E. N. S. T
المدرسة الوطنية العليا للتكنولوجيا
Ecole nationale supérieure de technologie

GEii
Génie Électrique et Infomatique Industrielle

الجمهورية الجزائرية الديمقراطية الشعبية

République Populaire Démocratique Algérienne

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Département : Génie Électrique Et Informatique Industrielle

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'ingénieur d'Etat en Electronique

Spécialité

Systèmes Embarqués

Implémentation de filtres adaptatifs sur DSP : Application aux signaux ECG

Réalisé par :

BOUKHATEM Cerine

NEDJAI Ayet Erahmene

Présenté et soutenu publiquement le 04/07/2023

Composition du Jury :

M	HARAOUBIA Brahim	Prof.	ENST	Président
Mme	BOUCHAMA Samira	M.C.B	ENST	Promotrice
M	CHERABIT Nouredine	M.A.A	ENST	Co-promoteur
M	BOUDIAR Toufik	M.C.A	ENST	Examinateur
Mme	BOUTARFA Souhila	M.C.B	ENST	Examinatrice

Année universitaire 2022–2023

Remerciements

Au terme de ce mémoire, nous adressons nos sincères remerciements tout particulièrement à :

. Madame BOUCHAMA Samira, notre encadrante pour son suivi et qui nous a consacré beaucoup de son temps afin de nous guider pour la réalisation de ce travail.

. Monsieur CHERABIT Nourdine qui nous a précieusement encadré et guidé, en nous apportant ses corrections et conseils.

. Dr. SEDDIKI Saber, cardiologue au centre cardiologue Birkhadem qui a bien voulu consacrer de son temps pour évaluer nos résultats et partager de son savoir avec nous.

Nous tenons également à remercier les membres de jury qui ont bien voulu examiner et évaluer ce travail, ainsi que le cadre administratif du département génie électrique et informatique industrielle de l'Ecole supérieure de technologie et l'ensemble des enseignants.

Enfin, nous souhaitons exprimer notre profonde gratitude envers Monisuer BOUKHATEM Abd El Rezak, ingénieur en éléctronique dont l'aide nous a été précieuse.

Table des matières

Liste des figures II
Liste des tableaux III
Liste des abréviations IV
Liste des symboles V

Introduction Générale **6**

I Filtrage des signaux numériques **7**
Introduction 8
I.1 Filtrage de signaux numériques 8
I.1.1 Filtres linéaires 8
I.1.1.1 Les filtres RII 8
I.1.1.2 Les filtres RIF 9
I.1.2 Filtres adaptatifs 9
I.1.2.1 Importance et applications du filtrage adaptatif 10
I.1.2.2 Identification 10
I.1.2.3 Modélisation inverse 11
I.1.2.4 Prédiction 11
I.1.2.5 Annulation d'interférence 12
I.2 Algorithmes du filtrage adaptatif 12
I.2.1 Filtre de Wiener 12
I.2.2 L'algorithme LMS 16
I.2.3 L'algorithme RLS 18
I.2.4 Autres Algorithmes de Filtrage Adaptatif 20
I.2.4.1 MCM normalisés : Normalized LMS 20
I.2.4.2 Moindres Carrés Moyens à pas variable : Variable Step-Size LMS 20
I.2.4.3 Algorithme de projection affine (APA) : Affine Projection Algorithm (APA) 21
Conclusion 21

II Les signaux ECG **22**
Introduction 23
II.1 Caractéristiques et Applications des Signaux ECG 23
II.1.1 Caractéristiques des Signaux ECG 23
II.1.1.1 Fréquence et régularité 24

II.1.1.2 Morphologie de l'onde P 24
II.1.1.3 Intervalle PR 25
II.1.1.4 Morphologie du complexe QRS 25
II.1.1.5 Morphologie du segment ST 25
II.1.1.6 Morphologie de l'onde T 26
II.1.1.7 Morphologie de l'onde U 26
II.1.1.8 Intervalle QTc 26
II.1.1.9 Rythme cardiaque 27
II.1.2 Applications des Signaux ECG 27
II.2 Bruits 28
II.2.1 Bruit d'interférence des lignes électriques (PLI) : 28
II.2.2 Errance de base (BW) 29
II.2.3 Bruit musculaire (EMG) 29
II.3 Traitement des signaux ECG 29
Conclusion 31

III La carte DSP TMS320c6713 32
Introduction 33
III.1 Présentation des DSP 33
III.1.1 Architecture Harvard modifiée 33
III.1.2 Les DSP à virgule flottante et à virgule fixe 34
III.1.2.1 Les DSP à virgule fixe 35
III.1.2.2 Les DSP à virgule flottante 35
III.2 Domaines d'applications 35
III.2.1 Traitement audio 35
III.2.2 Communications sans fil 35
III.2.3 Reconnaissance de la parole 35
III.2.4 Imagerie médicale 36
III.3 Architecture et fonctionnalités du DSKC6713 36
III.3.1 Module de traitement (CPU) 37
III.3.2 Mémoire 38
III.3.3 AIC23 Codec 39
III.3.4 Emulateur JTAG 40
III.3.5 Pipeline 40
III.4 Language de programmation et outil pour DSP 41
III.4.1 Language de programmation pour DSP 41
III.4.2 Outils de développement pour DSP 41
Conclusion 42

IV Implémentation du filtre sur la DSKc6713 43
Introduction 44
IV.1 Configuration de l'environnement de développement 44
IV.1.1 Interface et outils disponibles 44
IV.1.2 Memory Browser 46

IV.1.2.1 Présentation de la norme IEEE754 pour la représentation des nombres en virgule flottante . . . 46
IV.1.3 Création et affichage de graphiques dans CCS . . . 46
IV.1.4 Optimisation du code dans CCS . . . 47
IV.1.5 Configuration de la carte DSP6713 pour la réalisation de l'implémentation . . . 47
IV.2 Bases de Données . . . 49
IV.2.1 MIT-BIH Arrhythmia Database (mitdb) . . . 50
IV.2.2 ECG-ID Database (ecgiddb) . . . 50
IV.2.3 Utilisation des bases de donnée . . . 51
IV.3 Implémentation des filtres adaptatifs . . . 51
IV.3.1 Algorithmes de filtrage adaptatif utilisés . . . 51
IV.3.1.1 LMS . . . 51
IV.3.1.2 NLMS . . . 52
IV.3.1.3 RLS . . . 53
IV.3.2 Implémentation sur le DSP6713 . . . 54
IV.4 Résultats et Evaluation . . . 56
IV.4.1 Approches d'évaluation . . . 56
IV.4.2 Critères d'évaluation . . . 56
IV.4.3 Analyse temporelle . . . 57
IV.4.3.1 Erreur Quadratique Moyenne EQM . . . 57
IV.4.3.2 Ordre du filtre . . . 58
IV.4.3.3 Temps d'exécution . . . 59
IV.4.4 Analyse fréquentielle . . . 60
IV.4.5 Envoie des donnée vers le codec . . . 63
Conclusion . . . 64

Conclusion Générale **65**

Table des figures

I.1 Schéma d'un processus adaptatif 10
I.2 Système d'identification 11
I.3 Système de modélisation inverse 11
I.4 Système de prédiction 12
I.5 Système d'annulation d'interférence 12
I.6 Distribution uniforme du bruit 14
I.7 Distribution normale du bruit 14
I.8 Résultat du filtrage d'un signal bruité avec un bruit de distribution uniforme en utilisant un filtre de Wiener 15
I.9 Résultat du filtrage d'un signal bruité avec un bruit de distribution normale en utilisant un filtre de Wiener 15
I.10 Résultat du filtrage d'un signal bruité avec un bruit de distribution uniforme en utilisant un filtre de l'algorithme LMS. 17
I.11 Résultat du filtrage d'un signal bruité avec un bruit de distribution normale en utilisant un filtre de l'algorithme LMS. 18
I.12 Résultat du filtrage d'un signal bruité avec un bruit de distribution uniforme en utilisant un filtre de l'algorithme RLS. 19
I.13 Résultat du filtrage d'un signal bruité avec un bruit de distribution normale en utilisant un filtre de l'algorithme RLS. 20

II.1 Emplacements des électrodes des dérivations d'un signal ECG 23
II.2 Caractéristiques d'un signal ECG 24
II.3 Un signal ECG perturbé par une interférence de ligne électrique 28
II.4 Un signal ECG perturbé par l'érrance de base 29
II.5 Un signal ECG altéré par le bruit EMG 29
II.6 Traitement des Signaux ECG 30

III.1 Architecture : a) type Harvard modifiée, b) type Harvard. 34
III.2 Architecture de La TMS320C6713 37
III.3 Module de traitement (CPU) de la TMS320C6713 38
III.4 Memory Map, C6713 DSK 39
III.5 L'interface du codec sur la DSK TMS320C6713 40
III.6 Opération de Pipeline 41

IV.1 Interface de CCS 44
IV.2 La fenêtre de configuration du projet en CCS 45

IV.3 Le debogueur en CCS 46
IV.4 Accéder au menu Graph sur CCS 47
IV.5 Les Options d'inclusion dans le compilateur 47
IV.6 Les chemins dans l'éditeur de lien 48
IV.7 Configuration du cible 49
IV.8 Chaine de mesure 49
IV.9 Signal ECG désiré 50
IV.10Diagramme de flux de l'algorithme LMS 52
IV.11 Diagramme de flux de l'algorithme NLMS 53
IV.12Diagramme de flux de l'algorithme RLS 54
IV.13Compilation sans erreurs d'un projet sur ccs 55
IV.14dispositif matériel utilisé pour la mise en place de l'implémentation 55
IV.15Modèle final de suppression de bruit des signaux pris de la BDD mitdb . . 56
IV.16Modèle final de supression de bruit des signaux pris de la BDD ecgiddb . . 56
IV.17Signal ECG brute 58
IV.18Signal filtré tiré de la base de données 58
IV.19Signal ECG filtré par un filtre LMS d'ordre 37 58
IV.20Signal ECG filtré par un filtre LMS d'ordre 60 59
IV.21Sperctrogrammes du signal 109, le signal 109+bruit et le signal résultant après un filtrage LMS 62
IV.22Signal Person_09/rec_6 visualisé sur Zelscope après filtrage NLMS . . . 63

Liste des tableaux

III.1 Principales Caractéristiques du Processeur C6713 36

IV.1 Résultas du paramétre MSE pour l'algorithme LMS et NLMS pour ordre de filtre 25, 37 et 60 57

IV.2 Temps d'exécutions en cycle d'horloge et en secondes 59

Liste des abréviations

APA *Affine Projection Algorithm*

BW *Baseline Wander*

Bpm *Beat Per Minute*

CCS *Code Composer Studio*

CPU *Central Processing Unit*

DWT *Discrete Wavelet Transform*

DMA *Direct Memory Access*

DSP *Digital Signal Processor*

ECG *Électrocardiogramme*

EMG *Électromyogramme*

EQM *Erreur Quadratique Moyenne*

FIR *Réponse Impulsionnelle Finie*

JTAG *Joint Test Action Group*

LMS *Least Mean Square*

MCM *Moindre Carrés Moyens*

NLMS *Normalized Least Mean Square*

PLI *Power Line Interference*

PSN *Processeur de Signal Numérique*

RAM *Random Access Memory*

RII *Réponse Impulsionnelle Infinie*

RLS *Recursive Mean Square*

VLIW *Very Long Instruction Word*

VSLMS *Variable Step Least Mean Square*

Liste des symboles

x_k : valeurs successives du signal d'entrée

y_k : valeurs successives du signal de sortie

a_i, b_i : coefficients de la fonction de transfert du filtre

N, M : ordres du numérateur et du dénominateur de $H(Z)$, M est appelé l'ordre du filtre

z : variable complexe utilisée dans la fonction de transfert des filtres

$z_1, z_2, ..., z_N$: zéros de la fonction de transfert d'un filtre RII

$p_1, p_2, ..., p_M$: pôles de la fonction de transfert d'un filtre RII

$H(z)$: fonction de transfert d'un filtre RII

$d(n)$: signal désiré ou signal de sortie souhaité

$e(n)$: erreur entre le signal désiré et le signal de sortie réel

$w_0, w_1, ..., w_{N-1}$: poids du filtre dans l'algorithme

μ : paramètre de convergence dans l'algorithme LMS

R : valeur propre maximale de la matrice d'auto-corrélation dans l'algorithme LMS

α : vecteur d'innovation dans l'algorithme RLS

k : vecteur de gain dans l'algorithme RLS

W : vecteur de coefficients du filtre dans l'algorithme RLS

Q : matrice utilisée pour mettre à jour les coefficients dans l'algorithme RLS

ρ : paramètre dans l'algorithme VSLMS

$\eta(n)$: différence entre les poids mis à jour dans l'algorithme APA

ϵ : Paramètre de normalisation dans l'algorithme NLMS

λ : Facteur d'oubli dans l'algorithme RLS

Introduction Générale

Les maladies cardiovasculaires représentent un problème majeur de santé publique dans de nombreux pays, nécessitant un suivi régulier. Cependant, en raison du nombre insuffisant de spécialistes en cardiologie [1], il devient difficile d'assurer des soins adéquats à tous les patients. C'est là que la technologie peut jouer un rôle crucial en offrant de nouvelles possibilités, notamment la consultation à distance grâce à la télémédecine.

Dans le cadre des applications de systèmes embarqués dédiées à la télémédecine, les applications de classification automatique du signal électrocardiogramme (ECG) facilitent le diagnostic des maladies cardiovasculaires. Toutefois, les signnaux ECG sont souvent perturbés par du bruit généré par les techniques de mesure ou par des phénomènes physiologiques. Par conséquent, le filtrage de ces signaux devient essentiel pour extraire les informations pertinentes et assurer une interprétation correcte.

La non-stationnarité du signal ECG nous oriente vers l'utilisation des techniques de filtrage adaptatif. En utilisant des algorithmes d'adaptation, le filtre peut s'adapter aux changements du signal, garantissant ainsi une meilleure suppression des composantes indésirables tout en préservant les caractéristiques importantes de l'ECG.

La nature critique des maladies cardiaques rend nécessaire que le filtrage des signaux ECG soit effectué en temps réel, afin d'assurer une surveillance et une intervention rapides. Pour cela, l'utilisation des CPU dédié à ce type de calculs, tel que les processeurs de traitement de signal (DSP), est inévitable.

Dans le cadre de notre projet, nous avons entrepris une étude approfondie des filtres adaptatifs et du signal ECG. Afin d'évaluer leur performance sur ce signal, nous avons développé une implémentation des filtres sur le DSP C6713, en exploitant les bases de données disponibles. Parallèlement, nous avons effectué une analyse à la fois temporelle et spectrale pour obtenir une compréhension complète du sujet. Pour ce faire, nous avons organisé notre mémoire en quatre chapitres :

- Le premier chapitre aborde le filtrage des signaux numériques en présentant les algorithmes adaptatifs les plus utilisés.
- Le deuxième chapitre porte sur les signaux ECG et leurs caractéristiques, en mettant en évidence les types de bruit susceptibles de les affecter.
- Le troisième chapitre présente la carte utilisée, la DSK320C6713.
- Le dernier chapitre met en avant les résultats obtenus et propose des perspectives de recherche.

1. https ://www.who.int/fr.

Chapitre I

Filtrage des signaux numériques

Introduction

Dans une chaîne de traitement de signal, l'opération du filtrage est nécessaire pour éliminer ou réduire les signaux indésirables. Ce chapitre donne une perspective globale sur le filtrage des signaux numériques. Pour ce faire, nous commencerons par présenter les filtres linéaires, puis nous aborderons les filtres adaptatifs et leurs multiples applications. Enfin, le chapitre se conclura avec l'étude et la mise en œuvre des différents algorithmes de filtrage adaptatif.

I.1 Filtrage de signaux numériques

Le terme « filtre numérique » désigne un système linéaire permanent qui opère en temps discret et dont le but est de modifier de manière précise la distribution fréquentielle (spectrale) d'un signal numérique d'entrée afin d'obtenir le signal numérique de sortie avec la modification souhaitée. Les filtres numériques peuvent être classés en deux grandes catégories : les filtres linéaires et les filtres adaptatifs.

I.1.1 Filtres linéaires

Les filtres linéaires sont des techniques spécifiques d'interpolation, il existe deux types de tels filtres : le RII et le RIF.

I.1.1.1 Les filtres RII

Un filtre à réponse impulsionnelle infinie (RII, pour Infinite Impulse Response en anglais) est un type de filtre numérique dont la réponse impulsionnelle a une durée infinie.

Contrairement aux filtres à réponse impulsionnelle finie (FIR), les filtres IIR peuvent rétroagir la sortie filtrée vers leur propre entrée, ce qui leur permet de produire une réponse plus complexe et souvent avec une réponse en fréquence plus sélective [1]. Ces filtres sont caractérisés par l'utilisation de coefficients récurrents dans leur équation de différence I.1. Cette rétroaction permet aux filtres IIR d'avoir une mémoire interne, ce qui peut être avantageux dans certaines applications de filtrage.

Les filtres IIR sont largement utilisés dans le domaine du traitement du signal pour des applications telles que la conception de filtres passe-bas, passe-haut, passe-bande, ainsi que pour des tâches plus complexes telles que la compression audio ou la réduction du bruit. Cependant, en raison de leur nature récursive, les filtres IIR peuvent être plus sensibles aux instabilités numériques. Ce type de filtre est peu coûteux et d'une latence faible, caractéristique qui le rend idéal pour des applications embarquées. L'équation aux différences d'un tel filtre est donnée selon l'équation I.1 [1].

$$y_k = \sum_{i=0}^{N} a_i.x_{k-i} - \sum_{j=1}^{N} b_j.y_{k-j} \tag{I.1}$$

On appelle :
x_k : valeurs successives du signal d'entrée
a_i, b_i : coefficients de la fonction de transfert du filtre
y_k : valeurs successives du signal de sortie
N, M : ordres du numérateur et du dénominateur de H(Z), M est appelé l'ordre du filtre.

La fonction de transfert d'un filtre IIR est donnée selon l'équation I.2 :

$$H(z) = \frac{\sum_{i=0}^{N} a_i.z^{-i}}{\sum_{j=0}^{M} b_j.z^{-j}} \tag{I.2}$$

Une factorisation de l'équation I.2 donne l'équation I.3 :

$$H(z) = \alpha \frac{(z - z_1)(z - z_2)...(z - z_N)}{(z - p_1)(z - p_2)...(z - p_M)} \tag{I.3}$$

Où : z1, z2,..., zn sont les zéros, p1, p2,..., pn sont les pôles.

On dit qu'un filtre RII est spécifié par les valeurs des pôles et des zéros de H(z) (au coefficient de gain α près).

I.1.1.2 Les filtres RIF

Un filtre RIF (Réponse impulsionnelle finie) est un type de filtre numérique dont la réponse impulsionnelle est de durée finie. Il est caractérisé par l'utilisation d'une séquence finie de coefficients pour effectuer la convolution linéaire entre le signal d'entrée et ces coefficients. La réponse fréquentielle d'un filtre FIR est déterminée par les valeurs des coefficients utilisés [2]. Ils peuvent être conçus pour réaliser diverses opérations de filtrage, telles que la suppression de certaines fréquences, le passage de certaines fréquences, la réduction du bruit, etc.

Ces filtres sont favorisés pour les applications où la phase linéaire est importante et où une quantité décente de mémoire et de performance de calcul est disponible. L'équation d'un filtre FIR est définit à partir de l'équation I.1, pour laquelle tous les coefficients b_i sont nuls et $b_0 = 1$. L'équation est donc définit par I.4 [2].

$$y_i = \sum_{i=0}^{N} a_i.x_{k-i} \tag{I.4}$$

I.1.2 Filtres adaptatifs

Le filtrage adaptatif est une technique de traitement numérique de signal qui utilise un filtre adaptatif, dont les coefficients sont ajustés en temps réel pour s'adapter aux caractéristiques du signal d'entrée afin de minimiser la différence entre la sortie réelle et la sortie désirée, extraire le signal souhaité et par conséquent minimiser le bruit. La figure I.1 représente le schéma général d'un processus adaptatif.

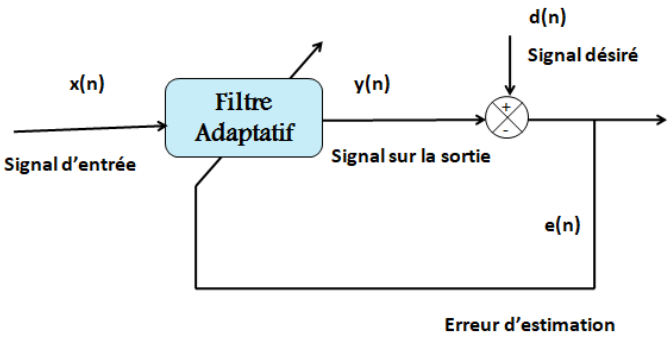


Figure I.1 – Schéma d'un processus adaptatif

I.1.2.1 Importance et applications du filtrage adaptatif

La contamination d'un signal par d'autres signaux indésirables, souvent plus importants, est un problème rencontré couramment dans de nombreuses applications. Dans ces cas, il est nécessaire que les caractéristiques du filtre soient variables ou qu'elles soient modifiées de façon intelligente. D'où la nécessité du filtrage adaptatif [3].

Le filtrage adaptatif est utilisé dans les cas suivants [4] [5] :

— Pour s'adapter aux conditions qui changent en temps réel.

— Lorsqu'il y a une superposition de bande entre le signal désiré et le bruit Si la bande occupée par le bruit est inconnue ou si elle varie avec le temps.

Le filtrage adaptatif présente une large gamme d'applications dans le traitement du signal, les systèmes de communication et les systèmes de contrôle. On distingue quatre catégories principales

I.1.2.2 Identification

Dans les applications traitant de l'identification, un filtre adaptatif est utilisé pour fournir un modèle linéaire qui représente le meilleur ajustement à un processus inconnu. Seuls les signaux d'entrée/sortie sont connus. Les paramètres de filtre seront estimés et la sortie du processus sera considérée comme la sortie souhaitée du filtre adaptatif [6]. La figure I.2 présente le schéma général d'un système d'identification.

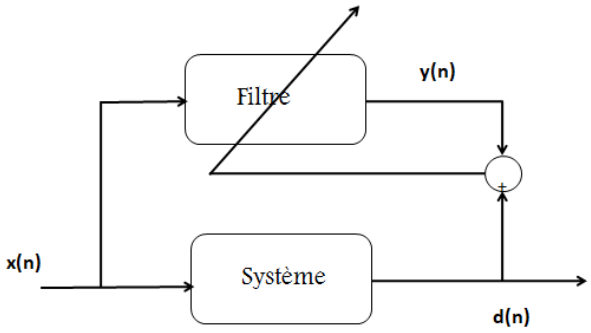


Figure I.2 – Système d'identification

I.1.2.3 Modélisation inverse

Dans cette deuxième classe d'applications, la fonction est d'estimer un système ou un filtre inverse qui peut être utilisé pour annuler l'effet d'un système ou d'un filtre connu. En d'autres termes, étant donné un signal d'entrée connu et le signal de sortie produit par un système ou un filtre, on estime le système ou le filtre inverse qui peut être utilisé pour produire le signal d'entrée à partir du signal de sortie[6]. La figure I.3 présente le principe de modélisation inverse.

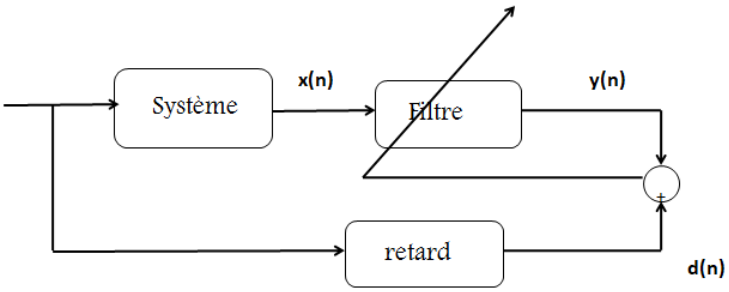


Figure I.3 – Système de modélisation inverse

I.1.2.4 Prédiction

Dans ce cas, la fonction du filtre adaptatif est de fournir la meilleure prédiction de la valeur future d'un signal aléatoire.

Les données d'entrée correspondent aux valeurs précédentes du signal, tandis que les données de sortie correspondent aux valeurs futures du signal[6]. Le principe de la prédiction est résumé sur la figure I.4.

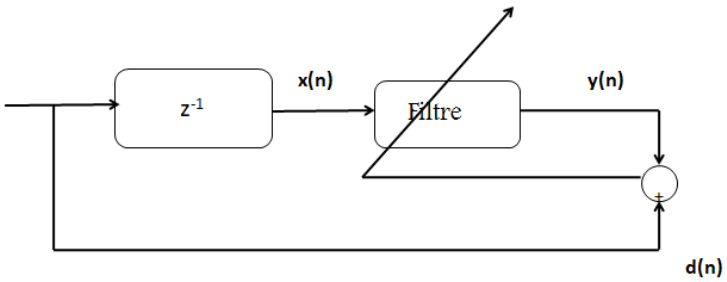


Figure I.4 – Système de prédiction

I.1.2.5 Annulation d'interférence

Dans cette dernière classe d'applications, le filtre adaptatif est utilisé pour annuler les interférences inconnues contenues dans un signal primaire. Ce dernier sera considéré comme la sortie désirée du filtre adaptatif. Un signal de référence est utilisé comme entrée[6]. La figure I.5 présente le principe d'annulation d'interférence.

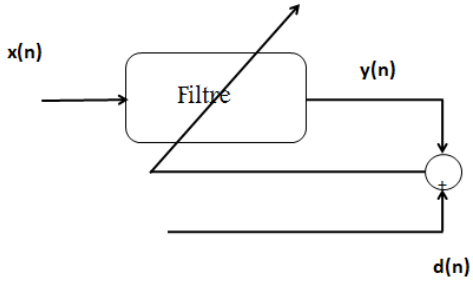


Figure I.5 – Système d'annulation d'interférence

I.2 Algorithmes du filtrage adaptatif

Plusieurs algorithmes peuvent être utilisés pour le filtrage adaptatif. Considérés comme des extensions du filtre de Wiener [7], il apparaît mieux de le présenter en premier lieu avant d'aborder ces derniers.

I.2.1 Filtre de Wiener

Le filtre de Wiener, nommé d'après Norbert Wiener, est un filtre linéaire couramment utilisé en traitement du signal pour supprimer le bruit d'un signal ou pour estimer un signal à partir de mesures bruitées [8].

La sortie $y(n)$ est une somme de ses N entrées pondérées. Voir l'équation (I.5)

$$y(n) = w_0 x(n) + w_1 x(n-1) + ... + w_{N-1} x(n-N+1) \tag{I.5}$$

On considère le cas d'un seul poids $y(n) = wx(n)$ et on note le signal d'erreur

$$e(n) = d(n) - wx(n) \tag{I.6}$$

Pour obtenir le poids optimal w^* , le carré du terme d'erreur est calculé. Le résultat est noté sur l'équation (I.7)

$$e^2(n) = (d(n) - wx(n))^2 = d^2(n) - 2d(n)wx(n) + w^2x^2(n) \tag{I.7}$$

L'espérance statistique s'écrit :

$$E(e^2(n)) = E(d^2(n)) - 2wE(d(n) * x(n)) + w^2E(x^2(n)) \tag{I.8}$$

En utilisant les notations de statistiques , On définit :

L'erreur Quadratique Moyenne :	$J = E(e^2(n)) = MSE$
$E(d^2(n)) = \sigma^2$;	La Puissance du Signal Bruité
$E(d(n)x(n)) = P$;	La corrélation entre $d(n)$ et $x(n)$
$E(x^2(n)) = R$;	L'auto-corrélation de $x(n)$

Pour N largement suffisant, l'équation (I.8) se réduit à :

$$J = \sigma^2 - 2wP + w^2R \tag{I.9}$$

Comme σ^2, P et R sont des constantes, J est une fonction quadratique de w et le poids optimal w^* est atteint lorsque le minimum de MSE J_{min} est atteint.

Pour obtenir le poids optimal, J est dérivé par rapport à w :

$$\frac{dJ}{dw} = -2P + 2wR = 0 \tag{I.10}$$

$$w^* = R^{-1}P \tag{I.11}$$

Nous avons réalisé une implémentation de ces équations en utilisant Matlab pour les appliquer à un signal sinusoïdal d'une durée de 500 millisecondes et d'une fréquence de 5 Hz, qui a été perturbé par deux types de bruits.

Le bruit utilisé se compose d'échantillons ayant une moyenne nulle et une variance unitaire, présentant des propriétés similaires à celles d'un bruit gaussien blanc. Deux distributions sont traitées, une distribution normale standard et une distribution uniforme.

Les figures I.6 et I.7 présentent respectivement la distribution uniforme et la distribution normale de tels bruits.

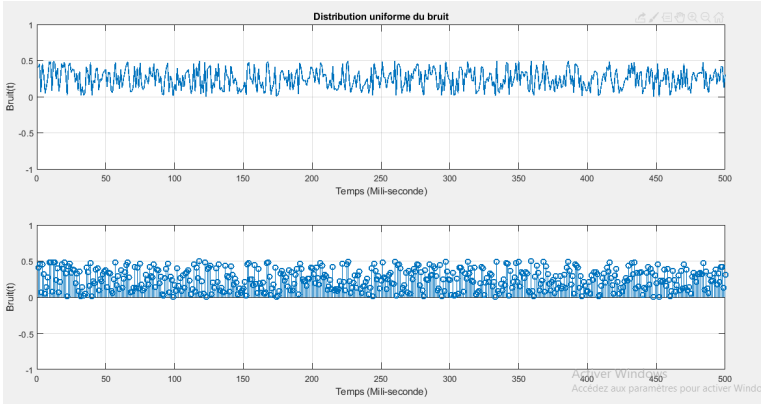


Figure I.6 – Distribution uniforme du bruit

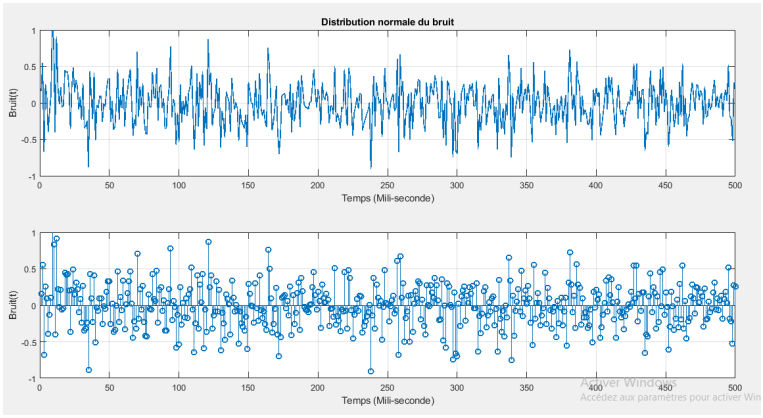


Figure I.7 – Distribution normale du bruit

D'après les résultats obtenus, on remarque que le bruit normal suit une distribution en forme de cloche, tandis que le bruit de distribution uniforme a une répartition constante sur la plage de données.

Les figures I.8 et I.9 montrent le résultat du filtrage d'un signal bruité avec un filtre de Wiener. La figure I.8 correspond à un signal avec un bruit de distribution uniforme, tandis que la figure I.9 correspond à un signal avec un bruit de distribution normale.

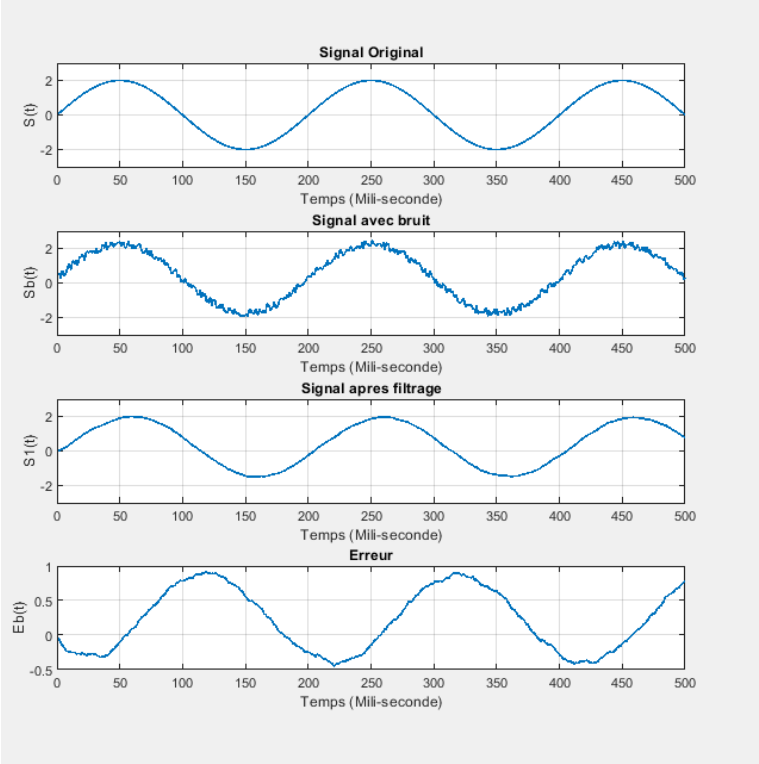


Figure I.8 – Résultat du filtrage d'un signal bruité avec un bruit de distribution uniforme en utilisant un filtre de Wiener

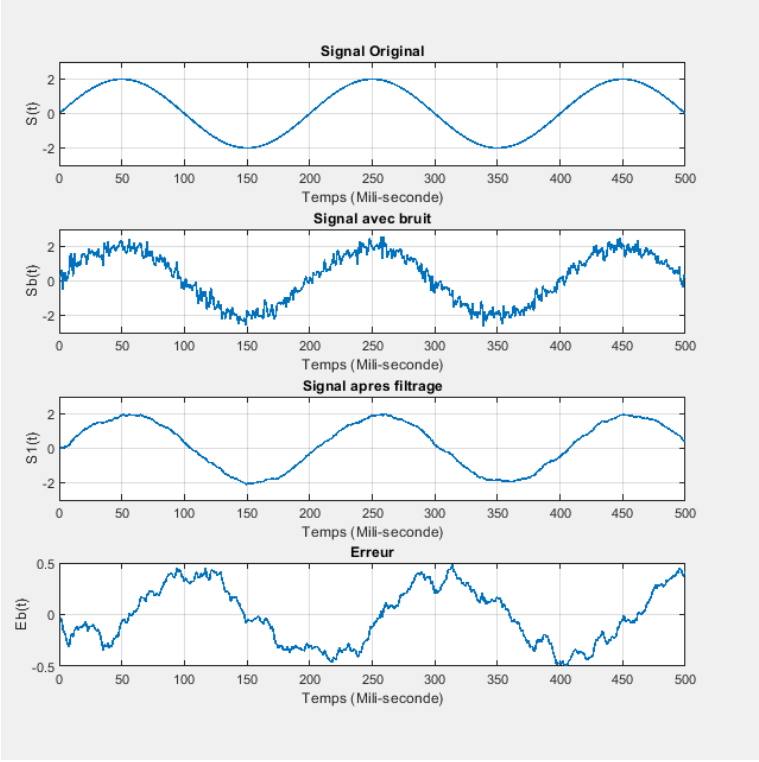


Figure I.9 – Résultat du filtrage d'un signal bruité avec un bruit de distribution normale en utilisant un filtre de Wiener

Le filtre de Winner est conçu pour atténuer les effets indésirables du bruit et améliorer la qualité du signal d'intérêt. Cela se fait en exploitant les caractéristiques statistiques du bruit pour l'estimation et la suppression [9].

I.2.2 L'algorithme LMS

Least Mean Square , ou méthode de moindres carrées a été présenté pour la première fois en 1960. Cet algorithme est le plus utilisé en filtrage adaptatif pour une large gamme d'applications grâce à sa simplicité et robustesse [10].

Le principe derrière cet algorithme est d'ajuster itérativement les poids d'une combinaison linéaire, afin de minimiser l'erreur entre la sortie réelle et la sortie désirée[8].

La sortie $y(n)$ est une somme de ses N entrées pondérées

$$y(n) = w_0x(n) + w_1x(n-1) + ... + w_{N-1}x(n-N+1) \tag{I.12}$$

L'objectif est de trouver les poids qui minimisent l'erreur quadratique moyenne (MSE) entre la sortie du filtre $y(n)$ et la sortie désirée $d(n)$ en mettant à jour les poids à chaque itération par l'équation suivante :

$$w_k(n+1) = w_k(n) + 2\mu e(n)x(n) \tag{I.13}$$

Où μ est le paramètre de convergence , il doit satisfaire $0 < \mu < (\frac{1}{R_{(max)}})$
R est la valeur propre maximale de la matrice d'auto-corrélation $E(x^2(n))$

$$0 < \mu < (\frac{1}{R}) \tag{I.14}$$

Les étapes suivantes sont requises pour appliquer l'algorithme [4] :

1. Initialiser $w_1, w_2, ..., w_{N-1}$ par des valeurs aléatoires.

2. Lire d(n) , x(n) et réaliser le filtrage numérique.

$$y(n) = w_0x(n) + w_1x(n-1) + ... + w_{N-1}x(n-N+1) \tag{I.15}$$

3. Calculer l'erreur sur la sortie.

$$e(n) = d(n) - y(n) \tag{I.16}$$

4. Mettre à jour les coefficients du filtre en utilisant la formule (I.13). Cet algorithme est obtenu par la minimisation de la fonction d'erreur quadratique moyenne.

$$MSE = J = E(e^2(n)) \tag{I.17}$$

En remplaçant $e(n) = d(n) - y(n)$ et $y(n) = w(n) * x(n)$ puis dérivant par rapport

à chaque poids :

$$\frac{dJ}{dw_k(n)} = 2E(x(n) * (d(n) - w(n) * x(n))) \tag{I.18}$$

Si $\frac{dJ}{dw(n)} = 0$ est résolut pour w(n) :

$$w_k(n+1) = w_k(n) + 2\mu e(n)x(n) \tag{I.19}$$

Nous avons implémenté sur Matlab ces équations pour les appliquer à un signal sinusoïdal d'une durée de 500 millisecondes et d'une fréquence de 5 Hz bruité par les deux bruits utilisés précédemment.

Les figures I.10 et I.11 illustrent le résultat du filtrage d'un signal bruité avec un filtre de l'algorithme LMS. La figure I.10 correspond à un signal avec un bruit de distribution uniforme, tandis que la figure I.11 correspond à un signal avec un bruit de distribution normale.

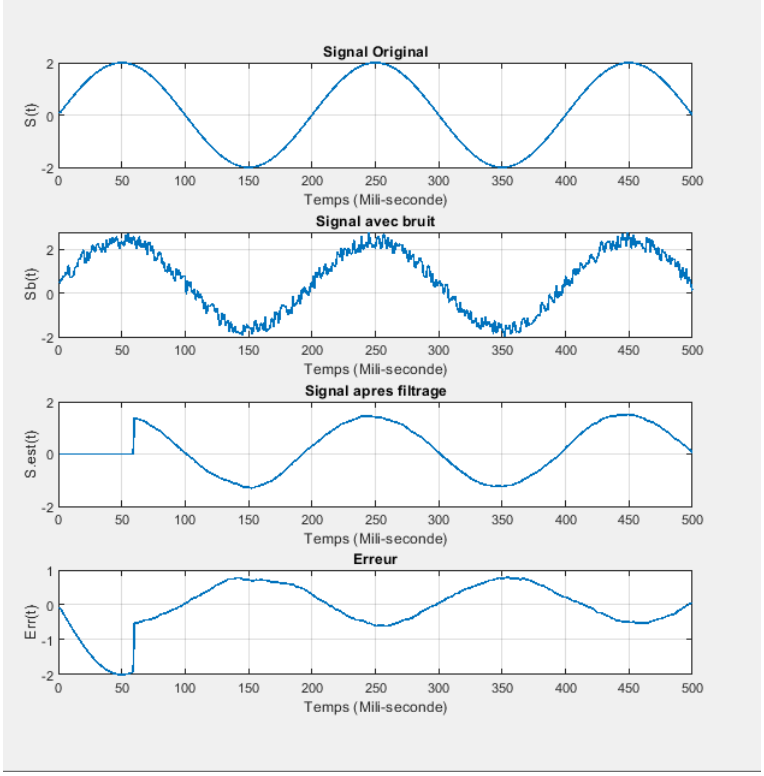


Figure I.10 – Résultat du filtrage d'un signal bruité avec un bruit de distribution uniforme en utilisant un filtre de l'algorithme LMS.

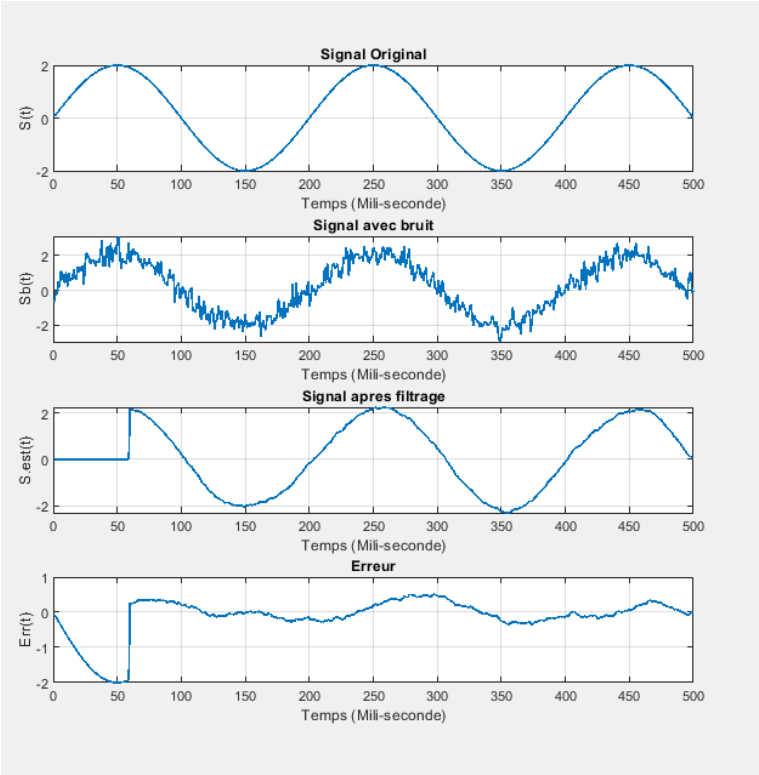


Figure I.11 – Résultat du filtrage d'un signal bruité avec un bruit de distribution normale en utilisant un filtre de l'algorithme LMS.

L'algorithme LMS parvient à réduire le bruit et à améliorer la qualité du signal d'intérêt. La première partie nulle du signal filtré est due à la période d'adaptation nécessaire au filtre pour ajuster ses paramètres en fonction du signal d'entrée. Une fois que la taille du filtre est atteinte, la sortie commence à converger et suit le signal d'entrée de manière plus précise.

I.2.3 L'algorithme RLS

Récursive mean squares , ou méthode des moindres carrés récursifs, est une variante de l'algorithme LMS, qui est utilisée dans plusieurs domaines, comme le traitement de signal, le contrôle et l'apprentissage automatique.

Le principe est de minimiser la somme des carrés de la différence entre le signal désiré et le signal de sortie du filtre utilisant les nouveaux échantillons du signal entrant. L'algorithme RLS calcule les coefficients sous une forme récursive à chaque itération.
Les étapes suivantes sont requises pour appliquer l'algorithme [8] :

1. Initialiser $W = [w_1, w_2, ..., w_{N-1}]$ par des valeurs aléatoires et $Q = PI$, I étant la matrice et P étant l'inverse de la puissance du signal d'entrée.
2. Calculer les vecteurs d'innovation α et de gain k

$$\alpha = d(n) - X^T(n)W \tag{I.20}$$

$$k = \frac{QX(n)}{\lambda + X^T(n)QX(n)} \tag{I.21}$$

3. Mettre à jour les coefficients du filtre

$$W = W + k\alpha \tag{I.22}$$

4. Mettre à jour la matrice Q , produit la sortie puis calculer l'erreur

$$Q = [Q - kX^T(n)Q]/\lambda \tag{I.23}$$

$$y(n) = W^T X(n) \tag{I.24}$$

$$e(n) = d(n) - y(n) \tag{I.25}$$

Nous avons implémenté sur Matlab ces équations pour les appliquer à un signal sinusoïdal d'une durée de 500 millisecondes et d'une fréquence de 5 Hz bruité par les deux bruits utilisés précédemment.

Les figures I.12 et I.13 illustrent respectivement le résultat du filtrage d'un signal bruité avec un filtre de l'algorithme RLS. La figure I.12 représente le résultat du filtrage d'un signal bruité avec un bruit de distribution uniforme, tandis que la figure I.13 représente le résultat du filtrage d'un signal bruité avec un bruit de distribution normale.

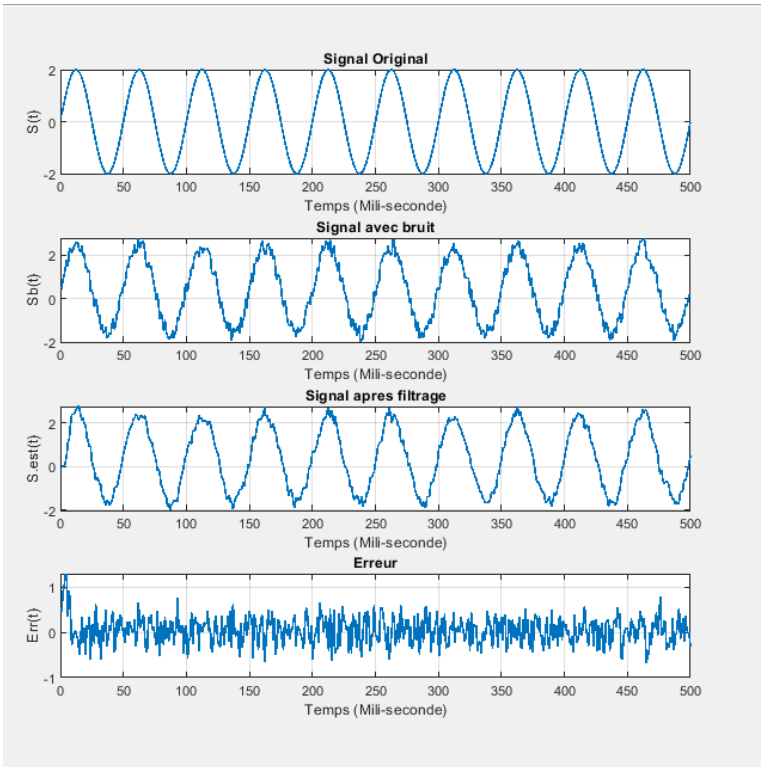


Figure I.12 – Résultat du filtrage d'un signal bruité avec un bruit de distribution uniforme en utilisant un filtre de l'algorithme RLS.

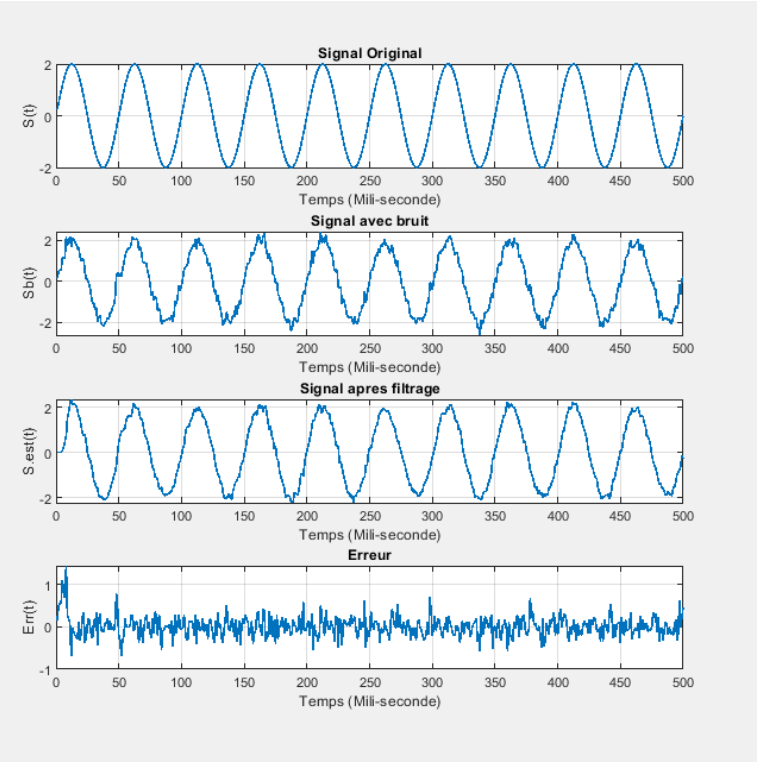


Figure I.13 – Résultat du filtrage d'un signal bruité avec un bruit de distribution normale en utilisant un filtre de l'algorithme RLS.

L'algorithme RLS parvient à réduire de manière similaire les effets du bruit, indépendamment de la distribution.

I.2.4 Autres Algorithmes de Filtrage Adaptatif

Il existe d'autres algorithmes de filtrage adaptatif, on peut citer les trois algorithmes suivants.

I.2.4.1 MCM normalisés : Normalized LMS

L'algorithme MCM normalisé est considéré comme une implémentation spéciale de l'algorithme LMS qui prend en considération la variation du niveau du signal à l'entrée du filtre et sélectionne un paramètre de taille de pas normalisé, ce qui résulte en une convergence plus rapide et plus stable [11].

I.2.4.2 Moindres Carrés Moyens à pas variable : Variable Step-Size LMS

Le paramètre de pas μ joue un rôle important dans le contrôle de performance de l'algorithme LMS, comme la vitesse de convergence est proportionnelle à ce paramètre, on peut aboutir aux deux résultats suivants :

— Un grand paramètre de pas peut être nécessaire pour minimiser le temps transitoire de l'algorithme LMS.
— Pour réaliser un petit déréglage, un petit paramètre de pas doit être utilisé.

L'algorithme VSLMS est une solution au problème de contradiction présenté par les deux résultats précédents [11].
La formule de mise à jour des poids est :

$$w_i(n+1) = w_i(n) + 2\mu_i(n)e(n)x(n-i) \tag{I.26}$$

$\mu_i(n)$ est mis à jour par la formule suivante :

$$\mu_i(n) = \mu_i(n-1) + \rho g_i(n)g_i(n-1) \tag{I.27}$$

Ou ρ est un paramètre positif très petit.

I.2.4.3 Algorithme de projection affine (APA) : Affine Projection Algorithm (APA)

L'algorithme LMS à projection affine (APLMS), ou NLMS généralisé, est une généralisation de l'algorithme NLMS. Il est obtenu par la résolution du problème d'optimisation suivant [11] :

Pour une entrée $x(n)$ et une sortie désirée $d(n)$, on choisit les poids de mise à jour $w(n)$ d'une façon à minimiser la norme euclidienne au carré de la différence :

$$\eta(n) = w(n+1) - w(n) \tag{I.28}$$

Conclusion

En conclusion de ce chapitre, nous avons introduit de manière concise le filtrage adaptatif, suivi de l'implémentation du filtrage d'un signal sinusoïdal bruité avec des bruits gaussiens. L'objectif ultime est d'appliquer ces algorithmes aux signaux ECG une fois qu'ils auront été mis en œuvre sur un processeur de traitement de signal (DSP). Les signaux ECG seront traités en détails dans le prochain chapitre.

Chapitre II

Les signaux ECG

Introduction

L'ECG est largement utilisé pour diagnostiquer les maladies cardiaques en analysant l'activité électrique du cœur et en identifiant les anomalies. Il est également utilisé dans les applications de systèmes embarqués pour le télédiagnostic, permettant des évaluations à distance, une surveillance continue des patients et des soins médicaux à distance. L'ECG enregistre l'activité électrique du cœur, traçant la tension en fonction du temps, fournissant des informations sur la fréquence cardiaque, la régularité et les intervalles d'activation électrique.

Ce chapitre offre une introduction générale aux signaux ECG. Nous commençons par définir ces signaux, puis examinons leurs caractéristiques principales et leurs applications. Enfin, nous abordons succinctement le traitement de ces signaux.

II.1 Caractéristiques et Applications des Signaux ECG

Les signaux ECG (électrocardiogramme) présentent plusieurs caractéristiques et applications importantes. Dans cette section, nous allons présenter un aperçu des principales caractéristiques et utilisations des signaux ECG particulièrement dans les systèmes embarqués.

II.1.1 Caractéristiques des Signaux ECG

L'interprétation d'un signal ECG standard à 12 dérivations nécessite l'identification et l'analyse de neuf caractéristiques clés. La figure II.1 présente les emplacements des électrodes des dérivations d'un signal ECG.

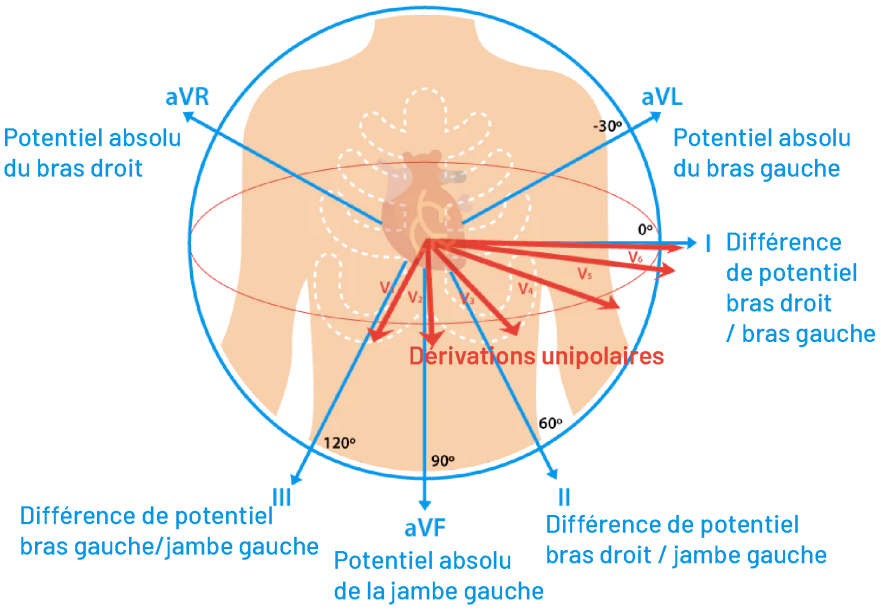


Figure II.1 – Emplacements des électrodes des dérivations d'un signal ECG

II.1.1.1 Fréquence et régularité

La fréquence cardiaque correspond au nombre de battements de cœur par minute (bpm : beat per minute). Elle se situe entre 60 et 100 bpm, mais elle peut varier en fonction de facteurs tels que l'âge, l'activité physique et l'état émotionnel.

Lorsqu'un signal ECG indique une fréquence cardiaque inférieure à 100 bpm, les grands carrés sur le papier ECG suffisent pour évaluer le rythme cardiaque. Toutefois, lorsque la fréquence est supérieure à 100 bpm (tachycardie), il faut également prendre en compte les petits carrés, car de légères variations peuvent altérer l'évaluation du rythme cardiaque [12].

La régularité du signal ECG fait référence à la cohérence de la chronologie entre les battements de cœur successifs. Dans un ECG normal, l'intervalle de temps entre les ondes R successives est relativement constant. Ces ondes R représentent la dépolarisation ventriculaire du cœur lors de sa contraction. Elles permettent d'évaluer la durée, l'amplitude et la forme de cette composante spécifique. Les changements dans les caractéristiques de l'onde R peuvent indiquer différentes affections cardiaques. L'analyse des ondes R, combinée à d'autres aspects de l'ECG, permet aux professionnels de la santé de diagnostiquer les problèmes cardiaques et de fournir des soins appropriés.

Les signaux ECG présentent une structure d'ondes caractéristique, comprenant les ondes P, QRS et T. Ces formes d'ondes fournissent des informations sur l'activité électrique normale ou anormale du cœur.

II.1.1.2 Morphologie de l'onde P

L'onde P est une petite onde arrondie qui précède normalement le complexe QRS sur un signal ECG. À des rythmes cardiaques lents ou normaux, l'onde P est clairement visible, mais à des rythmes plus rapides, elle peut se fondre avec l'onde T et devenir difficile à identifier [13].La figure II.2 présente les caractéristiques d'un signal ECG.

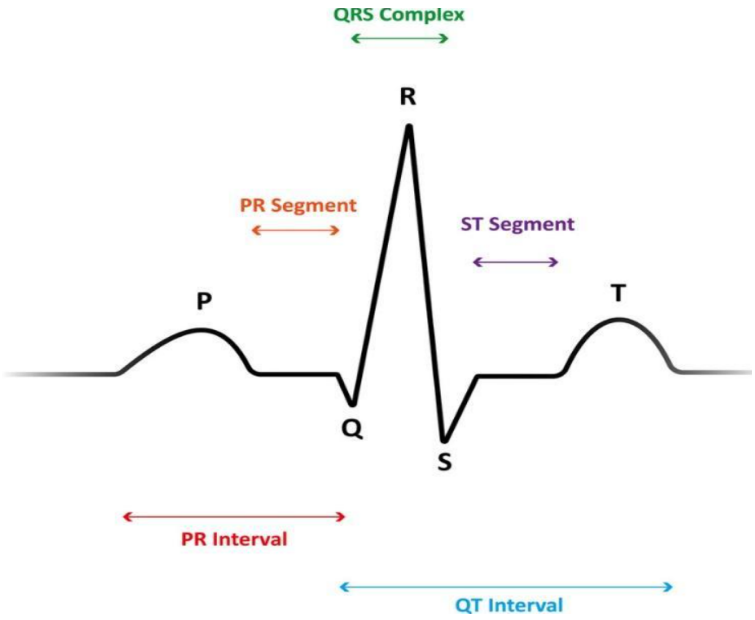


Figure II.2 – Caractéristiques d'un signal ECG

Pour définir la morphologie de l'onde P, quatre étapes doivent être suivies [13] :

— Contour général : Le contour général de l'onde P est généralement lisse et entièrement positif ou entièrement négatif, sauf dans la dérivation V1 où elle peut être biphasique.
— Durée : La durée de l'onde P est normalement inférieure à 0,12 seconde.
— Amplitudes positives et négatives : L'amplitude de l'onde P ne dépasse normalement pas 0,2 mV dans les dérivations des membres dans le plan frontal et de 0,1 mV dans les dérivations thoraciques dans le plan transversal.
— Axes dans les plan frontal et transversal : Dans le plan frontal, l'axe normal de l'onde P se situe entre 0 et +75 degrés. Dans le plan transversal, l'onde P est normalement verticale dans les dérivations I, II, aVF et V4 à V6 (voir figure II.1).

II.1.1.3 Intervalle PR

Le temps nécessaire pour qu'un signal électrique se propage des oreillettes aux ventricules du cœur est appelé l'intervalle PR. En général, cet intervalle dure de 0,10 à 0,21 seconde. La durée de l'intervalle PR est influencée par l'équilibre entre les systèmes nerveux sympathique et parasympathique, qui varie en fonction de la fréquence cardiaque. Lorsque la fréquence cardiaque est élevée, l'intervalle PR est plus court en raison de l'augmentation de l'activité du système nerveux sympathique [12].

II.1.1.4 Morphologie du complexe QRS

Pour définir la morphologie de l'onde QRS, quatre étapes doivent être suivies [13] :
— Contour : Le complexe QRS présente un contour pointu parce qu'il est composé de signaux de fréquence plus élevée que les ondes P et T.
— Durée : La durée du complexe QRS est appelée l'intervalle QRS, qui varie normalement de 0,07 à 0,11 seconde.
— Amplitudes positives et négatives : L'amplitude présente un large intervalle de limites normales et varie en fonction de l'âge et du sexe. L'amplitude globale du QRS est mesurée entre les crêtes des formes d'ondes positives et négatives les plus hautes du complexe. On parle d'amplitude QRS anormalement faible lorsque l'amplitude globale ne dépasse pas 0,5 mV dans l'une des dérivations des membres et 1,0 mV dans l'une des dérivations précordiales.
— Axes dans les plan frontal et transversal : L'axe QRS est la direction moyenne de la force totale produite par la dépolarisation des ventricules droit et gauche. L'axe QRS est généralement positif dans les dérivations du plan frontal. Les pôles positifs et négatifs des 6 dérivations des membres fournissent la circonférence complète de 360 degrés du système de référence hexagonal dans le plan frontal, et les pôles positifs et négatifs des 6 dérivations précordiales la fournissent dans le plan transversal.

II.1.1.5 Morphologie du segment ST

Le segment ST représente la période pendant laquelle le myocarde ventriculaire traverse les deux phases préliminaires de repolarisation après sa dépolarisation. Ce segment

forme typiquement un angle distinct avec la pente descendante de l'onde R ou la pente ascendante de l'onde S, puis se poursuit presque horizontalement jusqu'à ce qu'il s'incurve doucement dans l'onde T. La longueur du segment ST est influencée par des facteurs qui modifient la durée de l'activation ventriculaire. Le segment ST peut présenter une légère pente ascendante, descendante ou une dépression horizontale, ce qui peut être une variante normale. L'aspect du segment ST peut également être modifié en présence d'un complexe QRS anormalement prolongé.

II.1.1.6 Morphologie de l'onde T

Pour définir la morphologie de l'onde T, quatre étapes doivent être suivies [12] :

— Contour : L'onde T est lisse et arrondie et sa forme et son axe sont similaires à ceux de l'onde P. Elle a une direction positive dans toutes les dérivations, sauf dans l'aVR, où elle est négative, et dans le V1, où elle est biphasique. En outre, un léger "pic" de l'onde T peut être une variante normale.

— Durée : La durée de l'onde T n'est généralement pas mesurée, mais elle est incluse dans l'intervalle QT.

— Amplitudes positives et négatives : L'amplitude de l'onde T a des limites normales larges, tend à diminuer avec l'âge et est plus importante chez les hommes que chez les femmes. Elle tend à varier en fonction de l'amplitude du QRS et doit toujours être supérieure à celle de l'onde U qui l'accompagne. Les ondes T ne dépassent normalement pas 0,5 mV dans les dérivations des membres ou 1,5 mV dans les dérivations précordiales.

— Axes dans les plan frontal et transversal : L'évaluation de l'axe de l'onde T est essentielle et doit être effectuée par rapport à celui du complexe QRS. Il convient de noter que l'axe de l'onde T reste généralement constant tout au long de la vie, tandis que celui du complexe QRS évolue d'une position verticale à une position horizontale. Ainsi, chez l'enfant, l'axe de l'onde T est plus horizontal que celui du complexe QRS, tandis que chez l'adulte, l'axe de l'onde T devient plus vertical que celui du complexe QRS.

II.1.1.7 Morphologie de l'onde U

Sur un signal ECG, l'onde U se présente généralement comme une petite onde de forme arrondie qui suit l'onde T ou qui en est absente. L'onde U a généralement 10 % de l'amplitude de l'onde T, est orientée dans la même direction et est plus importante dans les dérivations V2 ou V3. À des fréquences cardiaques plus lentes, l'onde U est plus importante, et les ondes U et T réduisent leur taille et fusionnent avec l'onde P qui les suit à des fréquences cardiaques plus rapides [12].

II.1.1.8 Intervalle QTc

L'intervalle QT est une mesure de la durée de l'activation électrique et de la récupération du myocarde ventriculaire. Cette durée varie inversement avec la fréquence cardiaque et doit être corrigée pour cette dernière afin de déterminer sa "normalité". L'intervalle QT

corrigé (QTc) est utilisé dans l'analyse ECG de routine et a une limite supérieure d'environ 0,46 seconde (460 ms). Le QTc est légèrement plus long chez les femmes que chez les hommes et augmente légèrement avec l'âge. Une mesure précise de l'intervalle QTc ne peut être effectuée qu'après plusieurs cycles cardiaques réguliers et égaux [13].

II.1.1.9 Rythme cardiaque

Pour évaluer le rythme cardiaque, il faut prendre en compte les 8 autres caractéristiques de l'ECG. Des irrégularités de la fréquence, de la régularité, de l'onde P et de l'intervalle PR peuvent indiquer des anomalies du rythme cardiaque. De même, certaines irrégularités des 5 autres caractéristiques de l'ECG peuvent suggérer le potentiel de développement d'anomalies du rythme cardiaque. Par conséquent, toutes les caractéristiques de l'ECG doivent être examinées pour détecter les anomalies et évaluer le rythme cardiaque [12].

II.1.2 Applications des Signaux ECG

L'électrocardiogramme est un outil de diagnostic couramment utilisé dans la pratique clinique pour identifier et surveiller les pathologies cardiaques. Les signaux ECG sont utilisés dans diverses applications telles que le diagnostic des maladies cardiaques, notamment l'arythmie, l'infarctus du myocarde, l'insuffisance cardiaque et les anomalies structurelles [14].

En outre, les signaux ECG sont utilisés pour surveiller l'activité électrique du cœur pendant les activités quotidiennes et l'exercice, en particulier chez les patients souffrant d'une maladie cardiaque ou présentant un risque de développer une maladie cardiaque. Les signaux ECG sont également utilisés pour évaluer le risque de mort cardiaque subite chez certains patients, contrôler l'efficacité des médicaments affectant le cœur et étudier les effets de divers médicaments et interventions sur le cœur dans le cadre de la recherche [15].

L'exploitation des signaux ECG par les systèmes embarqués a ouvert de nouvelles perspectives passionnantes dans le domaine de la santé et du suivi médical. Dans une étude antérieure [16], les auteurs discutent de l'utilisation de la reconnaissance biométrique basée sur les signaux ECG comme solution alternative pour garantir la confidentialité des informations dans certaines applications. Ils mettent l'accent sur un système intégré de reconnaissance biométrique qui utilise des algorithmes avancés pour identifier les individus en temps réel. Ce système extrait des informations significatives du domaine de fréquence des signaux ECG et est implémenté sur un processeur ARM Cortex 4.

En outre, une autre étude menée par Dong, Zhang et Jia [17] présente un moniteur ECG intelligent portable qui peut être utilisé dans un système de surveillance à distance sans fil. Un prototype de cet appareil a été réalisé et utilisé pendant un mois dans un environnement clinique. Les résultats de l'expérience montrent que ce PIEM est cliniquement approuvé, précis dans ses mesures, hautement mobile, facile à utiliser, durable et léger.

Dans une autre publication [18], les auteurs proposent un nouveau système embarqué pour l'ECG qui intègre le raisonnement sur le signal ECG et l'identification RF pour surveiller un patient âgé. Leur méthode présente une bonne précision dans la reconnaissance

des battements cardiaques et permet une surveillance continue et une identification du patient âgé lorsqu'il est seul.

II.2 Bruits

Les bruits présents dans un électrocardiogramme (ECG) sont des perturbations additives qui se situent dans la gamme de fréquences pertinentes pour l'analyse du signal. Ces bruits se manifestent principalement sous la forme de caractéristiques morphologiques qui peuvent ressembler aux aspects naturels de l'ECG ou être spécifiques à une maladie [19]. Il est important de noter que ces bruits suivent généralement une distribution gaussienne, ce qui signifie que s'ils sont centrés, leur moyenne est nulle.

L'identification et la suppression des bruits sont essentielles pour obtenir un signal ECG précis et fiable, ce qui permet une interprétation correcte par les professionnels de la santé. Voici quelques-uns des bruits les plus courants rencontrés dans un ECG [19] :

II.2.1 Bruit d'interférence des lignes électriques (PLI) :

Les interférences des lignes électriques (PLI) sont un problème courant dans les équipements médicaux qui peuvent affecter de manière significative les enregistrements (ECG) en dominant parfois le signal et en le rendant difficile à analyser. La figure II.3 présente un exemple d'un signal ECG perturbé par une interférence de ligne électrique de basse fréquence.

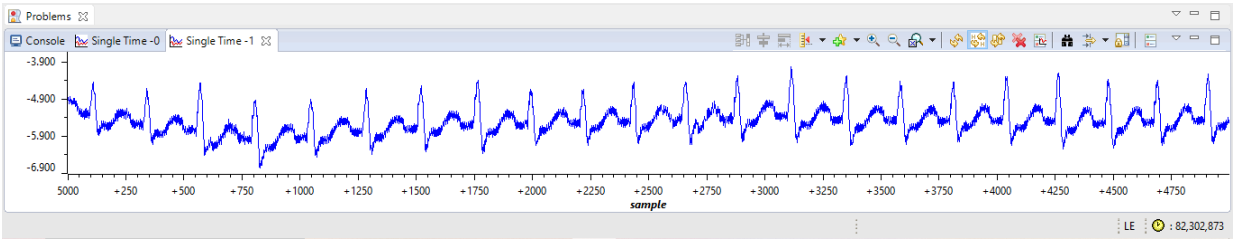


Figure II.3 – Un signal ECG perturbé par une interférence de ligne électrique

Le couplage capacitif et le couplage inductif sont des mécanismes qui causent l'interférence des lignes électriques. Les différences d'impédance des électrodes, les courants parasites à travers le patient, les câbles ou les instruments, ainsi que les entrées flottantes des instruments contribuent à la contamination des enregistrements ECG par cette interférence. Le couplage capacitif génère principalement un bruit à haute fréquence, tandis que le couplage inductif introduit plutôt un bruit à basse fréquence [20].

Pour minimiser les interférences des lignes électriques, il est important d'appliquer correctement les électrodes, de s'assurer qu'il n'y a pas de fils lâches et d'utiliser un blindage adéquat pour tous les composants.

II.2.2 Errance de base (BW)

L'errance de base (Baseline Wander) est un artefact à basse fréquence causé principalement par la respiration, les mouvements du corps, un mauvais contact des électrodes et l'impédance peau-électrode.

La magnitude de L'errance peut atteindre jusqu'à environ 15 % de la déviation à pleine échelle (FSD), qui correspond à l'amplitude de crête à crête du signal ECG dans une plage de fréquences de 0,15 à 0,3 Hz.

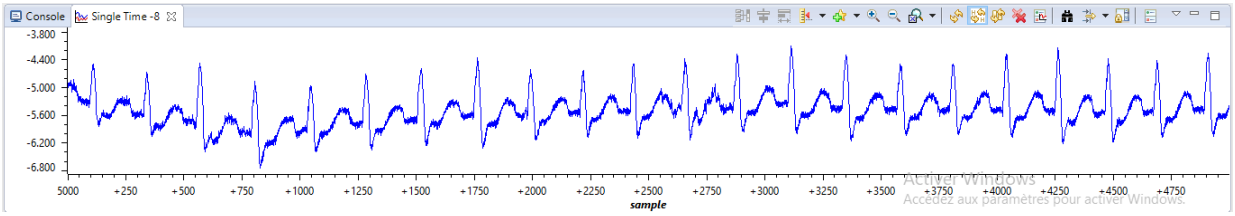


Figure II.4 – Un signal ECG perturbé par l'érrance de base

Sur la figure II.4, nous pouvons observer la présence de l'errance de base, qui déforme le segment ST. Cela peut conduire à un diagnostic incorrect d'infarctus du myocarde, de syndrome de Brugada et d'autres anomalies associées à ce dernier.

II.2.3 Bruit musculaire (EMG)

Le bruit musculaire est un problème majeur dans les enregistrements ECG, en particulier pendant l'exercice, car il masque les formes d'onde de faible amplitude. Contrairement à l'errance de la ligne de base, il ne peut pas être éliminé par un filtrage à bande étroite. Cela pose un problème car son contenu spectral chevauche celui du complexe PQRST [21].

La figure II.5 montre un signal ECG corrompu par le bruit EMG. La suppression efficace de ce bruit est essentielle pour une analyse précise des complexes PQRST.

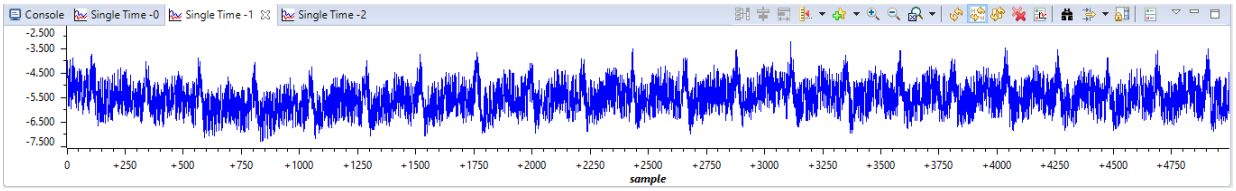


Figure II.5 – Un signal ECG altéré par le bruit EMG

II.3 Traitement des signaux ECG

Le filtrage est une technique couramment utilisée pour réduire le bruit dans le signal de l'électrocardiogramme (ECG) lors de l'analyse du signal.

Les filtres passe-bande sont généralement utilisés, mais leur efficacité dépend du type et du niveau de bruit présent dans le signal. Il est important de comprendre les processus

de bruit impliqués avant d'essayer de filtrer ou de prétraiter le signal ECG car il est sensible et même une petite quantité de bruit peut altérer ses caractéristiques [22]. La figure suivante montre un exemple d'un signal ECG filtré.

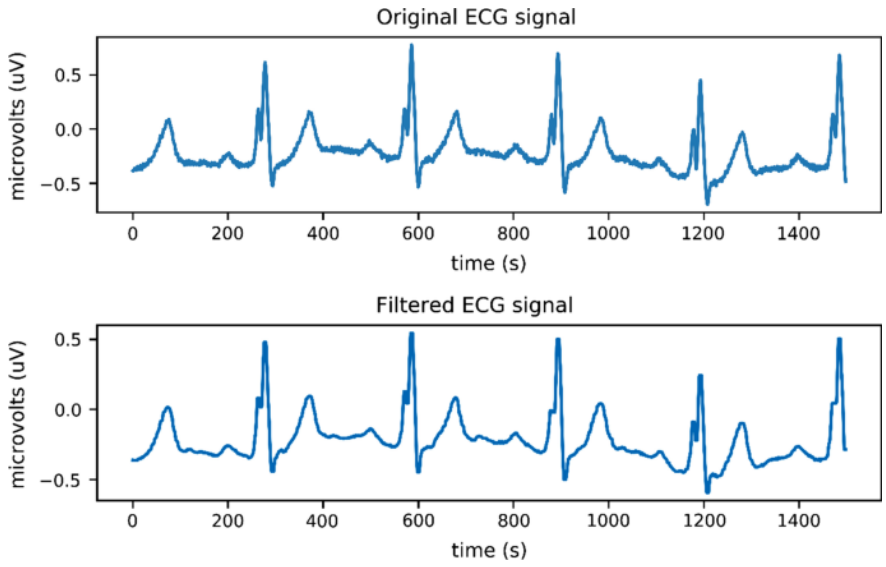


Figure II.6 – Traitement des Signaux ECG

La conception de filtres permettant d'éliminer l'errance de la ligne de base, les interférences des lignes électriques et le bruit musculaire des signaux ECG a fait l'objet d'une attention particulière [23].

La partie traitement des signaux ECG a impliqué une recherche approfondie sur les travaux existants concernant le filtrage adaptatif des signaux ECG. Plusieurs travaux ont mis en évidence différentes approches pour le filtrage adaptatif des signaux ECG en temps réel [24] [25] [26].

Dans [24], La méthode de filtrage utilisée combine la DWT pour la décomposition du signal en coefficients d'ondelettes et le seuillage pour éliminer le bruit haute fréquence et extraire les caractéristiques du signal. Ensuite, un réseau de neurones artificiels est utilisé pour filtrer les coefficients d'ondelettes restants et obtenir le signal ECG filtré en sortie. De plus, Laguna et al [25], proposent dans une méthode de filtrage en cascade utilisant un filtre coupe-bande passe-haut suivi d'un filtre adaptatif corrélatif d'impulsions pour éliminer les composantes indésirables du signal ECG, tout en préservant les composantes ST corrélées. Le filtre en cascade permet de supprimer le composant continu, les variations de ligne de base non corrélées et de conserver les caractéristiques pertinentes du signal ECG.

Dans [26], les auteurs proposent une structure de filtre adaptatif récurrent pour acquérir la réponse impulsionnelle du complexe QRS normal. L'entrée principale du filtre est le signal ECG à analyser, tandis que l'entrée de référence est un train d'impulsions coïncidant avec les complexes QRS. Cette méthode est appliquée à plusieurs problèmes de détection d'arythmies : détection des ondes P, complexes ventriculaires prématurés et reconnaissance de bloc de conduction, fibrillation atriale et rythme commandé. Cependant, il est important de noter que l'implémentation de ces filtres adaptatifs sur un processeur géné-

raliste ne s'adapte pas aux applications de systèmes embarqués.

Dans le cadre de ce projet, nous avons pris en compte ces considérations et nous avons choisi d'adopter une approche basée sur les filtres adaptatifs implémentée sur une carte DSP6713.

Conclusion

Dans ce chapitre, les signaux d'électrocardiogramme (ECG) et leurs caractéristiques sont présentés. Nous étudions également les différents types de bruit qui peuvent affecter les enregistrements ECG. Une attention particulière est accordée au traitement des signaux ECG, mettant en évidence l'importance des techniques de prétraitement et de filtrage. Parmi les approches discutées, les filtres adaptatifs sont présentés comme une méthode prometteuse pour réduire efficacement le bruit et améliorer la précision des enregistrements ECG. Dans le cadre de notre travail nous nous intéressons à l'implémentation des filtres adaptatifs sur un processeur de traitement de signal. La carte DSP 6713 est abordée dans le chapitre suivant.

Chapitre III

La carte DSP TMS320c6713

Introduction

Le traitement de signal est devenu indispensable pour de nombreuses applications dans divers domaines tels que la médecine, la surveillance et l'automobile.

Les processeurs de signal numérique (PSN) ou plus couramment désignés sous le nom (DSP), sont des processeurs optimisés pour le traitement de signal. En répondant aux contraintes de temps réel et de précision exigées par ces applications, les DSP jouent un rôle majeur dans le développement actuel.

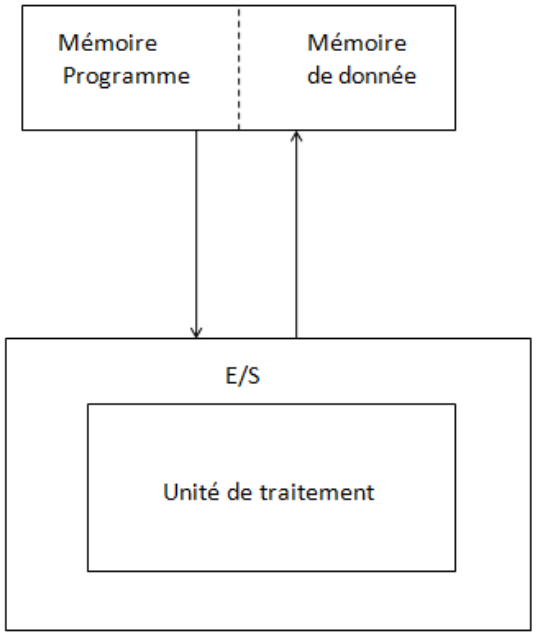
Dans le cadre de notre travail nous nous intéressons à l'exploitation des DSP dans le domaine du filtrage adaptatif des signaux biomedicaux, en particulier le signal ECG. A cet effet nous présentons dans ce chapitre la carte utilisée : DSK6713.

III.1 Présentation des DSP

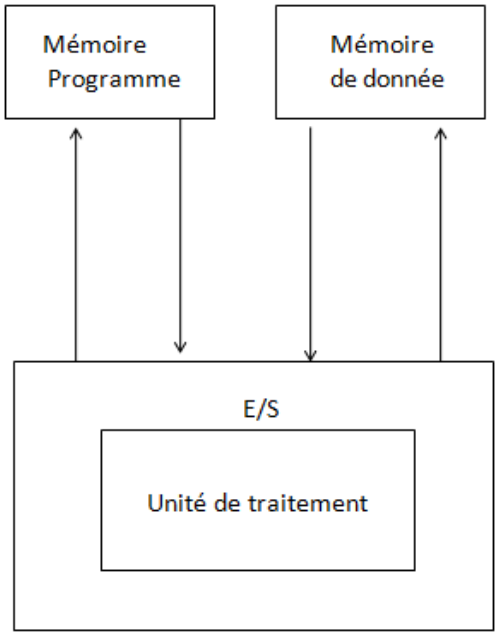
Les DSP sont des processeurs spécialisés conçus spécifiquement pour le traitement des signaux numériques. Leur architecture permet d'optimiser les opérations de calcul. En intégrant des techniques avancées telles que la segmentation de mémoire, ainsi que des pipelines, ces processeurs spécialisés garantissent une manipulation précise et rapide des signaux, permettant ainsi des applications variées allant de la communication sans fil au traitement audio et vidéo haute définition [27].

III.1.1 Architecture Harvard modifiée

La figure III.1.1 a) présente l'architecture de Harvard modifiée qui a été conçue pour combiner l'efficacité de l'architecture de Harvard, (voir figure III.1.1 b) tout en réduisant ses coûts. Ainsi, cette solution combine l'efficacité économique de l'architecture de Von Neumann avec les performances de l'architecture de Harvard [28].



a)



b)

Figure III.1 – Architecture : a) type Harvard modifiée, b) type Harvard.

Alors que l'architecture de Harvard classique sépare la mémoire en deux busses distincts pour les instructions et les données, l'architecture de Harvard modifiée permet à la mémoire d'être partagée entre les instructions et les données. Cela signifie que les instructions peuvent être traitées comme des données et sont manipulées de la même manière [29].

III.1.2 Les DSP à virgule flottante et à virgule fixe

On distingue deux types de représentation des données dans les DSPs :

III.1.2.1 Les DSP à virgule fixe

Les données sont représentées comme étant des nombres fractionnaires à virgule fixe, ou comme des entiers. Cette représentation facilite l'addition binaire de nombres aussi bien positifs que négatifs.

III.1.2.2 Les DSP à virgule flottante

Les données sont représentées en utilisant une mantisse et un exposant. Les DSP à virgule flottante offrent une très grande dynamique en termes de manipulation d'espace mémoire, mais ils sont plus chers et consomment plus d'énergie.

III.2 Domaines d'applications

Les DSP sont utilisés dans une variété d'applications, allant des communications sans fil à la reconnaissance de la parole, en passant par l'imagerie médicale et le traitement audio. Leur capacité à effectuer des calculs rapides sur des signaux en temps réel est essentielle pour ces applications. Dans ce chapitre, nous allons examiner certaines applications des DSP, ainsi que des exemples spécifiques de travaux effectués sur la TMS320C6713.

III.2.1 Traitement audio

Le traitement audio est l'une des applications les plus courantes des DSP. Les DSP peuvent être utilisés pour la suppression du bruit, l'amélioration de la qualité sonore et la compression audio.

Par exemple, une équipe de recherche a utilisé la TMS320C6713 pour concevoir un système de traitement en temps réel capable de supprimer efficacement le bruit de larsen (Howling Noise) d'un système de diffusion sonore. L'article mentionne que toute la mise en œuvre a été réalisée en se basant sur la fonctionnalité d'accès direct à la mémoire (DMA) du processeur DSP. L'algorithme basé sur le DMA pour l'acquisition et la mise en mémoire tampon des échantillons est plus efficace que l'algorithme fonctionnant sur les interruptions [30].

III.2.2 Communications sans fil

Les DSP sont utilisés dans les systèmes de communication sans fil pour le codage et le décodage des signaux, ainsi que pour la correction d'erreurs et la suppression du bruit.

III.2.3 Reconnaissance de la parole

La reconnaissance de la parole est une application exigeante en termes de puissance de traitement et de précision, et les DSP sont souvent utilisés pour cette tâche.

Par exemple, un projet de recherche a utilisé la TMS320C6713 pour développer un système de reconnaissance de la parole en temps réel [31].

III.2.4 Imagerie médicale

Les DSP sont largement utilisés dans le domaine de l'imagerie médicale, notamment pour le traitement d'images et de signaux provenant de scanners, d'IRM et d'autres équipements de diagnostic.

Par exemple, un projet de recherche a utilisé la TMS320C6713 pour implémenter la technique SCR pour opération de tumeur en utilisant le traitement d'images biomédicales [32].

Ces exemples illustrent la polyvalence des DSP et leur utilisation dans des applications de pointe. Le TMS320C6713 représente une plateforme particulièrement adaptée à ces travaux en raison de sa puissance de traitement et de sa flexibilité en matière de programmation.

III.3 Architecture et fonctionnalités du DSKC6713

Le TMS320C6713 est un processeur de signal numérique (DSP) de la famille TMS320C6000 de Texas Instruments (TI) [1]. Il est conçu pour offrir des performances de traitement de signal exceptionnelles et une faible consommation d'énergie pour les applications de traitement de signal en temps réel.

Architecture	VLIW
Précision	à virgule flottante
Fréquence d'horloge	225 MHz
Mémoire Flash	256 Ko
RAM	64 Ko
interface d'E/S intégrée	GPIO, SPI, UART,...
16-bit codec(AIC23B)	convertisseur A/D D/A

Table III.1 – Principales Caractéristiques du Processeur C6713

La figure III.2 montre l'architecture interne du TMS320C6713 [33].

1. https ://www.ti.com/

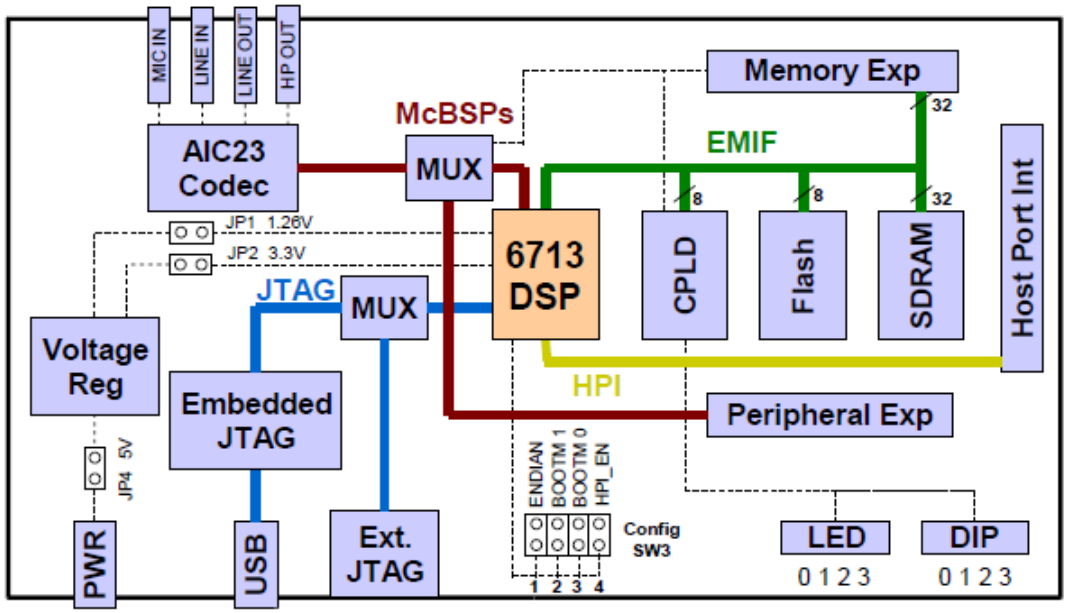


Figure III.2 – Architecture de La TMS320C6713

Le TMS320C6713 est un processeur numérique de signal (DSP) haute performance reconnu pour son architecture avancée et ses fonctionnalités polyvalentes (voir tableau III.1).

Il est également disponible intégré sur une carte de développement. La carte DSK DSPc6713 est équipée d'un DSPc6713, d'une horloge de 225 MHz et de 64 Ko de RAM pour les données. Elle dispose également d'une interface d'E/S intégrée pour la communication avec les périphériques externes, ainsi que d'un certain nombre de périphériques intégrés, tels qu'un codec audio stéréo, un contrôleur Ethernet et des connecteurs d'extension [34].

Afin de garantir une intégration aisée dans le flux de travail existant la carte est compatible avec des environnements de développement tiers, tels que MATLAB/Simulink de MathWorks en plus d'être accompagnée de logiciels de développement élaboré et puissant tel que code composer studio (CCS).

III.3.1 Module de traitement (CPU)

Considéré comme le cœur de la carte, ce module est responsable de l'exécution des algorithmes et du stockage des données nécessaires au traitement.

le CPU du DSK6713 est composé de huit unités fonctionnelles indépendantes réparties en deux chemins de données A et B [33]. Elles sont présentées dans la figure III.3.

1. Unité de multiplication (.M) : Cette unité est responsable des opérations de multiplication.
2. Unité arithmétique et logique (.L) : Cette unité est utilisée pour les opérations arithmétiques, logiques et les instructions de branchement.
3. Unité pour les opérations de branchement, de manipulation de bits et d'arithmétique (.S) : Cette unité gère les instructions de branchement, les manipulations de bits et les opérations arithmétiques.

4. Unité pour les opérations de chargement/stockage et d'arithmétique (.D) : Cette unité est utilisée pour les transferts de données, y compris les opérations de chargement et de stockage, ainsi que les opérations arithmétiques.

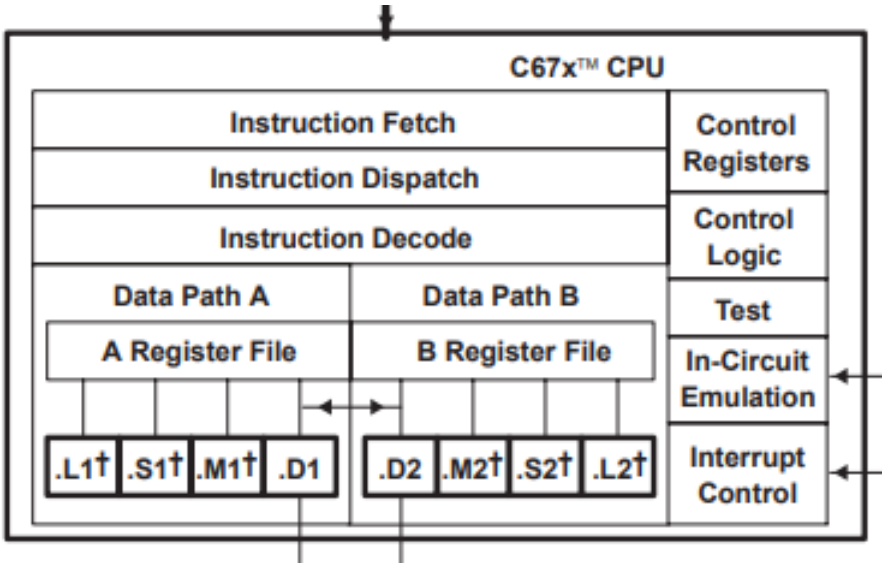


Figure III.3 – Module de traitement (CPU) de la TMS320C6713

III.3.2 Mémoire

Le DSP C6713 sur le DSK comprend 264 ko de mémoire interne, ainsi que 16 Mo de SDRAM externe. Le DSK comprend également 512 ko de mémoire Flash. Avec un bus d'adresse de 32 bits, l'espace mémoire total est de 4 Go, comprenant quatre espaces mémoire externes : CE0, CE1, CE2 et CE3 [33].

La famille de DSP C67xx dispose d'un vaste espace d'adressage mémoire. Le code du programme et les données peuvent être placés n'importe où dans l'espace d'adressage unifié (architecture Harvard modifié) et les adresses font toujours 32 bits de largeur. La figure III.4 montre l'adresse de la région memoire du c6713 à gauche, avec des détails spécifiques sur son utilisation à droite.

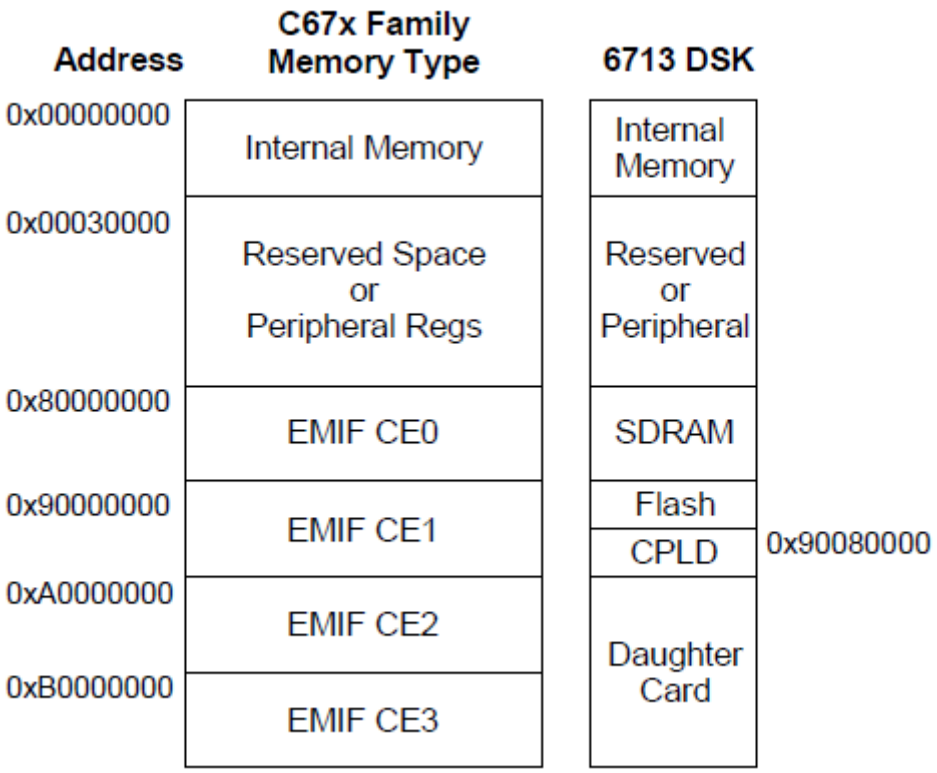


Figure III.4 – Memory Map, C6713 DSK

III.3.3 AIC23 Codec

Le codec AISC est un périphérique spécialisé qui fournit des capacités de conversion analogique-numérique et numérique-analogique. Il permet ainsi au DSP de se connecter à des périphériques audio externes, tels que des microphones et des haut-parleurs.La figure III.5 illustre l'interface du codec sur la DSK TMS320C6713.

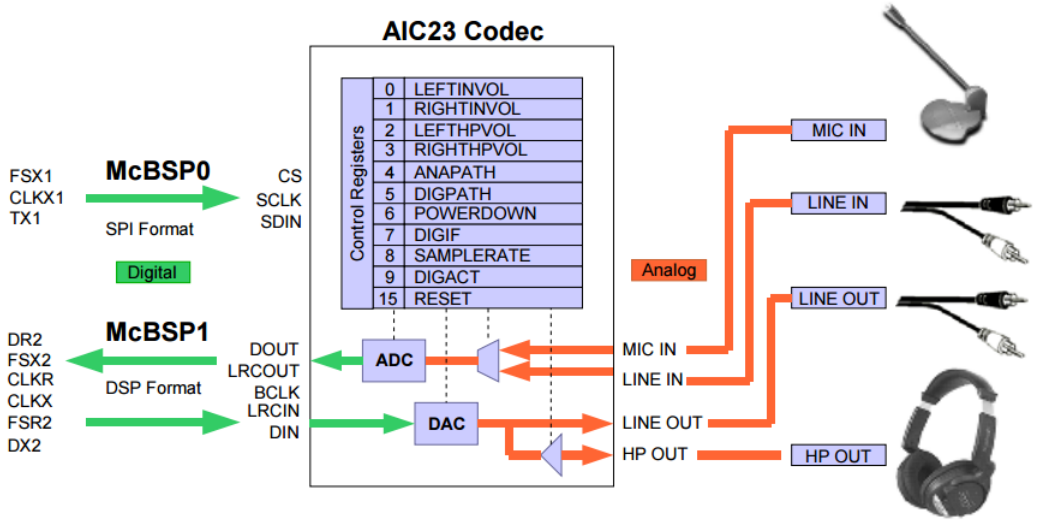


Figure III.5 – L'interface du codec sur la DSK TMS320C6713

III.3.4 Emulateur JTAG

L'émulateur JTAG fournit une interface haute vitesse pour charger des logiciels sur la carte et accéder à la mémoire interne et aux registres du DSP en assurant la connexion de la puce DSP à un ordinateur hôte, permettant le développement et le débogage de logiciels.

III.3.5 Pipeline

Les DSP utilisent la méthode du pipeline pour améliorer les performances du CPU, Le pipeline consiste à imposer un ordre dans le déroulement des instructions pour optimiser le traitement en rapidité d'exécution. Il y a trois étapes dans le Pipelining : l'extraction du programme (Fetch), le décodage (Decode), et l'exécution (Execute) [33].

1. L'étape (Fetch) se compose de quatre phases :
   - PG : génération de l'adresse du programme (dans le CPU) pour récupérer une adresse.
   - PS : program address send (vers la mémoire) pour envoyer l'adresse.
   - PW : program address ready wait (lecture de la mémoire) pour attendre les données.
   - PR : programme de réception d'un paquet de données (au niveau du CPU) pour lire le code d'opération dans la mémoire.
2. L'étape (Decode) se compose de deux phases :
   - DP : distribution de toutes les instructions d'un PF aux unités fonctionnelles appropriées.
   - DC : décodage des instructions.
3. L'étape (Execute) est composée de 6 phases (en virgule fixe) à 10 phases (en virgule flottante) en raison des délais d'exécution, associés aux instructions suivantes :
   - Instruction de multiplication.
   - Instruction de chargement.
   - Instruction de branchement.

Un exemple de l'enchaînement des étapes du pipeline est présenté dans la figure III.6.

Clock Cycle											
1	2	3	4	5	6	7	8	9	10	11	12
PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5	E6
	PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5
		PG	PS	PW	PR	DP	DC	E1	E2	E3	E4
			PG	PS	PW	PR	DP	DC	E1	E2	E3
				PG	PS	PW	PR	DP	DC	E1	E2
					PG	PS	PW	PR	DP	DC	E1
						PG	PS	PW	PR	DP	DC

Figure III.6 – Opération de Pipeline

III.4 Language de programmation et outil pour DSP

Pour programmer des DSP, il est important de comprendre les langages de programmation et les outils disponibles.

III.4.1 Language de programmation pour DSP

Le langage de programmation le plus couramment utilisé pour les DSP est le langage C, bien qu'il existe d'autres langages de programmation tels que le langage assembleur et le langage MATLAB. Le langage C est populaire car il permet une programmation de haut niveau, ce qui facilite la tâche du programmeur et permet une meilleure abstraction du matériel DSP.

III.4.2 Outils de développement pour DSP

Les outils de développement pour DSP sont des logiciels qui permettent la programmation, le débogage et la simulation du DSP. L'outil de développement le plus utilisé nos jours est le Code Composer Studio (CCS), qui est un environnement de développement intégré (IDE) développé par Texas Instruments.

CCS est un outil de développement complet qui permet de développer, déployer et déboguer des applications DSP. Il offre une large gamme de fonctionnalités telles que l'éditeur de code, le débogueur, le profilage et la simulation, et prend en charge plusieurs DSP de Texas Instruments, y compris le DSPC6713.

Outre CCS, il existe d'autres outils de développement tels que VisualDSP++ de Analog Devices et Keil µVision de ARM, qui sont également populaires.

Conclusion

Le DSP est devenu un outil essentiel dans de nombreux domaines en raison de sa capacité à effectuer des tâches de traitement de signal complexes avec une grande précision et rapidité. Dans ce chapitre, nous avons présenté la TMS320C6713, ainsi que les outils et les langages de programmation nécessaires pour l'exploiter.
En vue de ce qui a été élaboré, le prochain chapitre portera sur l'implémentation d'un filtre adaptatif sur le TMS320C6713.

Chapitre IV

Implémentation du filtre sur la DSKc6713

Introduction

Ce chapitre couvre les étapes clés pour réaliser l'implémentation. Il débute par la configuration de l'environnement de développement, notamment en examinant les interfaces et outils disponibles dans CCS (Code Composer Studio). Ensuite, les bases de données utilisées dans l'étude sont présentées : la MIT-BIH Arrhythmia Database (mitdb) et l'ECG-ID Database (ecgiddb). Dans la partie suivante, nous discutons des algorithmes de filtrage adaptatif tels que LMS,NLMS et RLS, et de leur implémentation sur le DSP6713. Enfin, les résultats et l'évaluation de l'implémentation sont exposés, y compris les approches d'évaluation et les critères pertinents utilisés.

IV.1 Configuration de l'environnement de développement

L'environnement de développement utilisé est **Code Composer Studio** de Texas Instruments. C'est un logiciel puissant pour programmer et déboguer des applications pour les processeurs numériques de signal (DSP), il offre une interface conviviale et de nombreux outils pour faciliter le développement de logiciels DSP.

IV.1.1 Interface et outils disponibles

L'interface de Code Composer Studio est bien organisée, facilitant la navigation et l'utilisation du logiciel.La figure IV.1 suivante montre l'nterface de CCS.

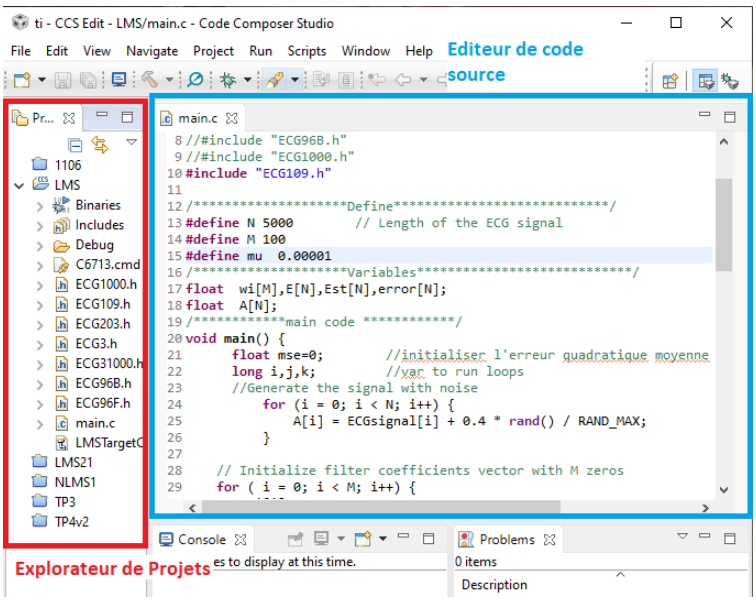


Figure IV.1 – Interface de CCS

La figure précédente met en évidence les différentes fonctionnalités essentielles de cette interface.

1. L'éditeur de code source :

   Pour bien structurer le code principal c (main.c), Il est possible de créer de nouveaux fichiers source (en-têtes, fonctions, etc), les ouvrir et les modifier à partir de cet éditeur.

2. L'explorateur de projet :

   Il affiche la structure du projet, y compris les fichiers source, les bibliothèques et les ressources associées.

3. La fenêtre de configuration du projet :

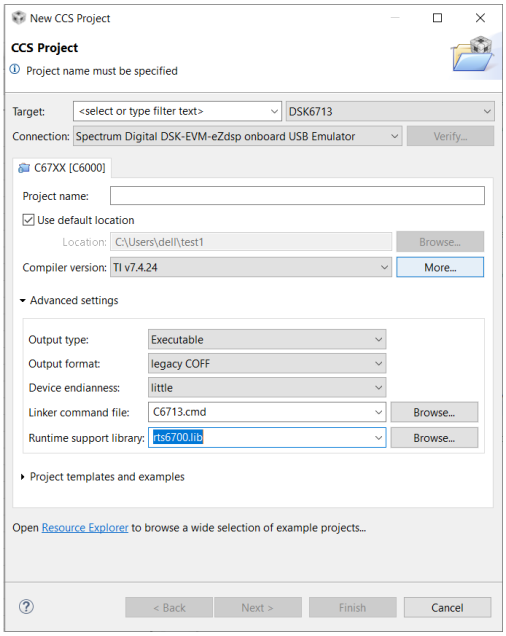


Figure IV.2 – La fenêtre de configuration du projet en CCS

Il est nécessaire de définir les options de configuration spécifiques au projet dans cette fenêtre. Cela inclut les paramètres de compilation, les options de liaison et les configurations matérielles. Afin de définir les options qui seront utilisées lors de la compilation, de la liaison et de l'exécution du projet.

Pour ouvrir cette fenêtre on sélectionne File→ New →CCS Project .

4. Le débogueur :

La figureIV.3 représente Le débogueur en CCS.

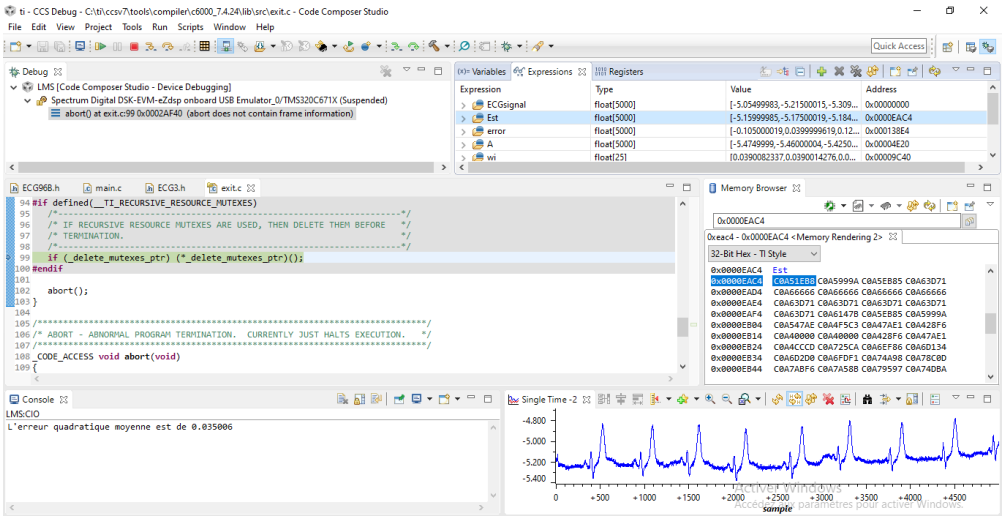


Figure IV.3 – Le debogueur en CCS

Le CCS offre un puissant débogueur permettant de suivre l'exécution du code pas à pas, de mettre des points d'arrêt (BreakPoints) et d'inspecter les variables en temps réel.

Pour accéder au débogueur, on sélectionne View→Debug.

IV.1.2 Memory Browser

Le débogueur de Code Composer Studio permet de visualiser le contenu de la mémoire à un endroit spécifique,d'inspecter et interpréter les données stockées dans la mémoire.

On peut accéder au Memory Browser en séléctionnant View→Memory Browser.

IV.1.2.1 Présentation de la norme IEEE754 pour la représentation des nombres en virgule flottante

Dans les DSP à virgule flottante comme la TMS320C6713, les données sont représentées en utilisant une mantisse et un exposant. La représentation de ces nombres s'effectue selon la formule suivante :

$$n = mantisse * 2^{exposant}.$$

Généralement, la mantisse est un nombre fractionnaire (-1.0 à +1.0), et l'exposant est un entier indiquant la place de la virgule en base 2.

IV.1.3 Création et affichage de graphiques dans CCS

Code Composer Studio intègre une interface d'analyse de signal avancée qui permet aux développeurs de contrôler les données de signal de manière critique et approfondie.

Le menu Graph contient de nombreuses options offrant une flexibilité dans l'affichage des données, y compris un graphe temps/fréquence permettant de visualiser les signaux dans le domaine du temps ou de la fréquence, comme indiqué sur la figure IV.4.

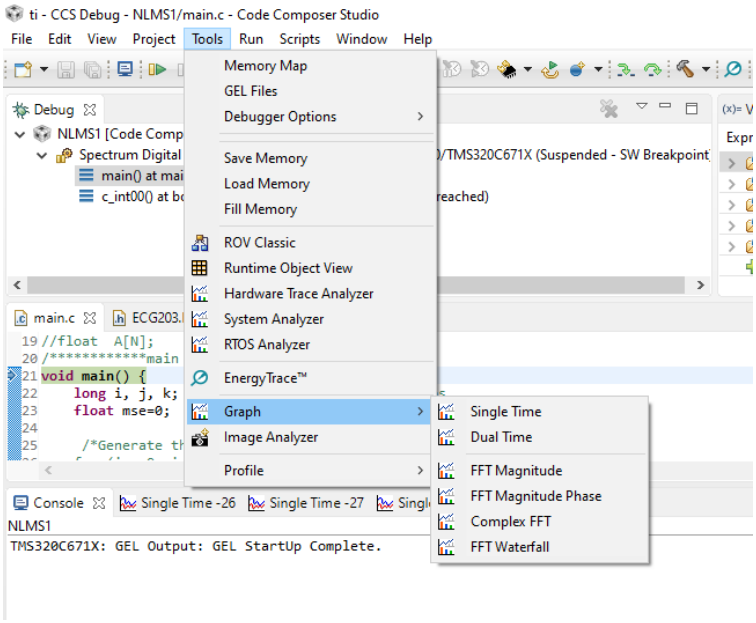


Figure IV.4 – Accéder au menu Graph sur CCS

Les graphiques de fréquence comprennent les graphiques FFT Magnitude, FFT cascade, FFT complexe et FFT magnitude et phase. Les graphiques du domaine temporel peuvent être configurés pour afficher une ou deux échelles de temps.

IV.1.4 Optimisation du code dans CCS

Différentes possibilités d'optimisation sont offertes par le Code Composer Studio (CCS). Nous mettrons en évidence les fonctionnalités et les outils intégrés dans CCS qui permettent d'améliorer les performances, l'efficacité et la qualité des applications embarquées.

IV.1.5 Configuration de la carte DSP6713 pour la réalisation de l'implémentation

Avant de commencer la programmation avec Code Composer Studio, il est nécessaire de le configurer afin qu'il soit compatible avec le processeur TMS320C6713 utilisé dans notre projet.

1. Paramètres de Compilateur

   Il est nécessaire de spécifier les chemins d'inclusion des fichiers d'en-tête requis comme représente la figure IV.5.

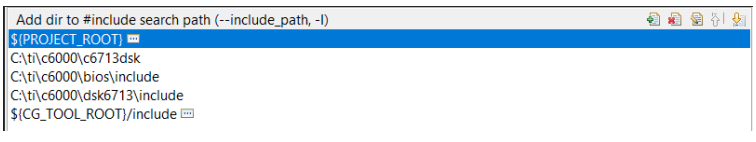


Figure IV.5 – Les Options d'inclusion dans le compilateur

2. Paramètres d'Editeur de Lien

   L'éditeur de lien est responsable de la création de l'exécutable final en liant les différents fichiers objets générés lors de la compilation. Parfois, il faut spécifier des chemins supplémentaires pour rechercher des bibliothèques ou des fichiers objets. La figure IV.6 représente les chemins dans l'éditeur de lien.

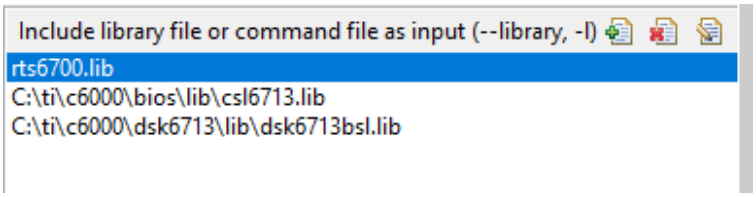


Figure IV.6 – Les chemins dans l'éditeur de lien

3. Symboles Prédéfinis :

   Dans Code Composer Studio (CCS), le symbole prédéfini CHIP_6713 fait référence au processeur DSP TMS320C6713 de Texas Instruments. Ce symbole est utilisé pour conditionner le code source en fonction du processeur cible, permettant ainsi d'inclure ou d'exclure des parties spécifiques du code en fonction de la plateforme de développement.

4. Configuration du cible :

   Son rôle est de permettre la configuration spécifique de la plateforme matérielle DSK6713 lors du développement du code.Pour établir une nouvelle Target Configuration comme celle présentée sur la figure IV.7, on sélectionne View→Target Configuration.

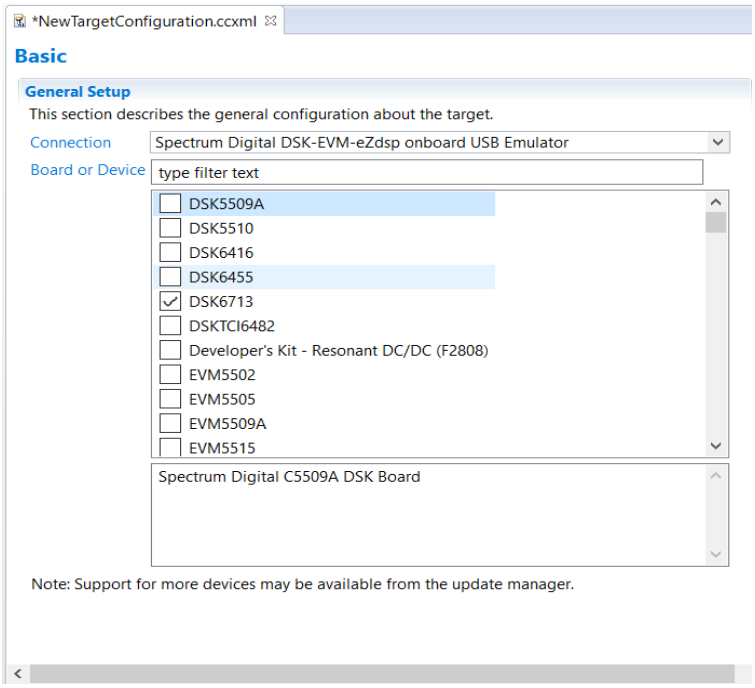


Figure IV.7 – Configuration du cible

IV.2 Bases de Données

Une chaîne de traitement du signal consiste à l'acquisition du signal, sa conversion en données numériques, et enfin son traitement(filtrage, compression, modulation…).
Le signal est acquis et converti en données numériques avant d'être traité. La figure IV.8 est un schéma simplifié d'une chaîne de mesure illustrant ce processus.
Ensuite, il peut être utilisé sous forme de données numériques ou converti à nouveau en format analogique, selon les besoins.

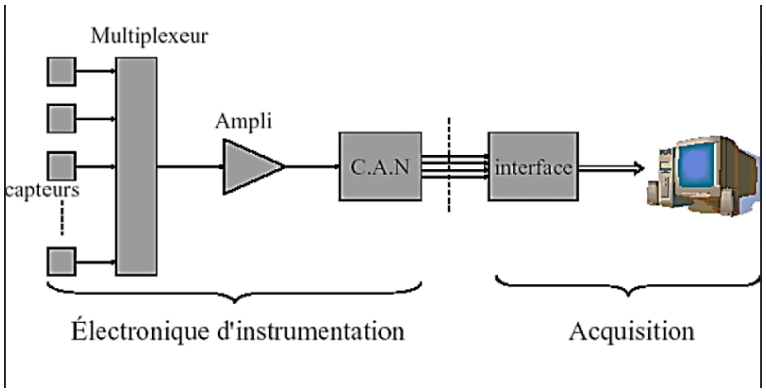


Figure IV.8 – Chaine de mesure

Dans ce présent travail, nous avons récupéré les signaux ECG à partir des bases de données mitdb [35] et ecgiddb[36] sur le répertoire Physionet.

Les bases de données jouent un rôle essentiel dans le domaine de la recherche biomédicale. Ces bases de donnée permettent un accès ouvert et gratuit à des enregistrements de longue durée et offre un vaste ensemble de données physiologiques.

IV.2.1 MIT-BIH Arrhythmia Database (mitdb)

La base de données MIT-BIH Arrhythmia (mitdb) [35] est une collection d'enregistrements d'électrocardiogrammes (ECG) provenant de patients atteints de divers types d'arythmies cardiaques. Chaque enregistrement contient des données ECG à 2 canaux, avec une fréquence d'échantillonnage de 360 Hz et une résolution de 11 à 12 bits. Chaque enregistrement est étiqueté avec des annotations précises indiquant les types spécifiques d'arythmies présents dans le signal ECG. Les signaux tirés de cette base de donné sont chosit soigneusement afin de traiter des cas particulier d'ECG.
L'enregistrement 109, indique une présence de Quinidine et révèle un bloc auriculo-ventriculaire du premier degré, ainsi que des PVC (contractions ventriculaires prématurées) multiformes et l'enregistrement 203 présente une variété de PVC de formes différentes de plus il est caractérisé par un bruit important, notamment des artefacts musculaires et des décalages de la ligne de base .

IV.2.2 ECG-ID Database (ecgiddb)

La base de données ecgiddb [36] propose des enregistrements ECG à plusieurs canaux, avec une fréquence d'échantillonnage de 250 Hz. Chaque enregistrement est associé à des informations démographiques et médicales (âge, sexe, etc) et comprend à la fois des signaux bruts et filtrés.

Les enregistrements tirés à partir de cette base de données sont choisit aléatoirement à l'exception du signal désiré. L'enregistrement brut du signal Person_09/rec_6 et l'enregistrement Person_90/rec_2, sont utilisé comme entrée du filtre et l'enregistrement filtré du signal Person_03/rec_1 est utilisé comme modèle (signal désiré).
Le signal modèle est celui d'une personne saine, illustré sur la figure IV.9

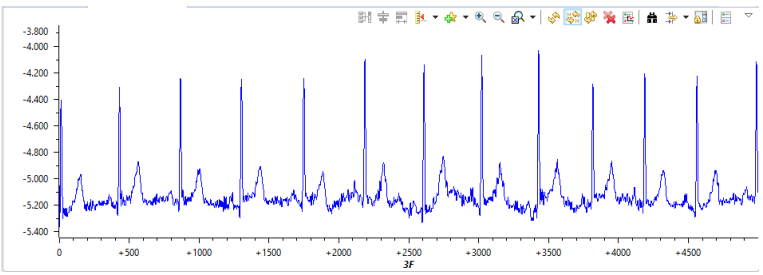


Figure IV.9 – Signal ECG désiré

IV.2.3 Utilisation des bases de donnée

Les signaux ont été téléchargés sous forme de fichiers .mat, puis convertis en utilisant un programme Matlab en fichiers d'en-tête.

En raison des descriptions des signaux abordées auparavant, les données de la base ecgiddb sont filtrées sans ajout de bruit, tandis que du bruit est ajouté au données de la base mitbih avant d'effectuer le filtrage.
Pour l'implémentation, des enregistrements d'une minute ont été considérés avec une haute précision, et les 5000 premières valeurs ont été sélectionnées parmi 21600.

IV.3 Implémentation des filtres adaptatifs

Discuté préalablement au chapitre I, l'idée principale est d'implémenter un filtre FIR à coefficients qui changent au cours du temps, en d'autre terme qui sont adapté et mise à jour en utilisant les équations des algorithmes adaptatifs.

L'approche pour la conception et l'implémentation de ces filtres adaptatifs sur le DSP opté pour est de coder en utilisant le langage c.

IV.3.1 Algorithmes de filtrage adaptatif utilisés

Dans le premier chapitre, une étude des algorithmes adaptatifs existants a été abordée et modélisée sur Matlab. En utilisant une approche similaire, les programmes C sont développés. Pour illustrer visuellement le fonctionnement de ces algorithmes, les diagrammes de flux correspondants sont présentés sur les figures IV.10, IV.11 et IV.12.

IV.3.1.1 LMS

Taux d'apprentissage μ

Dans un premier temps, le taux d'apprentissage μ doit satisfaire l'équation

$$0 < \mu < (\frac{1}{\lambda_(max)})$$

λ est la valeur propre maximale de la matrice d'auto-corrélation $E(x^2(n)) = R$
Pour les signal ECG choisi, la valeur maximale a été calculée par MATLAB .

$$0 < \mu < 0.000015312$$

En se basant sur cette information, plusieurs exécutions ont été effectuées en pratique en modifiant la valeur de mu pour trouver la valeur optimale.Ce était fait en observant la convergence et stabilité des graphes obtenus.

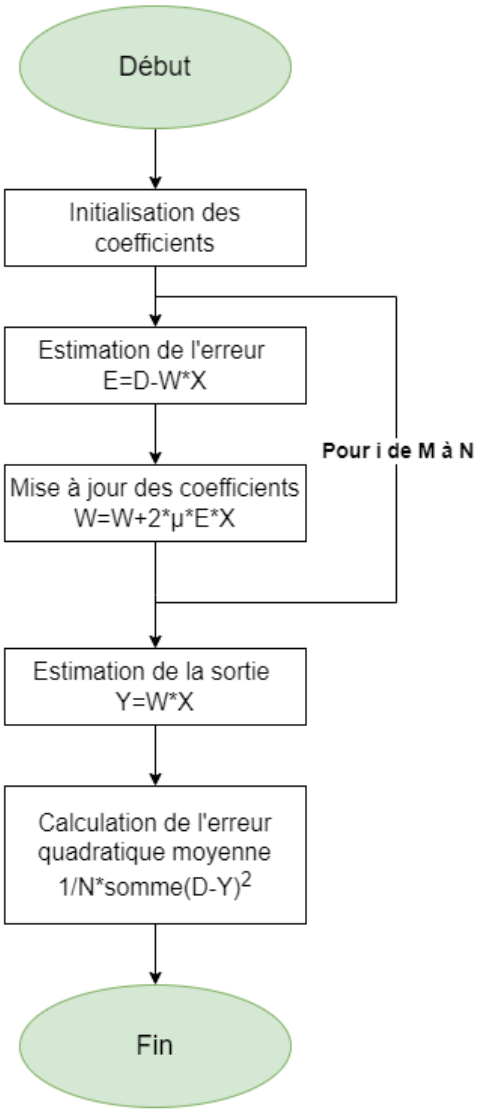


Figure IV.10 – Diagramme de flux de l'algorithme LMS

IV.3.1.2 NLMS

Facteur de normalisation ϵ

Il est généralement choisi comme une valeur très petite. Un choix courant pour ϵ est une valeur de l'ordre de 10^{-6} .

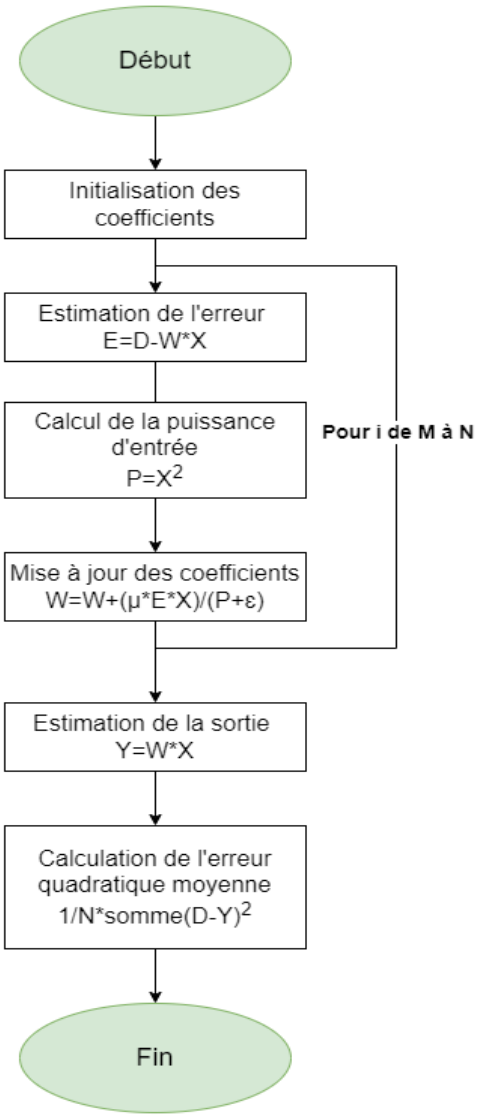


Figure IV.11 – Diagramme de flux de l'algorithme NLMS

IV.3.1.3 RLS

Facteur d'oubli λ

Lorsque λ est choisi comme une valeur inférieure à 1, cela crée un schéma qui accorde une plus grande importance aux échantillons les plus récents et tend à oublier progressivement les échantillons passés.

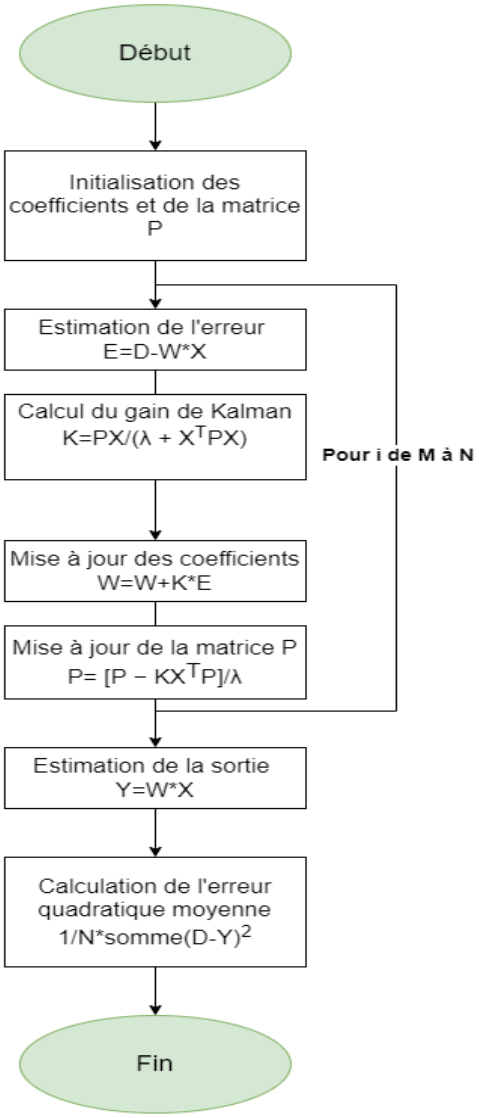


Figure IV.12 – Diagramme de flux de l'algorithme RLS

IV.3.2 Implémentation sur le DSP6713

Pour chaque algorithme un projet sur CCS est créé et configuré de manière adéquate avant que son programme principal (main) ne soit compilé. Une fois que la compilation est correctement effectuée, sans erreur (voir sur la figure IV.13), une session de débogage du projet est entreprise.

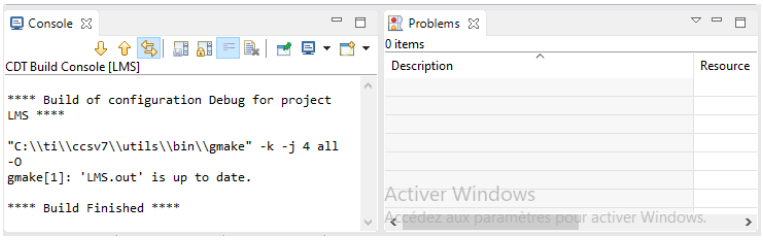


Figure IV.13 – Compilation sans erreurs d'un projet sur ccs

Plusieurs sessions de débogage sont effectuées pour un même projet dans le but de tester divers paramètres, d'évaluer les performances et d'optimiser les filtres numériques afin de tirer le meilleur parti du travail réalisé.

Les résultats des programmes peuvent être pleinement exploités grâce aux fonctionnalités avancées offertes par le CCS mentionnées auparavant.

Le dispositif matériel utilisé pour la mise en place de l'implémentation est illustré sur la figure IV.14.

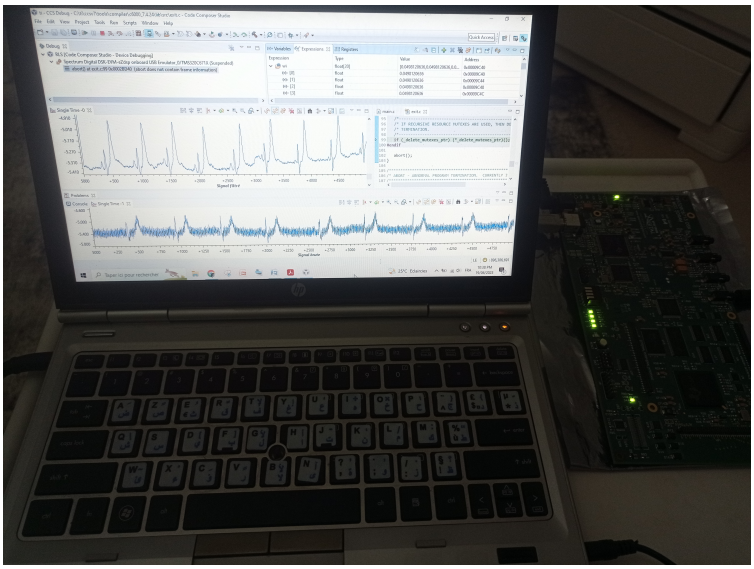


Figure IV.14 – dispositif matériel utilisé pour la mise en place de l'implémentation

IV.4 Résultats et Evaluation

Dans cette section , nous présentons et nous évaluons les résultats.

IV.4.1 Approches d'évaluation

Comme expliqué précédemment dans la section IV.2.3, deux approches de filtrage ont été utilisées :

Le modèle final implémenté pour filtrer les signaux pris de la base de données MIT-BIH Arrhythmia Database (mitdb) est illustré dans la figure IV.15

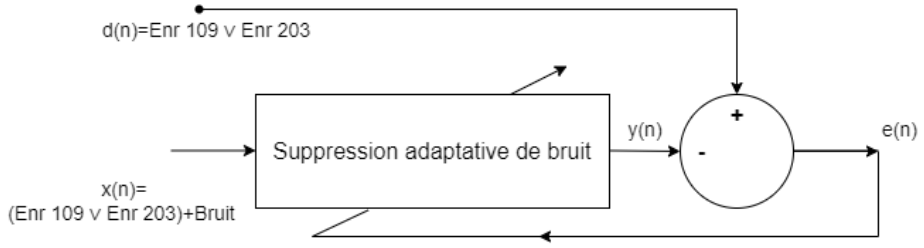


Figure IV.15 – Modèle final de suppression de bruit des signaux pris de la BDD mitdb

Le modèle final implémenté pour filtrer les signaux pris de la base de données ECG-ID Database (ecgiddb) est illustré dans la figure IV.16

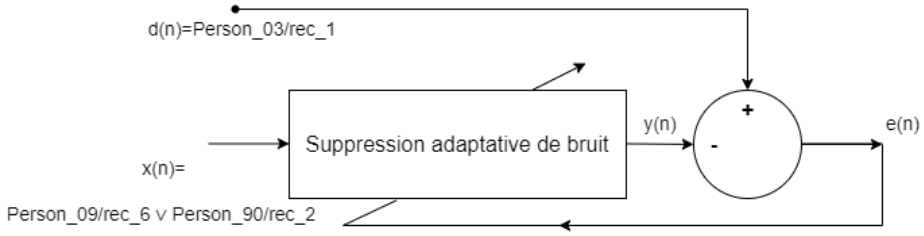


Figure IV.16 – Modèle final de supression de bruit des signaux pris de la BDD ecgiddb

IV.4.2 Critères d'évaluation

Deux critères sont utilisés pour évaluer les résultats des filtres et l'efficacité des algorithmes. Une analyse temporelle en utilisant l'erreur quadratique moyenne (EQM), l'effet apporté par l'ordre du filtre et le temps d'exécution des algorithmes ; Ainsi qu'une analyse fréquentielle faisant appel aux spectrogrammes des différents signaux. En outre on demande un avis médical pour se renseigner sur la qualité des résultats produits.

IV.4.3 Analyse temporelle

IV.4.3.1 Erreur Quadratique Moyenne EQM

EQM est une mesure couramment utilisée pour évaluer la performance des algorithmes de filtrage. C'est une mesure de la différence moyenne au carré entre les valeurs estimées produites par l'algorithme de filtrage et les valeurs désirées.

$$EQM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{IV.1}$$

ou :

- y_i représente les valeurs désirées,
- $\hat{y}_i$ représente les valeurs estimées produites par l'algorithme de filtrage,
- n est le nombre total d'échantillons.

Le taux d'apprentissage pour le programmes LMS et NLMS sont fixé à 0.00001 et à 0.005 respectivement. Les résultats sont notés dans le tableau IV.1.

EQM					
	Ordre du filtre	Person_90/rec_2	Person_09/rec_6	109	203
LMS	M= 25	0.174	0.172	0.199	0.258
LMS	M= 37	0.161	0.114	0.155	0.269
LMS	M= 60	0.048	0.046	0.104	0.165
NLMS	M= 25	0.174	0.151	0.196	0.261
NLMS	M= 37	0.157	0.096	0.159	0.278
NLMS	M= 60	0.030	0.028	0.116	0.178

Table IV.1 – Résultas du paramétre MSE pour l'algorithme LMS et NLMS pour ordre de filtre 25, 37 et 60

Les valeurs d'erreur quadratique moyenne sont légèrement plus faibles pour les filtres d'ordre plus grand.Néanmoins, les résultats restent globalement satisfaisants pour les deux algorithmes.

La moyenne des résultats obtenue pour un ordre de filtre de 37 appliqué sur la base de données ecgiddb (signaux brute) est de 0.126 pour NLMS et de 0.137 pour LMS. Dans [37] la valeur EQM moyenne pour un bruit composite (considéré brute) est de 0.1367 pour le filtre de Weiner et de 0.159 pour Kalman. Cette comparaison montre que le filtrage NLMS peut se montrer plus performant que l'approche Weiner, Kalman et LMS.

Nous avons également implémenté le filtre RLS sur PC et sur carte DSP,nous avons constaté qu'il présentait des performances limitées. En effet, sa convergence dépend fortement du choix de l'ordre et paramètres du filtre , cela est conforme aux résultats dans [38]. Par conséquent, dans notre analyse ultérieure, nous nous concentrerons sur les filtres LMS (Least Mean Squares) et NLMS (Normalized Least Mean Squares), pour lesquels nous avons obtenu des résultats significatifs.

IV.4.3.2 Ordre du filtre

Pour une interprétation plus approfnodie, le cas du signal _09/rec_6 est étudié à titre d'exemple.

Les graphiques affichant la forme brute du signal, sa version filtrée tirée de la base de données, ainsi que celle filtrée par notre filtre numérique LMS d'ordre 37 et d'ordre 60, sont présentés dans les figures IV.17, IV.4.4 IV.19, IV.20 respectivement.

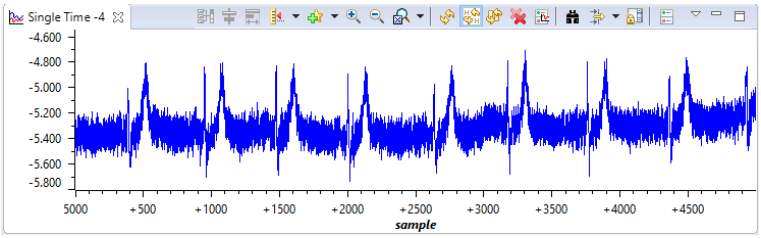
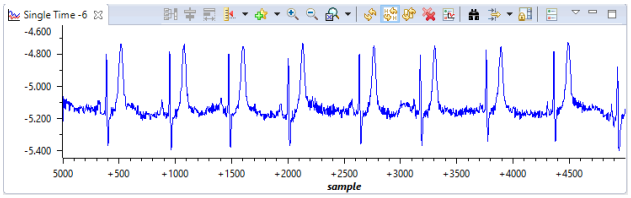
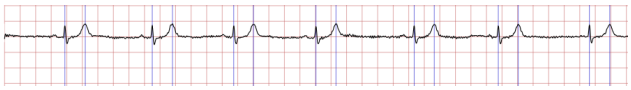


Figure IV.17 – Signal ECG brute



a) Visualisé sur CCS



b)Visualisé par LightWave

Figure IV.18 – Signal filtré tiré de la base de données

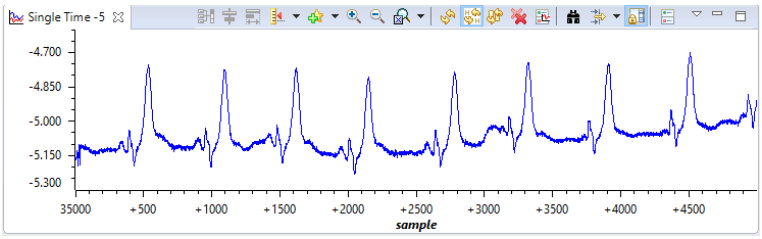


Figure IV.19 – Signal ECG filtré par un filtre LMS d'ordre 37

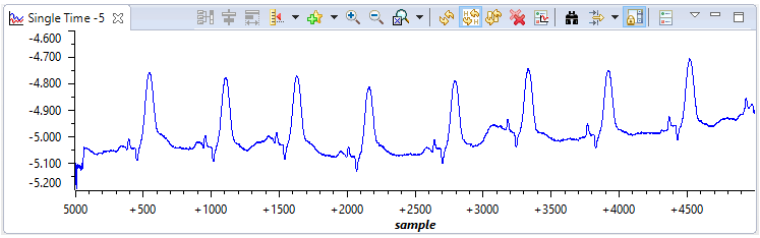


Figure IV.20 – Signal ECG filtré par un filtre LMS d'ordre 60

Il est notable que plus l'ordre du filtre augmente, plus la convergence est rapide et le débruitage devient plus fort.

Pour assurer un compromis entre la préservation des caractéristiques du signal, l'obtention d'un bon filtrage et une optimisation par rapport à la complexité du calcul, on choisit un ordre moyen de 37.

Un tracé ECG de bonne qualité doit permettre une lecture aisée des différentes ondes qui compose le signal. Un personnel médical nous a affimré que la visualisation de l'onde P à partir de IV.19 a été amélioré. D'une part le signal a été débruité mais on parvient aussi à généré un graphique de bonne qualité.

IV.4.3.3 Temps d'exécution

Le CCS offre la possibilité de calculer les cycles d'horloge nécessaires pour exécuter le programme. Il suffit d'activer le clock à partir de Run→Clock→Enable lors d'une session de débogage.
En utilisant cette fonctionnalité on cherche à comparer le temps d'exécution pour les deux algorithmes LMS et NLMS, les paramètres des algorithmes sont fixé et le signal 109 est utilisé.

Pour convertir les cycles d'horloge en secondes, on peut utiliser l'équation suivante :

$$\boxed{\text{Temps en secondes} = \text{Cycles d'horloge} \times \text{Période d'horloge}}$$

Mentionnée dans le chapitre 3, la période d'horloge du TMS320C6713 est de 225MHz.
Les résultats sont présentés dans le tableau IV.2suivant

	Ordre du filtre	Temps en Secondes	Nombre de Cycles d'horloge
LMS	M= 37	0.204	45977600
LMS	M= 60	0,294	66366605
NLMS	M= 37	1,71	384882461
NLMS	M= 60	2.73	614509551

Table IV.2 – Temps d'exécutions en cycle d'horloge et en secondes

- L'observation des temps d'exécution des algorithmes LMS et NLMS révèle que l'algorithme LMS est plus rapide à atteindre la convergence que l'algorithme NLMS.Cette différence de temps peut être expliquée par les variations dans les calculs et les mises à jour des poids du filtre adaptatif entre les deux algorithmes.

- Nous observons aussi que l'augmentation de l'ordre du filtre a un impact léger sur le temps d'exécution.
  En effet, à mesure que l'ordre du filtre augmente, le temps nécessaire pour effectuer les calculs et les opérations de filtrage augmente également. Cependant, cette augmentation reste modeste sur le DSP, ce qui indique que l'impact sur le temps d'exécution reste satisfaisant même avec des filtres d'ordre plus élevé.

IV.4.4 Analyse fréquentielle

En utilisant la fonction **specgram** de Matlab®, il est possible d'effectuer une analyse fréquentielle dépendante du temps qui fournit la densité de puissance du signal, elle représente visuellement les variations de fréquences dans le temps [39].

La Figure IV.21 présente trois spectrogrammes comparatifs d'un signal ECG 109, mettant en évidence différentes étapes de traitement. Le premier spectrogramme correspond au signal ECG 109 d'origine. Le deuxième spectrogramme représente le même signal, altéré par :

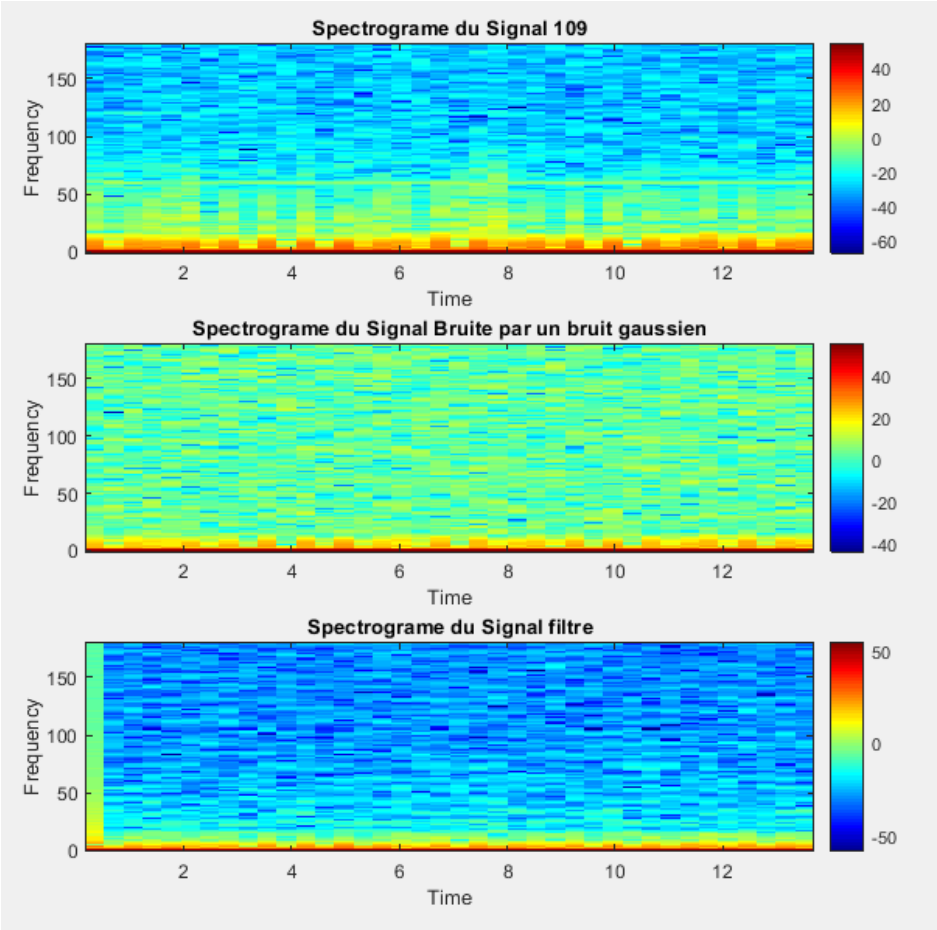
(a) Le Bruit gaussien.

(b) L'errance de base.

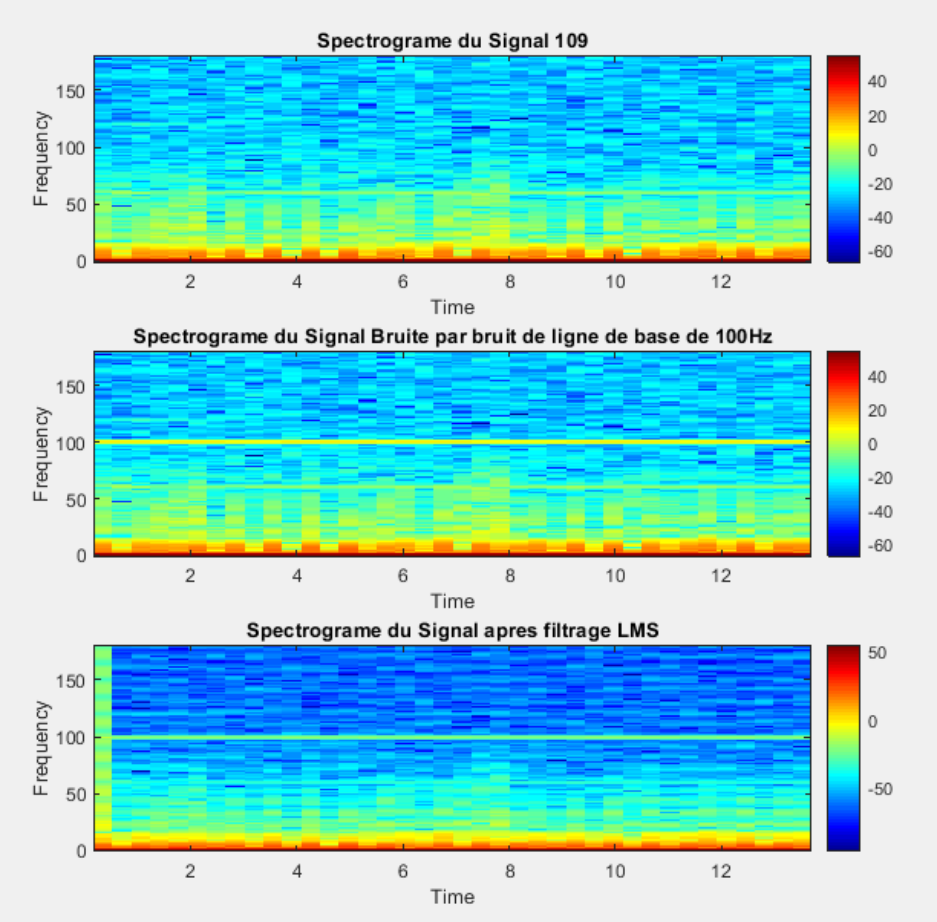
(c) L'interférence des lignes électriques.

(d) Le bruit EMG.

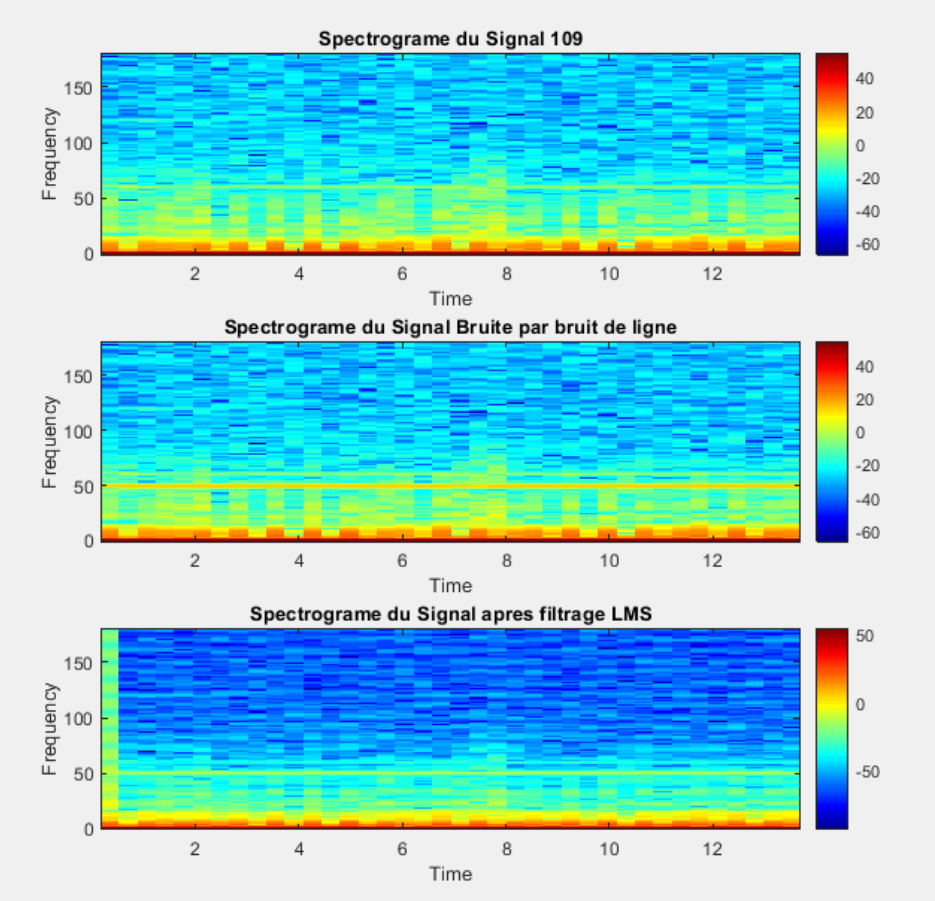
Enfin, le troisième spectrogramme présente le résultat du filtrage appliqué au signal bruité par LMS pour un ordre de filtrage de N=37, mu= 0.00001.



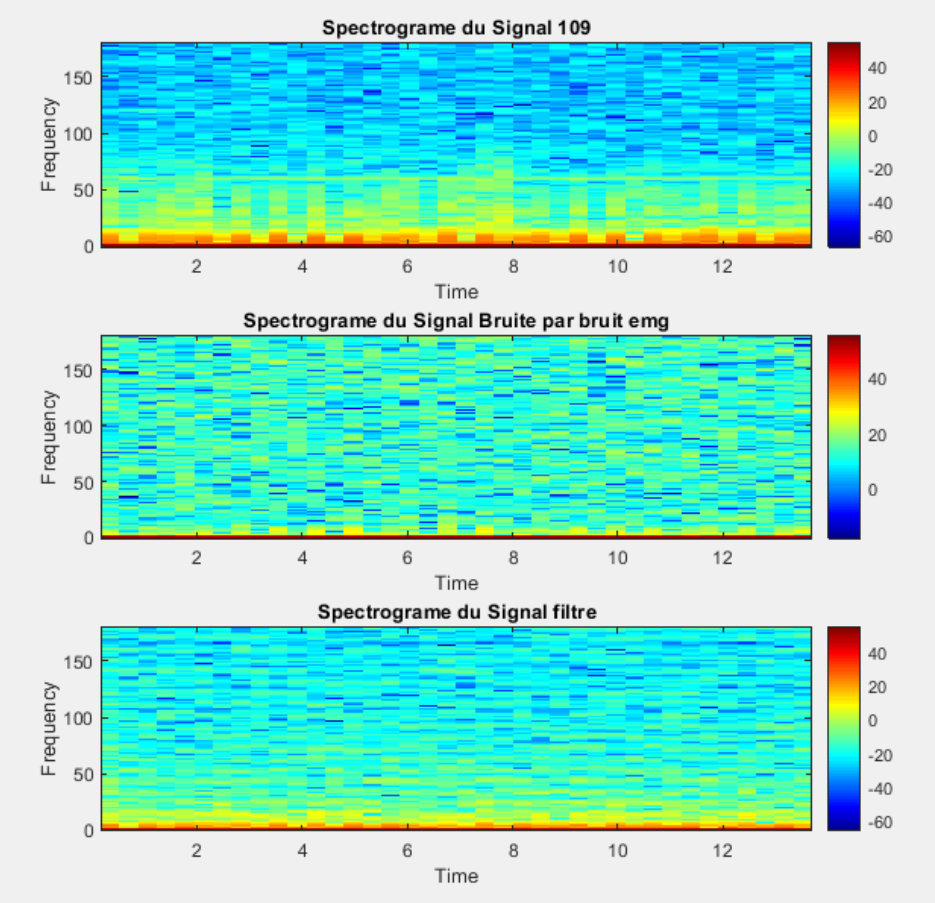
a) Bruit gaussien



b)Errance de base



c) Interférence des lignes électriques



d)Bruit EMG

Figure IV.21 – Sperctrogrammes du signal 109, le signal 109+bruit et le signal résultant après un filtrage LMS

— Le bruit gaussien et le bruit EMG apparaissent le long du spectre de fréquences et altèrent le spectrogramme du signal.Cependant, après le filtrage le bruit gaussien est complètement éliminé et le signal d'origine est presque entièrement récupéré, tandis que le bruit EMG persiste toujours.
— L'errance de base et l'interférence des lignes électriques sont visible comme des lignes distinctes autour des 100 Hz et 50 Hz respectivement. Après le filtrage, une légère altération dans les fréquences dominantes est observée au début du signal, due aux effets transitoires du filtrage. Ainsi, les deux composantes indésirables sont considérablement atténuées.
— De plus, on peut observer la conservation des informations pertinentes du signal. Les caractéristiques importantes, telles que les pics de fréquence ou les motifs distincts, demeurent présentes dans le spectrogramme filtré, indiquant que les informations essentielles du signal sont préservées malgré le processus de filtrage. Cela confirme l'efficacité du filtrage pour éliminer les composantes indésirables tout en maintenant les informations pertinentes du signal.

IV.4.5 Envoie des donnée vers le codec

On envisage dans ce qui suit à envoyer le signal filtré vers le codec de la carte DSP afin de reproduire le signal et l'exploiter sous sa forme analogique. La visualisation de ce dernier sur un logiciel d'analyse et de visualisation de signaux audio (Zelscope)[1]est présenté sur la figure IV.22.

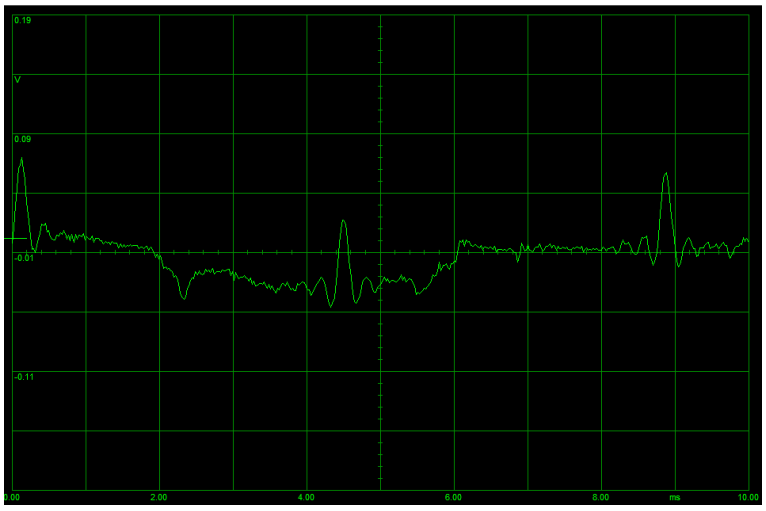


Figure IV.22 – Signal Person_09/rec_6 visualisé sur Zelscope après filtrage NLMS

1. https ://www.zelscope.com

Conclusion

Dans ce chapitre pratique, nous avons réalisé l'implémentation de filtres adaptatifs sur le DSKC6713 dans le but de débruiter des signaux ECG provenant de bases de données.

Nous avons d'abord expliqué le logiciel utilisé et les conditions requises pour mener à bien notre travail. Ensuite, nous avons présenté en détail notre approche, les algorithmes utilisés et les résultats obtenus, évalués selon plusieurs critères.

Il convient de noter que l'utilisation du filtre RLS ne s'adapte pas adéquatement aux signaux ECG, limitant ainsi son efficacité dans ce contexte spécifique. Malgré cela, nous avons réussi à procéder au filtrage des signaux avec succès. Une fois le filtrage effectué, nous avons également envoyé le résultat vers le codec de la carte pour démontrer la possibilité de le convertir en signal analogique et l'afficher sur un moniteur externe, en l'occurrence un oscilloscope .

Conclusion Générale

L'utilisation d'applications embarquées dédiées au télédiagnostique offre de nombreux avantages, notamment le suivi constant de l'état cardiaque et la possibilité de fournir des soins médicaux à distance. Dans ce contexte, notre travail a été axé sur le développement et l'implémentation sur DSP d'une solution de filtrage des signaux ECG basée sur des techniques de filtrage adaptatif.

Dans le cadre de notre travail, nous avons implémenté trois algorithmes de filtrage adaptatif(LMS,NLMS et RLS)à la fois sur un ordinateur personnel (sous Matlab) et sur une carte de traitement du signal numérique (TMS320C6713). L'évaluation de ces algorithmes a été réalisée en calculant l'erreur quadratique moyenne (EQM), étudier l'effet qu'apporte l'ordre du filtre sur le filtrage et en effectuant une analyse fréquentielles de spectres. De plus on tient compte des types de bruit qui ont pu altérer le signal.

Les résultats obtenus lors de notre étude ont démontré l'efficacité de notre approche. Nous avons réussi à implémenter des filtres adaptatifs capables de traiter les signaux ECG bruts de manière appropriée. Cependant, il convient de noter que, contrairement aux autres filtres, l'utilisation du filtre RLS ne s'adapte pas aux signaux ECG. Malgré cela, les performances des autres algorithmes ont été satisfaisantes, mettant en évidence le potentiel du filtrage adaptatif dans le prétraitement des signaux biomédicaux.

Cependant, le travail peut être étendu pour implémenter d'autres algorithmes, tels que les ondelettes adaptatives et établir une comparaison plus large entre les solutions possibles. Ces améliorations pourraient potentiellement faire l'objet d'une nouvelle étude.

Bibliographie

1. BOUKADOUM, Mounir ; MÉNARD, Michaël. *Filtres à réponse impulsionnelle infinie (RII),Support de cours*. 2019. Aussi disponible à l'adresse : `http://www.info2.uqam.ca/~boukadoum_m/MIC4220/Notes/7-MIC4220_RII_prof.pdf`. Consulté en Février 2023.
2. BOUKADOUM, Mounir ; MÉNARD, Michaël. *Filtres à réponse impulsionnelle finie (RIF), Support de cours*. 2019. Aussi disponible à l'adresse : `http://www.info2.uqam.ca/~boukadoum_m/MIC4220/Notes/7-MIC4220_RII_prof.pdf`. Consulté en Février 2023.
3. BRIZON, Ambre. *Compréhension et gestion des signaux faibles dans le domaine de la santé-sécurité*. 2009. Thèse de doctorat.
4. BERCHER, J.F. ; JARDIN, P. *Introduction au filtrage adaptatif,Support de cour*. 2003. Ecole de l'Innovation Technologique,2003.
5. IFEACHOR, Emmanuel C. ; JERVIS, Barrie W. *Digital Signal Processing : A Practical Approach*. 2nd. Pearson Education, 2002. isbn 0201596199.
6. HAYKIN, S. *Adaptive Filter Theory*. Prentice-Hall, Inc., 1986. isbn 0130040525.
7. PRATT, W. K. *Generalized Wiener Filtering Computation Techniques*. 1972.
8. TAN, Lizhe ; JIANG, Jean. *Digital Signal Processing : Fundamentals and Applications*. 3e éd. Academic Press, 2013.
9. WIENER, Norbert. *Information Theory*. T. 4. 1919. vol. 4, p. 249.
10. DOGARIU, Laura-Maria ; BENESTY, Jacob ; PALEOLOGU, Constantin. An Insightful Overview of the Wiener Filter for System Identification. *Appl. Sci.* 2021, t. 11, no 17, p. 7774. Disp. à l'adr. doi : `10.3390/app11177774`.
11. FARHANG-BOROUJENY, Behrouz. *Adaptive Filters : Theory and Applications*. 2e éd. Wiley, 2013.
12. SHENASA, Mohammad ; JOSEPHSON, Mark E. ; ESTES III, N.A. Mark. *The ECG Handbook of Contemporary Challenges*. Cardiotext Publishing, 2015.
13. WAGNER, Galen S. ; STRAUSS, David G. *Marriott's Practical Electrocardiography*. 12e éd. Lippincott Williams Wilkins, 2014.
14. ENDERLE, John D. ; BRONZINO, Joseph D. *Introduction to Biomedical Engineering*. Academic Press. Academic Press, 2012.
15. GOLDBERGER, Ary L. ; GOLDBERGER, Zachary D. ; SHVILKIN, Alexei. *Goldberger's Clinical Electrocardiography : A Simplified Approach*. 2017.

16. MATOS, André Cigarro ; LOURENÇO, André ; NASCIMENTO, José. Embedded System for Individual Recognition Based on ECG Biometrics. *Procedia Technology*. 2014, t. 17, p. 265-272. issn 2212-0173. Disp. à l'adr. doi : https://doi.org/10.1016/j.protcy.2014.10.236.

17. DONG, Jianwei ; ZHANG, Shi ; JIA, Xiaonan. A Portable Intelligent ECG Monitor Based on Wireless Internet and Embedded System Technology. 2008, t. 2, p. 553-556. Disp. à l'adr. doi : 10.1109/BMEI.2008.65.

18. SHIH, Dong-Her ; CHIANG, Hsiu-Sen ; LIN, Binshan ; LIN, Shih-Bin. An Embedded Mobile ECG Reasoning System for Elderly Patients. *IEEE Transactions on Information Technology in Biomedicine*. 2010, t. 14, n° 3, p. 854-865. Disp. à l'adr. doi : 10.1109/TITB.2009.2021065.

19. CHATTERJEE, Shubhojeet ; THAKUR, Rini Smita ; YADAV, Ram Narayan. Review of Noise Removal Techniques in ECG Signals. 2020.

20. LIMAYE, Hrishikesh ; DESHMUKH, V.V. ECG Noise Sources and Various Noise Removal Techniques : A Survey. 2016.

21. KHER, Rahul. Signal Processing Techniques for Removing Noise from ECG Signals. 2019.

22. NAYAK, Seema ; SONI, M.K. ; BANSAL, Dipali. Filtering Techniques for ECG Signal Processing. 2012.

23. SORNMO, Leif ; LAGUNA, Pablo. Electrocardiogram (ECG) Signal Processing. 2006.

24. POUNGPONSRI, Suranai ; YU, Xiao-Hua. An adaptive filtering approach for electrocardiogram (ECG) signal noise reduction using neural networks. *Neurocomputing*. 2013, t. 117, p. 206-213. issn 0925-2312. Disp. à l'adr. doi : https://doi.org/10.1016/j.neucom.2013.02.010.

25. LAGUNA, Pablo ; JANÉ, Raimon ; CAMINAL, Pere. Adaptive filtering of ECG baseline wander. 1992, t. 2, p. 508-509. Disp. à l'adr. doi : 10.1109/IEMBS.1992.5761083.

26. THAKOR, N.V. ; ZHU, Y.-S. Applications of adaptive filtering to ECG analysis : noise cancellation and arrhythmia detection. *IEEE Transactions on Biomedical Engineering*. 1991, t. 38, n° 8, p. 785-794. Disp. à l'adr. doi : 10.1109/10.83591.

27. BAJOT, Yann ; MEHREZ, Habib. Les systèmes de traitement numérique du signal. 2001.

28. CORMIE, Gabriel. *GELE5343 Chapitre 5 : Architectures*. 2013.

29. INSTRUMENTS, Texas. *TMS320C28x CPU and Instruction Set, Reference Guide*. 2015. Aussi disponible à l'adresse : https://www.ti.com/lit/ug/spru430f/spru430f.pdf.

30. TA, Jithin ; KB, Mohamed Salih K ; RC, Jayan A. Real Time Suppression of Howling Noise in Public Address System. 2015.

31. MANIKANDAN, J. ; VENKATARAMANI, B. ; GIRISH, K. ; KARTHIC, H. ; SIDDHARTH, V. Hardware Implementation of Real-Time Speech Recognition System Using TMS320C6713 DSP. In : *2011 24th Internatioal Conference on VLSI Design*. 2011, p. 250-255. Disp. à l'adr. doi : 10.1109/VLSID.2011.12.

32. LYSIYA Nethravathi, Harini. SCR technique for tumour operation using biomedical image processing and its implementation on TMS320C6713. *i-manager's Journal o Image Processing*. 2015.

33. CHASSAING, Rulph. *Digital Signal Processing and Applications with the C6713 and C6416 DSK*. Wiley-Interscience, 2004.

34. SPECTRUM DIGITAL, INC. *TMS320C6713 DSK Technical Reference*. 2003.

35. MOODY, GB ; MARK, RG. The impact of the MIT-BIH Arrhythmia Database. *IEEE Eng in Med and Biol*. 2001, t. 20, n° 8, p. 45-50.

36. LUGOVAYA, T.S. *Biometric human identification based on electrocardiogram*. 2005.

37. B, Manju ; M, Sneha. ECG Denoising Using Wiener Filter and Kalman Filter. *Procedia Computer Science*. 2020, p. 273-281.

38. AL-QAZZAZ, Noor. Comparison of the RLS and LMS Algorithms to Remove Power Line Interference Noise from ECG Signal. 2010.

39. HAQUE, A.K.M Fazlul ; ALI, Md. Hanif ; KIBER, Md. Adnan. Improved spectrogram analysis for ECG signal in emergency medical applications. *International Journal of Advanced Computer Sciences and Applications*. 2010, p. 39-44.

Annexe1 : Codes utilisés

```
%%============================%
clc;
clear all;
close all;
%%=====Weiner filter PROGRAM: applied on a sine wave
   ↪ ======================%
t= 0.001:0.001:0.5;

x= 2*sin(2*pi*5*t);                %Generate desired signal

%% Add noise to the signal
rng(0);                            % set random seed for
   ↪ reproducibility
n = 0.5*rand(size(x));            % generate Gaussian noise
y = x + n;

%% Estimate power spectrum of signal and noise
Px = periodogram(x);
Pn = periodogram(n);

%% Estimate signal-to-noise ratio (SNR)
SNR = Px./Pn;

%% Set hyperparameters for Weiner filter
lambda = 0.9;                      % regularization parameter
alpha = 0.5;                       % scaling parameter

%% Compute Weiner filter coefficients
%************ mean square error (MSE) approximative equation
   ↪ **************%
h = conj(SNR)./(conj(SNR) + 1/lambda);          %h = Rxy * inv(
   ↪ Rxx)         ; Rxx = conj(SNR) + 1/lambda
h = alpha * h;

%% Apply Weiner filter to noisy signal
y_hat = filter(h, 1, y);
y_resulted = y_hat * (max(x) / max(y_hat));

%% Computing Error signal
Err = y_resulted-x;

%% Plotting all the signals
```

```
figure();
t1=t*1000;
maxt=max(t1);
subplot(4,1,1); plot(t1,x,'LineWidth',1.25); ylim ([-3 3])
grid on;    title('Signal Original'); ylabel ('S(t)'); xlabel ('
   ↪ Temps (Mili-seconde)')
subplot(4,1,2); plot(t1,y,'LineWidth',1.25);ylim ([-3 3])
grid on;    title('Signal avec bruit');ylabel ('Sb(t)'); xlabel (
   ↪ 'Temps (Mili-seconde)')
subplot(4,1,3); plot(t1,y_resulted,'LineWidth',1.25);ylim ([-3
   ↪ 3])
grid on;    title('Signal apres filtrage');ylabel ('S1(t)');
   ↪ xlabel ('Temps (Mili-seconde)')
subplot(4,1,4); plot(t1,Err,'LineWidth',1.25);
grid on;    title('Erreur');ylabel ('Eb(t)'); xlabel ('Temps (
   ↪ Mili-seconde)')
```

```
%%==================================%
clc;
clear all;
close all;
%%=====RLS PROGRAM: applied to a sine wave
   ↪ ================================%
t= 0.001:0.001:0.5;
x = 2*sin(2*pi*20*t);                    %Generate desired signal

% Filter Parameters
order = 25;              % Filter order
lambda = 0.98;          % Forgetting factor
delta = 0.1;            % Regularization factor

% Add noise to the input signal
noise = 0.8* rand(size(x));
y = x + noise;
% Initialization
N = length(y);
weights = zeros(order, 1);
P = delta*eye(order);                % inverse of the
   ↪ autocorrelation matrix

outputSignal = zeros(N, 1);
estimationError = zeros(N, 1);
%% RLS Filtering
for n = 1:N
```

```
    inputVector = zeros(order, 1);        %set to null at the
       ↪ beginning of each iteration
    for i = 1:order
        if n - i > 0
            inputVector(i) = y(n - i);
        end
    end

    prediction = weights' * inputVector;        %estimate
       ↪ predection
    error = y(n) - prediction;                  %estimate error
    %Calculate the gain vector
    k = (P * inputVector) / (lambda + inputVector' * P *
       ↪ inputVector);

    weights = weights + k * conj(error);
    P = (P - k * inputVector' * P) / lambda;

    outputSignal(n) = prediction;
    Err(n) = error;
end

%% Plotting all the signals
figure();
t1=t*1000;
subplot(4,1,1); plot(t1,x,'LineWidth',1.25);
grid on;   title('Signal Original');ylabel ('S(t)'); xlabel ('
   ↪ Temps (Mili-seconde)')
subplot(4,1,2); plot(t1,y,'LineWidth',1.25);
grid on;   title('Signal avec bruit');ylabel ('Sb(t)'); xlabel ('
   ↪ Temps (Mili-seconde)')
subplot(4,1,3);  plot(t1,outputSignal,'LineWidth',1.25);
grid on;    title('Signal apres filtrage');ylabel ('S.est(t)');
   ↪ xlabel ('Temps (Mili-seconde)')
subplot(4,1,4);  plot(t1,Err,'LineWidth',1.25); ylabel ('Err(t)')
   ↪ ; xlabel ('Temps (Mili-seconde)')
grid on;    title('Erreur');
```

```
    %%============================%
clc;
clear all;
close all;
%%=====LMS filter PROGRAM: applied to a sine wave
   ↪ ========================%
```

```
t= 0.001:0.001:0.5;
x= 2*sin(2*pi*5*t);                          %Generate desired signal

%% Generate the signal with noise
n= numel(x);
A= x(1:n)+ 0.4*randn(1,n);
%% Set filter parameters and compute filter coefficients
M= 60;
w= zeros(1,M);          %initializes the filter coefficients
   ↪ vector with M zeros.
wi= zeros(1,M);         %initializes the weight update vector wi
   ↪ with M zeros
E= [];                  %initializes vector to store error
mu= 0.0005;             %set learning rate
for i=M:n
    E(i)= x(i) - wi*A(i:-1:i-M+1)';       %%estimate filter error
    wi= wi + 2*mu*E(i)*A(i:-1:i-M+1);     %%update weighs
end
%% Estimating the signal
Est= zeros(n,1);                             %initializes vector to
   ↪ store error
for i=M:n
    j= A(i:-1:i-M+1);                        %extract noisy signal
       ↪ samples
    Est(i)= ((wi)*(j)');                     %run noisy signal through
       ↪  the filter
end

%% Computing Error signal
Err = Est'-x;
t1=t*1000;
%% Plotting all the signals
subplot(4,1,1), plot(t1,x,'LineWidth',1.25),grid on;
title('Signal Original');ylabel ('S(t)'); xlabel ('Temps (Mili-
   ↪ seconde)')
subplot(4,1,2), plot(t1,A,'LineWidth',1.25),grid on;ylabel ('Sb(t
   ↪ )'); xlabel ('Temps (Mili-seconde)')
title('Signal avec bruit');
subplot(4,1,3), plot(t1,Est,'LineWidth',1.25),grid on;ylabel ('S.
   ↪ est(t)'); xlabel ('Temps (Mili-seconde)')
title('Signal apres filtrage');
subplot(4,1,4), plot(t1,Err,'LineWidth',1.25), grid on;ylabel ('
   ↪ Err(t)'); xlabel ('Temps (Mili-seconde)')
title('Erreur');
```

```
 //====================LMS============================/
/*This code generates diffrent types of noise wich are added to signal 109 from mitbih Data base*/
/*******************Include*****************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "ECG109.h"
/*********************Fonctions************************/
double generateBaselineWander(double amplitude, double samplingRate,
                              int sampleIndex) {
  double frequency = 100; // frequence
  return amplitude * sin(2.0 *3.14 * frequency * sampleIndex / samplingRate);
}


double generatePowerLineNoise(double frequency, double amplitude,  double samplingRate, int sampleIndex)
{
    return amplitude * sin(2.0 *3.14 * frequency * sampleIndex / samplingRate);
}

double generateEMGNoise()
{
    return ((double)rand() / RAND_MAX) * 2.0 - 1.0; // entre -1 et 1
}
/*******************Define*****************************/
#define N 5000        // Length of the ECG signal
#define M 37
#define mu  0.00001
/*******************Variables*****************************/
float  wi[M],E[N],Est[N],error[N];
float  A[N];
/************main code ************/
void main() {
      float mse=0;        //initialiser l'erreur quadratique moyenne
```

```
36      long i,j,k;          //var to run loops
37      double amplitude = 0.1;       // Amplitude
38      double samplingRate = 360.0; // frequence d'échantillonnage
39      double frequency = 50.0;       // Frequence
40
41
42  //Generate the signal with noise
43      srand(time(NULL));
44      /*****************************Gaussian Noise
45          for (i = 0; i < N; i++) {
46              A[i] = ECGsignal[i] + 0.4 * rand() / RAND_MAX;
47          }*/
48      /*****************************Baseline Noise
           ↪ ****************************/
49      /*
50
51
52      for ( i = 0; i < N; i++) {
53          double noise = generateBaselineWander(amplitude,
               ↪ samplingRate, i);
54          //printf("%f\n", noise);
55          A[i] = ECGsignal[i] + noise;
56        }*/
57        /*****************************Baseline Noise
             ↪ ****************************/
58        for (i = 0; i < N; i++){
59              //  double noise = generatePowerLineNoise(frequency
                   ↪ , amplitude, samplingRate, i);
60                double noise = generateEMGNoise();
61                A[i] = ECGsignal[i] + noise;
62            }
63    // Initialize filter coefficients vector with M zeros
64    for ( i = 0; i < M; i++) {
65        wi[i] = 0;
66    }
67
68    // Run loop M-n time
69    for ( i = M; i < N; i++) {
70        // Estimate filter error
71        E[i] = ECGsignal[i];
72        for ( j = 0; j < M; j++) {
73            E[i] -= wi[j] * A[i - j];
74        }
75
```

```
76          // Update weights
77          for (j = 0; j < M; j++) {
78              wi[j] += 2 * mu * E[i] * A[i - j];
79          }
80      }
81
82
83      // Estimating the signal
84      for ( i = M; i < N; i++) {
85          float j[M];
86          for ( k = 0; k < M; k++) {
87              j[k] = A[i - k];
88          }
89          Est[i] = 0.0;
90          for ( k = 0; k < M; k++) {
91
92              Est[i] += wi[k] * j[k];
93
94          }
95      }
96      // Computing Error signal
97                  for (i = 0; i < N; i++) {
98                      error[i] = Est[i] - ECGsignal[i];       //
                           ↪ calculer l'erreur d'estimation
99                      mse+=(error[i]*error[i]);               //
                           ↪ calculer l'erreur quadratique
100                 }
101
102                 mse/=N;                     //calculer la
                       ↪ moyenne de l'erreur quadratique
103                 //printf("L'erreur quadratique moyenne est de %
                       ↪ f\n", mse);       //print the MSE
104
105     //Print the results
106     for (i = 0; i < N; i++) {
107          printf("%f\n", A[i]);
108      }
109
110                   printf("done!\n");
111
112
113 }//end of code
```

```
1       //====================NLMS============================/
```

```
/*This code send the filtered signal to the codec board
to be perceived on oscilloscope */
//Codec related include and configuration
#include "DSK6713_AIC23.h" //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ; //set sampling rate
/********************Include*****************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "ECG3.h"
#include "ECG96B.h"

/********************Define******************************/
#define N 5000                // Length of the ECG signal
#define M 35
#define mu 0.005
#define epsilon 1e-6        // Small constant to avoid division by
   ↪  zero

/********************Variables***************************/
float  wi[M], E[N], Est[N], error[N];
/************main code ************/
void main() {
    long i, j, k;          //variables to run loops
    float mse=0;         //initialiser l'erreur quadratique
       ↪ moyenne

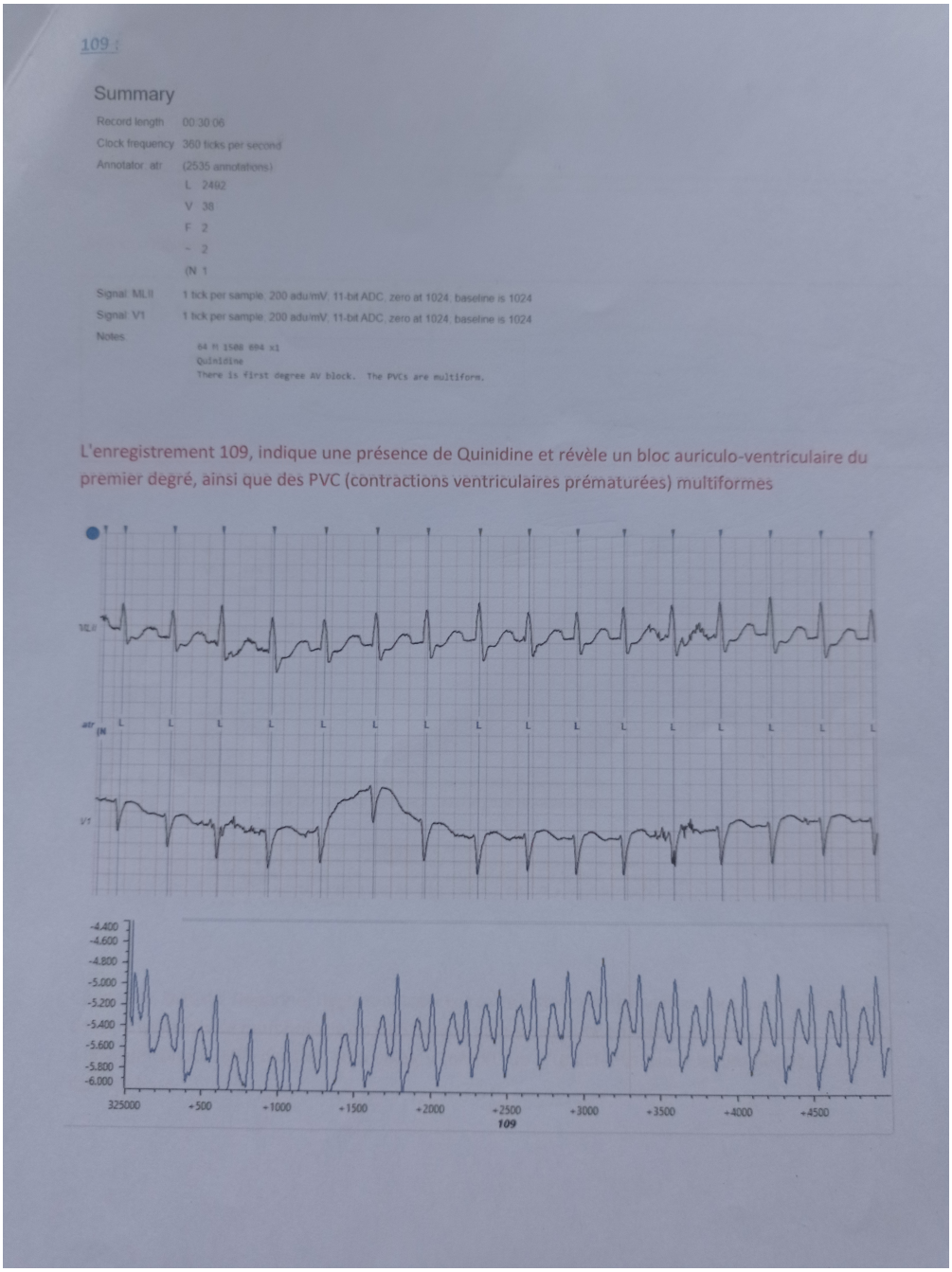
    // Initialize filter coefficients vector with M zeros
    for (i = 0; i < M; i++) {
        wi[i] = 0;
    }

    // Run loop M-n time
    for (i = M; i < N; i++) {
        // Estimate filter error
        E[i] = ECGsignal[i];
        for (j = 0; j < M; j++) {
            E[i] -= wi[j] * A[i - j];
        }

        // Compute input power
        float inputPower = 0.0;
        for (j = 0; j < M; j++) {
            inputPower += A[i - j] * A[i - j];
```

```
44          }
45
46          // Update weights using NLMS algorithm
47          for (j = 0; j < M; j++) {
48              float update =  mu * 2*E[i] * A[i - j] / (inputPower
                    ↪ + epsilon);
49              wi[j] += update;
50          }
51      }
52              comm_poll();                    //Initialiser le codec,
                    ↪ dsk et McBSP
53      // Estimating the signal
54      for (i = M; i < N; i++) {
55          float j[M];
56          for (k = 0; k < M; k++) {
57              j[k] = A[i - k];
58          }
59          Est[i] = 0.0;
60          for (k = 0; k < M; k++) {
61              Est[i] += wi[k] * j[k];
62              int codec[N];
63              codec[i]= Est[i]*1000;
64              output_left_sample((short) codec[i]);
65          }
66      }
67
68      // Computing Error signal
69                          for (i = 0; i < N; i++) {
70                              error[i] = Est[i] - ECGsignal[i];
                                    ↪ //calculer l'erreur d'estimation
71                              mse+=(error[i]*error[i]);
                                    ↪ //calculer l'erreur quadratique
72                          }
73
74      mse/=N;
75      printf("L'erreur quadratique moyenne est de %f\n", mse);
           ↪        //print the MSE
76
77      printf("done!\n");
78
79
80  }//end of program
```

Annexe2 : avis et notes médicales

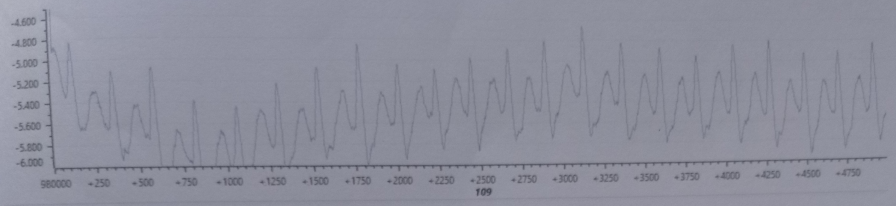


109 :

Summary

Record length	00:30:06
Clock frequency	360 ticks per second
Annotator atr	(2535 annotations)
	L 2492
	V 38
	F 2
	~ 2
	(N 1
Signal: MLII	1 tick per sample, 200 adu/mV, 11-bit ADC, zero at 1024, baseline is 1024
Signal: V1	1 tick per sample, 200 adu/mV, 11-bit ADC, zero at 1024, baseline is 1024
Notes:	64 M 1508 604 x1 Quinidine There is first degree AV block. The PVCs are multiform.

L'enregistrement 109, indique une présence de Quinidine et révèle un bloc auriculo-ventriculaire du premier degré, ainsi que des PVC (contractions ventriculaires prématurées) multiformes



Avis du médecin :

L'ECG est irrégulier.

Centre de Cardiologie Birkhadem
Dr. SEDDIKI Saber
Cardiologue
N° d'Ordre : 11552/16
Rue Klous Mohamed N° 8 Bir Khadem

203 :

N	2529
V	444
~	57
|	26
(AFIB	21
(VT	21
Q	4
a	2
(AFL	2
(T	1
F	1

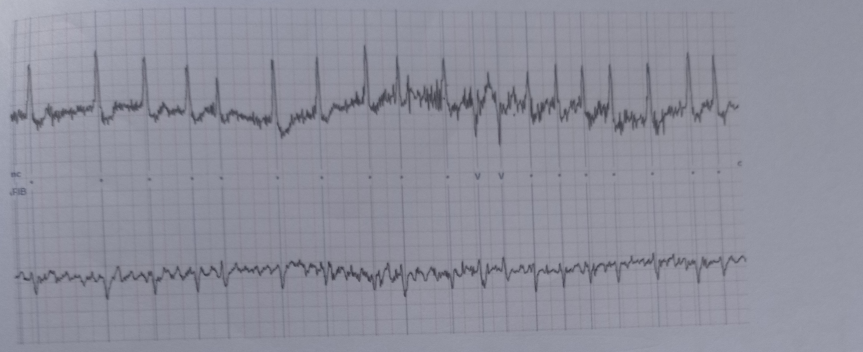
Signal: MLII 1 tick per sample, 200 adu/mV, 11-bit ADC, zero at 1024, baseline is 1024

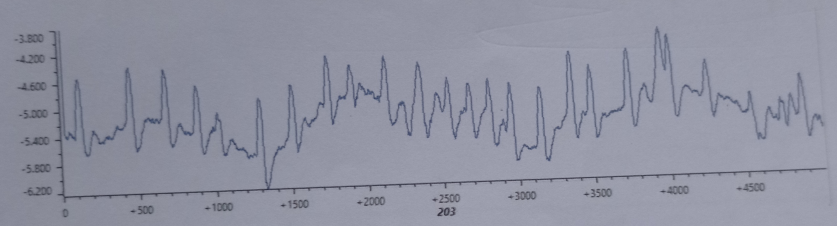
Signal: V1 1 tick per sample, 200 adu/mV, 11-bit ADC, zero at 1024, baseline is 1024

Notes:

```
43 M 1878 356 x1
Coumadin, Digoxin, Heparin, Hygroton, Lasix
The PVCs are multiform.  There are QRS morphology changes in the upper
channel due to axis shifts.  There is considerable noise in both channels,
including muscle artifact and baseline shifts.  This is a very difficult
record, even for humans!
```

Coumadin, Digoxin, Héparine, Hygroton, Lasix Les extrasystoles ventriculaires sont multiformes. Il y a des changements de morphologie du QRS dans la voie supérieure en raison de déviations de l'axe. Il y a un bruit considérable dans les deux voies, y compris un artefact musculaire et des décalages de ligne de base



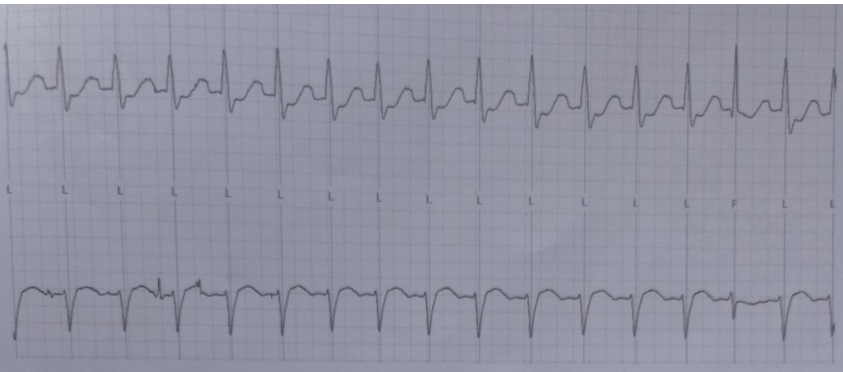


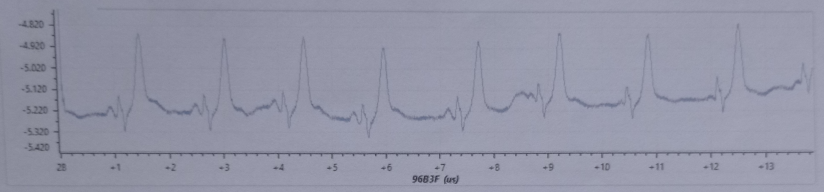
Avis du médecin :

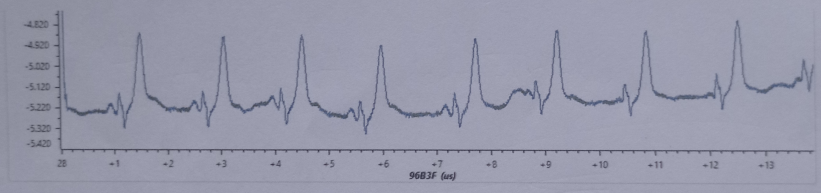
Pc Holter et interrogatoire.

Centre de Cardiologie Birkhadem
Dr SEDDIKI Saber
Cardiologue
N° d'Ordre : 11552/16
Rue Klous Mohamed N° 8 Birkhadem

Personne 9 rec6



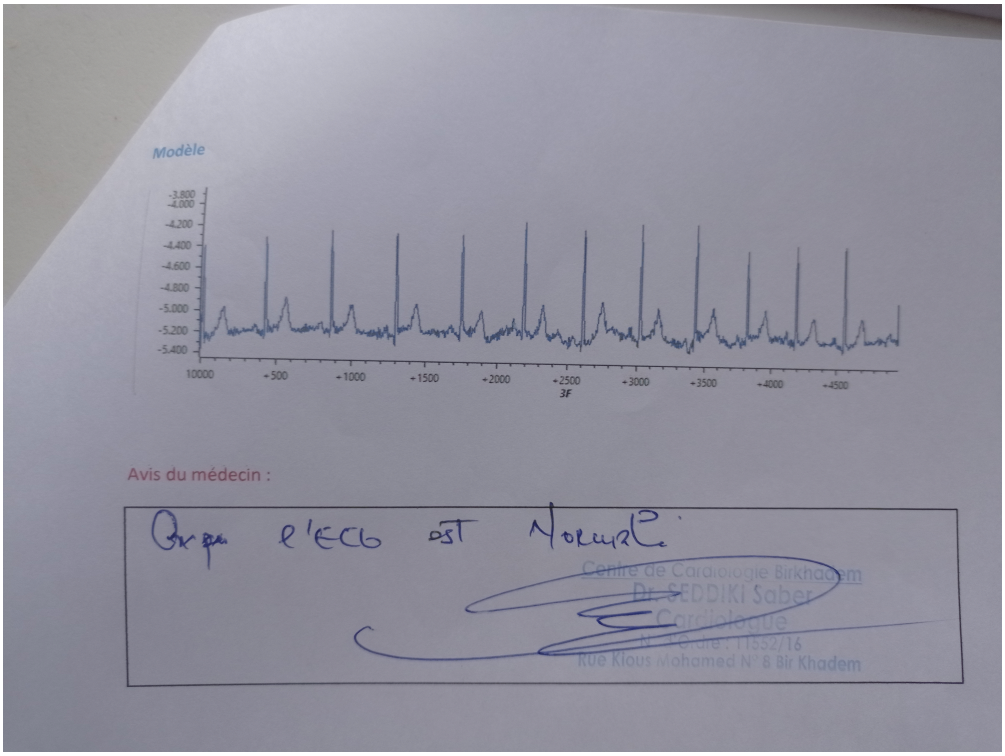




Avis du médecin :

- Le PGC a été posé exposition du segment fermé avec un stent de bon qualité.

Centre de Cardiologie Birkhadem
Dr. SEDD[illegible] Saber
Cardiologue
N° d'Ordre : 11552/16
Rue Kious Mohamed N° 8 Bir Khadem


Modèle
-3.800
-4.000
-4.200
-4.400
-4.600
-4.800
-5.000
-5.200
-5.400
10000
+500
+1000
+1500
+2000
+2500
+3000
+3500
+4000
+4500
3F
Avis du médecin :
l'ECG est Normal
Centre de Cardiologie Birkhadem
Dr SEDDIKI Saber
Cardiologue
Rue Klous Mohamed N° 8 Bir Khadem

ملخص

في سياق تطبيقات الأنظمة المضمنة في مجال الطب، وتحديدًا عمليات المعالجة الأولية لتحليل إشارات القلب الكهربائية باستخدام التصفية التكيفية، قمنا بتطبيق هذه الخوارزميات LMS و NLMSو RLS على إشارات القلب فيDSP C671. أظهر تطبيق مرشحات LMS و NLMS على إشارات ECG نتائج مرضية، بينما أظهر مرشح RLS أداءً ضعيفًا لهذا النوع من الإشارات.
يفتح هذا العمل آفاقًا مثيرة لتحسين تصفية ECG ويساهم في تصميم أجهزة ECG مثلا لتوفير تشخيص دقيق وتلقائي أو مراقبته بشكل مستدام.

الكلمات الرئيسية : **تصفية متكيفة، DSP C6713، LMS، NLMS، RLS**

Résumé

Dans le cadre des applications de systèmes embarqués dans le domaine médical, et les opérations prétraitement de l'analyse des signaux électriques cardiaques à l'aide d'un filtrage adaptatif, nous avons implémenté les filtres LMS, NLMS et RLS sur un processeur de traitement de signal, DSP C671. L'application des filtres LMS et NMLS aux signaux ECG a donné des résultats satisfaisants contrairement au filtre RLS qui est peu performant pour ce type de signaux. Ce travail ouvre des perspectives intéressantes pour l'amélioration du filtrage ECG et contribuent à la conception d'appareils ECG, tels que des moniteurs ECG, afin de fournir un diagnostic précis et automatisé.

Les mots-clés : **Filtrage adaptatif, DSP C6713, LMS, NLMS, RLS**

Abstract

In the context of embedded systems applications in the medical field, specifically the preprocessing operations of analyzing cardiac electrical signals using adaptive filtering, we have implemented the LMS, NLMS, and RLS filters on a DSP C671 signal processing processor. The application of LMS and NLMS filters to ECG signals has yielded satisfactory results, unlike the RLS filter, which has shown poor performance for this type of signals. This work opens interesting perspectives for improving ECG filtering and contributes to the design of ECG devices, such as ECG monitors, to provide accurate and automated diagnosis.

Keywords: **Adaptive filtering, DSP C6713, LMS, NLMS, RLS**