init;
18          sc_core::sc_out<bool> convert;
19          //!< clock
20          Clock clk;
21
22          SC_HAS_PROCESS(SignalGenerator);
23
24          /**
25          * @brief SignalGenerator
26          * @param name
27          * @param ACQ_F
28          * @param ratio N / M
29          * @param tetaUs
30          */
```

```
31          SignalGenerator(sc_core::sc_module_name name,
32                          double ACQ_F,
33                          int ratio,
34                          NodeCfg::TYPE_ENC type,
35                          double tetaS);
36
37          void process();
38
39  protected:
40          int _ratio;
41          int _counter;
42          double _tetaS;
43          NodeCfg::TYPE_ENC _type;
44          sc_core::sc_signal<bool> _clkOut;
45  };
46
47  #endif  // SIGNALGENERATOR_H
```

Code A.18 – Implémenation du module générateur de signaux de contrôle

```
1   #include <SignalGenerator.h>
2
3   SignalGenerator::SignalGenerator(sc_core::sc_module_name name,
4                                    double ACQ_F,
5                                    int ratio,
6                                    NodeCfg::TYPE_ENC type,
7                                    double tetaS)
8           : sc_core::sc_module(name),
9             clk("CLK", ACQ_F),
10            _ratio(ratio),
11            _counter(0),
12            _tetaS(tetaS),
13            _type(type) {
14
15          clk.out(_clkOut);
16
17          SC_THREAD(process);
18          sensitive_pos << _clkOut;
19          dont_initialize();
20
21          ctrl.initialize(false);
22          b_ctrl.initialize(true);
23          init.initialize(false);
24          convert.initialize(false);
25  }
26
27  void SignalGenerator::process() {
28          bool discard_first_simple = true;
```

```
29          while (true) {
30
31                  if (_type == NodeCfg::TYPE_ENC::DIGITAL ||
32                                  _type == NodeCfg::TYPE_ENC::NONE) {  //
                                        generete only the convert signal
33                          if (discard_first_simple) {
34                                  discard_first_simple = false;
35                          } else {
36                                  convert.write(true);
37                                  wait(_tetaS, sc_core::SC_SEC);
38                                  convert.write(false);
39                          }
40                  } else {
41                          ctrl.write(true);
42                          b_ctrl.write(false);
43
44                          if (_counter == 0) {
45                                  init.write(true);
46                          }
47
48                          wait(_tetaS, sc_core::SC_SEC);
49
50                          ctrl.write(false);
51                          b_ctrl.write(true);
52                          init.write(false);
53
54                          // start ADC convertion before triggering the
                                init signal
55                          if (_counter == (_ratio - 1)) {
56                                  wait(_tetaS, sc_core::SC_SEC);
57                                  convert.write(true);
58                                  wait(_tetaS, sc_core::SC_SEC);
59                                  convert.write(false);
60                          }
61
62                          ++_counter;
63                          if (_counter == _ratio) {
64                                  _counter = 0;
65                          }
66                  }
67
68                  wait();
69          }
70  }
```

A.5.3 Modèle du microcontrôleur

Code A.19 – Description du module microcontrôleur

```
#ifndef MICROCONTROLLER_H
#define MICROCONTROLLER_H

#include <systemc-ams>
#include <SignalGenerator.h>
#include <Packet.h>
#include <NodeCfg.h>

class Microcontroller : public sc_core::sc_module {
public:
        //!< input port
        sc_core::sc_fifo_in<sc_dt::sc_lv<16> > in;
        sc_core::sc_in<bool> endAcq;
        sc_core::sc_in<double> vp;
        sc_core::sc_in<double> vn;
        sc_core::sc_out<Packet> out;
        sc_core::sc_out<bool> txRq;

        SignalGenerator* signalGenerator;

        SC_HAS_PROCESS(Microcontroller);

        Microcontroller(sc_core::sc_module_name name,
                        int id,
                        double F_ACQ,
                        int ratio,
                        double tetaUs,
                        NodeCfg::TYPE_ENC type);

        ~Microcontroller();

        void processSample();

        void processEndAcq();

        void packUpData(sc_dt::sc_lv<16>& data);

private:
        int _id;
        Packet* _packet;
        int _dataIdx;
        int _packetNum;
        NodeCfg::TYPE_ENC _type;
        int _ratio;
        int _counterEncoder;
        sc_dt::sc_lv<16> _encodedData;
```

```
};

#endif  // MICROCONTROLLER_H
```

Code A.20 – Implémenation du module microcontrôleur

```
#include <Microcontroller.h>

Microcontroller::Microcontroller(sc_core::sc_module_name name,
                                 int id,
                                 double ACQ_F,
                                 int ratio,
                                 double tetaUs,
                                 NodeCfg::TYPE_ENC type)
        : sc_core::sc_module(name),
          _id(id),
          _dataIdx(0),
          _packetNum(0),
          _type(type),
          _ratio(ratio),
          _counterEncoder(0),
          _encodedData(sc_dt::sc_lv<16>(sc_dt::SC_LOGIC_0)) {

        signalGenerator = new SignalGenerator("SIG_GEN", ACQ_F, ratio,
           type, tetaUs);

        SC_METHOD(processSample);
        sensitive << in.data_written();
        dont_initialize();

        SC_METHOD(processEndAcq);
        sensitive << endAcq;
        dont_initialize();

        _packet = new Packet;
}

Microcontroller::~Microcontroller() {
        delete signalGenerator;
        delete _packet;
}

void Microcontroller::packUpData(sc_dt::sc_lv<16>& data) {
        sc_dt::sc_lv<8> highByte = data.range(15, 8);
        sc_dt::sc_lv<8> lowByte = data.range(7, 0);
        (*_packet)(_dataIdx) = highByte;
        ++_dataIdx;
        (*_packet)(_dataIdx) = lowByte;
```

```
42          ++_dataIdx;
43          if (_dataIdx == Packet::PAYLOAD) {
44                  // reset data index
45                  _dataIdx = 0;
46
47                  // fill up packet header
48                  _packet->setId(_id);
49                  _packet->setNum(_packetNum);
50                  ++_packetNum;
51                  _packet->setType(Packet::DATA);
52                  // send packet to the transmitter
53                  out.write(*_packet);
54                  txRq.write(!txRq.read());
55          }
56  }
57
58  void Microcontroller::processEndAcq() {
59          // fill up packet header
60          _packet->setId(_id);
61          _packet->setNum(_packetNum);
62          _packet->setType(Packet::STOP);
63          // send packet to the transmitter
64          out.write(*_packet);
65          txRq.write(!txRq.read());
66
67          signalGenerator->clk.stop();
68  }
69
70  void Microcontroller::processSample() {
71          sc_dt::sc_lv<16> sample;
72          in.nb_read(sample);
73
74          if (_type == NodeCfg::TYPE_ENC::DIGITAL) {
75                  // encode data
76                  _encodedData = _encodedData.to_uint() + sample.to_uint();
77                  ++_counterEncoder;
78
79                  if (_counterEncoder == _ratio) {
80                          // reset counter
81                          _counterEncoder = 0;
82                          packUpData(_encodedData);
83                          // reinitialize encoded data
84                          _encodedData = sc_dt::sc_lv<16>(sc_dt::SC_LOGIC_0
                                ));
85                  }
86          } else {
87                  packUpData(sample);
```

```
88          }
89  }
```

A.6 Modèle SystemC-AMS du module RF

Code A.21 – Description du module RF

```
1   #ifndef TRANSMITTER_H
2   #define TRANSMITTER_H
3
4   #include <systemc-ams>
5   #include <tlm>
6   #include <tlm_utils/simple_initiator_socket.h>
7   #include <Packet.h>
8
9   class Transmitter : sc_core::sc_module {
10  public:
11          static const double STB_CURRENT;
12          static const double S_CURRENT;
13          static const double OA_CURRENT;
14
15          static const double TOA;
16          static const double TS;
17
18          //!< TLM-2 socket, defaults to 32-bits wide, base protocol
19          tlm_utils::simple_initiator_socket<Transmitter> initiatorSocket;
20
21          sc_core::sc_in<Packet> inp;
22          sc_core::sc_in<bool> txRq;
23          sc_core::sc_out<double> current;
24
25          SC_HAS_PROCESS(Transmitter);
26
27          Transmitter(sc_core::sc_module_name name);
28
29          ~Transmitter();
30
31          void processing();
32
33          void currentConsumption();
34
35          double getEnergyComsumption();
36  private:
37          const double ENERGY_PER_PACKET = 14;
38          double _energy;
39  };
40
```

```
41 #endif  // TRANSMITTER_H
```

Code A.22 – Implémentation du module RF

```
1  #include <Transmitter.h>
2
3  const double Transmitter::STB_CURRENT = 0.022;
4  const double Transmitter::S_CURRENT = 8;
5  const double Transmitter::OA_CURRENT = 11.3;
6
7  const double Transmitter::TOA = 321;
8  const double Transmitter::TS = 130;
9
10 Transmitter::Transmitter(sc_core::sc_module_name name)
11         : sc_core::sc_module(name), _energy(0.0) {
12
13         SC_THREAD(processing);
14         sensitive << txRq;
15         dont_initialize();
16
17         SC_THREAD(currentConsumption);
18         sensitive << txRq;
19         dont_initialize();
20
21         current.initialize(STB_CURRENT);
22 }
23
24 Transmitter::~Transmitter() {}
25
26 void Transmitter::currentConsumption() {
27         while (true) {
28                 current.write(S_CURRENT);
29                 wait(TS, sc_core::SC_US);
30                 current.write(OA_CURRENT);
31                 wait(TOA, sc_core::SC_US);
32                 current.write(STB_CURRENT);
33
34                 _energy += ENERGY_PER_PACKET;
35
36                 wait();
37         }
38 }
39
40 double Transmitter::getEnergyComsumption() {
41         return _energy;
42 }
43
44 void Transmitter::processing() {
```

```
45          while (true) {
46                  Packet data = inp.read();
47                  int dataSize = sizeof(Packet);
48
49                  // TLM-2 generic payload transaction, reused across calls
                        to b_transport
50                  tlm::tlm_generic_payload* trans = new tlm::
                        tlm_generic_payload;
51                  sc_core::sc_time delay = sc_core::sc_time(10, sc_core::
                        SC_NS);
52                  tlm::tlm_command cmd = tlm::TLM_WRITE_COMMAND;
53
54                  // Initialize the transaction attributes
55                  trans->set_command(cmd);
56                  trans->set_data_ptr((unsigned char*)&data);
57                  trans->set_data_length(dataSize);
58                  trans->set_response_status(
59                                              tlm::TLM_INCOMPLETE_RESPONSE);
                                                    // Mandatory initial value
60
61                  initiatorSocket->b_transport(*trans, delay);  // Blocking
                        transport call
62
63                  // Initiator obliged to check response status and delay
64                  if (trans->is_response_error()) {
65                          std::cerr << "FATAL ERROR!" << std::endl;
66                  }
67
68                  wait();
69          }
70  }
```

A.7 Modèle SystemC-AMS d'un nœud

Code A.23 – Configuration d'un nœud

```
1  #ifndef ACSNODECFG_H
2  #define ACSNODECFG_H
3
4  #include <string>
5  #include <ADC.h>
6
7  /**
8  * @brief The NodeCfg struct
9  * @author aravelomanantsoa@gmail.com
10 */
11 struct NodeCfg {
```

```
12          enum TYPE_ENC {
13                  ANALOG,
14                  DIGITAL,
15                  NONE
16          };
17          int _id;                // Node ID
18          std::string _db_file;  // database file name path.
19          double _acq_f;  // Frequency at which the data contained in the
                given file
20          // were sampled.
21          double _unit;
22          double _gain;              // Gain of the amplifier.
23          double _offset;            // Gain of the amplifier.
24          ADC::RESOLUTION _adc_res;  // resolution of the ADC
25          double _v_ref_p;
26          double _v_ref_n;
27          TYPE_ENC _type_enc;
28          int _ratio;  // N / M, the encoder is disabled if RATIO is equal
                to 1
29          //------ SPECIFIC FOR LTSPICE -----------
30          std::string _spice_file;  // database file name path.
31          double _spice_acq_f;  // Frequency at which the data contained in
                 the given
32          // file were sampled.
33          double _spice_unit;
34          //---------------------------------------
35          double _t_s;           // sample time
36          double _c1;            // Capacitor 1 value
37          double _c2;            // Capacitor 2 value
38          double _r_on;          // On resistance of the switches
39          double _r_off;         // Off resistance of the switches
40          double _time_step_us;  // Simulation Time step.
41          NodeCfg() {}
42
43          NodeCfg(int id, std::string db_file, double acq_f, double unit,
44                  double gain = 1.0, double offset = 0.0,
45                  ADC::RESOLUTION res = ADC::R16, double v_ref_p = 2.5,
46                  double v_ref_n = -2.5, TYPE_ENC type_enc = NONE, int
                        ratio = 1,
47                  std::string spice_file = "", double spice_acq_f = 1e6,
48                  double spice_unit = 1.0, double t_s = 20e-6, double c1 =
                        0.1e-6,
49                  double c2 = 0.1e-6, double r_on = 10, double r_off = 10e6
                        ,
50                  double time_step_us = 2)
51                  : _id(id),
52                    _db_file(db_file),
```

```
                    _acq_f(acq_f),
                    _unit(unit),
                    _gain(gain),
                    _offset(offset),
                    _adc_res(res),
                    _v_ref_p(v_ref_p),
                    _v_ref_n(v_ref_n),
                    _type_enc(type_enc),
                    _ratio(ratio),
                    _spice_file(spice_file),
                    _spice_acq_f(spice_acq_f),
                    _spice_unit(spice_unit),
                    _t_s(t_s),
                    _c1(c1),
                    _c2(c2),
                    _r_on(r_on),
                    _r_off(r_off),
                    _time_step_us(time_step_us) {}
};

#endif  // ACSNODECFG_H
```

Code A.24 – Description du module nœud

```
#ifndef NODE_H
#define NODE_H

#include <systemc-ams>

#include <Sensor.h>
#include <Amplifier.h>
#include <CSEncoder.h>
#include <ADC.h>
#include <Microcontroller.h>
#include <Transmitter.h>
#include <NodeCfg.h>

/**
* @brief The Node class
* @author aravelomanantsoa@gmail.com
*/
class Node : sc_core::sc_module {
public:
        // internal modules
        Sensor* sens;
        Amplifier* lna;
        CSEncoder* encoder;
        ADC* adc;
```

```
25          Microcontroller* micro;
26          Transmitter* tx;
27
28          // internal signals
29          sc_core::sc_signal<double> signal;
30          sc_core::sc_signal<bool> endAcq;
31          sca_tdf::sca_signal<double> condSignal;
32          sca_tdf::sca_signal<double> encodedSignal;
33
34          sc_core::sc_signal<double> vp;
35          sc_core::sc_signal<double> vn;
36
37          sc_core::sc_signal<bool> ctrl;
38          sc_core::sc_signal<bool> b_ctrl;
39          sc_core::sc_signal<bool> init;
40          sc_core::sc_signal<bool> convert;
41
42          sc_core::sc_signal<Packet> txPacket;
43          sc_core::sc_signal<bool> txRq;
44          sc_core::sc_signal<double> current;
45
46          /**
47          * @brief Node constructor
48          * @param name Name of the module
49          * @param cfg Configuration of the node
50          */
51          Node(sc_core::sc_module_name name, NodeCfg& cfg);
52
53          ~Node();
54
55          sca_util::sca_trace_file* _trace;
56  };
57
58  #endif  // NODE_H
```

Code A.25 – Implémentation du module nœud

```
1   #include <Node.h>
2
3   Node::Node(sc_core::sc_module_name name, NodeCfg& cfg)
4           : sc_core::sc_module(name), vp("vp", cfg._v_ref_p), vn("vn", cfg.
              _v_ref_n) {
5           char tmp[60];
6           int id = cfg._id;
7           std::sprintf(tmp, "SENS_%d", id);
8
9           sens = new Sensor(tmp, cfg._db_file, cfg._acq_f, cfg._unit);
10          sens->endAcq(endAcq);
```

```
11          sens->out(signal);
12
13          std::sprintf(tmp, "LNA_%d", id);
14          lna = new Amplifier(tmp, cfg._gain, cfg._offset);
15          lna->in(signal);
16          lna->out(condSignal);
17          lna->set_timestep(cfg._time_step_us, sc_core::SC_US);
18
19          if (cfg._type_enc == NodeCfg::ANALOG) {
20                  std::sprintf(tmp, "ENCODER_%d", id);
21                  encoder = new CSEncoder(tmp, cfg._c1, cfg._c2, cfg._r_on,
                        cfg._r_off);
22                  encoder->in(condSignal);
23                  encoder->init(init);
24                  encoder->ctrl(ctrl);
25                  encoder->b_ctrl(b_ctrl);
26                  encoder->out(encodedSignal);
27          } else {  // bypass the encoder
28                  encoder = NULL;
29          }
30
31          std::sprintf(tmp, "ADC_%d", id);
32          adc = new ADC(tmp, cfg._adc_res);
33
34          if (cfg._type_enc == NodeCfg::ANALOG) {
35                  adc->in(encodedSignal);
36          } else {
37                  adc->in(condSignal);
38          }
39
40          adc->convert(convert);
41          adc->vp(vp);
42          adc->vn(vn);
43
44          std::sprintf(tmp, "MICRO_%d", id);
45          micro = new Microcontroller(
46                          tmp, cfg._id, cfg._acq_f, cfg._ratio, cfg._t_s,
                            cfg._type_enc);
47          micro->in(adc->fifo);
48          micro->endAcq(endAcq);
49          micro->vp(vp);
50          micro->vn(vn);
51          micro->signalGenerator->convert(convert);
52          micro->signalGenerator->ctrl(ctrl);
53          micro->signalGenerator->b_ctrl(b_ctrl);
54          micro->signalGenerator->init(init);
55          micro->out(txPacket);
```

```
56          micro->txRq(txRq);
57
58          std::sprintf(tmp, "TX_%d", id);
59          tx = new Transmitter(tmp);
60          tx->inp(txPacket);
61          tx->txRq(txRq);
62          tx->current(current);
63
64  #ifdef TRACE_ENABLE
65          std::sprintf(tmp, "trace_systemc_%d.dat", id);
66          _trace = sca_util::sca_create_tabular_trace_file(tmp);
67          if (cfg._type_enc == NodeCfg::ANALOG) {
68                  sca_util::sca_trace(_trace, encodedSignal, "v_out");
69          } else {
70                  sca_util::sca_trace(_trace, condSignal, "v_out");
71          }
72          sca_util::sca_trace(_trace, convert, "convert");
73          sca_util::sca_trace(_trace, current, "I");
74  #endif
75  }
76
77  Node::~Node() {
78          delete sens;
79          delete lna;
80          if (encoder != NULL) {
81                  delete encoder;
82          }
83          delete adc;
84          delete micro;
85          delete tx;
86  #ifdef TRACE_ENABLE
87          sca_util::sca_close_vcd_trace_file(_trace);
88  #endif
89  }
```

A.8 Modèle SystemC-AMS du routeur

Code A.26 – Description du routeur

```
1  #define SC_INCLUDE_DYNAMIC_PROCESSES
2
3  #ifndef ROUTER_H
4  #define ROUTER_H
5
6  #include <systemc-ams>
7  #include <tlm>
8  #include <tlm_utils/simple_target_socket.h>
```

```
9  #include <tlm_utils/simple_initiator_socket.h>
10 #include <Clock.h>
11 #include <Packet.h>
12 #include <map>
13
14 /**
15 * @brief The Router class
16 * @author aravelomanantsoa@gmail.com
17 */
18 class Router : sc_core::sc_module {
19 public:
20         // TLM-2 socket, defaults to 32-bits wide, base protocol
21         tlm_utils::simple_target_socket<Router>* targetSocketArray;
22         tlm_utils::simple_initiator_socket<Router>* initiatorSocketArray;
23
24         SC_HAS_PROCESS(Router);
25
26         Router(sc_core::sc_module_name name, int targetCount, int
              initiatorCount,
27                 std::map<int, std::vector<int> > & routingTable);
28
29         ~Router();
30
31         void b_transport(tlm::tlm_generic_payload& trans, sc_core::
              sc_time& delay);
32
33         void dataRateProcess();
34
35         double getDataRate();
36
37         int getTotalPacketNumber();
38
39 private:
40         int _targetCount;
41         int _initiatorCount;
42         std::map<int, std::vector<int> > _routingTable;
43         int _totalPacketNumber;
44         int _packetPerSecond;
45         double _dataRate;
46 };
47
48 #endif  // ROUTER_H
```

Code A.27 – Implémentation du routeur

```
1 #include <Router.h>
2
3 Router::Router(sc_core::sc_module_name name, int targetCount,
```

```
4                int initiatorCount, std::map<int, std::vector<int> > &
                     routingTable)
5          : sc_core::sc_module(name),
6            _targetCount(targetCount),
7            _initiatorCount(initiatorCount),
8            _routingTable(routingTable),
9            _totalPacketNumber(0),
10           _packetPerSecond(0) {
11
12         targetSocketArray = new tlm_utils::simple_target_socket<Router>[
               _targetCount];
13         // register socket
14         for (int i = 0; i < _targetCount; ++i) {
15                 targetSocketArray[i].register_b_transport(this, &Router::
                       b_transport);
16         }
17
18         initiatorSocketArray =
19                         new tlm_utils::simple_initiator_socket<Router>[
                               _initiatorCount];
20
21         // register processes
22         SC_THREAD(dataRateProcess);
23 }
24
25 void Router::b_transport(tlm::tlm_generic_payload& trans,
26                          sc_core::sc_time& delay) {
27         Packet* packet = (Packet*)trans.get_data_ptr();
28         // count only data packet
29         if (packet->getType() == Packet::TYPE::DATA) {
30                 ++_totalPacketNumber;
31                 ++_packetPerSecond;
32         }
33         // route packet
34         int id = packet->getId ();
35         std::vector<int> dest = _routingTable[id];
36         for(std::vector<int>::iterator it = dest.begin (); it != dest.end
               (); it++) {
37                 initiatorSocketArray[*it]->b_transport(trans, delay);
38         }
39
40 }
41
42 Router::~Router() {
43         delete[] targetSocketArray;
44         delete[] initiatorSocketArray;
45 }
```

```
46
47 void Router::dataRateProcess() {
48         while (true) {
49                 wait(1, sc_core::SC_SEC);
50                 double tmp = _packetPerSecond * Packet::PAYLOAD;
51                 if (tmp > _dataRate) {
52                         _dataRate = tmp;
53                 }
54                 _packetPerSecond = 0;
55         }
56 }
57
58 double Router::getDataRate() { return _dataRate / 1024; }
59
60 int Router::getTotalPacketNumber() { return _totalPacketNumber; }
```

A.9 Modèle SystemC-AMS décodeur

Code A.28 – Description du décodeur

```
1  #ifndef RECEIVER_H
2  #define RECEIVER_H
3
4  #define SC_INCLUDE_DYNAMIC_PROCESSES
5
6  #include <systemc-ams>
7  #include <tlm>
8  #include <tlm_utils/simple_target_socket.h>
9  #include <Packet.h>
10 #include <map>
11 #include <NodeCfg.h>
12 #include <Clock.h>
13
14 /**
15  * @brief The Receiver class
16  * @author aravelomanantsoa@gmail.com
17  */
18 class Receiver : sc_core::sc_module {
19 public:
20         // TLM-2 socket, defaults to 32-bits wide, base protocol
21         tlm_utils::simple_target_socket<Receiver> targetSocket;
22
23         SC_HAS_PROCESS(Receiver);
24
25         /**
26          * @brief Receiver
27          * @param name
```

```
28          * @param id
29          * @param cfgMap
30          * @param M
31          */
32         Receiver(sc_core::sc_module_name name, int id, std::map<int,
               NodeCfg>& cfgMap,
33                  int M);
34
35         ~Receiver();
36
37         void b_transport(tlm::tlm_generic_payload& trans, sc_core::
               sc_time& delay);
38
39         void processThead(Packet& packet);
40
41         double convertData(NodeCfg& cfg, sc_dt::sc_lv<8>& highByte,
42                            sc_dt::sc_lv<8>& lowByte);
43
44         sc_core::sc_out<bool> end;
45
46         double getRecTime();
47
48 private:
49         std::map<int, std::ofstream*> _outStreamPacketMap;
50         std::map<int, std::ofstream*> _outStreamDataMap;
51         std::map<int, std::vector<double>*> _tmpDataMap;
52         int _id;
53         std::map<int, NodeCfg> _nodeCfgMap;
54         int _M;
55         double _recTime;
56 };
57
58 #endif  // RECEIVER_H
```

Code A.29 – Implémentation du décodeur

```
1  #include <Receiver.h>
2  #include <Recovery.h>
3
4  Receiver::Receiver(sc_core::sc_module_name name, int id,
5                     std::map<int, NodeCfg>& cfgMap, int M)
6          : sc_core::sc_module(name),
7            _id(id),
8            _nodeCfgMap(cfgMap),
9            _M(M),
10           _recTime(0.0) {
11
12         // register socket
```

```
13          targetSocket.register_b_transport(this, &Receiver::b_transport);
14
15          // create output stream
16          for (std::map<int, NodeCfg>::iterator itNode = _nodeCfgMap.begin
                ();
17               itNode != _nodeCfgMap.end(); ++itNode) {
18                  int nodeId = itNode->first;
19                  NodeCfg cfg = itNode->second;
20
21                  char fileNamePacket[40];
22                  char fileNameData[40];
23
24                  sprintf(fileNamePacket, "%d_node_packet_%d.txt", _id,
                        nodeId);
25                  sprintf(fileNameData, "%d_node_data_%d.txt", _id, nodeId)
                        ;
26
27                  _outStreamPacketMap[nodeId] = new std::ofstream(
                        fileNamePacket);
28                  _outStreamDataMap[nodeId] = new std::ofstream(
                        fileNameData);
29
30                  // available only for compressed data
31                  if (cfg._type_enc != NodeCfg::TYPE_ENC::NONE) {
32                          _tmpDataMap[nodeId] = new std::vector<double>;
33                  }
34          }
35
36          end.initialize(false);
37  }
38
39  Receiver::~Receiver() {}
40
41  void Receiver::b_transport(tlm::tlm_generic_payload& trans,
42                             sc_core::sc_time& delay) {
43          Packet* packet = (Packet*)trans.get_data_ptr();
44
45          // Create a new process
46          sc_core::sc_spawn(sc_bind(&Receiver::processThead, this, *packet)
                );
47
48          // Obliged to set response status to indicate successful
                completion
49          trans.set_response_status(tlm::TLM_OK_RESPONSE);
50  }
51
52  double Receiver::convertData(NodeCfg& cfg, sc_dt::sc_lv<8>& highByte,
```

```
53                              sc_dt::sc_lv<8>& lowByte) {
54          // concatenate bytes
55          sc_dt::sc_lv<16> data;
56          data.range(15, 8) = highByte;
57          data.range(7, 0) = lowByte;
58          // start data conversion
59          int rawData = data.to_uint();
60          double step =
61                          (cfg._v_ref_p - cfg._v_ref_n) / (std::pow(2, (int
                               )cfg._adc_res) - 1);
62          return rawData * step + cfg._v_ref_n;
63  }
64
65  void Receiver::processThead(Packet& packet) {
66          char msg[60];
67
68          const int id = packet.getId();
69          Packet::TYPE type = packet.getType();
70
71          std::map<int, NodeCfg>::iterator itNodeCfg = _nodeCfgMap.find(id)
                ;
72          if (itNodeCfg == _nodeCfgMap.end()) {
73                  std::sprintf(msg, "INVALID␣NODE␣ID␣%d", id);
74                  SC_REPORT_INFO("Decoder", msg);
75                  return;
76          }
77
78          switch (type) {
79
80          case Packet::TYPE::DATA: {
81                  std::sprintf(msg, "%s␣RECEIVE␣DATA␣PACKET␣FROM␣%d", this
                        ->name(), id);
82                  SC_REPORT_INFO("Decoder", msg);
83
84                  // output packet
85                  std::map<int, std::ofstream*>::iterator itStreamPacket =
86                                          _outStreamPacketMap.find(id);
87                  if (itStreamPacket == _outStreamPacketMap.end()) {
88                          std::sprintf(msg, "STREAM␣NOT␣FOUND␣FROM␣%d", id)
                                ;
89                          SC_REPORT_FATAL("Decoder", msg);
90                          return;
91                  }
92
93                  *(itStreamPacket->second) << packet << std::endl;
94
95                  // retrive output stream corresponding to the node id
```

```
96                  std::map<int, std::ofstream*>::iterator itStreamData =
97                                  _outStreamDataMap.find(id);
98                  if (itStreamData == _outStreamPacketMap.end()) {
99                          std::sprintf(msg, "STREAM NOT FOUND FROM %d", id)
                               ;
100                         SC_REPORT_FATAL("Decoder", msg);
101                         return;
102                 }
103
104                 NodeCfg cfg = itNodeCfg->second;
105
106                 if (cfg._type_enc == NodeCfg::TYPE_ENC::NONE) {
107                         // directly save data into file
108                         for (int i = 0; i < Packet::PAYLOAD; i = i + 2) {
109                                 // convert data
110                                 double data = convertData(cfg, packet(i),
                                        packet(i + 1));
111                                 data = data - cfg._offset;
112                                 // remove offset and save converted data
                                        into file
113                                 *(itStreamData->second) << data << std::
                                        endl;
114                         }
115                 } else {
116                         // the received packet contains encoded data
117                         std::map<int, std::vector<double>*>::iterator
                            itTmpData =
118                                                 _tmpDataMap.find(id);
119                         if (itTmpData == _tmpDataMap.end()) {
120                                 std::sprintf(msg, "DATA NOT FOUND FROM %d
                                        ", id);
121                                 SC_REPORT_FATAL("Decoder", msg);
122                                 return;
123                         }
124
125                         std::vector<double>* tmpEncodedSignal = itTmpData
                            ->second;
126
127                         for (int i = 0; i < Packet::PAYLOAD; i = i + 2) {
128                                 // convert data
129                                 double data = convertData(cfg, packet(i),
                                        packet(i + 1));
130                                 // save encoded data in temporary memory
131                                 tmpEncodedSignal->push_back(data);
132                                 // wait until M encoded data have been
                                        received before starting the
133                                 // recovery process
```

```
134                                 if (tmpEncodedSignal->size() == (unsigned
                                        )_M) {
135                                         std::sprintf(msg, "%s START
                                            RECOVERY FROM %d", this->name
                                            (), id);
136                                         SC_REPORT_INFO("Decoder", msg);
137
138                                         int N = cfg._ratio * _M;
139
140                                         cs::Matrix Phi(_M, N);
141                                         cs::Matrix Psi(N, N);
142                                         cs::Matrix A(_M, N);
143
144                                         // generate sparsifying matrix
145                                         cs::Recovery::buildIDctMatrix(Psi
                                            );
146                                         // generate measurement matrix
147                                         cs::Recovery::buildSensingMatrix(
                                            Phi);
148
149                                         A = Phi * Psi;
150
151                                         double recoveredSignal[N];
152                                         double recoveredTransform[N];
153                                         double y[_M];
154                                         for (int i = 0; i < _M; ++i) {
155                                                 y[i] = (*tmpEncodedSignal
                                                    )[i];
156                                         }
157
158                                         // start recovery
159                                         double rec = cs::Recovery::
                                            findSparseSolution(A, y,
                                            recoveredTransform);
160                                         if (rec > _recTime) {
161                                                 _recTime = rec;
162                                         }
163
164                                         // reconstruct signal
165                                         for (int i = 0; i < N; i++) {
166                                                 double tmpProd = 0.0;
167                                                 for (int k = 0; k < N; k
                                                    ++) {
168                                                         tmpProd += Psi(i,
                                                             k) *
                                                            recoveredTransform
                                                            [k];
```

```
169                                                 }
170                                                 recoveredSignal[i] =
                                                        tmpProd;
171                                             }
172
173                                             // remove offset and save
                                                    recovered data into file
174                                             for (int i = 0; i < N; ++i) {
175                                                 *(itStreamData->second)
                                                        << (recoveredSignal[i]
                                                            - cfg._offset)
176                                                                                 <<
                                                                                    std
                                                                                    ::
                                                                                    endl
                                                                                    ;
177                                             }
178
179                                             // clear temporary memory
180                                             tmpEncodedSignal->clear();
181                                         }
182                                     }
183                                 }
184
185                                 break;
186                         }
187
188                         case Packet::TYPE::STOP: {
189                                 std::sprintf(msg, "%s RECEIVE STOP PACKET FROM ID %d",
                                        this->name(), id);
190                                 SC_REPORT_INFO("Decoder", msg);
191                                 // remove the configuration data from the map
192                                 _nodeCfgMap.erase(itNodeCfg);
193                                 if (_nodeCfgMap.size() == 0) {
194                                         end.write(true);
195                                 }
196
197                                 break;
198                         }
199
200                         default:
201                                 break;
202                         }
203 }
204
```

```
205 double Receiver::getRecTime() { return _recTime; }
```

Publications personnelles

Revues internationales à comité de lecture

1) **Ravelomanantsoa A.**, Rabah H. et Rouane A., « Compressed Sensing : a Simple Deterministic Measurement Matrix and a Fast Recovery Algorithm ». In : Instrumentation and Measurement, IEEE Transactions on (2015). doi:10.1109/TIM.2015.2459471.

2) **Ravelomanantsoa A.**, Rabah H. et Rouane A., « Simple and Efficient Compressed Sensing Encoder for Wireless Body Area Network ». In : Instrumentation and Measurement, IEEE Transactions on 63.12 (2014), p. 2973-2982. issn : 0018-9456. doi: 10.1109/TIM.2014.2320393.

Conférences internationales avec actes et comité de lecture

1) **Ravelomanantsoa A.**, Rabah H. et Rouane A., « Fast and Efficient Signals Recovery for Deterministic Compressive Sensing : Applications to Biosignals ». In : Conference on Design & Architectures for Signal & Image Processing (DASIP) 2015.

2) **Ravelomanantsoa A.**, Rabah H. et Rouane A., « Simple Deterministic Measurement Matrix : Application to EMG Signals ». In : Microelectronics (ICM), 2014 26th International Conference on. 2014, p. 76 - 79. doi:10.1109/ICM.2014.7071810.

3) **Ravelomanantsoa A.**, Rabah H. et Rouane A., « SystemC-AMS based virtual prototyping of wireless body sensor network using compressed sensing ». In : Microelectronics (ICM), 2013 25th International Conference on. 2013, p. 1-4.
doi:10.1109/ICM.2013.6734992.

Conférences nationales avec actes et comité de lecture

1) **Ravelomanantsoa A.**, Rabah H. et Rouane A., « Design and Implementation of Compressed Sensing Encoder ». In : Colloque National du GDR SOCSIP. 2014.

2) **Ravelomanantsoa A.**, Rabah H. et Rouane A., « SystemC-AMS modeling of an acquisition and a reconstruction systems based on the compressed sensing theory ». In : Colloque National du GDR SOCSIP. 2013.

Bibliographie

[AALK12] Moshaddique Al Ameen, Jingwei Liu, and Kyungsup Kwak. Security and privacy issues in wireless sensor networks for healthcare applications. *Journal of Medical Systems*, 36(1) :93–101, 2012. URL : http : //www.springerlink.com.bases - doc.univ - lorraine.fr/content/rvl702055u398g24/abstract/, doi:10.1007/s10916-010-9449-4. 16

[ACD+10] E.G. Allstot, A.Y. Chen, A.M.R. Dixon, D. Gangopadhyay, and D.J. Allstot. Compressive sampling of ecg bio-signals : Quantization noise and sparsity considerations. In *Biomedical Circuits and Systems Conference (BioCAS), 2010 IEEE*, pages 41–44, Nov 2010. doi:10.1109/BIOCAS.2010.5709566. 51

[AM11] A. Amini and F. Marvasti. Deterministic construction of binary, bipolar, and ternary compressed sensing matrices. *IEEE Transactions on Information Theory*, 57(4) :2360–2370, 2011. doi:10.1109/TIT.2011.2111670. 50

[AMH75] N. Ahmed, Paul J. Milne, and Stanley G. Harris. Electrocardiographic data compression via orthogonal transforms. *Biomedical Engineering, IEEE Transactions on*, BME-22(6) :484–487, Nov 1975. doi : 10.1109/TBME.1975.324469. 71

[AT97] G. Antoniol and P. Tonella. Eeg data compression techniques. *Biomedical Engineering, IEEE Transactions on*, 44(2) :105–114, Feb 1997. doi : 10.1109/10.552239. 33

[Bar07] R.G. Baraniuk. Compressive sensing [lecture notes]. *IEEE Signal Processing Magazine*, 24(4) :118 –121, July 2007. doi:10.1109/MSP.2007.4286571. 25, 50

[Bar10] M. Barnasconi. Systemc ams extensions—solving the need for speed. In *Open SystemC Initiative whitepaper*, may 2010. 80

[BD07] T. Blumensath and M.E. Davies. *On the difference between orthogonal matching pursuit and orthogonal least squares.* TechReport, March 2007. 29

[BD09] Thomas Blumensath and Mike E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3) :265 – 274, 2009. URL : http : //www.sciencedirect.com/science/article/pii/S1063520309000384, doi:http://dx.doi.org/10.1016/j.acha.2009.04.002. 31

[BDE09] Alfred M. Bruckstein, David L. Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals

and images. *SIAM Rev.*, 51(1) :34–81, February 2009. URL : `http://dx.doi.org/10.1137/060657704`, `doi:10.1137/060657704`. 23

[BDM+05] M. Briere, E. Drouard, F. Mieyeville, D. Navarro, I. O'Connor, and F. Gaffiot. Heterogeneous modelling of an optical network-on-chip with systemc. In *Rapid System Prototyping, 2005. (RSP 2005). The 16th IEEE International Workshop on*, pages 10–16, June 2005. `doi:10.1109/RSP.2005.25`. 80

[BDMS13] AS. Bandeira, E. Dobriban, D.G. Mixon, and W.F. Sawin. Certifying the restricted isometry property is hard. *IEEE Transactions on Information Theory*, 59(6) :3448–3450, June 2013. `doi:10.1109/TIT.2013.2248414`. 26

[BEG+11] M. Barnasconi, K. Einwich, C. Grimm, T. Maehne, and A. Vachoux. Advancing the systemc analog/mixed-signal (ams) extensions introducing dynamic timed data flow. In *Open SystemC Initiative whitepaper*, sep 2011. 80

[BHEE10] Z. Ben-Haim, Y.C. Eldar, and M. Elad. Coherence-based performance guarantees for estimating a sparse vector under random noise. *IEEE Transactions on Signal Processing*, 58(10) :5030–5043, October 2010. `doi:10.1109/TSP.2010.2052460`. 26, 29

[Blu] Health wellness market | bluetooth technology website. URL : `http://www.bluetooth.com/Pages/Health-Wellness-Market.aspx`. 14

[BRK12] M. Balouchestani, K. Raahemifar, and S. Krishnan. Wireless body area networks with compressed sensing theory. In *2012 ICME International Conference on Complex Medical Engineering (CME)*, pages 364 –369, July 2012. `doi:10.1109/ICCME.2012.6275663`. 2, 18

[BT13] B. Bah and J. Tanner. Vanishingly sparse matrices and expander graphs, with application to compressed sensing. *Information Theory, IEEE Transactions on*, 59(11) :7491–7508, Nov 2013. `doi:10.1109/TIT.2013.2274267`. 50

[BTG+11] M. Bykowski, D. Tracey, B. Graham, N. Timmons, and J. Morrison. A schema for the selection of network topology for wireless body area networks. In *Radio and Wireless Symposium (RWS), 2011 IEEE*, pages 390–393, Jan 2011. `doi:10.1109/RWS.2011.5725490`. 13

[BWTJ11] Maged N. Kamel Boulos, Steve Wheeler, Carlos Tavares, and Ray Jones. How smartphones are changing the face of mobile and participatory healthcare : an overview, with example from eCAALYX. *Biomed. Eng. Online*, 10, April 2011. `doi:10.1186/1475-925X-10-24`. 16

[CC13] Cesar F. Caiafa and Andrzej Cichocki. Computing sparse representations of multidimensional signals using kronecker bases. *Neural Computation*, 25(1) :186–220, January 2013. 29

[CCS10] Fred Chen, Anantha P. Chandrakasan, and Vladimir Stojanovic. *A Signal-agnostic Compressed Sensing Acquisition System for Wireless and Implantable Sensors*. Ieee, New York, 2010. 18

[CCS12] F. Chen, A.P. Chandrakasan, and V.M. Stojanovic. Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors. *IEEE Journal of Solid-State Circuits*, 47(3) :744–756, 2012. doi:10.1109/JSSC.2011.2179451. ix, 36, 44, 47

[CDS98] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1) :33–61, January 1998. URL : http : //epubs.siam.org/doi/abs/10.1137/S1064827596304010, doi:10.1137/S1064827596304010. 28

[CMR+14] R. Cavallari, F. Martelli, R. Rosini, C. Buratti, and R. Verdone. A survey on wireless body area networks : Technologies and design challenges. *Communications Surveys Tutorials, IEEE*, 16(3) :1635–1657, Third 2014. doi: 10.1109/SURV.2014.012214.00007. 13

[CR07] Emmanuel Candès and Justin Romberg. Sparsity and incoherence in compressive sampling. *Inverse Problems*, 23(3) :969, 2007. URL : http: //stacks.iop.org/0266-5611/23/i=3/a=008. 27, 32

[CRT05] E. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *ArXiv Mathematics e-prints*, March 2005. 32

[CRT06] E.J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles : exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2) :489–509, February 2006. doi: 10.1109/TIT.2005.862083. 2, 18, 22

[CRV12] A.J. Casson and E. Rodriguez-Villegas. Signal agnostic compressive sensing for body area networks : Comparison of signal reconstructions. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4497–4500, August 2012. doi: 10.1109/EMBC.2012.6346966. 2

[CT05] E.J. Candès and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12) :4203 – 4215, December 2005. doi: 10.1109/TIT.2005.858979. 26, 28

[CT06] E.J. Candès and T. Tao. Near-optimal signal recovery from random projections : Universal encoding strategies ? *IEEE Transactions on Information Theory*, 52(12) :5406 –5425, December 2006. doi:10.1109/TIT.2006.885507. 28

[CW08] E.J. Candès and M.B. Wakin. An introduction to compressive sam-

pling. *IEEE Signal Processing Magazine*, 25(2) :21 –30, March 2008. doi : 10.1109/MSP.2007.914731. 22, 28

[CW11] T.T. Cai and Lie Wang. Orthogonal matching pursuit for sparse signal recovery with noise. *IEEE Transactions on Information Theory*, 57(7) :4680 –4688, July 2011. doi:10.1109/TIT.2011.2146090. 24, 27

[CXZ09] T.T. Cai, Guangwu Xu, and Jun Zhang. On recovery of sparse signals via l1 minimization. *Information Theory, IEEE Transactions on*, 55(7) :3388–3397, July 2009. doi:10.1109/TIT.2009.2021377. 27

[DAGA12] A.M.R. Dixon, E.G. Allstot, D. Gangopadhyay, and D.J. Allstot. Compressed sensing system considerations for ECG and EMG wireless biosensors. *IEEE Transactions on Biomedical Circuits and Systems*, 6(2) :156 –166, April 2012. doi:10.1109/TBCAS.2012.2193668. 2, 35

[DB08] Michael E. Davies and Thomas Blumensath. Faster & greedier : algorithms for sparse reconstruction of large datasets. In *in Proceedings of the third International Symposium on Communications, Control and Signal Processing (ISCCSP*, 2008. 29

[DE03] David L. Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. *Proceedings of the National Academy of Sciences*, 100(5) :2197–2202, 2003. URL : http : //www.pnas.org/content/100/5/2197.abstract, doi:10.1073/pnas.0437847100. 25

[DE11] M.F. Duarte and Y.C. Eldar. Structured compressed sensing : From theory to applications. *Signal Processing, IEEE Transactions on*, 59(9) :4053–4085, Sept 2011. doi:10.1109/TSP.2011.2161982. 25, 27

[DET06] D.L. Donoho, M. Elad, and V.N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1) :6 – 18, January 2006. doi:10.1109/TIT.2005.860430. 29

[DGNT08] Thong T. Do, Lu Gan, Nam Nguyen, and Trac D. Tran. Sparsity adaptive matching pursuit algorithm for practical compressed sensing. In M. B. Matthews, editor, *2008 42nd Asilomar Conference on Signals, Systems and Computers, Vols 1-4*, pages 581–587. Ieee, New York, 2008. 29

[DMNC11] Wan Du, Fabien Mieyeville, David Navarro, and IanO Connor. Idea1 : A validated systemc-based system-level design and simulation environment for wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2011(1), 2011. URL : http://dx.doi.org/10.1186/1687-1499-2011-143, doi:10.1186/1687-1499-2011-143. 81

[Don06] D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4) :1289–1306, April 2006. doi:10.1109/TIT.2006.871582. 2, 18, 22

[DTDS12] D.L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck. Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 58(2) :1094–1121, Feb 2012. doi:10.1109/TIT.2011.2173241. 31

[DW10] M.A. Davenport and M.B. Wakin. Analysis of orthogonal matching pursuit using the restricted isometry property. *Information Theory, IEEE Transactions on*, 56(9) :4395–4401, Sept 2010. doi:10.1109/TIT.2010.2054653. 32

[Ere05] Halit Eren. *Wireless Sensors and Instruments : Networks, Design, and Applications*. CRC Press, November 2005. ix, 37

[FW14] Simon Fauvel and Rabab K. Ward. An energy efficient compressed sensing framework for the compression of electroencephalogram signals. *Sensors (Basel, Switzerland)*, 14(1) :1474–1496, January 2014. PMID : 24434840 PMCID : PMC3926621. URL : http : //www.ncbi.nlm.nih.gov/pmc/articles/PMC3926621/, doi : 10.3390/s140101474. 2, 17, 18

[FWL+12] Rong Fan, Qun Wan, Yipeng Liu, Hui Chen, and Xiao Zhang. Complex orthogonal matching pursuit and its exact recovery conditions. *arXiv :1206.2197*, June 2012. URL : http://arxiv.org/abs/1206.2197. 31

[GAD+14] D. Gangopadhyay, E.G. Allstot, A.M.R. Dixon, K. Natarajan, S. Gupta, and D.J. Allstot. Compressed sensing analog front-end for bio-sensor applications. *IEEE Journal of Solid-State Circuits*, 49(2) :426–438, 2014. doi: 10.1109/JSSC.2013.2284673. ix, xiii, 15, 41, 42, 47, 55

[GM97] A.P. Guerrero and C. Mailhes. On the choice of an electromyogram data compression method. In *Engineering in Medicine and Biology Society, 1997. Proceedings of the 19th Annual International Conference of the IEEE*, volume 4, pages 1558–1561 vol.4, Oct 1997. doi:10.1109/IEMBS.1997.757009. 71

[GP11] Sal Anand Gopalan and Jong-Tae Park. Energy-efficient MAC protocols for wireless body area networks : A survey. *China Communications*, 8(5) :1–10, September 2011. 17

[GV06] R. Gribonval and P. Vandergheynst. On the exponential convergence of matching pursuits in quasi-incoherent dictionaries. *Information Theory, IEEE Transactions on*, 52(1) :255–261, Jan 2006. doi:10.1109/TIT.2005.860474. 30

[HOH10] Zaixing He, T. Ogawa, and M. Haseyama. The simplest measurement matrix for compressed sensing of natural images. In *2010 17th IEEE Interna-*

tional Conference on Image Processing (ICIP), pages 4301–4304, 2010. doi:10.1109/ICIP.2010.5651800. 50, 52, 53, 55

[HPB+09] M.A. Hanson, H.C. Powell, A.T. Barth, K. Ringgenberg, B.H. Calhoun, J.H. Aylor, and J. Lach. Body area sensor networks : Challenges and opportunities. *Computer*, 42(1) :58–65, Jan 2009. doi:10.1109/MC.2009.5. ix, 8, 15

[HTCP09] D.C. Hoang, Y.K. Tan, H.B. Chng, and S.K. Panda. Thermal energy harvesting from human warmth for wireless body area network in medical healthcare system. In *Power Electronics and Drive Systems, 2009. PEDS 2009. International Conference on*, pages 1277–1282, Nov 2009. doi:10.1109/PEDS.2009.5385814. 16

[ICRV14] S.A. Imtiaz, A.J. Casson, and E. Rodriguez-Villegas. Compression in wearable sensor nodes : Impacts of node topology. *IEEE Transactions on Biomedical Engineering*, 61(4) :1080–1090, April 2014. doi:10.1109/TBME.2013.2293916. 43, 47

[JMBW15] Liu Jing, Li Ming, Yuan Bin, and Liu Wenlong. A novel energy efficient mac protocol for wireless body area network. *Communications, China*, 12(2) :11–20, Feb 2015. doi:10.1109/CC.2015.7084398. 17

[KLW+06] S. Kirolos, J. Laska, M. Wakin, M. Duarte, D. Baron, T. Ragheb, Y. Massoud, and R. Baraniuk. Analog-to-information conversion via random demodulation. In *Design, Applications, Integration and Software, 2006 IEEE Dallas/-CAS Workshop on*, pages 71–74, Oct 2006. doi:10.1109/DCAS.2006.321036. 37

[KR08] Stefan Kunis and Holger Rauhut. Random sampling of sparse trigonometric polynomials, orthogonal matching pursuit versus basis pursuit. *Found. Comput. Math.*, 8(6) :737–763, November 2008. URL : http://dx.doi.org/10.1007/s10208-007-9005-x, doi:10.1007/s10208-007-9005-x. 29

[KY10] Jamil Y Khan and Mehmet R Yuce. Wireless body area network (wban) for medical applications. *New Developments in Biomedical Engineering. INTECH*, 2010. xiii, 9, 10

[KYBH12] Jamil Yusuf Khan, Mehmet R Yuce, Garrick Bulger, and Benjamin Harding. Wireless body area network (wban) design techniques and performance evaluation. *Journal of medical systems*, 36(3) :1441–1457, 2012. 7

[LB99] Hanwoo Lee and K.M. Buckley. Ecg data compression using cut and align beats approach and 2-d transforms. *Biomedical Engineering, IEEE Transactions on*, 46(5) :556–564, May 1999. doi:10.1109/10.759056. 33

[LBM+11] Benoît Latré, Bart Braem, Ingrid Moerman, Chris Blondia, and Piet Demeester. A survey on wireless body area networks. *Wireless Net-*

works, 17(1) :1–18, 2011. URL : http://www.springerlink.com.bases-doc.univ - lorraine.fr/content/bx44263752724622/abstract/, doi:10.1007/s11276-010-0252-4. xiii, 1, 6, 11, 14

[LDB14] A. Liberale, E. Dallago, and A.L. Barnabei. Energy harvesting system for wireless body sensor nodes. In *Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE*, pages 416–419, Oct 2014. doi:10.1109/BioCAS.2014.6981751. 17

[LG14] Shuxing Li and G. Ge. Deterministic sensing matrices arising from near orthogonal systems. *IEEE Transactions on Information Theory*, 60(4) :2291–2302, April 2014. doi:10.1109/TIT.2014.2303973. 28, 50

[LKD+07] J.N. Laska, S. Kirolos, M.F. Duarte, T.S. Ragheb, R.G. Baraniuk, and Y. Massoud. Theory and implementation of an analog-to-information converter using random demodulation. In *IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007*, pages 1959–1962, May 2007. doi:10.1109/ISCAS.2007.378360. 22, 38, 46

[LLR10] Ming Li, Wenjing Lou, and Kui Ren. Data security and privacy in wireless body area networks. *Wireless Communications, IEEE*, 17(1) :51–58, February 2010. doi:10.1109/MWC.2010.5416350. 16

[LLXJ12] Dandan Li, Xinji Liu, Shutao Xia, and Yong Jiang. A class of deterministic construction of binary compressed sensing matrices. *Journal of Electronics (China)*, 29(6) :493–500, 2012. URL : http://dx.doi.org/10.1007/s11767-012-0870-3, doi:10.1007/s11767-012-0870-3. 50

[LT1] LT1002 - dual, matched precision operational amplifier - linear technology. URL : http://www.linear.com/product/LT1002. 116

[LTS] Linear technology - design simulation and device models. URL : http://www.linear.com/designtools/software/. 90

[LZB13] Dejan E. Lazich, Henning Zoerlein, and Martin Bossert. Low coherence sensing matrices based on best spherical codes. In *Systems, Communication and Coding (SCC), Proceedings of 2013 9th International ITG Conference on*, pages 1–6, Jan 2013. 26, 28

[LZX+14] Benyuan Liu, Zhilin Zhang, Gary Xu, Hongqi Fan, and Qiang Fu. Energy efficient telemonitoring of physiological signals via compressed sensing : A fast algorithm and power consumption evaluation. *Biomedical Signal Processing and Control*, 11(0) :80 – 88, 2014. URL : http://www.sciencedirect.com/science/article/pii/S1746809414000366, doi:http://dx.doi.org/10.1016/j.bspc.2014.02.010. 18, 43, 48

[MAX] MAX312, MAX313, MAX314 10Ω, quad, SPST,

CMOS analog switches - overview. URL : `http://www.maximintegrated.com/datasheet/index.mvp/id/1063`. 116

[MBE] mbed NXP LPC1768 - handbook | mbed. URL : `http://mbed.org/handbook/mbed-NXP-LPC1768`. 115

[MKAV11] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst. Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes. *IEEE Transactions on Biomedical Engineering*, 58(9) :2456–2466, 2011. `doi:10.1109/TBME.2011.2156795`. 2, 18, 27, 43, 46, 50

[MMT14] M.M. Mohades, A Mohades, and A Tadaion. A reed-solomon code based measurement matrix with small coherence. *Signal Processing Letters, IEEE*, 21(7) :839–843, July 2014. `doi:10.1109/LSP.2014.2314281`. 25, 28

[MZ93] S.G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12) :3397–3415, Dec 1993. `doi:10.1109/78.258082`. 28, 29

[NT09] D. Needell and J.A. Tropp. Cosamp : Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3) :301 – 321, 2009. URL : `http://www.sciencedirect.com/science/article/pii/S1063520308000638`, `doi:http://dx.doi.org/10.1016/j.acha.2008.07.002`. 31

[PB10] A. Pantelopoulos and N.G. Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C : Applications and Reviews*, 40(1) :1–12, January 2010. `doi:10.1109/TSMCC.2009.2032660`. xiii, 9, 13, 14

[Phy] PhysioBank ATM. URL : `http://www.physionet.org/cgi-bin/atm/ATM`. ix, x, 33, 59, 62, 93

[PRK93] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal matching pursuit : recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44 vol.1, Nov 1993. `doi:10.1109/ACSSC.1993.342465`. 28, 30

[Rez13] Y.A. Reznik. Relationship between dct-ii, dct-vi, and dst-vii transforms. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 5642–5646, May 2013. `doi:10.1109/ICASSP.2013.6638744`. 33

[RF2] nRF24L01 - 2.4GHz RF - products - nordic semiconductor. URL : `http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01`. 101, 107, 110

[RLN+08] T. Ragheb, J.N. Laska, H. Nejati, S. Kirolos, R.G. Baraniuk, and Y. Massoud. A prototype hardware for random demodulation based compres-

sive analog-to-digital conversion. In *51st Midwest Symposium on Circuits and Systems, 2008. MWSCAS 2008*, pages 37–40, August 2008. doi : 10.1109/MWSCAS.2008.4616730. 38, 46

[RR11] S. Rein and M. Reisslein. Low-memory wavelet transforms for wireless sensor networks : A tutorial. *Communications Surveys Tutorials, IEEE*, 13(2) :291–307, Second 2011. doi:10.1109/SURV.2011.100110.00059. 34

[SBTL05] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed topology construction of bluetooth wireless personal area networks. *Selected Areas in Communications, IEEE Journal on*, 23(3) :633–643, March 2005. doi:10.1109/JSAC.2004.842567. 14

[SC] SystemC - accellera systems initiative. URL : http : //www.accellera.org/downloads/standards/systemc. 80

[SC12] B.L. Sturm and M.G. Christensen. Comparison of orthogonal matching pursuit implementations. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 220–224, Aug 2012. 29, 31

[SCA] Systemc-ams and design of embedded mixed-signal systems. URL : http: //www.systemc-ams.org/. 80

[Spa] SparseLab. URL : https://sparselab.stanford.edu/. 55

[SSB98] Bo Shen, IK. Sethi, and V. Bhaskaran. Dct convolution and its application in compressed domain. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(8) :947–952, Dec 1998. doi:10.1109/76.736723. 33

[SSW+05] H.S. Savci, A. Sula, Zheng Wang, N.S. Dogan, and E. Arvas. Mics transceivers : regulatory standards and applications [medical implant communications service]. In *SoutheastCon, 2005. Proceedings. IEEE*, pages 179–182, April 2005. doi:10.1109/SECON.2005.1423241. 13

[SW12] Heping Song and Guoli Wang. Sparse signal recovery via ECME thresholding pursuits. *Mathematical Problems in Engineering*, 2012 :1–22, 2012. URL : http : //www.hindawi.com.bases - doc.univ - lorraine.fr/journals/mpe/2012/478931/, doi : 10.1155/2012/478931. 28

[TEA] Tea (technologie ergonomie appliquées). URL : http://teaergo.com. 106

[TG07] J.A. Tropp and A.C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12) :4655 –4666, December 2007. doi:10.1109/TIT.2007.909108. 29, 31, 57, 65, 66

[TLD+10] J.A. Tropp, J.N. Laska, M.F. Duarte, J.K. Romberg, and R.G. Baraniuk. Beyond nyquist : Efficient sampling of sparse bandlimited signals. *IEEE*

Transactions on Information Theory, 56(1) :520 –544, January 2010. doi : 10.1109/TIT.2009.2034811. 38, 52, 53, 55

[TLM] SystemC TLM (Transaction-level Modeling). URL : http : //accellera.org/activities/working-groups/systemc-tlm. 80

[Tro04] J.A. Tropp. Greed is good : algorithmic results for sparse approximation. *Information Theory, IEEE Transactions on*, 50(10) :2231–2242, Oct 2004. doi:10.1109/TIT.2004.834793. 27, 29, 30, 31

[TW10] J.A. Tropp and S.J. Wright. Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE*, 98(6) :948–958, June 2010. doi:10.1109/JPROC.2010.2044010. 30

[UHB+12] Sana Ullah, Henry Higgins, Bart Braem, Benoit Latre, Chris Blondia, Ingrid Moerman, Shahnaz Saleem, Ziaur Rahman, and Kyung Sup Kwak. A comprehensive survey of wireless body area networks. *Journal of medical systems*, 36(3) :1065–1094, 2012. 1, 11

[WBB10] Yan Wang, A Bermak, and F. Boussaid. Fpga implementation of compressive sampling for sensor network applications. In *Quality Electronic Design (ASQED), 2010 2nd Asia Symposium on*, pages 5–8, Aug 2010. doi : 10.1109/ASQED.2010.5548167. 41, 46

[WBN+12] M. Wakin, S. Becker, E. Nakamura, M. Grant, E. Sovero, D. Ching, Juhwan Yoo, J. Romberg, A. Emami-Neyestanak, and E. Candès. A nonuniform sampler for wideband spectrally-sparse environments. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 2(3) :516–529, Sept 2012. doi : 10.1109/JETCAS.2012.2214635. 40

[Wel74] L. Welch. Lower bounds on the maximum cross correlation of signals (corresp.). *Information Theory, IEEE Transactions on*, 20(3) :397–399, May 1974. doi : 10.1109/TIT.1974.1055219. 26

[WKC+12] M. Wagner, B. Kuch, C. Cabrera, P. Enoksson, and A. Sieber. Android based body area network for the evaluation of medical parameters. In *2012 Proceedings of the Tenth Workshop on Intelligent Solutions in Embedded Systems (WISES)*, pages 33 –38, July 2012. 16

[WR04] D.P. Wipf and B.D. Rao. Sparse bayesian learning for basis selection. *Signal Processing, IEEE Transactions on*, 52(8) :2153–2164, Aug 2004. doi : 10.1109/TSP.2004.831016. 28

[WTF+12] Yue Wang, Zhi Tian, Chunyan Feng, Shulan Feng, and P. Zhang. Performance analysis of generalized block diagonal structured random matrices in compressive sensing. In *Communications and Information Technologies (ISCIT), 2012 International Symposium on*, pages 793–797, Oct 2012. doi : 10.1109/ISCIT.2012.6381010. 50

[XSG+14] Xiaoling Xu, Lei Shu, Mohsen Guizani, Mei Liu, and Junye Lu. A survey on energy harvesting and integrated data sharing in wireless body area networks. *International Journal of Distributed Sensor Networks*, 2014. 16

[YBM+12] Juhwan Yoo, S. Becker, M. Monge, M. Loh, E. Candès, and A. Emami-Neyestanak. Design and implementation of a fully integrated compressed-sensing signal acquisition system. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5325–5328, 2012. doi:10.1109/ICASSP.2012.6289123. ix, 38, 39, 46

[YL13] NamYul Yu and Ying Li. Deterministic construction of fourier-based compressed sensing matrices using an almost difference set. *EURASIP Journal on Advances in Signal Processing*, 2013(1), 2013. URL : http://dx.doi.org/10.1186/1687-6180-2013-155, doi:10.1186/1687-6180-2013-155. 50

[ZCK00] Y. Zigel, Arnon Cohen, and A. Katz. The weighted diagnostic distortion (wdd) measure for ecg signal compression. *Biomedical Engineering, IEEE Transactions on*, 47(11) :1422–1430, Nov 2000. doi:10.1109/TBME.2000.880093. xiii, 36

[Zig] ZigBee specification overview. URL : http://www.zigbee.org/Specifications/ZigBee/Overview.aspx. 13

[ZJMR13] Zhilin Zhang, Tzyy-Ping Jung, S. Makeig, and B.D. Rao. Compressed sensing of eeg for wireless telemonitoring with low energy consumption and inexpensive hardware. *Biomedical Engineering, IEEE Transactions on*, 60(1) :221–224, Jan 2013. doi:10.1109/TBME.2012.2217959. 2

[ZMRE11] L. Zelnik-Manor, K. Rosenblum, and Y.C. Eldar. Sensing matrix optimization for block-sparse decoding. *Signal Processing, IEEE Transactions on*, 59(9) :4300–4312, Sept 2011. doi:10.1109/TSP.2011.2159211. 27

[ZSM+14] Jie Zhang, Yuanming Suo, S. Mitra, S.P. Chin, S. Hsiao, R.F. Yazicioglu, T.D. Tran, and R. Etienne-Cummings. An efficient and compact compressed sensing microsystem for implantable neural recordings. *Biomedical Circuits and Systems, IEEE Transactions on*, 8(4) :485–496, Aug 2014. doi:10.1109/TBCAS.2013.2284254. 33

Résumé

Le réseau sans fil sur le corps humain ou « *wireless body area network (WBAN)* » est une nouvelle technologie de réseau sans fil dédié à la surveillance des paramètres physiologiques d'une personne. Le réseau est composé de dispositifs électroniques miniatures, appelés nœuds, disposés aux alentours ou à l'intérieur du corps humain. Chaque nœud est doté d'un ou plusieurs capteurs mesurant les paramètres physiologiques de la personne, comme l'électrocardiogramme ou bien la température du corps, et les caractéristiques de l'environnement qui l'entoure. Ces nœuds sont surtout soumis à une contrainte énergétique importante puisque la miniaturisation a réduit les dimensions de leurs batteries. Puisque les nœuds consomment la majorité de l'énergie pour transmettre les données, une solution pour diminuer leur consommation consisterait à compresser les données avant la transmission.

Les méthodes classiques de compression ne sont pas adaptées pour le WBAN particulièrement à cause de la puissance de calcul requise et la consommation qui en résulterait. Dans cette thèse, pour contourner ces problèmes, nous utilisons une méthode à base de l'acquisition comprimée pour compresser et reconstruire les données provenant des nœuds. Nous proposons un encodeur simple et facile à mettre en œuvre pour compresser les signaux. Nous présentons aussi un algorithme permettant de réduire la complexité de la phase de reconstruction des signaux. Un travail collaboratif avec l'entreprise TEA (Technologie Ergonomie Appliquées) nous a permis de valider expérimentalement une version numérique de l'encodeur et l'algorithme de reconstruction. Nous avons aussi développé et validé une version analogique de l'encodeur en utilisant des composants standards.

Mots-clés: WBAN, acquisition comprimée, encodeur, algorithme de reconstruction, SystemC-AMS, ECG, EMG, EEG.

Abstract

A wireless body area network (WBAN) is a new class of wireless networks dedicated to monitor human physiological parameters. It consists of small electronic devices, also called nodes, attached to or implanted in the human body. Each node comprises one or many sensors which measure physiological signals, such as electrocardiogram or body heat, and the characteristics of the surrounding environment. These nodes are mainly subject to a significant energy constraint due to the fact that the miniaturization has reduced the size of their batteries. A solution to minimize the energy consumption would be to compress the sensed data before wirelessly transmitting them. Indeed, research has shown that most of the available energy are consumed by the wireless transmitter.

Conventional compression methods are not suitable for WBANs because they involve a high computational power and increase the energy consumption. To overcome these limitations, we use compressed sensing (CS) to compress and recover the sensed data. We propose a simple and efficient encoder to compress the data. We also introduce a new algorithm to reduce the complexity of the recovery process. A partnership with TEA (Technologie Ergonomie Appliquées) company allowed us to experimentally evaluate the performance of the proposed method during which a numeric version of the encoder has been used. We also developed and validated an analog version of the encoder.

Keywords: WBAN, compressed sensing, encoder, recovery algorithm, SystemC-AMS, ECG, EMG, EEG.