DOCTORAT BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE NANTES

THÈSE DE DOCTORAT DE

UNIVERSITÉ DE NANTES

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Électronique*

Par

David Pallier

SENTAUR

Sensor Enhancement to Augmented Usage and Reliability

Thèse présentée et soutenue à Université Gustave Eiffel - Campus Nantes, le 10/03/2021
Unité de recherche : IETR SYSCOM, Université Gustave Eiffel COSYS SII/I4S
Thèse N° :

Rapporteurs avant soutenance :

Daniela Dragomirescu	Professeur des universités, INSA Toulouse
Jean-Marie Gorce	Professeur des universités, INSA Lyon

Composition du Jury :

Président :	Olivier Sentieys	Professeur des universités, Université de Rennes 1
Examinateurs :	Daniela Dragomirescu	Professeur des universités, INSA Toulouse
	Jean-Marie Gorce	Professeur des universités, INSA Lyon
	Bastien Chapuis	Ingénieur de recherche, CEA
Dir. de thèse :	Sébastien Pillement	Professeur des universités, Université de Nantes
Encadr. de thèse :	Vincent Le Cam	Ingénieur divisionnaire des travaux publics de l'État, Université Gustave Eiffel

"There was nowhere to go but everywhere, so just keep on rolling under the stars"

Jack Kerouac, On the Road

Remerciements

Je souhaite tout d'abord remercier mes encadrants pour leur accompagnement pendant ces trois années de thèse. Merci à Vincent LE CAM, sans qui ce projet n'aurait jamais vu le jour, pour l'ensemble de nos échanges scientifiques et pour m'avoir encouragé tout au long de cette aventure. Merci à Sébastien PILLEMENT pour ses conseils avisés ainsi que ses nombreuses relectures, toujours très pertinentes, de mes travaux. Je tiens aussi à remercier chaleureusement Olivier SENTIEYS et Andrea PINNA pour leur expertise annuelle en tant que comité de suivi de thèse.

Au-delà de mes encadrants, des remerciements doivent également être adressés à tous les membres de mon jury de thèse : merci à Olivier SENTIEYS pour avoir présidé ce jury de thèse ; merci à Daniela DRAGOMIRESCU, à Jean-Marie GORCE, et à Bastien CHAPUIS pour leur relecture et leur analyse du manuscrit.

Il me faut aussi remercier les membres nantais de SII, et plus généralement de l'I4S. Merci à Qinghua ZHANG et Laurent MEVEL pour nos échanges qui m'ont permis de mieux comprendre certains aspects du filtrage de Kalman. Merci à David BETAILLE et Miguel ORTIZ de m'avoir fait profiter de leurs connaissances du système de positionnement GPS. Merci également à Jean Luc MANCEAU et Marc BRUNET (de l'IETR) pour leur soutien technique.

Je tiens aussi à remercier l'Université Gustave Eiffel, ex-IFSTTAR, et le RFI Wise pour avoir financé cette thèse.

De manière un peu moins formelle, je salue toute la troupe du laboratoire. Les "anciens" : Guillaume, Nico, Ludo "le bonhomme" et Antoine qui a dû me supporter quotidiennement pendant presque trois ans. Ainsi que ceux qui sont encore présents : Ivan "le jukebox" ; Martin dit "les bons plans" (courage tu y es presque) ; Quentin le dealer de jeux de société ; Thibaud avec qui je m'engage dans beaucoup trop de projets (promis un jour on en terminera un - ps j'attend toujours ma formation Emacs ;-) ; et Arthur fidèle compagnon de boisson et d'escapades outre-Atlantique. Merci pour tous ces bons moments.

Je ne peux pas oublier ici la joyeuse bande des Nantais : merci Val, Ben, Camille, Clem, Poulette, Ludo, Edwige et tous les autres pour tous ces moments, grandement nécessaires, de "décompression". Je remercie tout particulièrement mes parents, beaux-parents et frère et soeurs pour leur soutien, même dans les passages parfois compliqués, qui sont inhérents à la thèse.

Pour terminer, mes pensées vont vers toi Sophie : un grand merci pour ton soutien, ta patience et tes encouragements. Merci de m'avoir porté durant cette difficile dernière ligne droite qu'est la rédaction du manuscrit.

Glossaire

EAM Erreur Absolue Moyenne. 42, 43, 45, 50, 53, 54, 57, 81–83, 87–89, 101–106, 108, 109

FPGA Field Programmable Gate Arrays. 24, 25, 27, 33, 34, 38, 41, 44, 113–115, 124, 125, 128–131, 133, 157

FTSP Flooding Time Synchronization Protocol. 21–23, 25

GBF Générateur Basse Fréquence. 41, 43–45, 48, 98, 129

GNSS Géolocalisation et Navigation par un Système de Satellites. 3, 18, 27

GPS Global Positioning System. 3, 4, 13, 26–28, 31, 33, 36, 38, 39, 41, 43–50, 53, 57–69, 71, 72, 77, 78, 80, 81, 84–90, 93, 95, 96, 98, 99, 102, 105, 106, 110–112, 114–119, 124, 125, 127–130, 133

MHFC Modèle d'Horloge à Fréquence Constante. 67, 71, 77–79, 81, 82, 84, 85, 87–90, 96, 98, 100–103, 105, 108, 112, 118

MHFD Modèle d'Horloge à Fréquence Dynamique. 71, 72, 77–79, 81, 82, 85, 87, 88, 90, 96, 98, 100–103, 108, 112

NTP Network Time Protocol. 18, 19, 25

PPS Pulse Per Second. 27, 36, 37, 39, 41–45, 47, 48, 52, 53, 60, 64, 72, 74, 77, 79, 85, 92, 93, 96, 114–118, 122, 125, 129

PTP Precision Time Protocol. 18, 19, 24, 26

RBS Reference Broadcast Synchronization. 19, 25

RMSE Root-mean-square Error. 93, 94

RTC Real Time Clock. 14, 61, 114

SHM Structural Health Monitoring. 1, 4, 6–11, 27, 30, 33, 34

SPI Serial Peripheral Interface. 114, 116, 117

SPXO Simple Packaged Crystal Oscillator. 15, 58, 98, 99, 101, 102, 104–106, 108, 110, 111

TAI Temps Atomique International. 12, 13

TCXO Temperature Compensated Crystal Oscillator. 15–17, 97–101, 103–107, 109–111, 116, 121–125, 128

TPSN Timing-sync Protocol for Sensor Networks. 19, 21, 22

TTFF Time To First Fix. 60–62

UC Unité de Contrôle. 41, 42, 44

UH Unité d'Horodatage. 41–44, 48

USNO United States Naval Observatory. 13

UT1 Universal Time 1. 12, 13

UTC Temps Universel Coordonné. 12, 13, 18, 27, 31, 36–38, 48, 62, 64, 66

WSN Wireless Sensor Network. 11, 12, 17

Table des matières

Table des figures

Liste des tableaux

CHAPITRE 1

Introduction

Le développement exponentiel de l'électronique intégrée depuis une soixantaine d'années et les multiples champs d'applications qu'elle ouvre s'appliquent naturellement aux structures à enjeux : énergie, aéronautique, génie civil, santé, etc. Dans de multiples cas, le déploiement de capteurs et de systèmes de mesures sont nécessaires pour ausculter ponctuellement ou contrôler en continu le bon déroulement d'un processus industriel, le fonctionnement conforme d'un automatisme ou l'intégrité d'une structure. Dans le domaine du Contrôle et des Essais Non Destructifs (E/CND) comme celui de la surveillance en continu des structures (SHM), cela se traduit souvent par le déploiement de multiples noeuds de capteurs pour échantillonner des paramètres physiques tels que les vibrations à l'aide accéléromètres, les ondes mécaniques ou acoustiques grâce à des capteurs piézoélectriques, des déformations par des jauges, des températures, etc. Une fois digitalisés par une entité numérique (i.e. le plus souvent un Convertisseur Analogique Numérique associé à un micro-processeur), ces échantillons sont ensuite utilisés par des algorithmes qui déduisent ou contribuent à établir l'apparition d'un défaut structurel voire l'état global de santé actuel ou futur de la structure. Ces informations sont

stratégiques pour l'exploitant et le gestionnaire de la structure en ce qu'elles déclenchent ou conditionnent de lourds processus de maintenance et, dans les cas les plus extrêmes, peuvent conduire à l'arrêt d'exploitation partielle ou totale de la structure.

Dans la majorité des cas, les paramètres à enregistrer sont des signaux physiques dépendants du temps. Ils sont échantillonnés par des noeuds de capteurs répartis sur la structure à surveiller. Dans la plupart des situations de surveillance, l'exploitation de ces signaux par des algorithmes (ou même visuellement par un opérateur) n'a de sens que s'ils sont corrélés temporellement. Ainsi, les échantillons provenant des différents noeuds d'un réseau de capteurs, ont besoin d'être synchrones c'est à dire d'appartenir à la même base de temps. Les erreurs acceptables de mesures liées au temps dépendent de la précision relative de synchronisation des noeuds des capteurs à l'intérieur du réseau. Ci-dessous quelques exemples d'applications ainsi que la précision temporelle chiffrée nécessaire :

- Les fréquences naturelles des structures du Génie Civil (câble, tablier de pont...) sont typiquement comprises entre 0 et 100 Hz. Aussi, pour une bonne restitution des modes lors de l'analyse modale, les accélérations sont généralement échantillonnées à 1kHz. L'erreur de synchronisation entre les capteurs doit donc être inférieure à 1 milliseconde.
- L'échantillonnage des ondes acoustiques par des accéléromètres est souvent utilisé pour localiser l'origine d'un défaut ou d'une rupture de fil dans les câbles des ponts (à haubans, de suspente ou de pré-contraints). La méthode dite TDOA (Time Difference Of Arrival) permet de localiser l'origine d'un défaut (typiquement une rupture de fil) en comparant les horodatages de l'arrivée d'une onde acoustique entre deux noeuds de capteurs fixés aux extrémité du câble. Pour une vitesse de propagation typique de 5000 m/s et une précision de localisation souhaitable de +/- 15 cm, l'erreur de synchronisation entre deux capteurs doit toujours être inférieure à 30 microsecondes.
- La méthode TDOA est aussi utilisée pour localiser les défauts sur les lignes hautes tensions provoquées par la foudre, les chutes d'arbres, etc. Pour une vitesse de propagation des ondes électriques de 200 000 km/s et une précision de localisation de +/- 50 m, l'erreur de

synchronisation entre deux capteurs doit être inférieure à 250 nanosecondes.

La surveillance de grandes infrastructures telles que les ponts, rails ou lignes HT impliquent de très grands réseaux de mesure au sein desquels les capteurs peuvent être espacés de plusieurs kilomètres. Les protocoles de synchronisation classiquement utilisées pour les réseaux de capteurs sans-fil [1][2][3] ne sont pas capables, en général, de faire face à de telles contraintes. Certains protocoles de synchronisations basées sur les normes 802.11 [4] et 802.15.4 [5] permettent d'atteindre une précision de synchronisation de l'ordre de la nanoseconde mais ne sont pas prévus pour être déployés sur de telles distances. Des solutions filaires telles que celle développées pour le projet white rabbit [6] peuvent être utilisées pour synchroniser des noeuds avec une grande précision et sur de grandes distances. Pour le sans-fil, seule l'utilisation d'un système de navigation satellite avec des récepteurs sur les noeuds de capteurs tel que le GPS [7] permet d'obtenir une grande précision de synchronisation sur de grandes distances. Couplé à un système de remontée de données ponctuel par 3G, 4G, Nb-Iot, LTE-M, LoRa ou SigFox, cette solution de synchronisation permet un déploiement aisé et robuste sur une structure car sans contrainte de topologie réseau. Cependant, les récepteurs GNSS sont énergivores (typiquement de 100 à 200 mW) pour des noeuds alimentés par batteries ou utilisant la récupération d'énergie. C'est sur cette problématique que les travaux de cette thèse sont basés : offrir un niveau élevé de synchronisation dans un réseau de capteur sans-fil tout en optimisant la consommation des noeuds qui le compose.

Objectifs

L'objectif principal de ces travaux de thèse est de développer, d'implémenter et de tester un mécanisme de synchronisation par GPS capable d'adapter le niveau de synchronisation en fonction d'une précision de synchronisation donnée. Cet objectif peut être découpé en plusieurs sous-objectifs :

- Développer une architecture de noeud de capteurs permettant l'analyse de données en

hors ligne et l'implémenter sur deux noeuds prototypes. Valider les prototypes et générer des jeux de données à partir des prototypes et d'un générateur d'événements. Cette approche permet l'évaluation de différents paramètres de synchronisation à partir des jeux de données.

- Développer une solution de synchronisation adaptative par GPS et analyser cette solution à l'aide des jeux de données.
- Modéliser le comportement de l'horloge des noeuds et évaluer différentes méthodes de prédiction et de filtrage pour estimer les paramètres du modèle. Évaluer l'impact de la température sur la solution de synchronisation ainsi que l'utilisation d'un oscillateur compensé en température. L'ensemble de ces modèles ayant vocation à maximiser le temps d'extinction du récepteur GPS.

Structure du document

Le chapitre 2 définit le domaine du Structural Health Monitoring (SHM), décrit l'utilisation des noeuds de capteurs dans ce domaine et introduit les notions d'étalons de fréquences et d'oscillateurs à quartz. Ce chapitre comprend une revue de littérature sur les différentes méthodes de synchronisation des noeuds de capteurs. L'architecture des noeuds de capteurs pour l'acquisition des jeux de données est décrite dans le chapitre 3. La description des méthodes expérimentales ainsi que la validation des prototypes sont aussi présentes dans ce chapitre. Le chapitre 4 est consacré à le description du protocole de synchronisation adaptative et du modèle d'horloge. Plusieurs méthodes d'estimation des paramètres ainsi que l'utilisation d'un filtre de Kalman sont analysés dans ce chapitre. L'influence de la température sur l'estimation des paramètres du modèle ainsi que des gains apportés par l'utilisation d'un oscillateur compensé en température sont étudiés dans le chapitre 5. Enfin le chapitre 7 permet de conclure ce document en résumant les résultats obtenus, les limites de ces travaux ainsi que les pistes de recherche pour de futurs développements.

CHAPITRE 2

Contrôle de santé des structures et réseaux de capteurs

L'installation de réseaux de noeuds de capteurs sans-fil a tendance à remplacer ou compléter les inspections périodiques dans le domaine du contrôle de santé des structures de génie civil. L'utilisation de capteurs sans-fil lève cependant des enjeux techniques liés à la précision de leurs mesures et à leur autonomie. La précision de l'échantillonnage de données réparti sur plusieurs noeuds dépendant du temps est grandement corrélée à la synchronisation entre ces noeuds. Cette synchronisation entre les noeuds se doit d'être fiable, facilement déployable sur les structures et le plus économe en énergie possible.

Sommaire

2.1 Contrôle de santé des structures

2.1.1 Définition

Le présent rapport de thèse rapporte des travaux dont la principale inspiration en matière d'application est celle du SHM. Bien que les applications et bénéfices possibles d'une synchronisation de noeuds de capteurs sans-fil soient plus larges, les cas d'applications du SHM ont permis de bien définir des cas d'usages critiques au sein desquels la qualité de synchronisation des noeuds est fondamentale. Aussi, nous proposons ici de mieux définir la notion de SHM.

Le SHM est un acronyme signifiant *Structural Health Monitoring* communément traduit en français comme le Suivi des Santé des Structures. On peut affirmer que le SHM ne constitue

pas en soi une discipline scientifique fermée et auto-définie mais la mise en facteur cohérente et à visée applicative d'un ensemble de champs disciplinaires allants de l'électronique aux mathématiques en passant par la physique des matériaux.

Par *structures* on entend généralement les structures à enjeux c'est-à-dire celles pour lesquelles des vies sont en jeu (aéronautique, ouvrages d'art...) et/ou pour lesquelles des conséquences matérielles d'une défaillance sont dramatiques (déraillement d'un train, effondrement d'un pont, arrêt d'une centrale nucléaire). Par exemple les temps d'immobilisation dus à une réparation d'une ligne à haute tension ou au contournement d'un pont effondré sont extrêmement coûteux pour les gestionnaires et l'usager en général. Aussi on trouve des applications SHM s'applique plus particulièrement aux structures telles que (sans que cette liste ne soit exhaustive) : les ponts et ouvrages du Génie Civil [8], [9], les rails [10], [11], les systèmes de transports ou de production d'énergie telle que les éoliennes [12], [13]. Les techniques de SHM sont aussi appliquées dans le domaine de l'aviation pour l'inspection des appareils [14], [15] *on board* comme *off board*.

En règle générale, l'usage d'un système de SHM vise à équiper une structure de divers équipements matériels, dont essentiellement des capteurs, et d'utiliser des algorithmes de suivi permettant de suivre son état de santé en continu. Cette vision du SHM peut être perçue en opposition, ou en complément, des campagnes d'inspections arbitraires ou périodiques ne permettant pas nécessairement de détecter en temps réel l'apparition d'un défaut.

Ainsi le déploiement d'un système SHM est synonyme d'installation d'un ensemble complet et parfois complexe de briques matérielles et logicielles qui se répartissent, physiquement sur la structure, en passant par des médias de (télé) communication jusqu'au serveur ou système d'information de l'exploitant, sur lesquels des opérateurs du gestionnaire basent leur décision quant à l'exploitation de la structure. L'objectif commun de tous les systèmes SHM est de délivrer une information la plus pertinente possible sur l'état de santé actuel ou futur de la structure suivant l'échelle de temps visée. Par exemple, dans le domaine du Génie Civil, le principe de détection et localisation des ruptures de fils dans les câbles des ponts peut aler-

ter le gestionnaire en temps réel (si un nombre important de ruptures de fils sont détectées) ou conditionner un acte futur de maintenance (si ces ruptures sont rares mais co-localisées pouvant témoigner d'une faiblesse grandissante en un point particulier du câble).

Parvenir à une une estimation de l'état de santé actuel et/ou de la durée de vie résiduelle de la structure [16] est donc le but global des systèmes SHM. Cela peut constituer un changement de paradigme dans les stratégies de maintenance puisqu'il s'agit de remplacer ou de compléter des inspections périodiques réalisées par des opérateurs par une surveillance, en continu, à partir d'une instrumentation intégrée à la structure. Les données fournies par un système SHM cherchent à garantir une utilisation optimale de la structure qu'il surveille. Il réduit les temps d'immobilisation (au sens arrêt de l'exploitation de l'ouvrage), il peut permettre de contrôler des zones inaccessibles ou nécessitant de lourdes opérations de démontage pour y accéder et apporte une sécurité visant à réduire le nombre de ruptures catastrophiques (exemple : surveillance par ultrasons de l'état des torons dans les parties inaccessibles des câbles de précontrainte).

La principale motivation pour le développement de systèmes SHM est économique grâce à la réduction des temps d'immobilisation, l'augmentation intrinsèque de la sureté de la structure et l'augmentation des intervalles entre deux campagnes de maintenance. A long terme, le SHM doit permettre de mettre en place une stratégie de maintenance conditionnelle (CBM : Condition Based Maintenance) et non plus à intervalles prédéfinis. D'autres visions peuvent être tirées des systèmes SHM comme, par exemple, celle consistant à instrumenter et suivre une structure représentative d'un parc de structures identiques et d'en déduire des campagnes de maintenances basées sur la similarité (exemple : suivi et maintenance groupée des moteurs de Passage à niveau sur le réseau ferroviaire). Enfin, les acteurs académiques poussent à faire évoluer la perception du SHM en intégrant les systèmes SHM *ad initio* c'est-à-dire dès la conception et la fabrication de la structure.

2.1.2 Exemples industriels

D'origine essentiellement académique, le SHM intéresse désormais tous les acteurs de la chaîne de valeur : des industriels, aux utilisateurs finaux en passant par les exploitants.

L'état de santé de certaines structures à enjeux est constaté ou jugé comme préoccupant dans de nombreuses zones du globe : états des ponts et routes en Amérique du Nord, rapport sénatorial sur les ouvrages d'art en France, drames mortels du pont de Gênes en Italie ou de Mirepoix en France), etc. Le besoin en suivi de santé est énorme, et ne se limite pas aux structures nouvelles (exemples d'une rupture de câble sur le pont de l'île de Ré en 2018 en figure 2.1). La prolongation de la durée de vie des centrales nucléaires, le télérelevé de mesures en cas de crise (exemple du phénomène en accentuation des crues soudaines), le vieillissement du réseau ferroviaire ou l'inspection de structures peu accessibles (telles que les éoliennes *offshores*) constituent autant de sujets d'actualité pour le SHM. Mais il ne faudrait pas penser que le SHM industriel n'existe pas. De nombreuses solutions émergent en silo. On peut citer plusieurs succès industriels du SHM très récents. En aéronautique, un premier système SHM basé sur le Comparative Vacuum Monitoring (CVM) [17] a été certifié après plusieurs années d'analyses par les autorités américaines (Federal Aviation Administration, FAA) comme alternative à l'inspection périodique visuelle d'un composant peu accessible dans les Boeing 737. Dans le secteur pétrolier, Emerson a racheté pour plus de 40 M€ la société Permasense, issue de travaux de l'Imperial College à Londres, qui a développé une solution de monitoring sans fil pour le suivi de corrosion et d'érosion des conduites [18] (représentée en figure 2.2). Ce système est désormais déployé dans des raffineries. La société SERCEL, leader historique en technologie d'exploration pétrolière, se diversifie fortement dans le SHM et propose maintenant des produits visant au suivi de l'intégrité des structures béton par analyse vibratoire.

La mise en œuvre industrielle des systèmes SHM pose toutefois la question de leur certification. Le cas du CVM est, à ce titre, assez emblématique puisque la procédure de certification par la FAA a pris plusieurs années, notamment parce qu'il n'y a pas encore aujourd'hui de méthodologie établie pour quantifier et garantir les performances d'un système SHM. Pour pallier

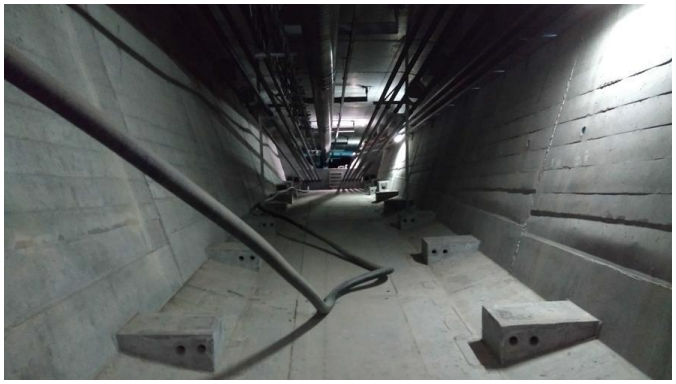


FIGURE 2.1 – Rupture d'un câble du pont de l'ile de Ré

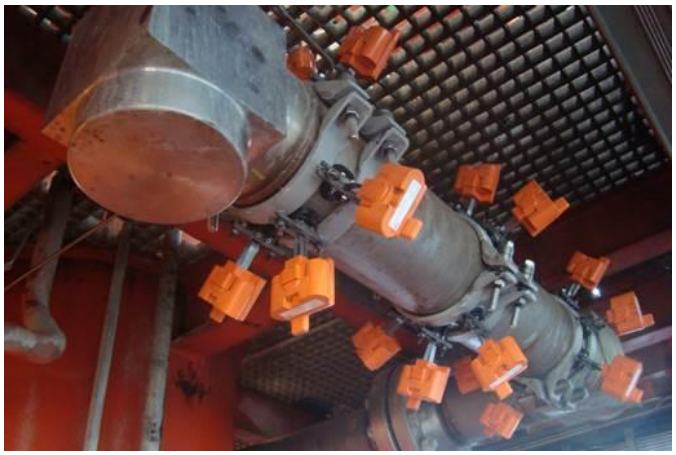


FIGURE 2.2 – Système SHM de Pemarsense

ce manque un groupe de travail dédié s'est constitué en 2008 au sein de la SAE International : l'Aerospace Industry Steering Committee on Structural Health Monitoring and Management (AISC-SHM). En génie civil , la simulation et la validation numérique des systèmes SHM demeure un enjeu majeur. On ne saurait détruire tout ou partie d'une éolienne ou d'un pont pour vérifier et qualifier à grande échelle la conséquence de cette dégradation dans, par exemple, la variation de ses modes propres. Il y a donc encore de fort enjeux et de fortes attentes en matière de modélisation, de méthodologie de simulation et de validation des systèmes SHM.

Un travail technologique et scientifique demeure considérable mais, à l'instar des exemples donnés ci-dessus, le monde industriel s'empare du sujet et ce, parfois, collectivement. Citons,

par exemple, la COFREND (Confédération Française pour les Essais Non Destructifs) dont le pôle scientifique c'est donné pour mission d'unir ses membres autour du SHM en vue de structurer la filière SHM au niveau national.

2.1.3 Les réseaux de capteurs sans fil

Les sciences et techniques de l'électronique, de l'informatique industrielle et des télécommunications permettent, désormais, de proposer des réseaux de noeuds de capteurs sans fil (Wireless Sensor Network : WSN) tout à fait pertinents pour les applications du SHM. L'intégration de ces WSN dans les systèmes SHM est souvent avantageuse d'un point de vue financier. Dans le domaine du Génie Civil par exemple, il est très régulièrement fait appel à des techniques de surveillance dans le cadre d'appels d'offre de surveillance renforcée des structures. Ces marchés de surveillance ont des fourchettes typiques de 100 k€ à 600 k€ dont la part matérielle (i.e. achat de capteurs et diverses fournitures) demeure marginale par rapport à leur installation, au câblage et à leur exploitation. Aussi toute démarche permettant de rendre ces systèmes sans fil (i.e. éviter de longues et coûteuses câbleries) et assurant une surveillance continue (c'est-à-dire y compris entre deux campagnes physiques d'auscultation) représente des gains financiers considérables.

L'utilisation des WSN présente aussi de nombreux challenges, et plus particulièrement dans le domaine du SHM ou la fréquence d'échantillonnage, la bande passante et la puissance de calcul nécessaires peuvent être élevés [19]-[21]. Dans [22], [23], plusieurs plateformes de WSN pouvant être utilisées pour le SHM sont décrites. Pour toutes ces plateformes, les enjeux sont typiquement les mêmes : les noeuds de capteurs doivent être fiables et disposer de ressources de calcul suffisantes pour pouvoir effectuer une partie du traitement des échantillons récoltés ; la couche radio et le protocole de communication utilisé ainsi que la topologie de réseau doivent permettent une bande passante suffisante par rapport à la fréquence d'échantillonnage des capteurs ; la consommation énergétique des noeuds doit être minimisée du fait de leur autonomie.

Mais outre ces dimensions assez généralistes, l'étude de cas d'application des WSN dans le contrôle de santé des structures fait émerger d'autres enjeux moins souvent abordés dans la littérature tels que : le placement optimal de capteurs pour minimiser leur nombre, ou leurs communications, tout en ayant accès à l'intégralité de l'information nécessaire au diagnostic de la structure; le vieillissement métrologique ou la confiance dans la qualité métrologique des données produites par un capteur après des mois et des années; la synchronisation des noeuds d'un réseau de capteurs, dont il est question dans cette thèse. Mais avant d'aborder la problématique de la synchronisation il faut d'abord définir ce qu'est un étalon de fréquence et comment est compté le temps dans un noeud de capteurs.

2.2 Etalon de fréquence et oscillateur à quartz

2.2.1 Etalon atomique, TAI et UTC

Un étalon de fréquence est une source de fréquence stable et connue, permettant de mesurer le temps en comptant des oscillations. Il existe plusieurs étalons de fréquence permettant de compter le temps. L'étalon atomique est celui qui permet de définir la seconde (au sens unité de base du Système international). La seconde atomique est définie comme suit : "La seconde est la durée de 9.192.631.770 périodes de la radiation correspondant à la transition entre les deux niveaux hyperfins de l'état fondamental de l'atome de Césium 133" [24]. En 1971, le Temps Atomique International (TAI), calculé par le Bureau International de l'Heure (BIH) à partir d'horloges atomiques, est officiellement reconnu comme référence de temps lors de la $14^{ième}$ conférence générale des poids et mesures [25]. D'un autre côté, l'échelle de temps historiquement utilisée pour la navigation est le Universal Time 1 (UT1), contrairement au TAI, cette échelle de temps est basée sur le jour solaire moyen (période de rotation de la terre autour de son axe, moyennée sur un an). Du fait des variations de la vitesse de rotation, le temps UT1 dérive vis-à-vis du temps TAI. Afin de garder une échelle de temps basée sur la rotation de la terre, l'échelle de temps UTC (mélange de Universal Coordinated Time et Temps Universel

Coordonné) est créée. Le UTC est basé sur le TAI mais avec des secondes intercalaires ajoutées ou retranchées pour le maintenir à moins d'une seconde du temps UT1. La dernière seconde intercalaire a été ajoutée le 31 décembre 2016. Actuellement le UTC est en retard de 37 secondes par rapport au TAI.

Le UTC United State Naval Observatory (USNO) est une échelle de temps maintenue par le USNO à l'aide de ses propres horloges atomiques pour le système GPS. Ce temps est synchronisé sur le temps UTC à 1 μs près. On appelle GPS System Time le temps global du système GPS. Il est géré par le Control Segment (CS) du système GPS [26] qui contrôle les horloges atomiques des satellites GPS. Ce temps est synchronisé sur le temps UTC(USNO) modulo une seconde pour pouvoir être uniforme (pas de secondes intercalaires à ajouter ou retirer). Le temps global GPS était identique au temps UTC(USNO) le 6 janvier 1980 à minuit. Aujourd'hui ils diffèrent de 18 secondes. Dans les spécifications du système GPS [26], l'erreur de déphasage entre UTC(USNO) et le temps global GPS doit être inférieure à 40 ns 95% du temps. Dans les faits, cette erreur est beaucoup plus basse. Sur l'année 2015, le 95e centile de l'erreur pour chaque mois était compris entre 1.026 ns et 2.571 ns [27]. Les satellites GPS transmettent le temps au format epoch constitué du nombre de secondes écoulées depuis le dernier dimanche à minuit et du nombre de semaines écoulées depuis le 6 janvier 1980. Les satellites GPS transmettent aussi le nombre de secondes intercalaires pour pouvoir synchroniser les récepteurs GPS au temps UTC(USNO). Dans la suite du document, les références à la version USNO du temps UTC seront implicites dès qu'il s'agit du système GPS.

2.2.2 L'Oscillateur à Quartz

L'étalon de fréquence le plus utilisé pour l'électronique embarquée est l'oscillateur à quartz. Les horloges basées sur les oscillateurs à quartz sont moins précises que les horloges atomiques, mais beaucoup moins onéreuses, moins encombrantes et moins énergivores. Le quartz (SiO 2) est une espèce minérale présente à l'état naturel sous forme de cristaux qui possède des propriétés piézoélectriques. Ces propriétés permettent au quartz de vibrer à une fréquence stable

lorsqu'il est soumis à un signal électrique. Les oscillateurs sont composés d'une partie résonante (le cristal de quartz pour les oscillateurs à quartz) et d'une partie amplification permettant d'entretenir les oscillations.

Les oscillateurs à quartz présentent plusieurs sources d'erreurs [28]-[30]. La première source provient du cristal de quartz en lui-même. En effet, celui-ci peut être considéré comme un filtre très sélectif permettant d'obtenir une fréquence pure ; dans la réalité cette fréquence n'est pas "pure" et la sélectivité de l'oscillateur (aussi appelée facteur de qualité Q) dépend de la qualité et du dimensionnement du cristal. Les cristaux de quartz sont aussi sensibles à la température et leur fréquence évolue en fonction de celle-ci. Leur comportement en température dépend de l'angle de coupe du cristal. La coupe en diapason est plutôt utilisée pour les RTC (Real Time Clock) basses fréquence, leur dérive en fréquence est liée à la température par une fonction polynomiale de degré 2. Le cristal est souvent dimensionné de telle sorte que la fonction soit centrée sur 25 °C pour obtenir la fréquence théorique à 25 °C et s'en éloigner quand la température monte ou diminue. Les cristaux de quartz des oscillateurs plus haute fréquence (> 3 MHz) que l'on retrouve fréquemment pour cadencer les circuits numériques sont principalement de coupe dite "AT". Pour cette coupe les dérives en fréquences sont liées à la température par une fonction polynomiale du troisième degré. Encore une fois, le point d'inflexion de la courbe est généralement placé à 25 °C pour obtenir la fréquence théorique de l'oscillateur à cette température. De plus ce placement du point d'inflexion permet d'obtenir une zone plus ou moins affine entre 0 et 40 °C. La fréquence d'un oscillateur à quartz va aussi changer avec le vieillissement du composant ; ce vieillissement entraîne un décalage de la fréquence compris entre 10 ppm et 0.01 ppm par an en fonction de la matière de l'enveloppe du cristal et de la qualité de la mise sous vide de celui-ci [28], [29]. D'autres phénomènes tels que les vibrations, les accélérations, la pression, l'humidité, ou le rayonnement électromagnétique peuvent aussi modifier la fréquence de l'oscillateur. Généralement on distingue deux familles de sources d'erreurs : le bruit de phase causé par la sélectivité de l'oscillateur et l'étage d'amplification ; la dérive en fréquence liée à l'environnement de l'oscillateur et plus principalement sa température.

2.2.3 Types d'oscillateurs à quartz

Différents types d'oscillateur à quartz, plus ou moins stables, plus ou moins cher et plus ou moins énergivores sont disponibles sur étagère [31]-[34] :

- XO (Crystal Oscillator) : Oscillateur à quartz. Ne comprend que le crystal de Quartz et pas l'étage d'amplification.
- SPXO (Simple packaged crystal oscillator) : Oscillateur à Quartz incluant le crystal et l'étage d'amplification dans le même package. Généralement plus stables que les XO car les deux parties de l'oscillateur sont intégrées dans le même composant.
- VCXO (Voltage Controlled Crystal Oscillator) : La fréquence d'un oscillateur à quartz est liée à la réactance du circuit. Il est ainsi possible d'ajuster la fréquence de l'oscillateur en modifiant la réactance du circuit (à l'aide par exemple d'une diode varicap). Plus la gamme de fréquence ajustable est grande plus le bruit de phase de l'oscillateur est grand. Comme le SPXO, le VCXO comprend un crystal de quartz et un étage d'amplification, mais il possède aussi un circuit permettant de faire varier la réactance pour ajuster la fréquence de l'oscillateur en fonction de la tension appliquée.
- TCXO (Temperature Compensated Crystal Oscillator) : Mêmes composants que le SPXO (quartz et amplificateur), mais une boucle de contrôle est ajoutée pour compenser les variations de fréquence en fonction de la température. Différentes techniques peuvent être appliquées, les plus simples sont entièrement analogiques et consistent en un circuit de régulation de la réactance en fonction de l'impédance de thermistances intégrées au circuit. La méthode la plus précise consiste à numériser les variations de température puis à appliquer une tension calculée par un composant numérique (généralement un ASIC) à une diode varicap connectée à l'oscillateur. Cette méthode permet d'évaluer, pendant une phase de calibration, la fonction qui à la température associe la dérive en fréquence et donc d'enregistrer les coefficient de la fonction polynomiale caractérisant le plus précisément les paramètres $\frac{\Delta f}{f} = f(Temp)$ de chaque oscillateur.

- VCTCXO (Voltage Controlled Temperature Compensated Crystal Oscillator) : Même principe que le TCXO, mais avec une entrée permettant de faire varier la réactance en fonction de la tension appliquée pour ajuster la fréquence de l'oscillateur.
- MCXO (Microcontroller Compensated Crystal Oscillator) : L'idée est la même que pour les TCXO : compenser les dérives en fréquences en fonction de la température. Ces oscillateurs évaluent la température du quartz en fonction des différences d'évolution des modes harmoniques du quartz plutôt que par thermographie, ce qui permet d'avoir plus de précision sur la température du quartz. Ils utilisent des coupes de quartz spécifiques (SC-Cut) permettant d'avoir deux modes simultanément. La fréquence est ensuite ajustée numériquement soit en pilotant le quartz (toujours à l'aide de la réactance) soit en utilisant un synthétiseur de fréquence. Ces oscillateurs sont plus précis, mais plus coûteux que les TCXO.
- OCXO (Oven Controlled Crystal Oscillator) : Ici l'oscillateur est enfermé dans un four contrôlé pour maintenir une température constante, ainsi l'oscillateur n'est pas sujet aux variations de températures. Ces oscillateurs sont plus précis que les TCXO et les MCXO, mais plus énergivores et plus coûteux.
- DOCXO (Double Oven Controlled Crystal Oscillator) : Même principe que l'OCXO, mais avec un double four pour que la température soit encore plus constante. Plus précis, mais plus coûteux et plus énergivore qu'un OCXO.
- VCOCXO (Voltage Controlled Oven Controlled Oscillator) : Même principe que l'OCXO, mais avec une entrée permettant de faire varier la réactance en fonction de la tension appliquée pour ajuster la fréquence de l'oscillateur.

Les caractéristiques d'une partie de ces oscillateurs, disponibles sur étagère, sont regroupées dans le tableau 2.1. Tous les oscillateurs à quartz présentés dans ce tableau sont des 10 MHz et ils sont tous alimentés en 3.3V. Les caractéristiques des standards atomiques, tirés de [28] sont regroupés dans le tableau2.2. Les standards atomique sont plusieurs ordres de grandeur au dessus des oscillateurs à quartz en matière de vieillissement, stabilité en température

et stabilité court terme, mais ils ne sont pas du tout prévus pour les mêmes usages. Le coût financier, énergétique et les dimensions des standards atomique les limitent aux équipements de laboratoires ou à des cas très particuliers de systèmes embarqués tels que les satellites. Au sein des oscillateurs à quartz, les TCXO permettent un bon compromis entre la stabilité en fréquence et la consommation énergétique, pour des systèmes embarqués autonomes en énergie (typiquement les noeuds dans les WSN).

TABLE 2.1 – Caractéristiques des oscillateurs à quartz

	SPXO	TCXO	TCXO	OCXO
Référence	ASV [31]	T100F [32]	AST3TQ [33]	AOCJYR [34]
Vieillissement	+/- 5 ppm (/an à 25 °C)	+/- 3 ppm (/20 ans)	+/- 1 ppm (/an à 25 °C)	+/- 1 ppm (/an)
Stabilité en température	+/- 100 ppm (-20 °C à +70 °C)	+/- 100 ppb (0 à +70 °C)	+/- 280 ppb (-40 °C à +85 °C)	+/- 25 ppb (-40 °C à +85 °C)
Stabilité, $\sigma_y(\tau)$ (τ = 1s)	ND	ND	1e-10	< 1e-10
Taille	35 mm^2	24 mm^2	35 mm^2	73 mm^2
Puissance	6.6 mW (Typique)	21.45 mW (Maximale)	9.9 mW (Typique)	400 mW (Régime permanent à 25 °C)
Prix	0.5€	18€	22€	30€

TABLE 2.2 – Caractéristiques des standard à atomiques tiré de [28]

	Rubidium	RbXO	Cesium
Vieillissement	+/- 0.2 ppb (/an)	+/- 0.7 ppb (/an)	+/- 0.02ppb (/an)
Stabilité en température	+/- 0.3 ppb (-55 °C à +68 °C)	+/- 0.5 ppb (-55 à +85 °C)	+/- 0.02 ppb (-28 °C à +65 °C)
Stabilité, $\sigma_y(\tau)$ (τ = 1s)	3e-12	5e-12	5e-11
Taille	800 cm^2	1200 cm^2	6000 cm^2
Puissance	20 W	650 mW	60 W
Prix	$8k	$10k	$40k

2.3 Synchronisation des noeuds de capteurs

Les horloges locales des noeuds de capteurs dérivent différemment les unes des autres du fait des imperfections des oscillateurs énumérées précédemment. Ainsi, pour pouvoir maintenir une référence de temps commune aux noeuds de capteurs du réseau, ceux-ci ont besoin d'être synchronisés de manière périodique. Cette référence de temps peut être une référence externe au réseau et commune à tous les noeuds tels que le temps UTC propagé par les satellites des systèmes de géolocalisations si tous les noeuds sont équipés d'un récepteur GNSS (Géolocalisation et Navigation par un Système de Satellites). Cette référence peut aussi être externe et propagée au sein du réseau par un ou plusieurs noeuds équipés de récepteurs GNSS les autres noeuds doivent alors se synchroniser au(x) noeud(s) référent(s) à l'aide d'un protocole de communication RF. Enfin la source de synchronisation peut être interne au réseau, dans ce cas elle peut être l'horloge locale d'un des noeuds du réseau ou de plusieurs noeuds du réseau si un système de consensus est utilisé. Les noeuds se synchronisent entre eux, là encore, à l'aide d'un protocole de communication RF.

La synchronisation dans les systèmes distribués a tout d'abord été formalisée par Lamport [35]. Depuis, de nombreuses solutions ont vu le jour pour résoudre ce problème dans les réseaux filaires. Le Network Time Protocol (NTP) à été développé par Mills [36] [37] (dernière version [38]) afin de fournir une solution de synchronisation universelle pour synchroniser les horloges locales d'ordinateurs via internet. Le Precision Time Protocol (PTP) a ensuite été créé, puis normalisé sous la norme IEEE 1588 (dernière version [39]), afin d'améliorer la précision de synchronisation du NTP. Les versions actuelles du NTP et du PTP sont les standards de facto sur les machines connectées à internet, elles permettent une précision de synchronisation sous la microseconde, jusqu'à 100 ns suivant les implémentations. Une extension du PTP, le projet white rabbit [6][40] (intégré dans la version 2019 de la norme IEEE 1588) développé par le CERN pour ses équipements scientifiques permet d'atteindre une précision de synchronisation sous la nanoseconde. Ces protocoles sont cependant peu adaptés aux noeuds de capteurs sans-fil

qui disposent de ressources matérielles moindres, doivent limiter leur utilisation d'énergie et n'utilisent pas les mêmes topologies de réseau [41].

2.3.1 Synchronisation sans-fil par échanges de paquets

Afin de pallier les inconvénients des protocoles NTP et PTP Elson *et al* ont développé le Reference Broadcast Synchronization (RBS) [1] spécifiquement conçu pour la synchronisation des noeuds de capteurs. Ganeriwal *et al* présentent eux aussi une solution spécialisée pour la synchronisation des noeuds de capteurs : Timing-sync Protocol for Sensor Networks (TPSN) [2]. Ces deux protocoles de synchronisations se basent sur l'échange de paquets horodatés entre les noeuds de capteurs pour synchroniser le réseau. Les noeuds qui ne sont pas à portée du noeud "origine" de la synchronisation sont progressivement synchronisés à mesure que la synchronisation se diffuse dans le réseau par sauts. Ces sauts correspondent aux passages par des noeuds intermédiaires pour que les noeuds situés aux extrémités du réseau (par rapport au noeud "origine") puissent être synchronisés. Ces échanges de paquets sont sujets à de nombreux délais déterministes et stochastiques qui diminuent la précision de la synchronisation. Ces délais ont été formalisés par Kopetz and Ochsenreiter dans [42] et étendus dans [2] et [3]. La figure 2.3 tirée de [2] représente ces délais, ils sont les suivants :

1. Temps d'envoi : Temps nécessaire à l'assemblage du message et à l'envoi de la demande d'émission à la couche MAC du côté émetteur. Ce délai dépend de l'architecture et de l'OS/-logiciel du noeud de capteur. Généralement il n'est pas déterministe, car il dépend des délais d'interruption et de traitement de l'OS/logiciel utilisé sur le noeud de capteurs.
2. Temps d'accès : Temps d'attente pour l'accès au canal de transmission. Non déterministe, car il dépend du trafic réseau.
3. Temps de transmission : Temps nécessaire à l'émetteur radio pour transmettre les bits du message. Ce délai dépend de la vitesse de transmission de la radio ainsi que de la longueur du message. Il est déterministe.

4. Temps de propagation : Temps de propagation du message sous forme d'onde électromagnétique entre l'émetteur et le récepteur. Ce temps est déterministe en théorie, mais dans la pratique l'environnement des noeuds et les obstacles physiques potentiellement présents entre l'émetteur et le récepteur affectent le temps de propagation.
5. Temps de réception : Temps nécessaire au récepteur radio pour récupérer les bits du message et les transmettre à la couche MAC. Ce temps est déterministe.
6. Temps d'horodatage : Temps nécessaire au paquet pour remonter de la couhe MAC à la couche application du noeud pour être déchiffré et horodaté. Comme le temps d'envoi, ce délai n'est pas déterministe car il dépend des délais d'interruption et de traitement de l'OS/logiciel utilisé sur le noeud de capteurs.

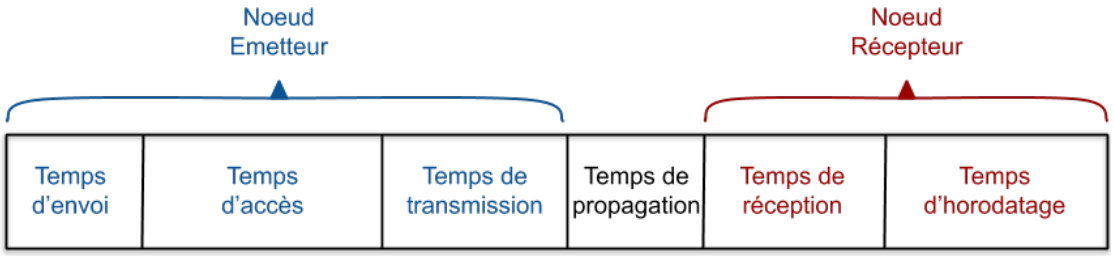


FIGURE 2.3 – Délais lors de l'échange de paquets tiré de [2]

La distinction entre les erreurs déterministes et non déterministes est importante car les erreurs déterministes peuvent souvent être éliminées si elles sont identiques sur deux noeuds ce qui est le cas lorsque celles-ci sont générées par un matériel radio identique (au sens même référence de composants) sur les deux noeuds. Pour supprimer une partie de ces erreurs plusieurs stratégies d'échanges de messages sont possibles. Dans l'échange bidirectionnel, développé dans [2], le noeud à synchroniser envoi un message horodaté à la référence qui horodate l'arrivée de ce message et renvoie un message horodaté contenant les deux autres horodatages. En horodatant l'arrivée de la réponse, le récepteur possède quatre horodatages qui lui permettent de calculer le déphasage de son horloge par rapport à l'horloge de référence en éliminant le temps de propagation (en partant du principe que celui-ci est symétrique) et en di-

minuant les temps de réception et de transmission du fait de leur soustraction. Dans l'échange entre récepteurs, développé dans [1], un noeud tierce diffuse un message non horodaté à tous les récepteurs en même temps. Ces derniers horodatent l'arrivée du message et s'échangent ensuite les horodatages pour calculer leur déphasage. Ce mécanisme permet de supprimer les temps d'envoi, d'accès et de transmission. Dans [43] et [2] l'horodatage au niveau de la couche MAC est proposé pour diminuer les erreurs d'envoi et de temps d'accès. L'horodatage au niveau de la couche MAC consiste à déplacer la génération et l'encapsulation de l'horodatage, de la couche application à la couche MAC, pour être le plus près possible de la transmission du message. Cette technique n'est pas applicable sur tous les émetteurs-récepteurs radio. Maroti *et al* ont décrit le Flooding Time Synchronisation Protocol dans [3], basé sur la synchronisation expéditeur à récepteur, comme TPSN, pour ajouter de la robustesse aux défaillances de noeuds ou de liaisons entre les noeuds. Contrairement à TPSN qui est basé sur une topologie en arbre et découvre le réseau lors d'une phase spécifique avec un protocole STP (Spanning Tree Protocol), FTSP est basé sur une topologie maillée avec un mécanisme d'élection périodique du noeud de référence. Dans le protocol FTSP, la synchronisation s'effectue par diffusion en partant de la référence. Celle-ci diffuse un paquet qu'elle horodate à l'aide de son horloge locale. Tous les noeuds à portée reçoivent le paquet et horodatent l'arrivée de celui-ci avec leur propre horloge locale. En faisant la différence entre les deux horodatages, les noeuds récepteurs peuvent calculer le déphasage de leur horloge avec celle du noeud émetteur et ainsi se synchroniser à celle-ci. Une fois les noeuds synchronisés ils diffusent à leur tour les messages de synchronisation pour les diffuser dans le réseau. De plus, le protocole FTSP est le premier à introduire un mécanisme pour réduire les erreurs d'interruptions en utilisant plusieurs horodatages par paquets.

Basés sur ces protocoles de nombreux protocoles de synchronisations par échanges de paquets spécialement conçus pour la synchronisation dans les réseaux de noeuds de capteurs ont vu le jour. Swain *et al* proposent une comparaison de la plupart de ces protocoles dans [44]. Une comparaison des implémentations existantes de ces protocoles est publiée dans [45]. Chen *et al*

proposent une implémentation d'un protocole de synchronisation sécurisé basé sur TPSN avec un chiffrement AES 128 bits [46]. Comme pour TPSN, l'horodatage MAC est utilisé. Du fait d'une fréquence d'oscillateur plus basse, l'implémentation présentée permet une synchronisation de l'ordre de la milliseconde. Le protocole R4Syn proposé par Djenouri *et al* dans [47] est basé sur une méthode d'échange entre récepteurs avec un changement périodique du noeud de référénce à l'aide de l'algorithme d'ordonnancement Round-Robin. Leur implémentation permet une erreur de synchronisation moyenne inférieure à 3 μs en moyenne pour un saut et 20 μs en moyenne pour 6 sauts. Dans [48], Lenzen *et al* présentent le protocole PulseSynch, similaire à FTSP mais optimisé pour synchroniser le réseau plus rapidement et diminuer la croissance de l'erreur de synchronisation quand le nombre de sauts augmente. Ce protocole est plus adapté aux réseaux comprenant un grand nombre de noeuds, l'implémentation décrite permet d'atteindre une erreur de synchronisation de 19 μs maximum (2.06 μs en moyenne) avec 31 noeuds.

Bien que les protocoles précédents utilisent tous un noeud référent (même si il peut changer), d'autres types de protocoles de synchronisation n'utilisent pas de références mais un temps global obtenu par un processus de consensus entre les noeuds du réseau. Cela a pour avantage l'utilisation de noeuds identiques, dans le sens où ils utilisent tous le même algorithme, et l'absence de processus de découverte du réseau et d'élection de référence. Ainsi les protocoles par consensus peuvent s'adapter à n'importe quelle topologie et sont donc robustes aux pertes de noeuds ou de liaisons. Dans [49], les noeuds se synchronisent localement par échange de paquets horodatés. Même si les noeuds ne sont pas tous synchronisés avec la même précision d'un bout à l'autre du réseau, il est démontré que toutes les horloges des noeuds du réseau convergent vers un temps global. L'implémentation présentée, testée sur un banc de test de 20 noeuds, présente une erreur de synchronisation plus grande que le FTSP globalement mais plus petite localement. Une amélioration du protocole est décrite dans [50] pour diminuer le nombre de messages une fois que la dérive de l'horloge locale est fixée. Leur implémentation permet d'économiser 82% de l'énergie utilisée dans [49]. Dans [51], deux algorithmes de consensus sont utilisés. Le premier syntonise les horloges et le second les synchro-

nise. Leur implémentation permet une erreur de synchronisation légèrement plus faible que le FTSP avec 9 noeuds disposés en grille de 3x3. Ces méthodes par consensus sont robustes et faciles à déployer sur n'importe quelle topologie mais souffrent d'un temps de synchronisation du réseau plus lent que les protocoles avec références et nécessitent plus d'échange de paquets. Dans [52], une approche hybride avec des consensus par clusters est décrite pour tenter d'améliorer le temps de convergence et diminuer le nombre d'échanges mais aucune implémentation réelle n'est testée.

Bien que ces protocoles offrent des mécanismes de synchronisation relativement simple à implémenter et à faible consommation, leurs performances sont limitées en matière de précision de synchronisation. Cette précision est d'autant plus limitée du fait de la courte portée des couches physiques utilisées entraînant l'utilisation de beaucoup de noeuds pour relayer les paquets de synchronisation d'un bout à l'autre du réseau avec une augmentation de l'erreur de synchronisation à chaque relais. Ces protocoles peuvent avoir leur utilité pour des applications vibratoires ou acoustiques avec un déploiement de noeuds sur une surface relativement faible sous contrainte d'une période de synchronisation de l'ordre de la seconde ou de la minute. Cependant, leur faible précision de synchronisation et leur courte portée de communication les rend peu adaptés au contrôle de santé des grandes structures telles que les ponts, le réseau ferroviaire ou les lignes haute tension qui nécessitent des topologies particulières souvent linéaire et avec de grandes distances entre les noeuds.

Certaines implémentations de protocoles de synchronisation avec des communications sur les bandes libres 868 MHz (ou 915 Mhz au USA) permettent de plus grandes distances entre les noeuds. Dans [53] un protocole de synchronisation par échange de messages est proposé pour le contrôle du réseau d'égouts d'une ville américaine. La particularité de leur implémentation réside dans le fait qu'ils utilisent une modulation à étalement de spectre par saut de fréquence (FHSS) sur la bande de fréquence 902-928 MHz. Cette particularité leur permet d'atteindre des distances de communications de 700 mètres en zone urbaine, leur protocole de synchronisation permet une erreur de synchronisation moyenne de 16.77 μs pour un saut et sans horodatage

au niveau MAC (leur émetteur-récepteur radio ne le permet pas). Dans [54], les protocoles de communication LoRaWAN et NB-Iot sont utilisés pour l'analyse vibratoire de machines industrielles. Les noeuds communiquent avec une passerelle en LoRaWAN et la passerelle renvoie les données sur un serveur avec NB-Iot. Les noeuds sont synchronisés à l'aide de messages horodatés envoyés par la passerelle. Les noeuds sont synchronisés avec une erreur inférieure à 5 μs.

2.3.2 Synchronisation avec les normes IEEE 802.11 et IEEE 802.15.4

D'autres approches pour la synchronisation de noeuds se basent sur des normes déjà existantes telles que les normes IEEE 802.11 (WiFi) pour les réseaux sans fils locaux [55] ou IEEE 802.15.4 [56] pour les LRWPAN (Low Rate Wireless Personal Area Network), qui définissent des spécifications pour la couche physique et la couche MAC. Mahmood *et al* comparent plusieurs implémentations de protocoles de synchronisation conformes à la norme IEEE 802.11 ou utilisant un émetteur-récepteur conforme à la norme IEEE 802.11b dans [57]. Les mécanismes de synchronisations conformes à la norme 802.11 sont la TSF (Timing Synchronization Function), la TM (Timing Measurement), et la TA (Timing Advertisement). La TSF et la TM sont basées sur la diffusion de messages horodatés par le point d'accès en mode infrastructure. La TM est basée sur un échanges bidirectionnel entre les stations en mode ad-Hoc. L'implémentation de la TSF permet une erreur de synchronisation moyenne de 4 μs avec un écart-type inférieur à 1 μs tandis que l'implémentation de la TA permet d'obtenir une erreur moyenne de 0.5 μs avec un écart-type de 2.5 μs [57], [58]. Le PTP (IEEE 1588) a été implémenté avec un chip radio 802.11b dans [59]-[62] sur des ordinateurs sous Windows et Linux et sur des FPGA. L'implémentation sur FPGA permet l'horodatage dans les couches basses tandis que les versions sur Windows et Linux doivent se contenter des compteurs et interruptions proposés par les systèmes d'exploitation. Les versions avec horodatage matériel (FPGA) permettent d'atteindre des erreurs de synchronisation de l'ordre de la nanoseconde tandis que les versions purement logicielles sont de l'ordre de la microseconde. Exel *et al* [4] proposent une amélioration de la version ho-

rodatage matériel en prenant en considération les délais sur la chouche physique. Leur solution permet une erreur de synchronisation inférieure à la nanoseconde avec 240 ps d'erreur moyenne et 531 ps d'écart-type. Similairement dans [63], Mahmood *et al* améliorent la version logicielle sous Linux en prenant en compte les délais d'interruptions ainsi que les délais déterministes du chip radio. Leur solution permet une erreur de synchronisation moyenne de 59 ns avec un écart-type de 0.46 μs. Le NTP a été implémenté avec la norme IEEE 802.11 dans [64] de manière purement logicielle. Leur implémentation permet une erreur de synchronisation de l'ordre de la milliseconde. L'implémentation décrite dans [65] permet une amélioration de la synchronisation avec une erreur moyenne de 0.5 ms (écart-type de 5.27μs) en implémentant l'horodatage au niveau driver. Dans [66], une implémentation logicielle du protocole de synchronisation RBS avec la norme IEEE 802.11 permet d'atteindre une erreur de synchronisation moyenne de 0.2 μs avec un écart-type de 0.18 μs. Ces méthodes de synchronisation permettent une meilleure précision que les protocoles par échange de paquets décrits précédemment au coût d'une implémentation plus complexe nécessitant un chip radio IEEE 802.11, du matériel capable de supporter les piles logicielles pour le 802.11 et un FPGA pour les implémentations avec horodatage matériel.

La norme IEEE 802.15.4 est plus adaptée aux réseaux de noeuds de capteurs du fait de son implémentation moins complexe et de sa plus faible consommation énergétique. Cependant le débit de données de 250 kbits/s est moins important que les débits atteignables par certaines spécifications de la norme IEEE 802.11. Dans [67] une implémentation de cette norme est utilisée et l'interférence constructive est exploitée pour synchroniser les noeuds de capteurs. Leur protocole de synchronisation permet une synchronisation du réseau plus rapide que le FTSP [3] et le protocole PulseSync [68] avec une erreur de synchronisation de l'ordre des microsecondes après huits sauts. Dans [69] le protocole FTSP est utilisé avec le protocole de communication ZigBee (lui-même basé sur une implémentation IEEE 802.15.4 auquel il ajoute des couches de plus haut niveau) avec une erreur de synchronisation maximale de 61 μs. L'utilisation de la technologie UWB (Ultra Wide Band) dans la couche physique a été intégré dans la

version 2011 de la norme 802.15.4. Les systèmes UWB à impulsions courtes, ou Impulse Radio Ultra Wide Band (IR UWB), sont particulièrement adaptés à la géolocalisation et à la synchronisation du fait de l'estimation plus aisée du temps de vol des impulsions [70]-[72]. Beluch *et al* utilisent la modulation IR UWB avec un schème CLD (Cross Layer Design [73]) pour propager la synchronisation dans un réseau a topologie en arbre dans [5]. Leur solution permet d'atteindre une erreur moyenne de 374 ps entre deux noeuds avec un écart-type de 677 ps. Dans [74], un protocole de synchronisation par consensus est développé pour les réseaux communicants en UWB. Une implémentation de leur protocole est testée dans [75] et obtient une erreur de synchronization moyenne de l'ordre de la nanoseconde avec un écart-type de 3 nanosecondes. L'utilisation de l'UWB pour les couches basses de l'IEEE 802.15.4 permet d'obtenir des erreurs de synchronisation du même ordre de grandeur que les implémentations du PTP IEEE 802.11 avec une complexité réduite et une plus faible consommation d'énergie mais la portée des communications est plus faible.

2.3.3 Synchronisation par GNSS

Le principe de positionnement par satellite fut conceptualisé dans les années 60 avec la création du GPS (Global Positioning Sytem). La position est déterminée par trilatération avec les mesures de distances entre un récepteur et les satellites. Ces distances sont mesurées par le récepteur en multipliant la différence entre la date d'émission et de réception du signal par la vitesse de celui-ci. La position du récepteur $\{x_u, y_u, z_u\}$ est liée à la distance d'un satellite i par l'équation suivante tirée de [76] :

$$\rho_i = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} + ct_u$$

Avec ρ_i la pseudo-distance du récepteur par rapport au satellite, $\{x_i, y_i, z_i\}$ la position du satellite, c la célérité et t_u le déphasage entre les horloges du récepteur et du satellite. Le système peut être résolu à partir de l'observation de quatre satellites. Chaque satellite diffuse un mes-

sage de navigation contenant des informations sur la constellation des satellites et sa trajectoire pour que le récepteur puisse calculer sa position. Ces trajectoires sont diffusées pour tous les satellites de façon grossière dans l'almanach. Les éphémérides correspondent à la trajectoire plus précise d'un satellite ; chaque satellite diffuse ses propres éphémérides. Étant donné que les satellites sont synchronisés au temps UTC ; une fois le déphasage t_u calculé ; l'horloge du récepteur est synchrone au temps UTC et celui-ci peut délivrer un signal de synchronisation PPS (Pulse Per Second) de fréquence 1 Hz.

La synchronisation par GNSS est relativement peu présente dans la littérature sur la synchronisation des noeuds de capteurs du fait du coût financier et énergétique des récepteurs GNSS [1]-[3], qui sont nécessaires sur tous les noeuds du réseau. Cependant, la synchronisation par GNSS présente deux avantages pour les applications SHM, tout d'abord la précision de synchronisation est inférieure à la centaine de nanosecondes sur la plupart des récepteurs simple fréquence (inférieure à 60 ns sur récepteur GPS ublox NEO6T [77]) et ensuite la simplicité de déploiement. Les capteurs ne se synchronisant pas entre eux mais à une référence commune, la synchronisation est agnostique de la topologie réseau et ne nécessite pas de découverte du réseau ni de processus de changement de référence. Contrairement aux méthodes par consensus, la synchronisation est quasiment immédiate et simultanée pour tous les noeuds du réseau, sans échange de messages. La synchronisation avec un récepteur GNSS est utilisée dans les VANET (Vehicular Ad-hoc Network) [78]-[80], du fait que les noeuds sont souvent déjà équipé d'un récepteur GNSS, que la topologie du réseau évolue et que la synchronisation doit être précise et fiable. La synchronisation par GPS est aussi utilisée pour le SHM dans [7] et [81]. Cette approche permet une précision de synchronisation des événements avec une erreur inférieure à 60 nanosecondes grâce à l'utilisation d'un FPGA pour avoir un horodatage matériel du signal de synchronisation et des évènements à horodater. Cependant le coût énergétique d'un récepteur GPS tout le temps allumé est considérable (120mW sous 3V avec une antenne passive [77]). Dans [82], Sazonov *et al* proposent une solution hybride avec une décomposition du réseau en cluster ; chaque cluster possède un noeud spécifique synchronisé par GPS qui est chargé de dif-

fuser les messages horodatés au reste du cluster. Leur méthode présente un compromis entre l'efficacité énergétique d'un protocole de synchronisation par diffusion d'un message horodaté et l'adaptabilité de la synchronisation GPS à tous les types de topologies. Une méthode similaire est employée dans [83] pour le contrôle de santé d'un pont ferroviaire avec une erreur de synchronisation inférieure à 50 microsecondes sur les horodatages des échantillons.

L'approche de la synchronisation par GPS décrite dans le reste du document est différente. Plutôt que de diminuer la consommation avec une méthode hybride, les noeuds possèdent tous un récepteur GPS mais celui-ci n'est allumé que périodiquement en fonction de la précision de synchronisation attendue. Du fait de la grande précision de synchronisation par GPS, il est possible de laisser dériver les horloges plus longtemps que les méthodes par échange de messages et ainsi d'espacer les synchronisations. Le GPS étant éteint entre deux synchronisations, la consommation énergétique dépend de la dérive des horloges et du niveau de synchronisation attendu. Dans la suite de ce manuscrit, la période pendant laquelle le GPS est éteint est appelée *l'apnée*.

2.3.4 Fréquence de synchronisation

Un autre aspect de la synchronisation des noeuds est la fréquence de synchronisation, c'est-à-dire le nombre de synchronisations par seconde (en réalité les synchronisations sont espacées de plus d'une seconde dans tous les protocoles, la fréquence est donc toujours inférieure à 1). En effet, comme il a été expliqué dans la section précédente, le temps local des noeuds de capteurs dérive du fait des sensibilités et des oscillateurs. Les erreurs d'horodatage d'événements ou d'échantillons vont donc augmenter entre les synchronisations. Pour ralentir la dérive ou augmenter le temps entre deux synchronisations consécutives, des modèles d'horloges peuvent être utilisés afin de prédire la fréquence et le déphasage de l'horloge. Un modèle d'horloge basique est le modèle linéaire ou la fréquence de l'oscillateur de l'horloge est considérée constante. Étant donnée que la fréquence réelle de l'oscillateur n'est pas connue, le temps compté par l'horloge dérive du vrai temps. Ce modèle est présenté dans [84] sous la

forme suivante :

$$C(t) = \theta + f.t \tag{2.1}$$

Avec C(t) le temps local donné par l'horloge au "vrai" temps t, θ le déphasage et f le skew. À partir de deux points de synchronisation, les paramètres θ et f peuvent être estimés. Ce modèle est utilisé dans une partie des protocoles de synchronisations vus précédemment avec l'idée suivante : si la fréquence évolue lentement, le modèle est valide pour un certain laps de temps et la fréquence ainsi que le temps local peuvent être estimés plus précisément entre deux synchronisations en extrapolant la courbe. Dans [1], [3], [47]-[49], [51], [85], des régressions linéaires sont utilisés pour estimer les paramètres θ et f et diminuer la dérive entre les synchronisations. Dans [86], l'estimation du skew et du déphasage sont séparés, le calcul du déphasage des horloges ne sont pas calculés par les noeuds mais par la référence à l'aide d'un échange bidirectionnel inversé. Le skew est par contre calculé sur les noeuds avec une diffusion classique. Cette méthode a pour objectif de diminuer la charge de calcul des noeuds en augmentant la charge de la référence qui est considérée non autonome.

Les variations de fréquences étant principalement dues aux conditions environnementales des oscillateurs, plusieurs travaux portent sur l'inférence de la fréquence à partir de l'environnement du noeud. Dans [87]-[89] la température ambiante est utilisée pour estimer la fréquence entre deux synchronisations. Dans [90], c'est la tension d'alimentation qui est utilisée pour estimer la fréquence. Rowe *et al* procèdent différemment dans [91]. Plutôt que d'estimer la fréquence, les horloges sont syntonisées sur la fréquence du courant dans les lignes hautes tension, utilisées pour le transport de l'électricité, censée être plus stable sur le long terme.

Pour aller plus loin, certains protocoles de synchronisation permettent d'adapter la fréquence de synchronisation en fonction de l'erreur d'horodatage maximale admise sur les noeuds. Dans [92], Jain et Chang introduisent un algorithme pour optimiser la fréquence d'échantillonnage de capteurs dans les systèmes de mesures réparties afin d'optimiser l'utilisation de la

bande passante. Leur algorithme se base sur les différences normalisées entre les prédictions d'un modèle et les observations pour augmenter ou diminuer la fréquence d'échantillonnage en fonction d'une consigne normalisée. Hamilton *et al* décrivent un protocole de synchronisation adaptatif en modifiant l'algorithme de Jain et Chang [93] pour converger vers la consigne (correspondant à l'erreur maximale autorisée) plus rapidement. Un filtre de Kalman est utilisé afin d'estimer les paramètres du modèle d'horloge à partir des messages horodatés, la période de synchronisation est adaptée dynamiquement en fonction de la matrice de variance-covariance calculée dans le filtre, des horodatages observés et de la consigne. Une boucle de rétroaction est aussi utilisée dans [85] pour faire converger l'erreur de synchronisation vers une consigne mais avec un modèle d'horloge linéaire sans filtre pour calculer les prédictions de déphasage. Une extension du protocole FTPS est décrite dans [94] pour faire évoluer la fréquence de synchronisation en fonction d'une consigne sur l'erreur. Une boucle de rétroaction est toujours utilisée, et la stabilité en fréquence des horloges des noeuds voisins est utilisée pour le calcul de la fréquence de synchronisation.

2.4 Conclusion

Dans le chapitre précédent, les contraintes liées à l'implémentation de systèmes SHM avec des réseaux de capteurs ont été décrites. La solution de synchronisation attendue pour cet usage doit être robuste, permettre un déploiement des noeuds de capteurs aisé sur de grandes structures et doit permettre un niveau de synchronisation pouvant descendre à la centaine de nanosecondes. Les principales solutions de synchronisations passées en revues dans ce chapitre sont répertoriées dans le tableau 2.3.

Les solutions par échanges de paquets "classiques" [1]-[3] sont limitées du point de vue de la précision même si certaines implémentations [53], [54] sur les bandes de fréquences libres (868 MHz EU ou 915 MHz USA) permettent de couvrir de grandes distances. Les solutions basées sur le standard IEEE 802.11 permettent une synchronisation à la nanoseconde mais sont com-

plexes à implémenter surtout pour les réseaux de capteurs sans-fil. La norme IEEE 802.15.4 permet aussi des précisions de synchronisation de l'ordre de la nanoseconde avec l'utilisation de la technologie IR-UWB [5], [75], mais elle est compliquée à déployer sur de grandes infrastructures du fait de la faible portée des noeuds. Les récepteurs GPS ont l'inconvénient d'être énergivores mais ils permettent d'accéder au temps UTC avec une grande précision de l'ordre de la dizaine de nanosecondes et leur déploiement sur de grandes infrastructures est très simple [81]-[83]. Dans la suite de ce document nous nous intéresserons à la synchronisation par GPS et son optimisation énergétique en fonction du besoin de synchronisation de l'application.

Protocole	Distance inter-noeuds théorique	Erreur de synchronisation
TPSN [2]	$\approx$ 100 à 300 m (Berkley motes [95])	16.9 μs avg
RBS [1]	$\approx$ 100 à 300 m (Berkley motes [95]) ou 20 à 150 m (802.11)	6.29 μs avg (802.11) 29.1 μs avg (Mica)
FTSP [3]	$\approx$ 100 à 300 m (Berkley motes [95])	1.48 μs avg
802.11 TSF [55]	$\approx$ 20 à 150 m	4 μs avg (< 1 μs std dev)
802.11 TA [55]	$\approx$ 20 à 150 m	0.5 μs avg (2.5 μs std dev)
PTP avec 802.11 [4]	$\approx$ 20 à 150 m	240 ps avg (531 ps std dev)
Synchronisation avec UWB [5]	< 10 m	374 ps avg (677 ps std dev)
Synchronisation avec LoRa [54]	$\approx$ 1 à 15 kms	5 μs max
Synchronisation par GPS	illimitée	< 60 *ns*

TABLE 2.3 – Distance et erreurs de synchronisation des différents protocoles

CHAPITRE 3

Conception et validation d'une architecture de noeud de capteurs pour l'évaluation hors ligne de stratégies de synchronisation

Le chapitre précédent a fait émerger le besoin de mécanismes de synchronisation par GPS, avec une consommation énergétique optimisée, pour certaines applications de SHM. Pour ce faire, une extinction périodique du récepteur GPS paramétrée par la précision de synchronisation a été proposé. Dans le but d'évaluer ce mécanisme de synchronisation, une architecture de noeud de capteurs est proposée. Ce chapitre se concentre dans un premier temps sur la description de cette architecture synchronisé par GPS. Dans un second temps, l'implémentation de cette architecture sur un noeud expérimental composé d'un FPGA Spartan 6 et d'une carte Raspberry Pi 3 est validée. Enfin, trois filtres sont étudiés dans le but d'améliorer la précision de la synchronisation en supprimant une partie des erreurs générées par le récepteur GPS. Les expérimentations décrites dans ce chapitres permettent la constitution d'un jeu de données qui sera utilisé pour l'évaluation de différents scénarios de contrôle du récepteur GPS dans le chapitre suivant.

Sommaire

3.1 Description de l'architecture des noeuds

3.1.1 Architecture du noeud de capteurs

L'architecture présentée dans cette section correspond à des noeuds de capteurs tels qu'ils sont classiquement utilisés pour les applications SHM, c'est-à-dire pour acquérir et éventuellement traiter ou renvoyer à un superviseur des échantillons horodatés de signaux physiques. Ces noeuds de capteurs ne sont pas prévus pour être des *actionneurs*, dans le sens ou ils ne servent pas à piloter des systèmes. Même si certaines applications nécessitent l'injection de signaux dans le matériau observé, le but de ces injections est l'auscultation du matériau et non la commande d'un système. Les noeuds de capteurs propsés ici sont basés sur une unité d'horodatage implémentée sur un circuit logique programmable (type CPLD ou FPGA) et connectée à une unité contrale. L'unité centrale est classiquement un microcontrolleur ou un SoC exécutant

l'application principale. Le terme application principale réfère ici au code d'acquisition, et d'exploitation ou de traitement des données. L'unité d'horodatage sert à dater les fronts des signaux numériques en entrée du noeud de capteurs. Ces fronts correspondent à des événements qui ont besoin d'être datés pour pouvoir être exploités par l'application principale, typiquement ce sont des sorties de comparateurs ou signaux de validations en sortie d'un Convertisseur Analogique Numérique (CAN). L'utilisation de matériel dédié à l'horodatation présente plusieurs avantages :

- Pas de surcoût logiciel : L'utilisation d'un circuit logique programmable supprime "l'overhead" propre aux mécanismes d'interruption sur les microprocesseurs. Ainsi le délai entre la détection d'un front sur une entrée et l'enregistrement du temps est déterministe dans le sens ou il sera toujours égal au même nombre de coups d'horloge. D'autre part, cet "overhead" est le même pour deux noeuds possédant la même implantation de l'unité d'horodatage. Dans un contexte de datation et comparaison d'un même signal par N noeuds identiques, cet overhead s'annule par comparaison relative.
- Adaptation de l'unité d'horodatage : Sur la majorité des microcontrolleurs le nombre de compteurs (nécessaires pour la datation d'évènements) est limité et la granularité temporelle est fixée par la fréquence du microcontrolleur. L'utilisation d'un circuit logique programmable offre de la flexibilité en permettant de définir le nombre d'entrées de capture souhaité ainsi que la granularité temporelle sur les circuits intégrant une PLL.
- Indépendance de l'unité centrale : Le fait de séparer l'horodatage du reste de l'application permet de libérer l'unité centrale des contraintes de temps-réel lorsque le noeud est utilisé uniquement en mode capteur. L'unité d'horodatage stocke les horodatages et l'unité centrale vient les lires quand nécessaire, par paquet et donc de manière asynchrone. La charge de temps-réel dur étant portée par le circuit logique, le microcontroleur/CPU/SoC peux éxecuter des applications et/ou OS sans contraintes de temps-réel.
- Bien que l'architecture utilisée soit dédiée, les choix d'implémentation demeurent volontairement générique afin que les briques validées lors de ce travail de thèse soient trans-

posables à d'autres implémentations.

Un récepteur GPS est utilisé pour synchronizer le noeud de capteurs. Ce récepteur est connecté à l'unité d'horodatage via une liaison série et un signal PPS. La liaison série permet de transférrer la date UTC sous forme d'epoch (i.e. le nombres de secondes écoulées depuis le 6 janvier 1980) et le signal PPS marque le début de la seconde UTC. La date epoch future est transférée chaque seconde avant d'être validée au prochain front montant du signal PPS comme illustré Figure 3.1. Le schéma 3.2 représente deux noeuds de capteurs A et B dans le cadre d'une surveillance acoustique.

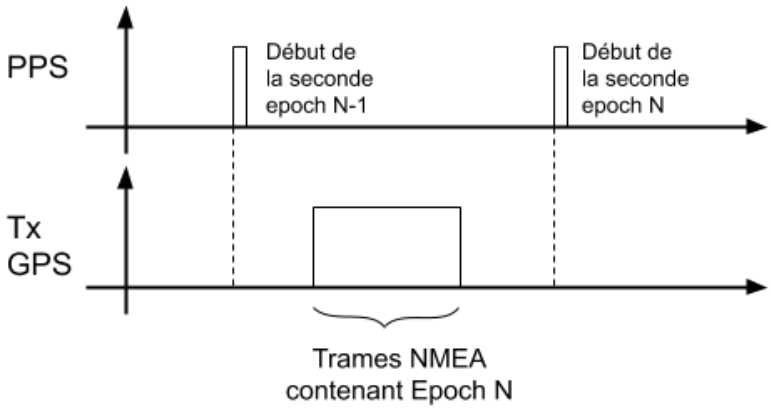


FIGURE 3.1 – Synchronisation GPS

3.1.2 Calcul des horodatages

Un signal "fake PPS" est défini comme étant une version locale (au noeud) du signal PPS. Cette version locale est créée à l'aide d'un compteur qui compte les battements de l'oscillateur du noeud. Ce compteur compte jusqu'à $N_{ftheorique}$, qui correspond au nombre de coups d'horloges pendant une seconde à la fréquence théorique de l'oscillateur local. Le déphasage θ du noeud de capteurs par rapport au temps GPS à la n^{ieme} seconde UTC correspond au temps écoulé entre le front montant (ou descendant) du signal PPS provenant du récepteur GPS et le front montant (ou descendant) du signal "fake PPS". Le paramètre Δ correspond au délai entre un front montant du signal "fake PPS" et un front montant sur une entrée numérique (i.e. un

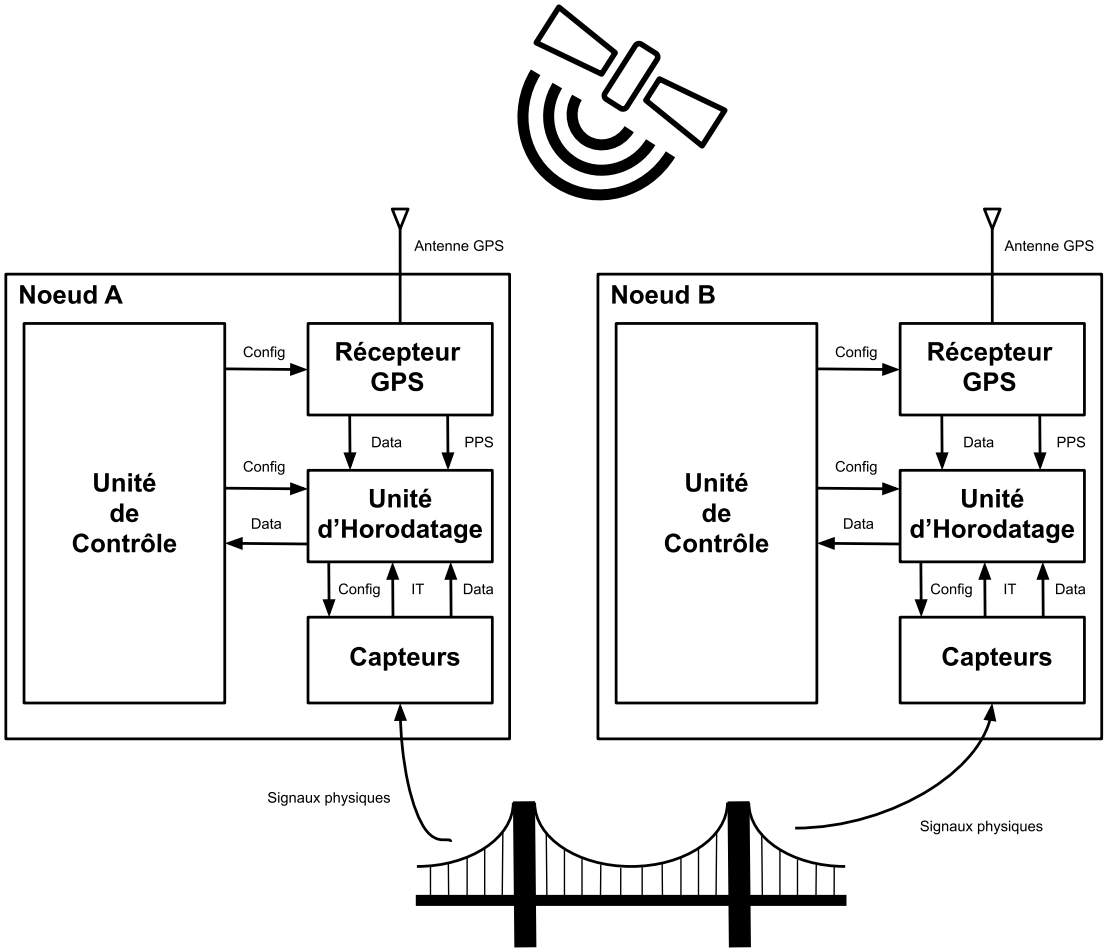


FIGURE 3.2 – Architecture de deux noeuds A et B

événement à horodater). Les variables θ et Δ sont représentés sur le chronogramme en Figure 3.3. La date $t[n]$ d'un évènement survenant pendant la n^{ieme} seconde locale est calculée à partir de la date UTC, récupérée dans l'epoch, plus le délai entre le front montant du signal PPS et la détection de l'évènement. Ce délai correspond à la différence de θ et Δ, donc $t[n]$ s'exprime par :

$$t[n] = t_{pps}[n] + \theta[n] - \Delta[n] \tag{3.1}$$

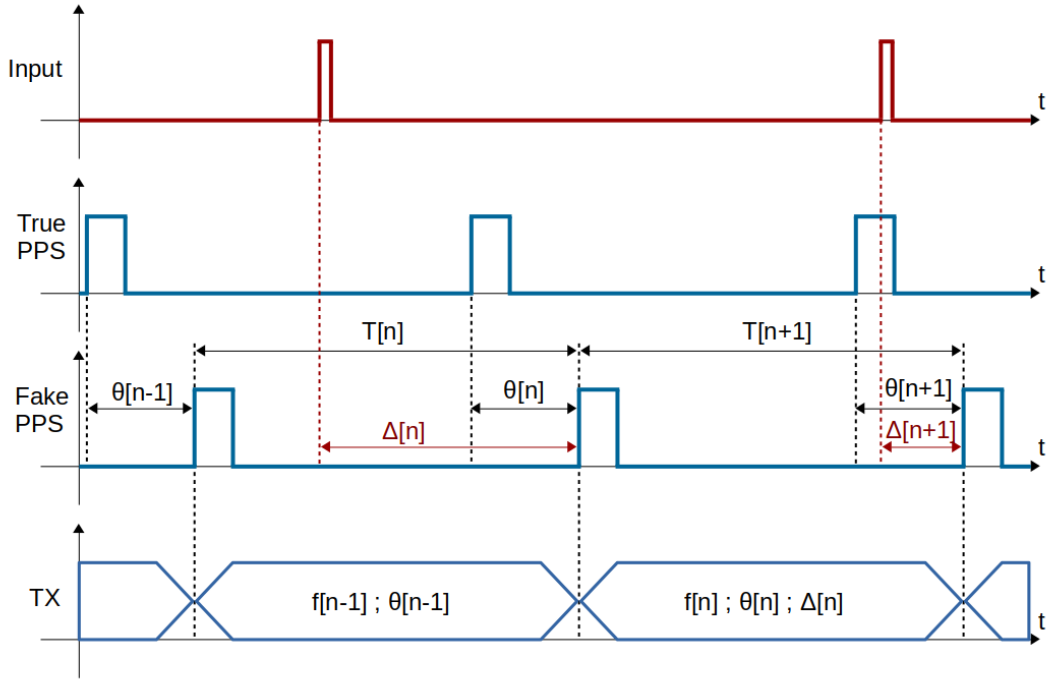


FIGURE 3.3 – Chronogramme des signaux

3.1.3 Architecture de l'unité d'horodatage

Pour pouvoir tester l'horodatage des noeuds, l'unité d'horodatage a été implémentée sur un FPGA Spartan 6 [96]. L'architecture de cette unité est décrite sur la Figure 3.4. Elle comprend plusieurs compteurs pour pouvoir mesurer des durées (Δ et θ), un compteur pour mesurer la fréquence, un diviseur de fréquence pour générer le signal "fake PPS" et un bloc interface pour pouvoir communiquer avec l'extérieur (récepteur GPS, unité de contrôle et capteur de température). Étant donné que l'objectif des expérimentations est d'acquérir des jeux de données pour pouvoir tester différents modèles d'oscillateurs avec différents temps d'apnée (temps pendant lequel le GPS est éteint cf 2), le déphasage entre l'horloge locale et le temps UTC n'est pas compensé mais mesuré. Ainsi les horodatages peuvent être recalculés hors ligne à l'aide de l'équation 6.1, depuis le jeu de données, pour différents paramètres de synchronisation tel que la durée d'apnée ou le modèle d'horloge utilisé pendant l'apnée. L'horloge locale n'est donc

pas asservie sur le signal PPS. Le compteur de l'horloge est borné par N_0, un entier correspondant au nombre de coups d'horloge qui devraient être comptés pendant une seconde si l'oscillateur fonctionnait parfaitement à sa fréquence nominale f_0. Le déphasage de l'horloge locale est observé à chaque seconde en mesurant le décalage entre le signal "fake PPS" et le vrai signal PPS issu du récepteur GPS (ici un recepteur Ublox NEO6T [77]). Les paramètres θ

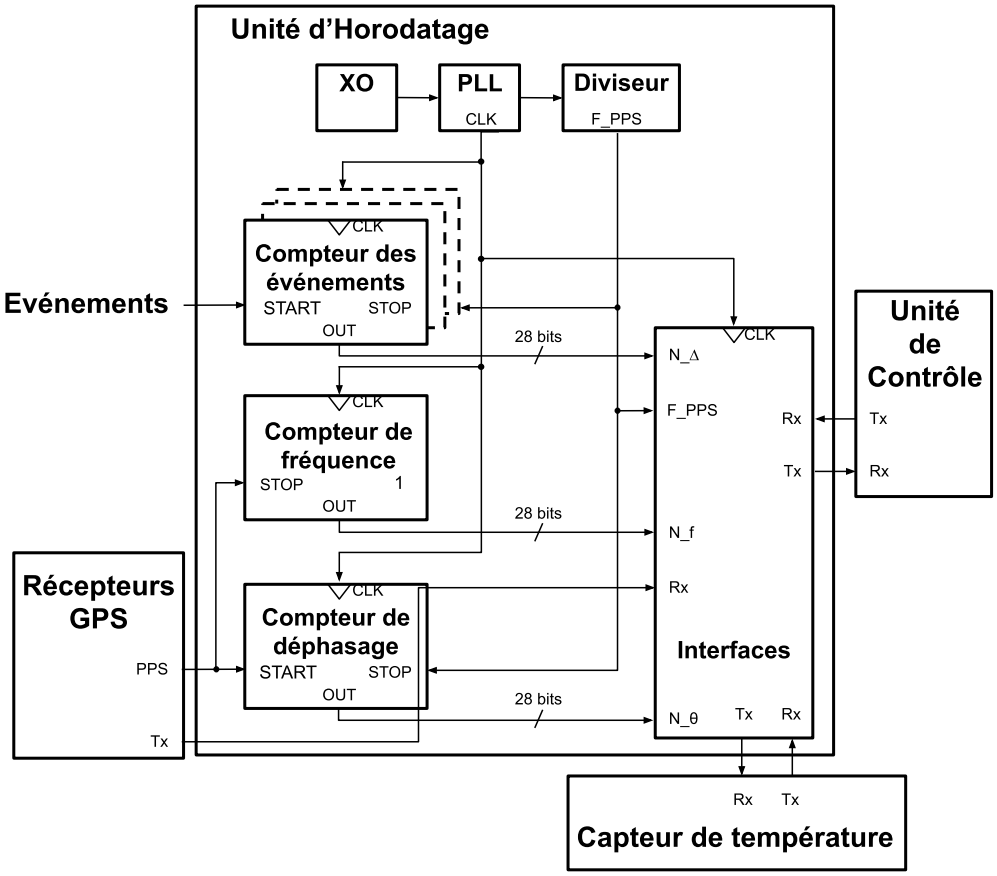


FIGURE 3.4 – Unité d'horodatage

et Δ de l'équation 6.1, permettant de calculer les horodatages, ne sont pas directement mesurables dans l'unité d'horodatage. L'unité d'horodatage permet d'obtenir N_θ et N_Δ à l'aide des compteurs de déphasage et d'événements. Ces paramètres sont des entiers correspondant aux nombres de coups d'horloge locale mesurés pendant les temps θ et Δ. Pour les convertir en

temps il faut utiliser la fréquence de l'oscillateur local $f[n]$, ainsi l'Eq 1 devient :

$$t[n] = t_{pps}[n] + \frac{1}{f[n]}(N_\theta[n] - N_\Delta[n]) + \varepsilon[n] \tag{3.2}$$

Avec ε représentant les erreurs de quantification des compteurs. Ces erreurs sont dûes au fait que les compteurs ne peuvent compter qu'un nombre entier de coups d'horloge. La fréquence $f[n]$ non plus n'est pas mesurée, c'est le nombre de coups d'horloge N_f pendant la durée T_{GPS} qui est mesurée par un compteur dans l'unité d'horodatage. La fréquence moyenne de l'oscillateur local pendant la $n^{ième}$ seconde s'exprime alors par :

$$f[n] = \frac{N_f[n]}{T_{GPS}[n]} \tag{3.3}$$

Ainsi l'équation 3.2 devient :

$$t[n] = t_{pps}[n] + \frac{T_{GPS}}{N_f[n]}(N_\theta[n] - N_\Delta[n]) + \varepsilon[n] \tag{3.4}$$

La température ambiante est également échantillonnée à chaque seconde par l'unité d'horodatage via un capteur de température numérique relié à l'unité par une liaison série. L'ensemble de ces paramètres est transmis à l'unité de contrôle via une liaison série à chaque cycle du signal "fake PPS". Acquis sur plusieurs minutes à plusieurs heures, ces données constituent la base de données brutes nécessaire à l'évaluation et la validation, en post traitement, de différents scénarios de synchronisations. Dans la suite de ce chapitre, toutes les analyses sont réalisées à l'aide du logiciel de calcul numérique GNU Octave à partir des jeux de données.

3.2 Validation des noeuds

3.2.1 Validation de l'unité d'horodatage

Afin de valider la capacité d'un noeud de capteurs à horodater des événements à l'aide de l'unité d'horodatage, un premier banc de test à été réalisé : deux noeuds de capteurs A et B, chacun constitué d'une unité d'horodatage (UH) implémenté sur un kit de développement FPGA Spartan 6 [97] et d'une unité centrale (UC) basés sur un Raspberry Pi 3 sont connectés à deux générateurs basse fréquence (GBF). Le premier GBF simule le signal PPS issu d'un récepteur GPS et le deuxième GBF des événements à horodater. Le signal PPS simulé et le signal des événements (*Events*) sont respectivement connectés aux entrées PPS et événements (*Events*) des noeuds de capteurs A et B comme illustré en Figure 3.5. Ainsi les unités d'horodatage des deux noeuds A et B recoivent le même signal de synchronization et les mêmes événements et doivent donc fournir, a priori, les mêmes horodatages aux unités centrales aux erreurs de quantification près.

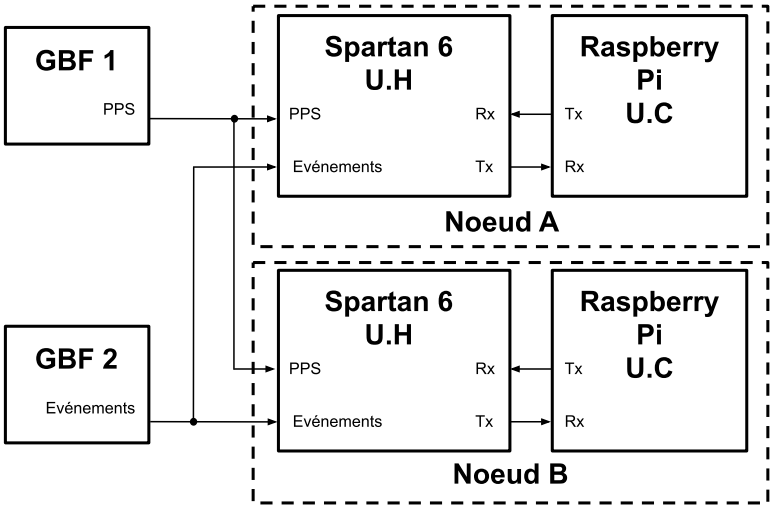


FIGURE 3.5 – Schéma bloc du banc de test avec GBF

La période d'apparition des événements (i.e la durée entre deux fronts montants consécutifs sur le signal *Events*) a été arbitrairement fixé à 2.34 secondes afin de créer un décalage entre

les événements et le signal PPS permettant de tester les compteurs de l'unité d'horodatage sur toute leur plage. Les deux UH des noeuds de capteurs ont enregistré et retransmis aux UC les variables N_θ, N_Δ et N_f pendant plus d'une heure (4982 secondes). Les horodatages ont ensuite été calculés hors ligne à l'aide de l'équation 3.4 pour les deux noeuds de capteurs. L'erreur de synchronisation entre les deux capteurs pour l'horodatage d'un événement n correspond à la différence entre les horodatages des noeuds A et B tel que :

$$err[n] = t_A[n] - t_B[n] \tag{3.5}$$

La moyenne, la médiane, l'écart type ainsi que les valeurs minimales (au sens mathématique du terme) et maximales des erreurs pour la totalité de l'expérimentation sont donnés dans le tableau 3.1. Étant donné que les deux unités d'horodatage sont synchronisées par le même signal de référence, les erreurs de synchronisation sur les horodatages sont principalement dues aux erreurs de quantification des compteurs. L'histogramme des erreurs est représenté en figure 3.6.

Minimum	Moyenne	Médianne	Ecart type	Maximum
-9.71 ns	31.58 ps	34.17 ps	3.02 ns	8.82 ns

TABLE 3.1 – Statistiques des erreurs de synchronisation - GBF

Le signe des erreurs n'est pas utile dans l'évaluation de la précision de synchronisation car il permet uniquement de savoir si le noeud A est en avance ou en retard sur le noeud B. Pour cette raison, la moyenne, la médiane, l'écart type, et le maximum de la valeur absolue des erreurs sont répertoriés dans le tableau 3.2. Les erreurs de synchronisation sur les horodatages sont toutes inférieures à 10 ns et l'EAM (erreur moyenne absolue) est de 2.42 ns.

Minimum	Moyenne	Médianne	Ecart type	Maximum
0.43 ps	2.42 ns	2.09 ns	1.80 ns	9.71 ns

TABLE 3.2 – Statistiques des erreurs de synchronisation absolues - GBF

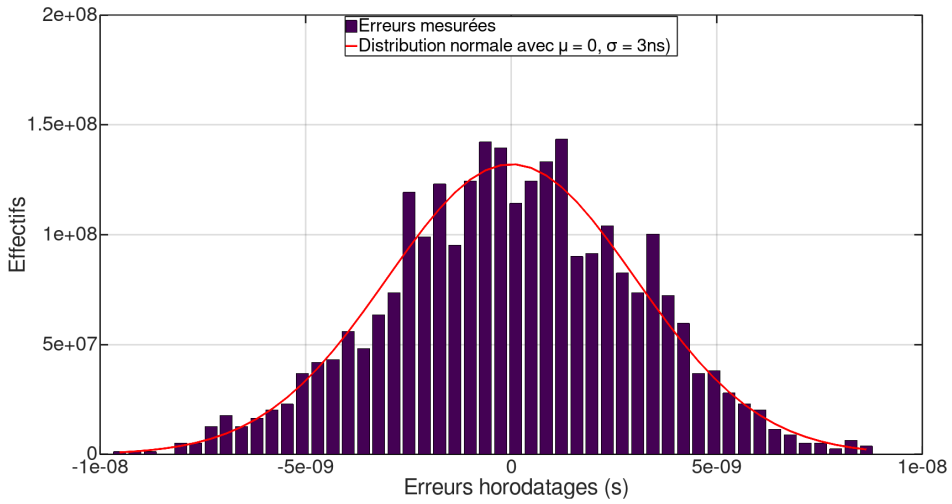


FIGURE 3.6 – Histogramme des erreurs - GBF

Le tableau 3.3 comptabilise le nombre d'erreurs de synchronisation strictement inférieures à un seuil donné. Pendant cette expérimentation, quasiment la moitié des erreurs de synchronisation a été inférieure à 2 ns.

Seuil	Quantité
2 ns	48%
4 ns	81%
6 ns	95%
8 ns	99%
10 ns	100%

TABLE 3.3 – Répartition des erreurs de synchronisation - GBF

D'après les résultats de l'expérimentation, l'implémentation de l'unité d'horodatage est fonctionnelle et permet une EAM de synchronisation de 2.42 ns lorsque les UH sont reliées au même signal physique PPS (ici émulé par un GBF).

3.2.2 Validation des noeuds avec synchronisation GPS

Après avoir validé le fonctionnement des unités d'horodatage, la synchronisation des noeuds de capteurs avec un récepteur GPS peut être évaluée. Afin de valider la capacité d'un noeud de

capteurs à horodater des événements à l'aide de l'unité d'horodatage et du signal PPS d'un récepteur GPS, un deuxième banc de test à été réalisé : deux noeuds de capteurs A et B, chacun constitué d'une unité d'horodatage (UH) implémenté sur un FPGA Spartan 6, d'un récepteur GPS Ublox NEO6T [77] et d'une unité centrale (UC) basés sur un Raspberry Pi 3 sont connectés à un même générateur basse fréquence (GBF) pour la génération d'événements (*Events)*. Chaque récepteur GPS est connecté à une antenne active positionnée sur le toit du batîment (i.e. conditions de réceptions "outdoor" idéales et identiques pour les deux noeuds). Le signal de sortie du GBF est connecté aux entrées événements des noeuds de capteurs A et B comme illustré en figure 3.7. Ainsi les unités d'horodatage des deux noeuds recoivent un signal PPS délivré par leur récepteur GPS respectif et les mêmes événements. Ils doivent donc fournir les mêmes horodatages aux unités centrales plus ou moins les erreurs de quantification de l'UH et les erreurs de synchronisation des récepteurs GPS qui s'appliquent au signal PPS qu'ils délivrent.

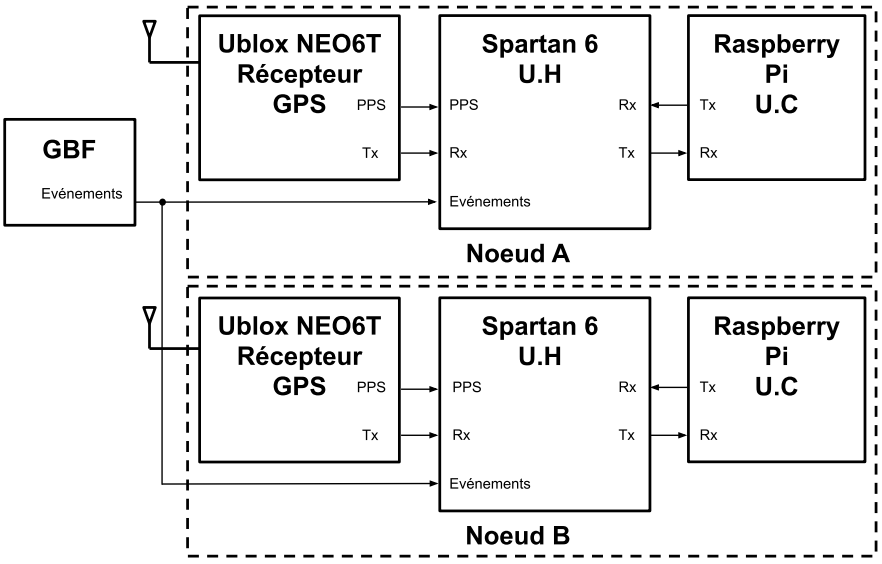


FIGURE 3.7 – Schéma bloc du banc de test avec GPS

Comme pour l'expérimentation précédente et pour les mêmes raisons, la période de génération des événements est fixée à 2.34 secondes. Les deux UH des noeuds de capteurs ont enregistré et retransmis aux UC les variables N_θ, N_Δ et N_f pendant plus d'une semaine (1 mil-

lion de secondes). Les timestamps ont ensuite été calculés hors ligne à l'aide de l'équation 3.2 pour les deux noeuds de capteurs.

La moyenne, la médiane, l'écart type ainsi que les valeurs minimales et maximales des erreurs pour la totalité de l'expérimentation sont données dans le tableau 3.4. Les erreurs sont plus importantes que lors de l'utilisation d'un signal de référence issu d'un GBF du fait de la gigue des signaux PPS. Les erreurs de synchronisation sur les horodatages sont toutes inférieures à 41 ns et l'EAM (absolute mean error) est de 7.72 ns. La moyenne de l'erreur est de -1.82 ns, cette erreur résulte du terme d'erreur constant dans les récepteur GPS (l'erreur GPS est étudiée dans la prochaine section). Cette erreur peut être supprimée en calibrant les récepteurs GPS des noeuds avant de les déployer. Cette calibration consiste à choisir un noeud comme référence, puis de mesurer le terme d'erreur constant de chaque noeud vis à vis de cette référence. Les récepteurs GPS peuvent ensuite être paramétrés pour prendre en compte cette erreur constante. L'histogramme des erreurs est représenté en figure 3.8.

	Minimum	Moyenne	Médianne	Ecart type	Maximum
Erreur	-40.84 ns	-1.82 ns	-1.6 ns	9.44 ns	35.05 ns
Erreur absolue	0 s	7.72 ns	6.61 ns	5.74 ns	40.84 ns

TABLE 3.4 – Statistiques des erreurs de synchronisation - GBF

Le tableau 3.5 comptabilise le nombre d'erreurs de synchronisation strictement inférieures à un seuil donné. D'après les résultats de l'expérimentation, l'architecture de noeud de capteurs basé sur la synchronisation GPS est fonctionnelle et permet une EAM de synchronisation de 7.72 ns. D'après les résultats de cette expérimentation, l'implémentation de l'architecture du noeud de capteurs synchronisé par GPS permet la synchronisation entre deux noeuds avec une EAM de 7.72 ns et une erreur maximum inférieure à 41 ns sur les horodatages. Comme attendu, les erreurs de synchronisation sont statistiquement plus élevées (EAM multiplié par 3.2 et maximum absolu multiplié par 4.2) que lors des tests de validation de l'unité d'horodatage avec GBF du fait de la gigue (*jitter*) sur les signaux PPS. Cette gigue est due aux erreurs de

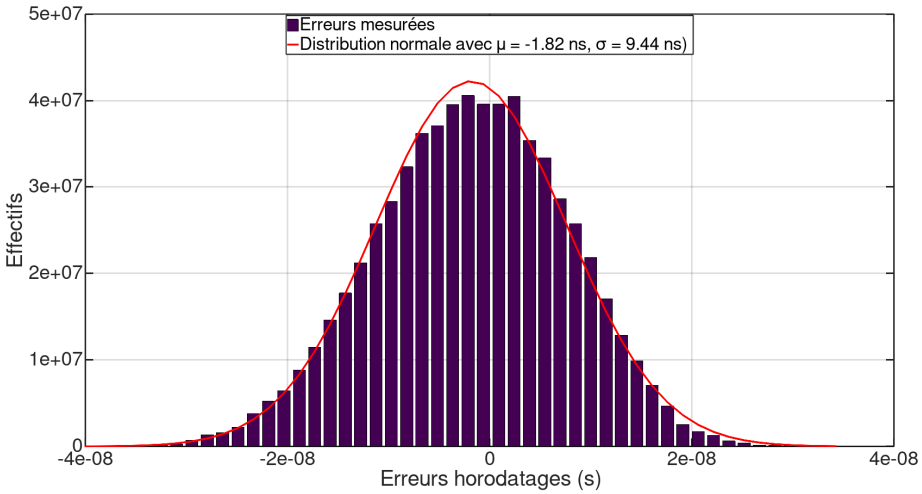


FIGURE 3.8 – Histogramme des erreurs - GPS

Seuil	Quantité
5 ns	38.8%
10 ns	69.9%
20 ns	96.3%
30 ns	99.9%
41 ns	100%

TABLE 3.5 – Répartition des erreurs de synchronisation - GPS

synchronisation des récepteurs GPS, étudiées ci-après.

3.3 Erreurs de synchronisation du récepteur GPS

3.3.1 Bruit GPS

D'après la note d'application du constructeur des récepteurs GPS **applicationnoteUblox**, les erreurs de synchronisation sont décomposables en trois composantes :

- Erreur constante : Cette erreur est dûe au délai causé par la longueur du cable de l'antenne et à un délai dans le récepteur GPS. Elle peut être prise en compte en calibrant le récepteur GPS à l'aide d'une référence plus stable.
- Erreur court terme sur le signal PPS : cette erreur est causée par l'erreur de quantification dans le récepteur GPS. L'oscillateur qui permet de générer le signal PPS est limité à une résolution de 20.83 ns étant donné qu'il oscille à 48MHz. Cette erreur peut être en partie compensée par le récepteur GPS qui transmet l'erreur de quantification du prochain PPS via les trames constructeur sur la liaison série. Cette erreur est appelé qerror [77].
- Erreurs dûes au multi-trajet (*multipath*) et aux temps de propagation des signaux dans la ionosphère : Si le multipath est trop important, l'erreur peut être réduite en configurant le récepteur GPS pour tracker le minimum de satellites et uniquement les satellites avec un angle d'élévation important.

Pour améliorer la synchronisation d'un récepteur GPS lorsque celui-ci est immobile, les récepteurs tel que le Ublox Neo6t proposent un mode *fixed* [77], dans lequel un seul satellite est traqué. Lorsque le récepteur GPS est immobile et que la position de l'antenne GPS n'est pas connue a priori, le mode *survey-in* est utilisé. Ce mode d'observation est utilisé par le récepteur GPS pour apprendre la position de l'antenne. Deux seuil sont alors utilisés pour ce mode : un seuil pour la précision de localisation (exprimé en mm^2, diamètre du cercle de la précision souhaitée autour de l'antenne) et un seuil temporel (en s, durée maximale autorisée pour converger vers une position). Si le récepteur GPS n'a pas convergée vers une position en dessous des seuils de localisation et temps donnés alors le *survey-in* a échoué, sinon la position de l'antenne est déterminée et peut être utilisée pour le mode *fixed*. C'est ce mode de fonctionne-

ment qui a été utilisé pour l'acquisition des données expérimentales. D'après [77] l'erreur de synchronisation est de 30 ns RMS sans compensation des *qerrors* et de 15 ns avec compensation.

Soit $e_{PPS}[n]$ le déphasage du $n^{ième}$ front montant du signal PPS par rapport à la $n^{ième}$ seconde du temps UTC, résultant des erreurs GPS. En prenant en compte cette erreur, la durée entre deux fronts de PPS n'est plus exactement d'une seconde. Elle devient :

$$T_{GPS}[n] = 1 + e_{PPS}[n] - e_{PPS}[n-1] \tag{3.6}$$

L'équation 3.4 permettant de calculer les horodatages devient :

$$t[n] = t_{pps}[n] + e_{PPS}[n] + \frac{1 + e_{PPS}[n] - e_{PPS}[n-1]}{N_f[n]}(N_\theta[n] - N_\Delta[n]) \tag{3.7}$$

Les erreurs e_{PPS} sont observables sur la mesure de fréquence N_f, enregistrée par l'UH de l'oscillateur local du noeud de capteurs. En effet, en remplacant T_{GPS} dans l'équation 3.3 on obtient :

$$N_f[n] = f[n](1 + e_{PPS}[n] - e_{PPS}[n-1]) \tag{3.8}$$

Les fréquences observées sur une partie des jeux de données de l'expérimentation GBF, puis de l'expérimentation GPS sont représentés en figure 3.9. Même si la durée T_{GPS} n'est pas exactement égale à une seconde sur l'expérimentation GBF, l'erreur de déphasage d'une seconde à l'autre est faible en comparaison de l'erreur e_{PPS}. Sur ce graphique le bruit dû aux e_{PPS} apparaît sous formes de sauts d'environ de quelques hertz lors de l'utilisation du signal PPS comme référence pour mesurer les N_f.

Lorsque la réception des signaux GPS est bonne, comme c'est le cas dans notre expérimentation, l'erreur sur le signal PPS proviens principalement du bruit de quantification. Comme énoncé précédemment, la partie de l'erreur du PPS due au bruit de quantification peut être corrigée en tenant compte des *qerrors*. Ces *qerrors* ont été enregistrés dans l'UH pendant toute

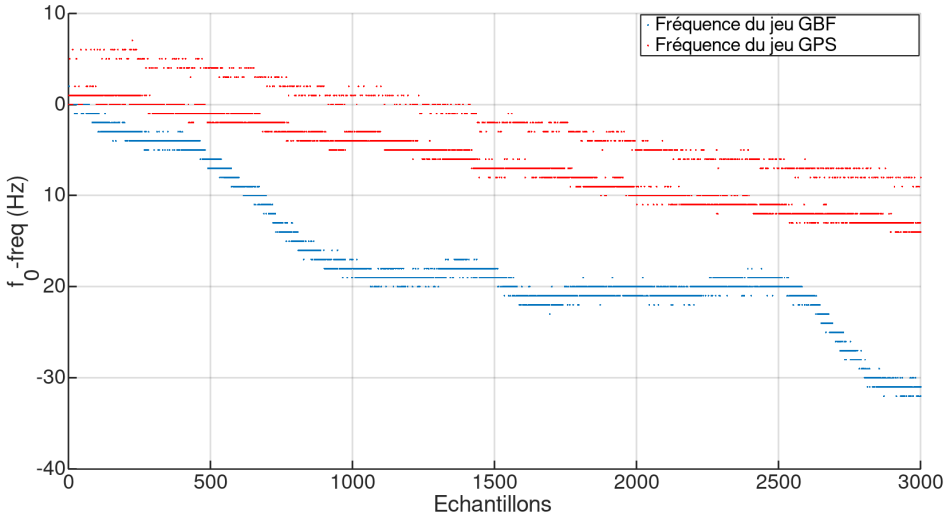


FIGURE 3.9 – Comparaison des fréquences avec synchronisation GBF et GPS

la durée de l'expérimentation GPS, une partie de ces *qerrors* est représentée en figure 3.10. Pour la mesure de fréquence, l'équation 3.3 est utilisée en tenant compte des *qerrors* tel que :

$$f[n] = \frac{N_f[n]}{1 + e_{PPS}[n] - e_{PPS}[n-1]} \tag{3.9}$$

$$\text{avec} \quad e_{PPS}[n] \approx qerror[n] \tag{3.10}$$

La figure 3.11 représente une partie des mesures N_f d'un noeud de capteurs avec et sans les corrections *qerrors*. Les *qerrors* fournies par le récepteur GPS semblent améliorer la mesure de fréquence pour la majorité des échantillons mais augmenter le déphasage sur certains échantillons. Cela se confirme sur les horodatages corrigés avec les *qerrors* suivant l'équation 3.7. Une partie des erreurs de datation entre les deux cartes avec les corrections *qerrors* est tracée en figure 3.12.

La moyenne, la médianne, l'écart type ainsi que les valeurs minimales et maximales des erreurs pour la totalité de l'expérimentation avec les corrections *qerrors* sont donnés dans le tableau 3.6. L'histogramme des erreurs est représenté en figure 3.13. Les erreurs de synchroni-

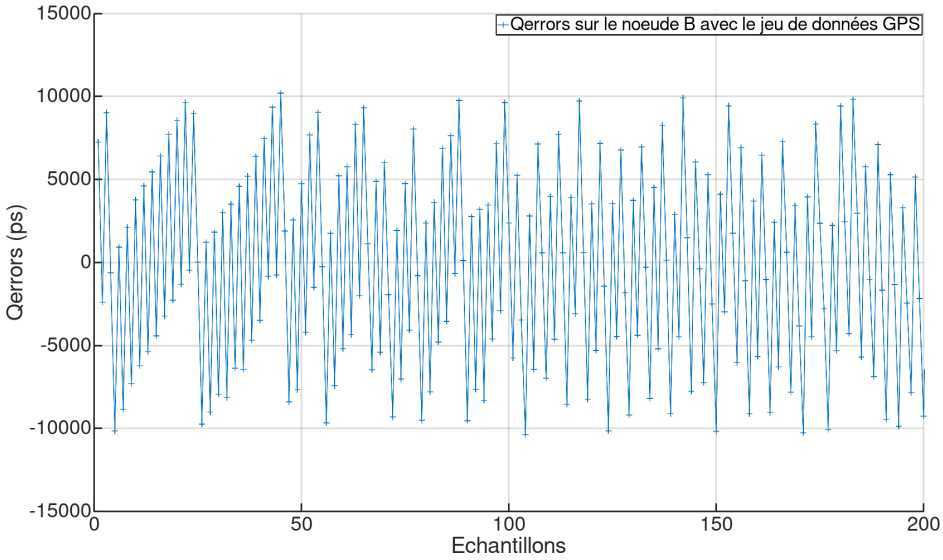


FIGURE 3.10 – *QErrors* d'une partie du jeu de données sur le noeud A

sation sur les horodatages sont toutes inférieures à 49 ns et l'EAM (absolute mean error) est de 5.54 ns. La moyenne des erreurs est similaire à la moyenne des erreurs sans la correction, ce qui est du à l'erreur constante des récepteur GPS et donc corrigable par calibration. L'écart-type des erreurs est diminué de 22.6% avec les *qerrors* mais les erreurs maximales sont plus grandes de 18.5%. La distribution des erreurs n'est plus tout à fait normale lorsque les corrections sont utilisées, les "queues" de la distribution sont alourdies.

	Minimum	Moyenne	Médianne	Ecart type	Maximum
Erreur	-48.15 ns	-1.81 ns	-1.76 ns	7.92 ns	48.43 ns
Erreur absolue	0 s	5.54 ns	3.48 ns	5.94 ns	48.43 ns

TABLE 3.6 – Statistiques des erreurs de synchronisation - GPS et qerrors

Le tableau 3.10 comptabilise le nombre d'erreurs de synchronisation strictement inférieures à un seuil donné. En comparaison des seuils obtenus sans les *qerrors*, les gains sont surtout obtenus en proportion d'erreurs de synchronisation inférieures à 5 et 10 ns. A partir de 20 ns, les proportions d'erreurs sont similaires et même légèrement dégradées pour les plus grandes erreurs. L'utilisation des *qerrors* permet donc de diminuer l'erreur moyenne mais dégrade lé-

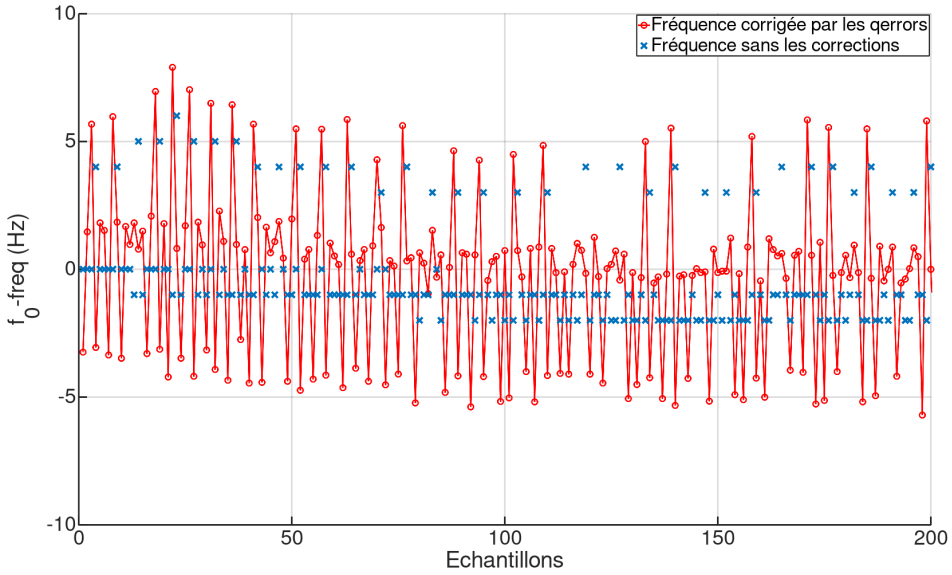


FIGURE 3.11 – Fréquence brute et fréquence corrigée par les qerrors

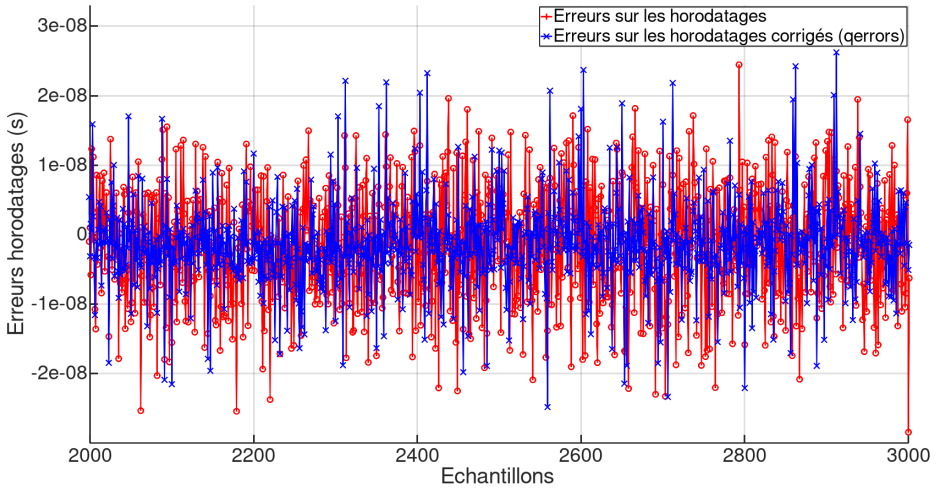


FIGURE 3.12 – Erreurs de synchronisation avec et sans qerrors

gèrement l'erreur maximale.

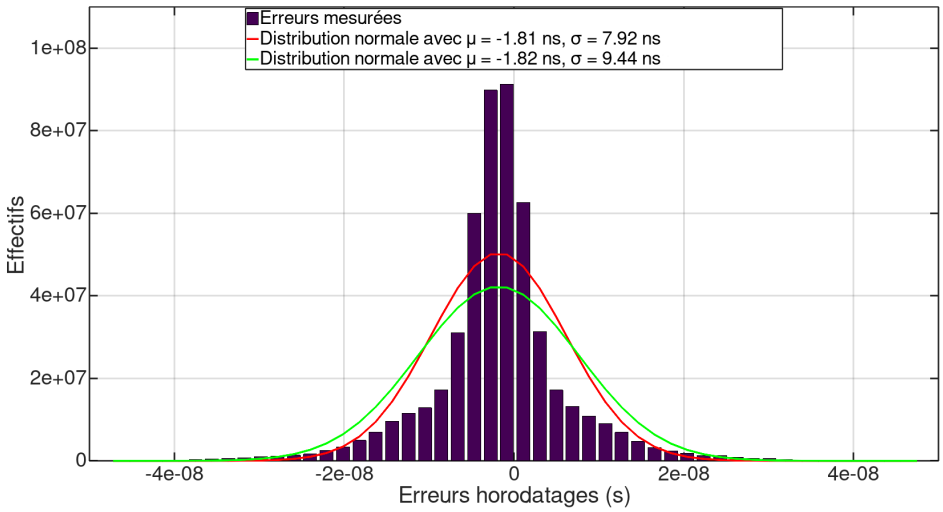


FIGURE 3.13 – Histogramme des erreurs - GPS et *qerrors*

Seuil	Quantité
5 ns	63.4%
10 ns	82.9%
20 ns	96.4%
30 ns	99.3%
49 ns	100%

TABLE 3.7 – Répartition des erreurs de synchronisation - GPS et qerrors

3.3.2 Filtrage du signal PPS

Afin de diminuer le bruit sur le signal PPS, trois filtres numériques sont évalués. Le bruit sur le signal PPS se répercute sur la mesure de la fréquence à partir de N_f et la mesure de déphasage à partir de N_θ. L'évolution théorique de ces paramètres étant relativement lente, le bruit peut être filtré à l'aide de filtres numériques simples. Les deux premiers filtres utilisés sont des filtres numériques par convolution à une seule dimension de la forme :

$$y[n] = \sum_{i=0}^{N-1} h[i]x[n-i] \tag{3.11}$$

Avec x l'entrée du filtre, y la sortie du filtre et h le noyau de convolution de dimension $N \times 1$. Ce filtre introduit un retard de $(N-1)/2$, les N sont donc choisis impairs et la sortie est retardée tel que :

$$y[n] = \sum_{i=-P}^{P} h[i+P]x[n+i] \quad \text{avec} \quad P = \frac{1}{2}(N-1) \tag{3.12}$$

Ce filtre n'est pas causal et ne peut donc pas être utilisé en temps réel, cependant il peut servir à lisser les données en hors-ligne pour éliminer les bruits causés par la gigue sur le signal PPS ou directement sur un noeud de capteurs si un délai de P secondes sur l'acquisition des données est tolérable par l'application visée. Le premier filtre testé est un simple filtre moyenneur, les coefficients du noyau de convolution sont tous égaux à $1/N$. Le deuxième filtre est un filtre gaussien, le noyau de convolution est défini par :

$$h[i] = f(i-P) \quad \text{avec} \quad f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{x^2}{2\sigma^2}) \tag{3.13}$$

Le noyau est ensuite normalisé pour que la somme de ses coefficients soit égale à 1. La taille du noyau N est choisie de sorte à ce que $P = \lceil 3\sigma \rceil$. Le troisième filtre testé est un filtre médian tel que $y[n]$ prend la valeur de la médiane des $x[i]$ pour i allant de $n-P$ à $n+P$. Comme les deux premiers filtres, ce filtre ne peut pas être utilisé en temps réel. Il peut être utilisé en hors-ligne ou bien avec un délai P dans l'acquisition des données.

3.3.3 Evaluation des filtres

Les différents filtres ont été appliqués sur les N_f et N_θ des données de l'expérimentation GPS avec et sans la prise en compte des corrections *qerrors*. Les horodatages des événements sont ensuite calculés d'après l'équation 3.4. Les EAM des horodatages sont illustrés en figure 3.14 en fonction de la taille du noyau du filtre pour les trois filtres avec et sans les *qerrors*. Les résultats du filtre médian tendent vers une constante quand la taille du noyau augmente. Les résultats des filtres moyenneur et gaussien dépendent de la taille du noyau. On observe une

taille optimale pour laquelle les EAM sont minimales pour les filtres moyenneur et gaussien. Ces tailles sont données pour les deux filtres avec et sans les *qerrors* dans le tableau 3.8.

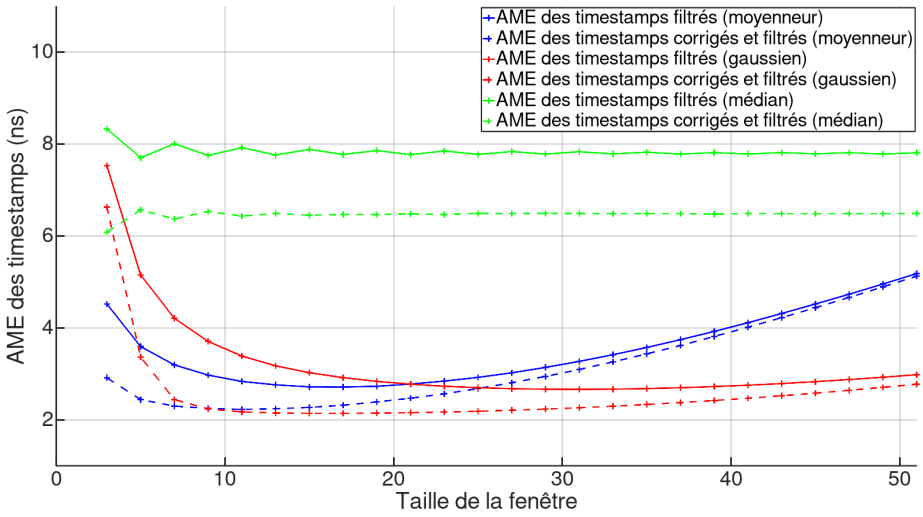


FIGURE 3.14 – EAM des horodatages en fonction de la taille N des filtres

N optimal	filtre moyenneur	filtre gaussien
Sans qerrors	17	31
Avec qerrors	11	17

TABLE 3.8 – Taille N optimale

Les statistiques des erreurs absolues pour la totalité de l'expérimentation et pour tous les filtres, sont donnés dans le tableau 3.9. Le filtre médian avec et sans les *qerrors* n'améliore pas les erreurs sur les horodatages. Les filtres moyenneur et gaussien diminuent les erreurs sur les horodatages avec et sans les *qerrors*. Les résultats des filtres gaussien et moyenneur sont a peu près équivalent avec un léger avantage pour le filtre gaussien mais pour une taille de noyau presque deux fois plus grande et donc un retard sur les données plus important dans le cas d'une application "temps réel". Les meilleurs résultats sont obtenus en utilisant les corrections *qerrors*.

Les distributions des erreurs pour les filtres gaussiens et moyenneurs avec et sans les *qerrors* sont représentées en figure 3.15. L'utilisation du filtre permet de supprimer l'alourdissement

	Moyenne	Ecart type	Maximum
Non filtré	7.72 ns	5.74 ns	40.84 ns
Filtre moyenneur	2.76 ns	2.14 ns	17.26 ns
Filtre gaussien	2.71 ns	2.10 ns	17.37 ns
Filtre médian	7.81 ns	5.57 ns	32.97 ns
Non filtré avec qerrors	5.54 ns	5.94 ns	48.43 ns
Filtre moyenneur avec qerrors	2.30 ns	1.65 ns	10.81 ns
Filtre gaussien avec qerrors	2.22 ns	1.56 ns	9.46 ns
Filtre médian avec qerrors	6.49 ns	7.50 ns	44.38 ns

TABLE 3.9 – Statistiques des erreurs de synchronisation absolues - GPS et filtrage

des queues de distributions observé lors de l'utilisation des corrections avec les *qerrors*. L'écart type n'est donc pas fortement impacté par l'utilisation des *qerrors*, contrairement aux valeurs maximales et minimales. L'utilisation du filtre moyenneur et des *qerrors* permet de diviser l'erreur absolue maximale par 4.48 par rapport à l'utilisation des *qerrors* sans filtre.

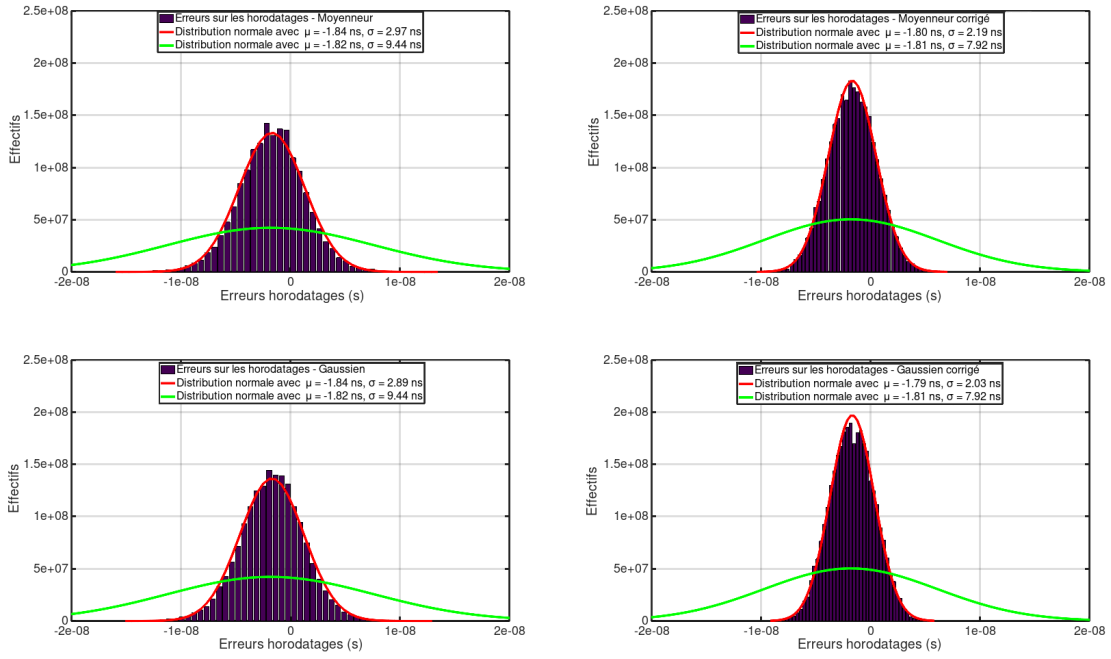


FIGURE 3.15 – Histogramme des erreurs - GPS et Filtrage

Une partie des différences d'horodatages entre les deux noeuds de capteurs est représentée

figure 3.16 avec les données brutes et les données corrigés mais non filtrées et les données non corrigées mais filtrées par le filtre moyenneur. Les erreurs sont bien plus centrés sur l'erreur moyenne de -1.8 ns avec le filtre et les grandes erreurs engendrées par les corrections *qerrors* ne sont plus présentes. La différence entre le filtre moyenneur avec correction et sans correction sur une partie des erreurs d'horodatages est illustré en figure 3.17. L'utilisation des *qerrors* permet de ramener les erreurs maximales et minimales sous la barre des 10 ns.

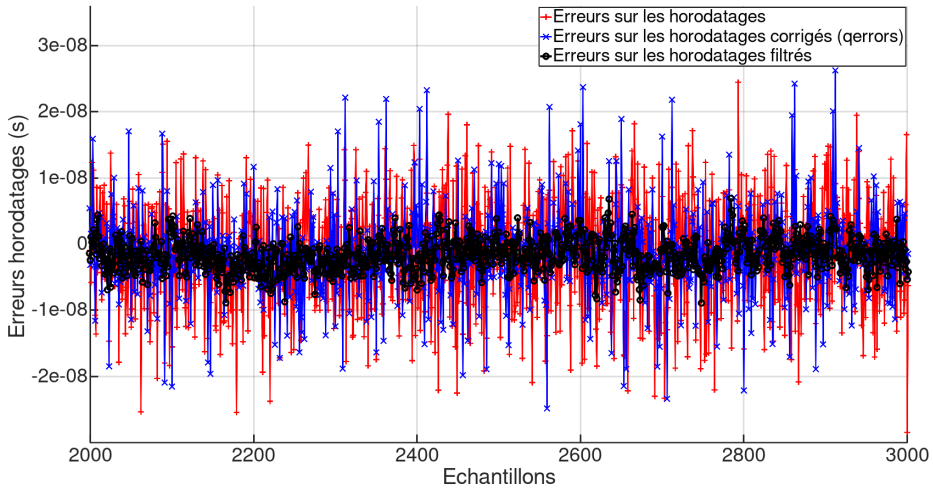


FIGURE 3.16 – Erreurs de synchronisation avec et sans qerrors/filtrage

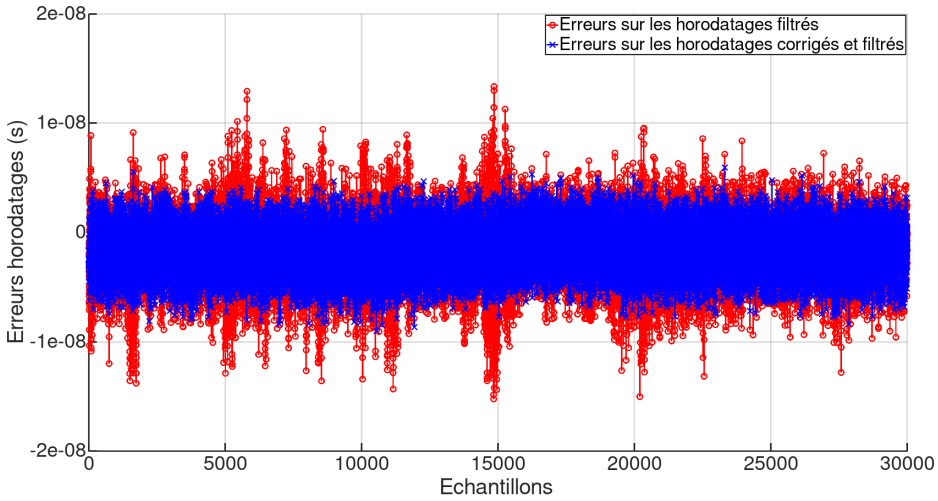


FIGURE 3.17 – Influence des qerrors sur les erreurs de synchronisation avec le filtrage

Le tableau 3.10 comptabilise le nombre d'erreurs de synchronisation strictement inférieures à un seuil donné. D'après les résultats de l'expérimentation, l'utilisation d'un filtre moyenneur avec les *qerrors* permet de diviser l'EAM par 2.4 et l'erreur maximale par 4.48 avec un délai d'acquisition de 5 secondes pour le traitement des horodatages. Avec ce filtre, il est alors possible d'obtenir une erreur d'horodatage maximale entre deux noeuds de capteurs d'environ 10 ns avec 92.6% des erreurs inférieures à 5 ns. Ce niveau de synchronisation est très proche du niveau de synchronisation obtenus en connectant deux noeuds au même signal de synchronisation (voir section 3.2.1) ce qui prouve l'efficacité du filtre moyenneur pour l'élimination des erreurs de synchronisation dûes au récepteur GPS. Il reste l'erreur constante, ici mesurée à 1.8 ns, qui peut être mesurée avant un déploiement des capteurs sur le terrain pour être enlevée directement dans le calcul des horodatages.

Seuil	Quantité						
	Filtre	Sans		Moyenneur		Gaussien	
	Qerrors	Sans	Avec	Sans	Avec	Sans	Avec
5 ns		38.8%	63.4%	85.4%	92.6%	86.1%	94.2%
10 ns		69.9%	82.9%	99.3%	100%	99.4%	100%
20 ns		96.3%	96.4%	100%	100%	100%	100%
30 ns		99.9%	99.3%	100%	100%	100%	100%
40 ns		100%	99.9%	100%	100%	100%	100%
50 ns		100%	100%	100%	100%	100%	100%

TABLE 3.10 – Répartition des erreurs de synchronisation - GPS et filtrage

3.4 Conclusion

L'objectif des travaux présenté dans ce chapitre était double, premièrement valider le bon fonctionnement de l'architecture de noeud de capteurs synchronisé par GPS et deuxièmement, acquérir un jeu de données important et exploitable en post-traitement pour pouvoir évaluer des scénarios de contrôle du récepteur GPS. Nous avons tout d'abord prouvé que la synchronisation par GPS avec l'utilisation d'une unité d'horodatage hardware cadencé à 240 MHz permet

d'obtenir une erreur de datation d'événements inférieure à 50 ns et de 5.54 ns en moyenne ($\approx$ 1.2 ticks). Ensuite, il a été démontré expérimentalement que l'ajout d'un filtre moyenneur pour diminuer les erreurs de synchronisation du récepteur GPS permet de passer cette erreur sous les 10 ns avec une moyenne de 2.3 ns ($\approx$ 0.55 ticks). Ce filtre peut facilement être implémenté sur des microcontrôleur dipsosant d'une FPU (Foating Point Unit) du fait de sa simplicité : N multiplications et $N - 1$ additions (N étant la taille du filtre). Les résultats présentés dans ce chapitre ayant été obtenus en précision double, une implémentation sur un microcontrôleur pourrait possiblement diminuer les gains du filtre, du fait du passage en simple précision. Le jeu de données enregistré lors de l'expérimentation avec GPS permet de recaler les horodatages à postériori comme décrit dans la section 3.1.2. Ce jeu de données nous permettra dans le chapitre suivant d'évaluer des scénarios de ON/OFF GPS en post-traitement en fonction de différents paramètres tels que la durée d'apnée, ou le modèle de l'horloge. La température ambiante étant le paramètre d'influence le plus important sur les oscillateurs SPXO, elle a aussi été enregistrée durant les expérimentations pour pouvoir être exploitée dans les scénarios ON/OFF GPS.

CHAPITRE 4

Évaluation en post-traitement de la stratégie de synchronisation cyclique par GPS

Le chapitre précédent a spécifié le fonctionnement des noeuds de capteurs ainsi que le protocole expérimental ayant permis l'acquisition d'un jeu de données. Dans ce chapitre, l'objectif est d'évaluer le principe de synchronisation ON/OFF introduit à la fin du chapitre 2. Ce mode ON/OFF alterne les périodes d'apnées pendant lesquelles le récepteur GPS est éteint et les périodes ou il est allumé pour resynchroniser le noeud. Dans un premier temps le principe de synchronisation avec ON/OFF GPS est détaillé. Ensuite, le jeu de données obtenu précédemment, est utilisé pour modéliser le comportement de l'horloge du noeud de capteur. Deux modèles sont évalués pour la prédiction des paramètres de l'horloge pendant la période d'apnée. Ces modèles seront utilisés pour la synchronisation "Temps-réel" rejouée en post-traitement. Finalement, la synchronisation a posteriori est évaluée. La synchronisation a posteriori est elle aussi évaluée en post-traitement mais contrairement à la synchronisation "Temps-réel", les noeuds ne calculent pas les horodatages toutes les secondes, ils sont calculés à la fin de la période d'apnée.

Sommaire

4.1 Synchronisation périodique par GPS

Comme démontré précédemment (chapitre 3), la synchronisation des noeuds de capteur par GPS permet d'atteindre une précision de synchronisation inférieure à 10 ns. Cependant le coût énergétique d'un récepteur GPS allumé tout le temps est considérable dans un contexte *embarqué* (typiquement 120mW continu [77]), auquel il faut ajouter le coût énergétique de l'antenne si celle-ci est active (typiquement de 3 à 20 mA [98]). Lorsque la précision de synchronisation attendue est moindre, la synchronisation par le récepteur GPS peut être utilisée de manière plus espacée : le récepteur peut être éteint entre deux périodes de resynchronisation. Cependant, le délai entre le démarrage d'un récepteur GPS et la génération d'un signal PPS synchrone au temps GPS n'est pas nul. Ce délai est constitué du temps nécessaire au récepteur pour l'obtention de la localisation et du temps, appelé Time To First Fix (TTFF), et du délai pour générer un signal physique synchrone au temps GPS. Sous condition de bonne réception des signaux GPS, le TTFF dépend de l'état du récepteur GPS au moment du rallumage, c'est à dire

la présence ou non d'un almanach et d'éphémérides valides en mémoire ainsi qu'une information, même approximative, de sa position et de son déphasage par rapport aux satellites. La plupart des récepteurs GPS disposent d'un mode de mise en sommeil permettant de maintenir en fonctionnement seulement la mémoire (dans le cas d'une mémoire volatile) et une RTC. Le dernier almanach, les dernières éphémérides et la dernière position sont stockés dans cette mémoire avant l'extinction du récepteur pour pouvoir être utilisés lors du redémarrage. Typiquement on distingue trois types de démarrages :

- Démarrage à froid : Le récepteur GPS ne dispose d'aucune information. En théorie il doit attendre d'avoir un almanach complet pour pouvoir tracker au moins quatre satellites visibles et obtenir un "fix". En pratique la plupart des récepteur GPS disposent aujourd'hui d'au moins autant de canaux que de satellites disponibles dans la constellation, ils peuvent donc tenter de tracker tous les satellites avant d'avoir un almanach complet. Sur ces récepteurs, le TTFF est le même pour le démarrage à chaud que pour le démarrage tiède mais le courant consommé peut être plus important du fait de l'utilisation d'un plus grand nombre de canaux. D'après [77] le TTFF moyen d'un récepteur ublox neo6t est de 27s dans ce mode.
- Démarrage à tiède : Le récepteur GPS dispose d'un almanach valide, d'une position approximative (souvent la dernière position connue avant extinction) et de son temps approximatif (à l'aide de sa propre RTC) mais pas des éphémérides. Dans ce mode le récepteur GPS peut directement tracker 4 satellites visibles pour obtenir leur éphémérides et calculer un "fix". Dans ce mode, et sous condition de bonne réception GPS, le temps de verrouillage d'un canal sur le signal d'un satellite est négligeable devant la durée du message de navigation. Le TTFF maximal correspond donc à peu près au temps maximum pour l'obtention des *subframes* 1 à 3 du message de navigation contenant les éphémérides [26], c'est à dire 36s. D'après [77] le TTFF moyen d'un récepteur ublox neo6t est aussi de 27s dans ce mode, ce qui a du sens du fait des 32 canaux disponibles sur le récepteur.
- Démarrage à chaud : Le récepteur GPS dispose d'un almanach valide, d'éphémérides va-

lides, d'une position approximative et de son temps approximatif. C'est le mode qui offre le meilleur TTFF car la position et le temps peuvent êtres calculés dès que les mesures de pseudo distances de quatre satellites sont disponibles. D'après [77] le TTFF moyen d'un récepteur ublox neo6t est de 1s dans ce mode.

Pour qu'un récepteur GPS puisse continuer à calculer sa position et son déphasage par rapport au temps GPS, il lui faut au moins une partie des informations du message de navigation. Ce message est constitué de 25 frames de 30 secondes, chacune divisé en 5 subframe de 6 secondes. La structure du message de navigation est représentée en figure 4.1.Les subframes 1 à 3 sont répétées au début de chaque frame, le contenu des frames 4 et 5 change entre chaque frame (aussi appelées pages des subframes). La subframe 1 du message de navigation contient les informations sur l'état du satellite ainsi que les paramètres de correction de son horloge. Ses paramètres sont nécessaires pour obtenir le temps GPS sur le récepteur. Les subframes 2 et 3 contiennent les éphémérides, qui sont nécessaires à la résolution du système d'équations (cf chapitre 2). Ces paramètres sont calculés pour des intervalles de quatre heures et sont actualisés toutes les 2 heures dans le message de navigation diffusé par le satellite [26]. Il faut donc récupérer les subframes 1 à 3 du message de navigation au minimum toutes les deux heures pour que le récepteur GPS puisse fonctionner normalement. De plus, les récepteurs GPS simple bande L1 C/A tel que celui utilisé ici peuvent éliminer une partie des erreurs de synchronisation/localisation dues à la ionosphère. Contrairement au récepteurs bi-bandes qui peuvent utiliser les différences de fréquence porteuses pour modéliser ces perturbations, les récepteurs mono-bande utilisent un modèle basé sur la météorologie spatiale. Pour le GPS, c'est le modèle de Klobuchar [99] qui est utilisé. Les coefficients du modèle sont diffusés dans la subframe 4 du message de navigation. Cette subframe contient aussi les paramètres permettant de convertir le temps GPS en temps UTC. Les paramètres UTC sont valides pour 70 heures après leur réception [26], les coefficients de Klobuchar sont rafraîchis une fois par jour. Afin de profiter des corrections ionosphériques (50% d'amélioration RMS sur l'erreur due à la ionosphère [99]) et d'une synchronisation au temps UTC, il est nécessaire d'obtenir les pages 13 et 18 de la sub-

frame 4 [26] au moins une fois par jour. Par souci de simplification on estimera par la suite que le message de navigation doit être réceptionné totalement une fois par jour. La subframe 5 contient une partie de l'almanach et des informations sur la constellation. Ces informations ne sont pas essentielles si le récepteur possède déjà les éphémérides des satellites en vue.

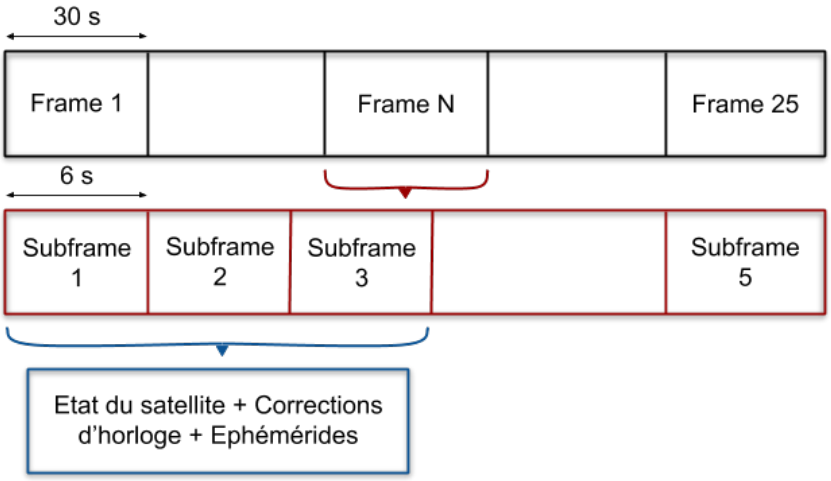


FIGURE 4.1 – Découpage du message de navigation GPS

En dehors des périodes d'acquisition des éphémérides et du message de navigation, le récepteur GPS peut donc être alternativement allumé et éteint pour économiser de l'énergie. On définit k_{on} comme la durée pendant laquelle le GPS est allumé par cycle et k la durée totale du cycle ON/OFF. La durée de la période éteinte $(k - k_{on})$, va grandement déterminer la dérive de l'horloge et donc la précision de la synchronisation ; d'autant plus que les facteurs environnementaux tels que la température, évoluent pendant cette période. La figure 4.2 représente les différentes phases du récepteur GPS sur une période de 24 heures (86 400 secondes) pour l'écoute d'un message de navigation complet par jour et un rafraîchissement des éphémérides toutes les deux heures afin de rester en mode rallumage "à chaud". À partir de ces informations, l'équation permettant de calculer le ratio temps allumé par jour du récepteur GPS a été définie :

$$r = \frac{1}{86400}(T_{NAV} + 11.T_{EPH} + \frac{k_{on}}{k}(86400 - T_{NAV} - 11.T_{EPH})) \tag{4.1}$$

En prenant un temps d'écoute des éphémérides T_{EPH} de 36 secondes, un temps d'écoute du message de navigation T_{NAV} de 756 secondes (le message de navigation dure 12.5 minutes) et un temps additionnel de 2 secondes à chaque rallumage du récepteur pour obtenir un "fix" et délivrer un signal PPS synchrone, le ratio minimum est de 1.33%. Pour un récepteur GPS nécessitant 120 mW ce ratio permet d'arriver à une puissance de 1.6 mW. Si les corrections ionosphériques ne sont pas utilisées et le temps UTC n'est pas requis, il est même théoriquement possible d'atteindre les 0.6 mW car seules les éphémérides sont utiles et donc $T_{NAV} = 0$.

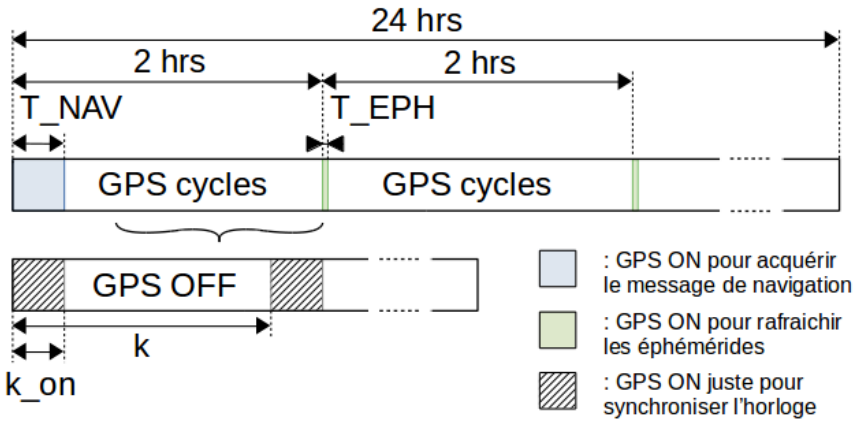


FIGURE 4.2 – Protocole de synchronisation cyclique par GPS

4.2 Modélisation

4.2.1 Modèle d'horloge

Le déphasage de l'horloge par rapport au temps de référence a besoin d'être estimé pendant la période d'apnée pour pouvoir continuer d'horodater des événements le plus précisément possible. Pour ce faire, il est possible de modéliser le comportement de l'oscillateur. Soit $C(t)$

la date locale d'un noeud en fonction du temps t tel que :

$$C(t) = \int_0^t \omega(\tau)d\tau + \theta_0 \tag{4.2}$$

Avec ω le *clock rate* et θ_0 le déphasage à l'origine (c'est à dire $t = 0$). Avec une horloge dont l'oscillateur oscille exactement à sa fréquence théorique, on obtient $C(t) = t + \theta_0$; si cette même horloge est parfaitement synchronisée par rapport au temps de référence à $t = 0$, on obtient une horloge parfaite : $C(t) = t$. Dans la pratique, le modèle $C(t) = \omega.t + \theta$ est souvent utilisé [1][3]. Pour ce modèle, le *clock rate* est constant mais diffère de 1 ce qui signifie que la fréquence réelle de l'oscillateur est différente de sa fréquence théorique mais qu'elle est considérée constante sur la période d'apnée. Bien que l'hypothèse d'une fréquence d'oscillation constante soit fausse pour tous les oscillateurs, elle peut être justifiée en fonction de la précision de synchronisation attendue et de la fréquence d'actualisation des paramètres ω et θ. Le déphasage $\theta(t)$ à un temps t d'une horloge correspond à la différence entre la date locale $C(t)$ et le "vrai" temps t. Il est exprimé de la façon suivante :

$$\theta(t) = t - C(t) \tag{4.3}$$

$$= t - \int_0^t \omega(\tau)d\tau + \theta_0 \tag{4.4}$$

Étant donné que le déphasage est mesuré une fois par seconde à l'aide du GPS, un modèle discret est nécessaire. L'équation 4.4 devient :

$$\theta[n] = \sum_{i=0}^{n-1} \tau[i] - \sum_{i=0}^{n-1} \omega[i].\tau[i] + \theta_0 \tag{4.5}$$

Avec τ une durée correspondant à une seconde du point de vue de l'horloge locale. Cette durée dépend de la fréquence de l'oscillateur tel que :

$$\tau[n] = \frac{N_0}{f[n]} = \frac{N_0}{N_f[n]}.T_{GPS}[n] \tag{4.6}$$

Le clock rate peut être calculé à partir du nombre de ticks de l'horloge locale pendant la durée T_{GPS} :

$$\omega[n] = \frac{N_f[n]}{N_0} \tag{4.7}$$

Ainsi l'équation 4.5 devient :

$$\theta[n] = \sum_{i=0}^{n-1} \tau[i] - \sum_{i=0}^{n-1} T_{GPS}[i] + \theta_0 \tag{4.8}$$

Si l'on considère que le signal est parfaitement synchronisé à UTC, la durée T_{GPS} dure exactement une seconde et donc l'équation 4.8 devient :

$$\theta[n] = \sum_{i=0}^{n-1} \frac{N_0}{N_f[i]} - n + \theta_0 \tag{4.9}$$

Ce modèle d'horloge discret peut être décrit en représentation d'état. Soit $\mathbf{x}$ le vecteur d'état comprenant les deux variables d'états τ et θ. L'équation d'état décrivant l'évolution du modèle est la suivante :

$$\mathbf{x}[n+1] = F\mathbf{x}[n] + \mathbf{u} \tag{4.10}$$

$$\text{Avec} \quad \mathbf{x}[n] = \begin{bmatrix} \tau[n] \\ \theta[n] \end{bmatrix}, F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad \text{et} \quad \mathbf{u} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \tag{4.11}$$

4.2.2 Prédiction de l'état pendant la période d'apnée

Durant l'apnée, τ n'est pas mesurable car le récepteur GPS est éteint. Comme l'évolution de τ n'est pas observable et que l'on sait que la fréquence de l'horloge dérive au cours du temps (cf chapitre 2), il est possible de modéliser l'évolution de ce paramètres τ comme une marche

aléatoire tel que :

$$\tau[n+1] = \tau[n] + w[k] \quad \text{avec} \quad w \sim \mathcal{N}(0, \sigma_w^2) \tag{4.12}$$

Ce modèle nous permet d'estimer la matrice de variance-covariance P de l'estimation pendant la période d'apnée. D'après les équations 4.10 et 4.12, on obtient :

$$P[n+1] = FP[n]F^T + Q \quad \text{avec} \quad Q = \begin{bmatrix} \sigma_w^2 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.13}$$

La méthode d'estimation de τ la plus simple pendant l'apnée, consiste à utiliser la dernière valeur observée quand le GPS était allumé. Soit n le dernier instant avec le récepteur GPS allumé, k secondes plus tard les paramètres prédits par le modèle définit en équation 4.10 sont les suivants :

$$\tau[n+k] = \tau[n] \tag{4.14}$$

$$\theta[n+k] = \theta[n] + k(\tau[n] - 1) \tag{4.15}$$

Dans la suite de ce document, ce modèle sera désigné par l'acronyme MHFC (pour Modèle d'Horloge à Fréquence Constante). En utilisant l'équation 4.13, les variances sur les paramètres τ et θ, prédites par le modèle, k secondes après le dernier allumage GPS sont définies par :

$$p_\tau[n+k] = p_\tau[n] + (k+1)\sigma_w^2 \tag{4.16}$$

$$p_\theta[n+k] = p_\tau[n]k^2 + p_\theta[n] + \frac{\sigma_w^2}{6}k(k+1)(2k+1) \tag{4.17}$$

Étant donné que l'erreur commise sur le déphasage à un instant t correspond à la somme des erreurs commises sur τ jusqu'à t, cette erreur est, a priori, d'autant plus croissante que l'on s'éloigne de la vraie valeur de τ. La précision de synchronisation dépend donc de la durée de

l'apnée, de la dérive de τ pendant l'apnée et de la précision sur l'observation des paramètres à la dernière synchronisation.

Afin de tester la précision des prédictions, la procédure suivante a été utilisée à partir du jeu de données. Le modèle décrit par les équations 4.14 et 4.15 a été utilisé pour prédire le déphasage sur des cycles de k secondes. Tous les k secondes, le récepteur GPS est considéré "allumé" et le vecteur d'état x est défini à l'aide des observations. La matrice de variance-covariance de l'état P est alors remise à 0 (matrice nulle). Ensuite le noeud est en apnée pendant les $k-1$ secondes suivantes. Le vecteur d'état et la matrice de variance-covariance sont estimés sans les observations GPS. Ici les mesures servent de référence, les erreurs de prédictions sur le déphasage sont donc calculées en faisant la différence entre le vecteur d'état et les mesures sur tout le jeu de données. Les erreurs de prédiction de déphasage pour un cycle de 600 secondes sont illustrées en figure 4.3. Comme prévu, l'erreur croit pendant les périodes d'apnée car la prédiction diverge des mesures en accumulant les erreurs de fréquence, et à chaque "rallumage" GPS (tous les k secondes) l'erreur est nulle (car les mesures servent de référence). En figure 4.4, la fréquence observée et la fréquence prédite par le modèle sont illustrées pour le même cycle de 600 secondes.

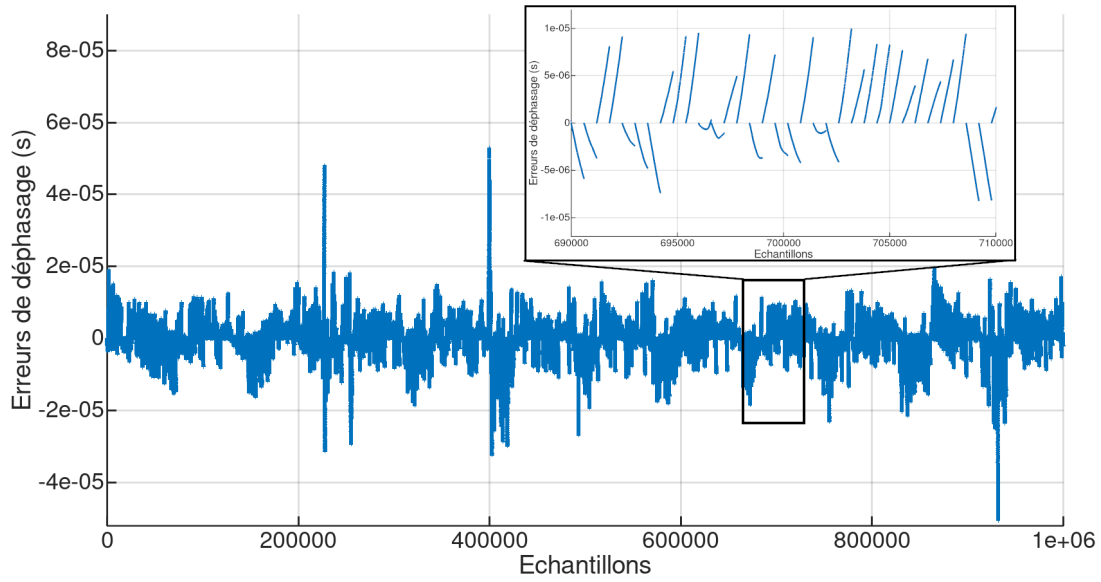


FIGURE 4.3 – Erreurs de prédiction du déphasages pour k=600 s

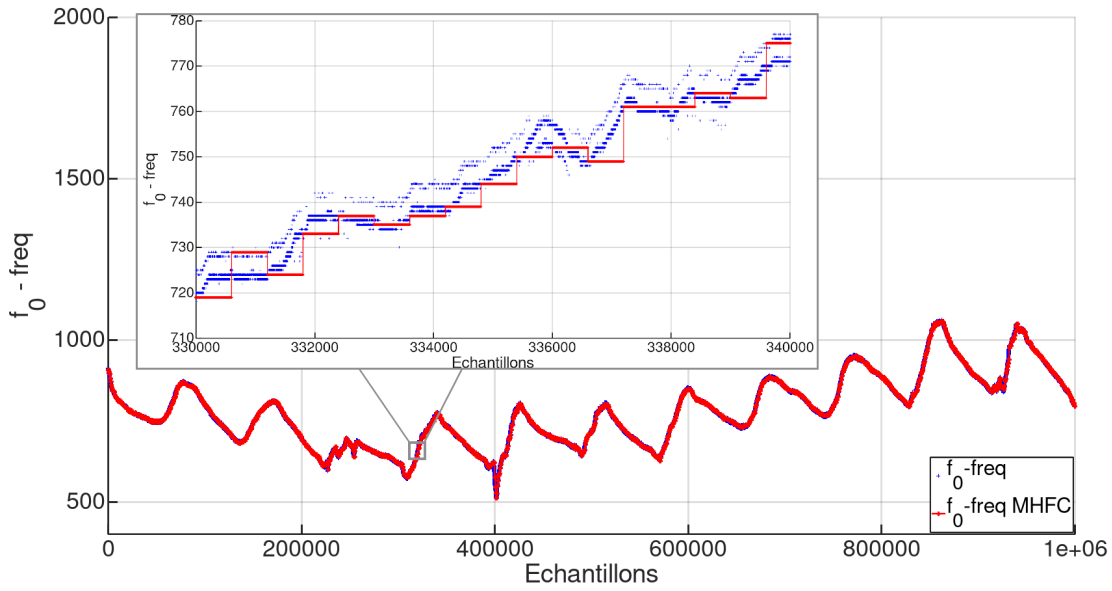


FIGURE 4.4 – Fréquence prédite par le modèle pour k=600 s

La moyenne, l'écart type et le maximum absolu des erreurs de prédictions juste avant le rallumage sur τ et θ et pour différentes durées de cycle sont répertoriés dans le tableau 4.1. Pour toutes ces durées de cycle le temps GPS ON est toujours de 1 seconde et le temps d'apnée de $k-1$ secondes. Afin de valider l'estimation des écart-types des paramètres prédits par le

modèle, les paramètres σ_w ont été calculés pour les différentes périodes d'apnées à l'aide de l'équation 4.16 puis injectés dans l'équation 4.17 pour calculer l'écart-type $\sqrt{\rho_\theta}$, prédit sur le déphasage à la fin de chaque cycle. Pour ces deux écarts-types, les conditions initiales ont été choisies nulles car les erreurs sont calculé par rapport aux observations. Ces écarts-types sont répertoriées dans le tableau 4.2. La modélisation de τ par une marche aléatoire dont l'écart-type du terme d'erreur dépend de la durée de l'apnée semble être adéquate car les écarts-types de déphasage prédits sont assez proche des écarts-types mesurés, présentés dans le tableau 4.1.

k	Erreurs sur τ (s)			Erreurs sur θ (s)		
	Moyenne	Ecart-type	Max abs	Moyenne	Ecart-type	Max abs
10	-4.1209E-11	1.2357E-08	4.1667E-08	-2.0417E-10	8.8391E-08	2.6087E-07
25	5.6667E-11	1.2488E-08	4.5834E-08	1.2064E-09	2.2069E-07	7.2758E-07
50	1.5833E-11	1.2589E-08	4.5834E-08	1.3746E-09	4.4495E-07	1.4108E-06
100	1.5917E-10	1.3074E-08	7.9167E-08	1.0506E-08	9.0713E-07	3.5338E-06
250	2.2500E-10	1.5119E-08	9.1667E-08	4.2914E-08	2.4774E-06	1.0121E-05
500	2.8334E-10	1.9236E-08	1.5833E-07	7.7867E-08	5.8549E-06	3.5907E-05
1000	2.0833E-10	2.9846E-08	3.0000E-07	2.7216E-07	1.6745E-05	1.4990E-04
2000	8.0001E-10	5.0075E-08	2.9167E-07	7.3627E-07	5.2113E-05	2.7207E-04
3000	-3.2533E-10	7.1861E-08	3.9167E-07	-8.0278E-07	1.1866E-04	9.3400E-04

TABLE 4.1 – Erreurs sur τ et θ

k	σ_w (s)	$\sqrt{p_\theta}$ (s)
10	3.9076E-9	6.5968E-8
25	2.4976E-9	1.7483E-07
50	1.7804E-9	3.5796E-07
100	1.3074E-9	7.4916E-07
250	9.5621E-10	2.1757E-06
500	8.6026E-10	5.5446E-06
1000	9.4381E-10	1.7219E-05
2000	1.1197E-9	5.7800E-05
3000	1.3120E-9	1.2444E-04

TABLE 4.2 – Prédiction de la variance

Il est aussi possible de considérer que τ évolue, principalement à cause des variations de

la température ambiante, et que son évolution peut être estimée à l'aide d'une approximation linéaire par morceaux. Ainsi pour chaque intervalle de k secondes, τ évolue linéairement tel que :

$$\tau[n+1] = \tau[n] + u \tag{4.18}$$

La raison de cette suite peut être mesurée pour chaque période d'apnée :

$$u = \frac{\tau[n] - \tau[n-k]}{k} \tag{4.19}$$

Sous l'hypothèse que u évolue lentement, le dernier u observé peut être utilisé pour prédire τ durant la prochaine apnée. L'estimation des paramètres est alors régie par les équations suivantes :

$$\tau[n+k] = \tau[n] + ku \tag{4.20}$$

$$\theta[n+k] = \theta[n] + \sum_{i=1}^{k} \tau[i] - k \tag{4.21}$$

Dans la suite de ce document, ce modèle sera désigné par l'acronyme MHFD pour Modèle d'Horloge à Fréquence Dynamique. Les erreurs de prédiction de déphasage pour un cycle de 600 secondes sont illustrées en figure 4.5. En figure 4.6, la fréquence observée et la fréquence prédite par le modèle sont illustrées pour le même cycle de 600 secondes. Les erreurs de déphasages semblent plus importantes avec le MHFD qu'avec le MHFC. Cela est aussi observable sur les prédictions de fréquence qui semblent diverger plus fortement avec le MHFD sur certains cycles.

La moyenne, l'écart type et le maximum absolu des erreurs de prédictions juste avant le rallumage sur τ et θ et pour différentes durées de cycle sont répertoriés dans le tableau 4.6. Pour toutes ces durées de cycle le temps GPS ON est toujours de 1 seconde et le temps d'apnée de $k-1$ secondes. Les écarts type de ces erreurs sont tracés en fonction de la durée du cycle en figure 4.7. Le MHFD améliore légèrement l'écart type (jusqu'à 2% pour $k = 3000$ secondes)

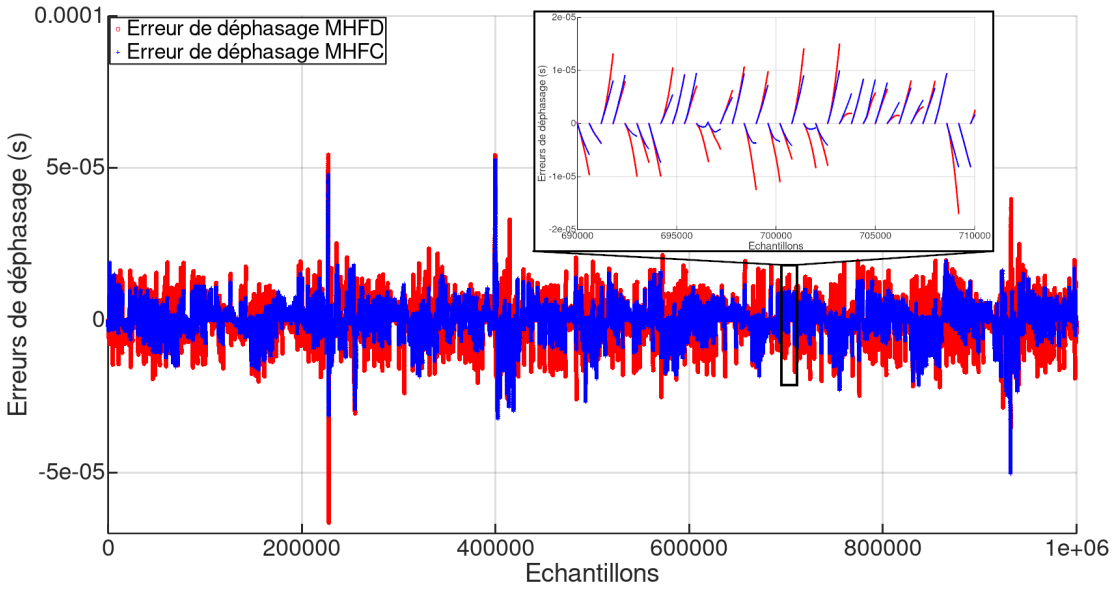


FIGURE 4.5 – Erreurs de prédiction du déphasages pour k=600 s - MHFC vs MHFD

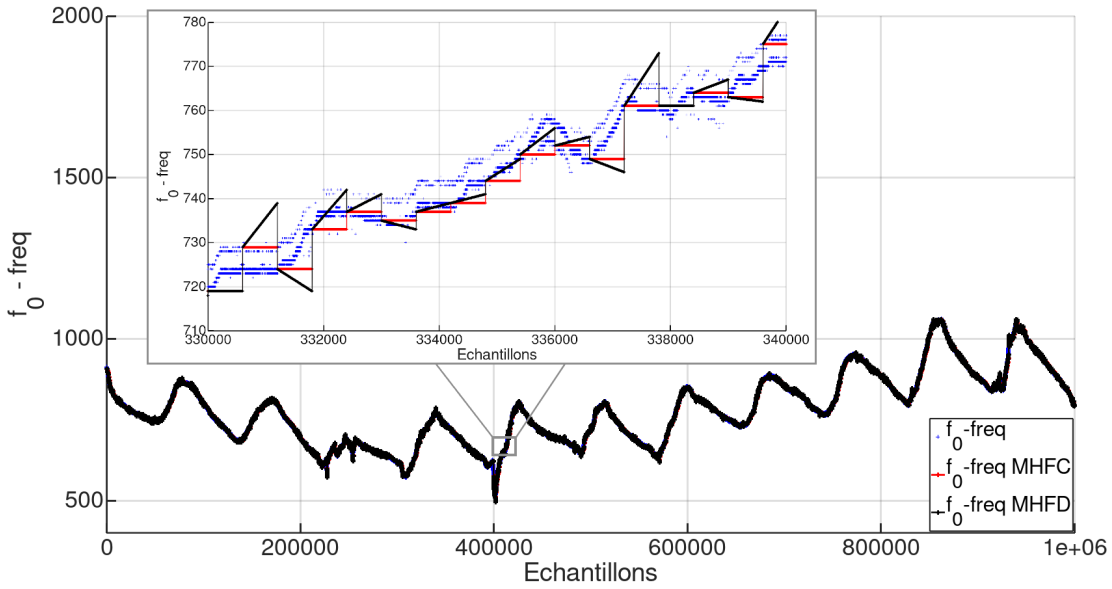


FIGURE 4.6 – Fréquence prédite par le modèle pour k=600 s - MHFC vs MHFD

des prédictions sur θ pour des durée d'apnées supérieures à 1500 secondes. La faiblesse des améliorations apporté par le MHFD peut en partie être expliqué par le bruit sur le signal PPS (cf chapitre 3) qui impacte les observations de τ quand le GPS est rallumé. Le calul de la pente

est sensible aux erreurs sur ces observations. Le mise en place d'un filtre de Kalman est détaillé dans la section suivante afin de vérifier cette hypothèse.

k	Erreurs sur τ (s)			Erreurs sur θ (s)		
	Moyenne	Ecart-type	Max abs	Moyenne	Ecart-type	Max abs
10	-4.6000E-11	2.1561E-08	7.0834E-08	-2.2172E-10	1.4556E-07	4.6438E-07
25	4.4688E-11	2.1584E-08	7.0834E-08	1.0728E-09	3.5350E-07	1.1905E-06
50	-7.9164E-12	2.1647E-08	7.0834E-08	8.1277E-10	7.0450E-07	1.9072E-06
100	1.1208E-10	2.2029E-08	9.5834E-08	8.2146E-09	1.4183E-06	4.4376E-06
250	1.0625E-10	2.3295E-08	9.1667E-08	2.8237E-08	3.6723E-06	1.1724E-05
500	5.0001E-11	2.5217E-08	1.6667E-07	1.9864E-08	7.7934E-06	3.7991E-05
1000	-2.4583E-10	3.1810E-08	3.0833E-07	4.5741E-08	1.8445E-05	1.5407E-04
2000	-7.4995E-11	4.8469E-08	3.4583E-07	-1.3754E-07	5.0248E-05	3.7209E-04
3000	-1.4139E-09	7.0579E-08	7.2917E-07	-2.4339E-06	1.1638E-04	1.1504E-03

TABLE 4.3 – Erreurs sur τ et θ - MHFD

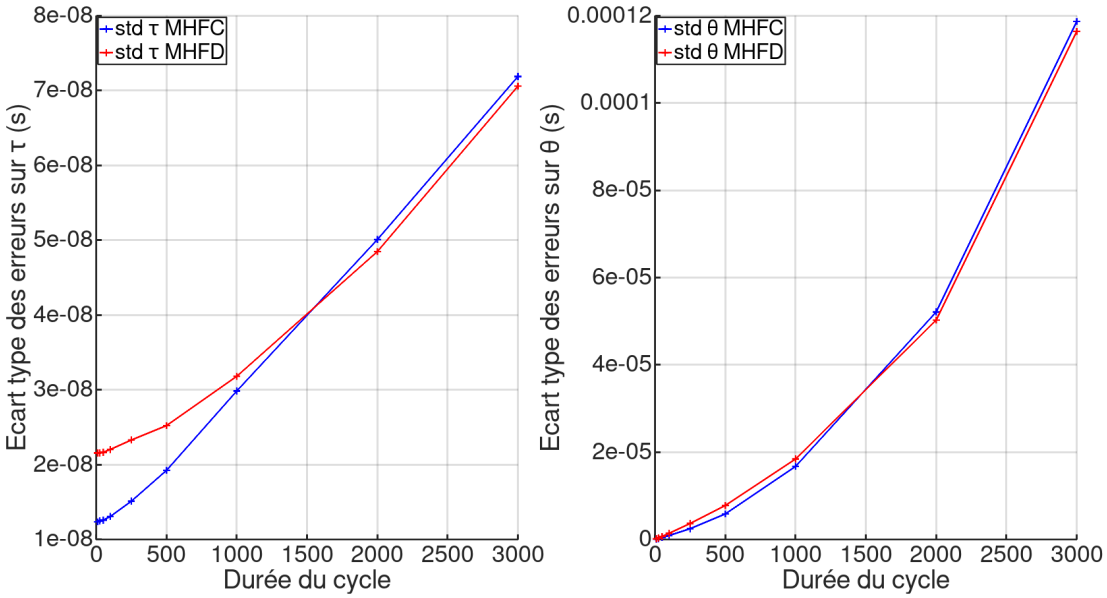


FIGURE 4.7 – Ecarts types des erreurs d'estimation des paramètres - MHFC vs MHFD

4.3 Filtrage de Kalman

4.3.1 Filtrage de Kalman avec récepteur GPS allumé

Comme il a été vu dans le chapitre précédent, le bruit sur le signal PPS peut être atténué à l'aide de filtres. La représentation d'état du modèle discret décrit par l'équation 4.10 se prête bien au filtrage de Kalman [100]. Ce filtre à réponse impulsionnelle infinie permet d'estimer l'état d'un système linéaire à partit d'observations en présence de bruits blancs. Ici, il nous permet d'estimer les paramètres du modèle τ et θ à partir des mesures bruitées par la gigue sur le signal PPS. Soit $z[n]$ la mesure de θ avec le signal PPS bruité. Le bruit sur la mesure est modélisé par une loi normale telle que :

$$z[n] = \theta[n] + v[n] \quad \text{avec} \quad v[n] \sim N(0, \sigma_v^2) \tag{4.22}$$

Les équations du processus de filtrage de Kalman discret [101] peuvent être décomposées en trois parties. Dans un premier temps, l'état futur est prédit à partir de l'équation du modèle. Ensuite, l'innovation et le gain de Kalman sont calculés. L'innovation correspond à la différence entre l'état prédit et les observations ; le gain de Kalman est calculé à partir de la covariance de la prédiction et de la covariance de l'innovation. Enfin, la prédiction de l'état est corrigée en lui ajoutant le produit de l'innovation et du gain de Kalman.

- Prédiction de l'état (x) et de la covariance de l'état (P) :

$$\hat{x}_{n+1|n} = F\hat{x}_{n|n} + u \tag{4.23}$$

$$P_{n+1|n} = FP_{n|n}F^T + Q \tag{4.24}$$

- Calcul de l'innovation (ν), de la covariance de l'innovation (S) et du gain de Kalman (K) :

$$\nu_{n+1} = z_{n+1} - H\hat{x}_{n+1|n} \tag{4.25}$$

$$S_{n+1} = HP_{n+1|n}H^T + R \tag{4.26}$$

$$K_{n+1} = P_{n+1|n}H^T S_{n+1}^{-1} \tag{4.27}$$

- Correction des prédictions de l'état (x) et de la covariance de l'état (P) :

$$\hat{x}_{n+1|n+1} = \hat{x}_{n+1|n} + K_{n+1}\nu_{n+1} \tag{4.28}$$

$$P_{n+1|n+1} = (I - K_{n+1}).P_{n+1|n} \tag{4.29}$$

$$\text{Avec} \quad Q = \begin{bmatrix} \sigma_w^2 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{et} \quad R = \sigma_v^2 \tag{4.30}$$

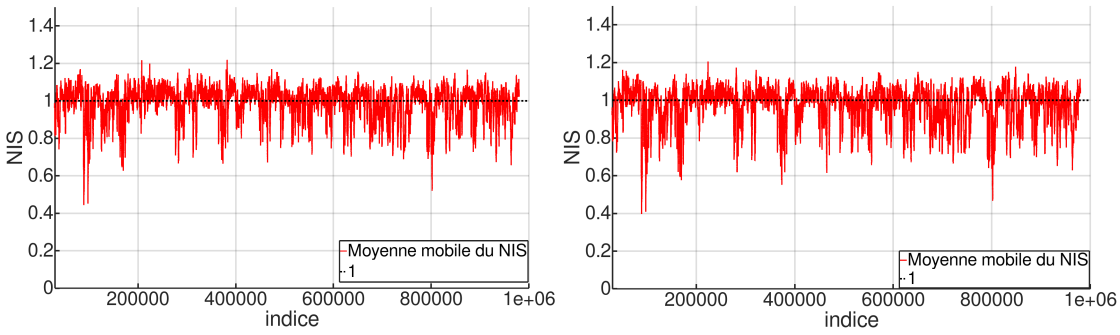
Afin de tester le filtre, une implémentation sous GNU Octave est utilisée en post-traitement. Le filtre est initialisé avec les observations bruitées et les paramètres τ et θ estimés sur l'intégralité du jeu de données. Etant donné que les véritables valeurs des paramètres (i.e non bruités) ne sont pas connues, le réglage des variances σ_w^2 et σ_v^2 se fait empiriquement à partir des estimations de l'innovation ν. Si le filtre est bien réglé et les bruits gaussiens, alors l'innovation est un bruit blanc de variance S. Pour vérifier que l'innovation possède bien les caractéristiques d'un bruit blanc, il est possible de procéder à plusieurs tests [101] : Vérifier que les 68e, 95e et 99.7e centiles des innovations correspondent bien à $\sqrt{S}$, $2\sqrt{S}$ et $3\sqrt{S}$, respectivement ; vérifier l'absence de biais en validant que la moyenne mobile des NIS (Normalized Innovations Squared) est égale au nombre de degrés de liberté (ici 1 car ν est un scalaire) ; vérifier l'absence d'auto-corrélation.

Le tableau 4.4 regroupe les centiles des innovations. Les NIS sont tracés en figure 4.8a et 4.8b pour les deux noeuds de capteurs. L'auto-corrélation est tracée en figure 4.9a et 4.9b pour les deux capteurs. On observe que l'innovation est non biaisée, présente peu d'autocorrélation

et sa distribution semble approximativement gaussienne de variance S. Le filtre semble donc fonctionner et les paramètres sont correctement réglés.

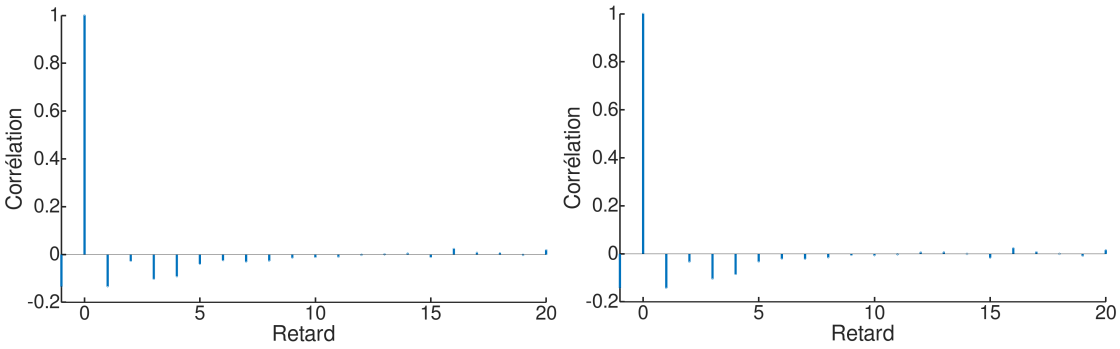
	Noeud 1	Noeud 2
$\lvert\nu\rvert < \sqrt{(S)}$	66.17 %	65.75 %
$\lvert\nu\rvert < \sqrt{(S)}$	96.05 %	95.68 %
$\lvert\nu\rvert < \sqrt{(S)}$	99.89 %	99.89 %

TABLE 4.4 – Distribution de l'innovation dans le filtre de Kalman



(a) NIS du noeud 1 (b) NIS du noeud 2

FIGURE 4.8 – Evolution du NIS



(a) Auto-corrélation des innovations sur le noeud 1 (b) Auto-corrélation des innovations sur le noeud 2

FIGURE 4.9 – Auto-corrélation des innovations sur le noeud 2

Pour évaluer les performances du filtre, les horodatages ont été calculés pour les deux noeuds de capteurs à partir des paramètres τ et θ filtrés. Les horodatages des deux noeuds ont ensuite été soustraits pour obtenir les erreurs de synchronisation entre les deux noeuds

de capteurs. Les statistiques de ces erreurs sont regroupés dans le tableau 4.5. En comparaison les filtres vu au chapitre précédent offrent un léger gain de 5 ns sur l'erreur maximale, mais contrairement à ceux-ci, la sortie du filtre de Kalman n'est pas retardée. Les paramètres filtrés sont disponibles immédiatement après l'acquisition des mesures, au temps de calcul près.

Après l'initialisation, il faut 30 secondes au filtre pour atteindre son régime permanent avec les paramètres proposés précédemment. Cette durée peut paraître relativement longue dans le cadre des cycles ON/OFF GPS étant donné que l'on cherche à minimiser la durée pendant laquelle le GPS est allumé, mais les erreurs dues à la gigue du signal PPS sont diminuées avant d'atteindre le régime permanent. Le compromis entre le temps de filtrage et les gains sur les erreurs est donc étudié dans la section 4.4.1 après avoir mesuré les performances du filtrage dans le mode ON/OFF.

	Minimum	Moyenne	Médianne	Ecart type	Maximum
Erreur	-15.16 ns	-1.67 ns	-1.69 ns	3.12 ns	13.29 ns
Erreur absolue	0 s	2.81	2.39 ns	2.15 ns	15.16 ns

TABLE 4.5 – Statistiques des erreurs de synchronisation

4.3.2 Filtrage de Kalman avec extinction périodique du récepteur GPS

Après avoir démontré l'efficacité du filtre de Kalman pour filtrer l'erreur du PPS, on évalue son impact sur les prédictions. Ce filtre a été décrit dans la section 4.3.1 pour améliorer la précision de synchronisation quand le récepteur GPS est tout le temps allumé. Ici il est utilisé uniquement pendant les phases ON pour estimer plus précisément la fréquence et le déphasage avant la période d'apnée. Comme vu en section 4.2.2, une meilleure connaissance des paramètres avant la période d'apnée devrait permettre une plus petite erreur au rallumage et donc une apnée plus longue pour la même erreur que les versions non filtrés des modèles prédictifs MHFC et MHFD.

En utilisant le filtre avec une durée de GPS allumé k_{on} de 30 secondes, le jeu de donnée est

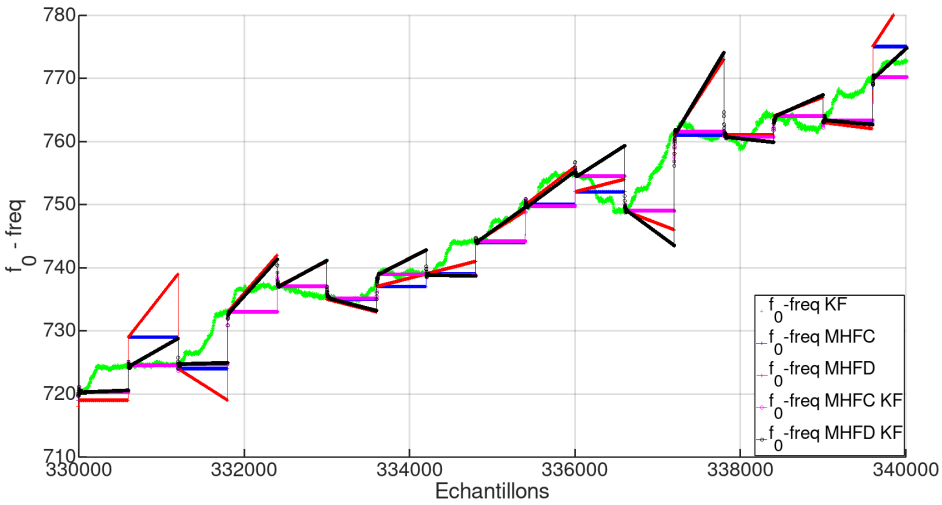
soumis aux mêmes scénarios que dans la section 4.2.2. Les erreurs de prédiction de déphasage pour un cycle de 600 secondes sont illustrées en figure 4.10a. Ces erreurs correspondent à la différence entre le déphasage prédit par le modèle et le déphasage observé et filtré avec le GPS allumé tout le temps (cf section précédente). En figure 4.10b, la fréquence observée et la fréquence prédite par le modèle sont illustrées pour le même cycle de 600 secondes. Les erreurs de déphasages semblent moins importantes avec les versions filtrées. Cela est aussi observable sur les prédictions de fréquence qui semblent diverger plus fortement en l'absence de filtrage. On observe sur la fréquence que le filtrage pendant la période ON permet bien de "recoller" à la fréquence de référence (GPS tout le temps allumé avec filtrage) avant le début de l'apnée.

La moyenne, l'écart type et le maximum absolu des erreurs de prédictions juste avant le rallumage sur τ et θ et pour différentes durées de cycle sont répertoriés dans les tableaux 4.6 et 5.4 pour le MHFC et le MHFD, respectivement. Pour toutes ces durées de cycle le temps GPS ON est toujours de 30 secondes et le temps d'apnée de $k - 30$ secondes.

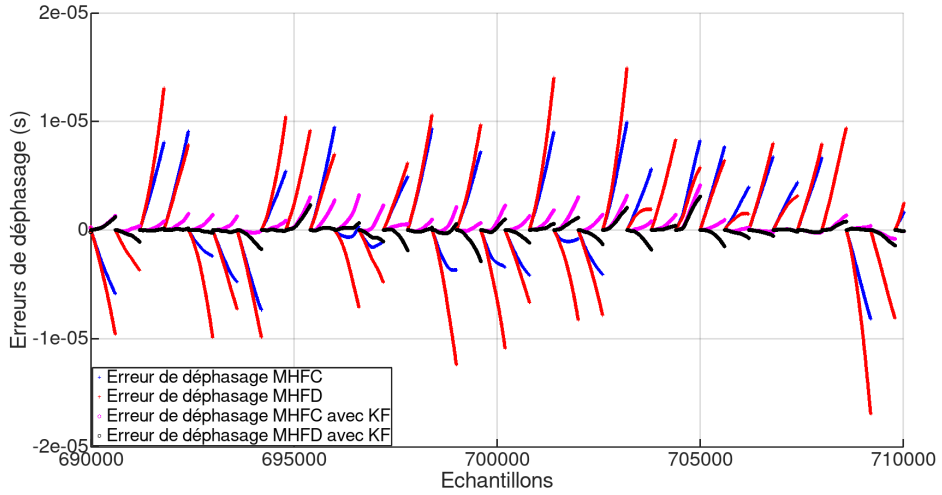
k	Erreurs sur τ (s)			Erreurs sur θ (s)		
	Moyenne	Ecart-type	Max abs	Moyenne	Ecart-type	Max abs
10	-2.4448E-11	8.7480E-09	2.6934E-08	-1.5010E-10	9.6327E-09	5.9916E-08
25	5.6322E-12	8.7389E-09	2.5500E-08	-1.3066E-10	9.6307E-09	4.2601E-08
50	-1.4754E-11	8.8604E-09	3.8569E-08	-1.0732E-10	2.4439E-08	3.6256E-07
100	6.1277E-11	9.3297E-09	5.7981E-08	7.8065E-10	1.2683E-07	1.5907E-06
250	8.0406E-11	1.1683E-08	8.1651E-08	7.3376E-09	8.7927E-07	8.5828E-06
500	1.5335E-10	1.6512E-08	1.4742E-07	1.3557E-08	3.4185E-06	3.4204E-05
1000	1.6818E-10	2.7765E-08	2.8910E-07	2.3227E-07	1.3335E-05	1.3909E-04
2000	9.1196E-10	4.8723E-08	2.8994E-07	9.5975E-07	4.7730E-05	2.5691E-04
3000	6.8650E-10	7.0751E-08	3.8076E-07	2.2318E-06	1.1346E-04	9.0139E-04

TABLE 4.6 – Erreurs sur l'estimation des paramètres - MHFC

Les écarts-types de ces erreurs sont tracés en fonction de la durée du cycle en figure 4.11. Le filtrage est bénéfique pour les deux modèles, mais c'est le MHFD qui en bénéficie le plus. Pour les versions filtrées, le seuil de distinction entre le MHFC et le MHFD est située aux alentours de $k = 200s$. En dessous de ce seuil le MHFC donne de meilleurs résultats, au dessus de ce seuil le MHFD améliore l'écart type (23 % maximum observé pour $k = 2000s$) des prédictions sur θ.



(a) Erreur de fréquence



(b) Erreur de déphasage

FIGURE 4.10 – Estimation du déphasage et de la fréquence avec KF

Cela confirme l'hypothèse émise à la fin de la section 4.2.2 ; les faibles performances du MHFD sont en partie dues à la plus forte sensibilité du modèle au bruit PPS. En utilisant un filtrage pour limiter l'erreur due au bruit lors des périodes de synchronisations, ce modèle se démarque plus nettement du MHFC pour les cycles supérieurs à 200 secondes. Pour les cycles plus courts, on peut conjecturer que les variations aléatoire court terme de la fréquence ne permettent pas de valider l'hypothèse de la variation lente du paramètre u (cf section 4.2.2).

k	Erreurs sur τ (s)			Erreurs sur θ (s)		
	Moyenne	Ecart-type	Max abs	Moyenne	Ecart-type	Max abs
10	-2.4427E-11	8.7479E-09	2.6934E-08	-1.4989E-10	9.6316E-09	4.4363E-08
25	5.6181E-12	8.7389E-09	2.5500E-08	-1.3071E-10	9.6306E-09	4.2601E-08
50	-3.8188E-11	8.9955E-09	3.0602E-08	-3.2099E-10	3.0003E-08	2.9167E-07
100	1.4659E-11	9.4073E-09	4.7560E-08	-8.0952E-10	1.2968E-07	1.2259E-06
250	-3.5734E-11	1.1242E-08	1.0602E-07	-5.3325E-09	7.6669E-07	1.0350E-05
500	-7.6670E-11	1.4286E-08	1.4140E-07	-4.0302E-08	2.6091E-06	2.9132E-05
1000	-2.8064E-10	2.3903E-08	2.8825E-07	1.4945E-08	1.0786E-05	1.3868E-04
2000	5.6651E-11	4.2538E-08	3.2328E-07	1.1774E-07	3.8702E-05	3.4074E-04
3000	-3.9995E-10	6.6976E-08	7.0304E-07	6.1891E-07	1.0633E-04	1.0943E-03

TABLE 4.7 – Erreurs sur l'estimation des paramètres - MHFD

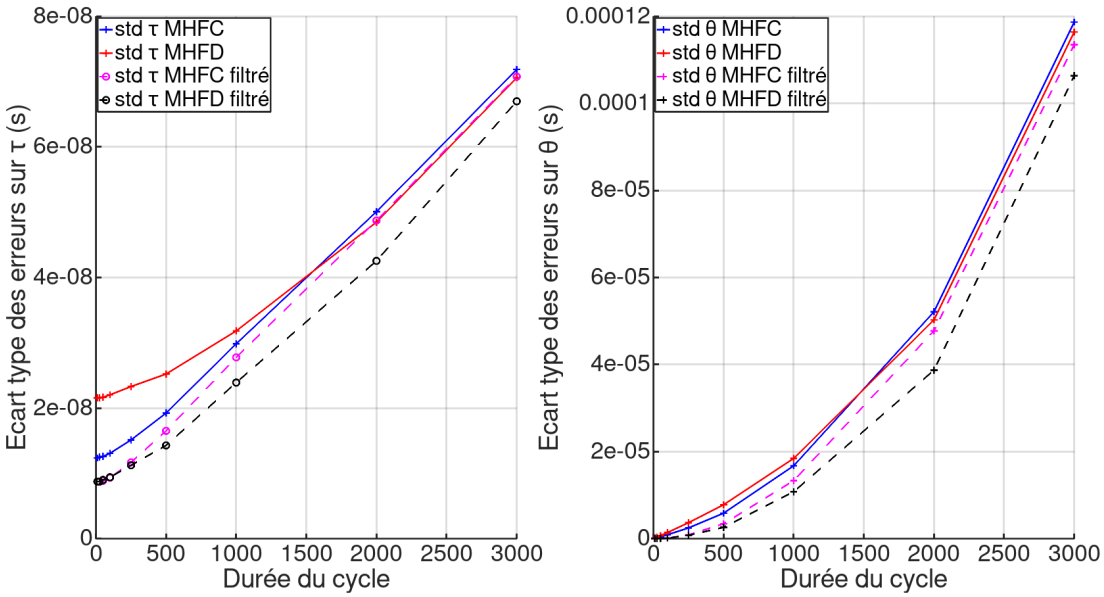


FIGURE 4.11 – Ecarts types des erreurs d'estimation - MHFC vs MHFD avec et sans KF

4.4 Évaluation expérimentale des modèles

4.4.1 Évaluation de l'impact sur les horodatages

Afin de valider l'impact du filtrage et de comparer les modèles, les erreurs d'horodatages doivent être analysées. Ici, la référence de synchronisation correspond aux horodatages calculés par un noeud utilisant le filtre de Kalman avec le récepteur GPS tout le temps allumé. Les

erreurs de synchronisations correspondent aux différences entre la référence et les horodatages calculés sur ce même noeud avec le mode ON/OFF.

Comme il a été vu précédemment, le régime permanent du filtre de Kalman est obtenu après 30 échantillons quand il est initialisé sur les observations. Ces 30 secondes doivent donc êtres 30 secondes de GPS allumé ce qui représente un certain coût énergétique. Afin de minimiser ce coût, il est possible de diminuer le temps allumé, ce qui aura aussi pour conséquence de diminuer le temps de filtrage et donc la précision de l'estimation. Pour trouver le meilleur compromis entre le temps de filtrage et la consommation énergétique, les erreurs absolues moyenne des horodatages (EAM) ont été calculées pour différentes durées ON et comparées à leur ratio durée de GPS allumé/24 heures. Ces ratios sont calculés à l'aide de l'équation 4.1 avec les paramètres définis dans la section 4.1. Les figures 4.12 et 4.13 illustrent le ratio en fonction de l'erreur moyenne absolue de synchronisation pour plusieurs durées k_{on} avec le MHFC et le MHFD. On observe bien que l'EAM augmente quand le ratio diminue du fait de temps d'apnées plus longs pour une même durée de GPS allumé. A partir de ces tests ont observe aussi que toutes les versions filtrées offrent de meilleurs ratio que les versions non filtrées jusqu'à une EAM d'environ 2 μs (cycle d'un peu moins de 500 secondes). Au delà, les version filtrées ne sont plus rentables d'un point de vue ratio. Un k_{on} de 2 secondes permet d'obtenir les meilleurs ratios pour des EAM de synchronisations comprises entre 30 ns et 2 μs. En dessous des 30 ns, les durées k_{on} plus élevées deviennent progressivement plus rentables à mesure que l'EAM diminue dans le sens ou elles sont seules à pouvoir atteindre ce niveau de synchronisation.

En matière de précision de synchronisation pour une durée de cycle donné, les versions filtrées donnent toujours de meilleurs résultats que les versions non filtrées. Cependant, quand on prend en compte le ratio, la solution la plus précise peut être une version non filtrée avec un cycle plus court comme illustré en figure 4.12 et 4.13. Le ratio est calculé pour toutes les combinaisons filtre/modèle et tracé en fonction de l'EAM des horodatages entre les deux noeuds en Figure 4.14. Les filtres sont utilisés avec un k_{on} de 2 secondes. Pour toutes les combinaisons, le ratio décroît de façon exponentielle avec l'augmentation de l'EAM pour tendre vers le ratio

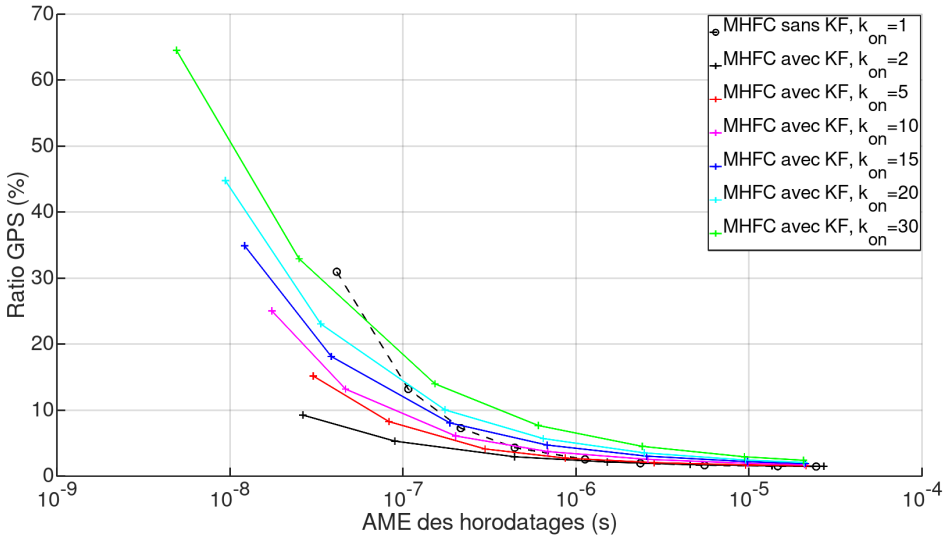


FIGURE 4.12 – Ratio GPS en fonction de l'EAM des erreurs d'horodatages - MHFC

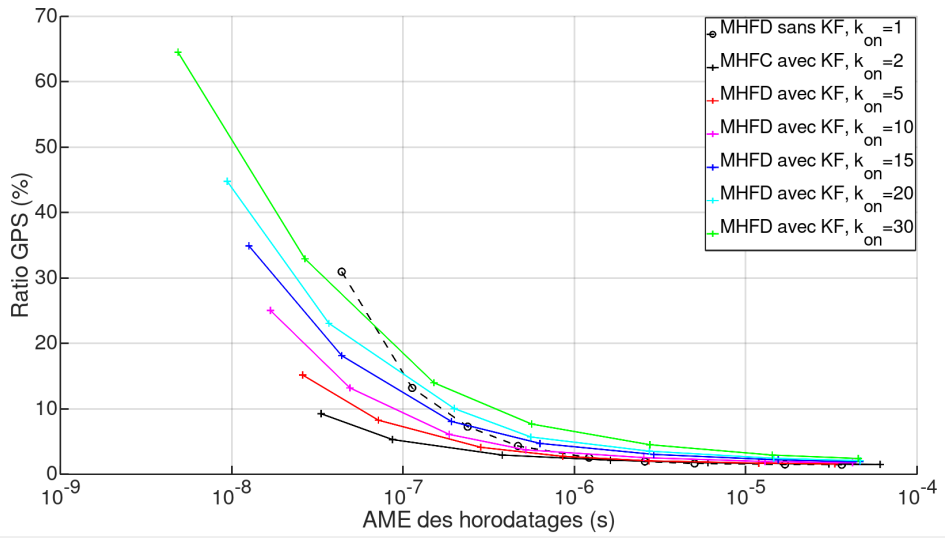


FIGURE 4.13 – Ratio GPS en fonction de l'EAM des erreurs d'horodatages - MHFD

limite de 1.33% (cf section 4.1). Le filtre de Kalman permet de diminuer le ratio d'un facteur 3 pour une EAM de 30 ns malgré un temps ON plus important. Cette amélioration diminue progressivement à mesure que l'EAM tend vers 2 μs, au-delà le filtrage n'est plus nécessaire. Le MHFC permet une amélioration du ratio de 7% par rapport au modèle MHFD sous les 200s. Les améliorations apportées par le MHFD au dessus des 200 secondes restent marginales à 0.5% du

fait de la décroissance exponentielle du ratio. Les erreurs maximales entre deux capteurs sont tracés en figure 4.15, comme pour les EAM, le filtre permet un meilleur ratio à une précision donnée.

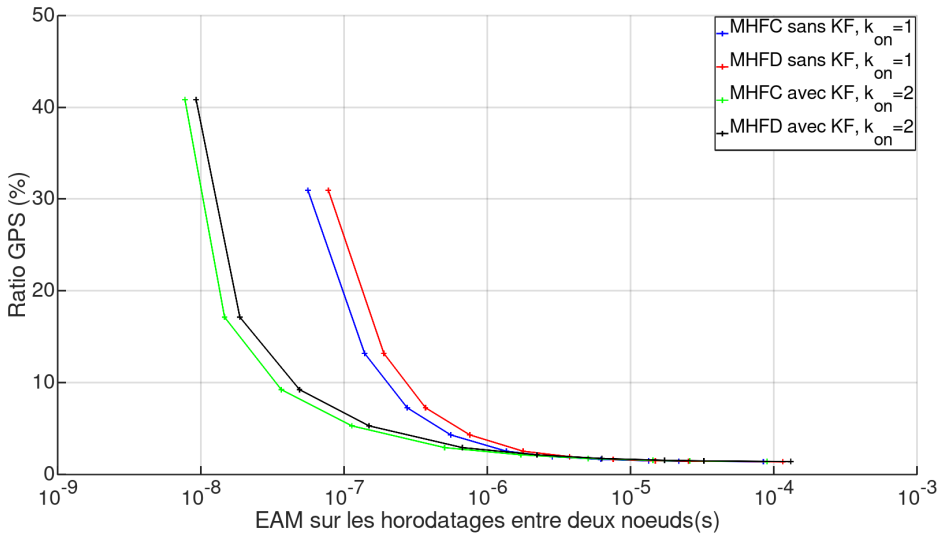


FIGURE 4.14 – Ratio GPS en fonction des EAM - temps réel

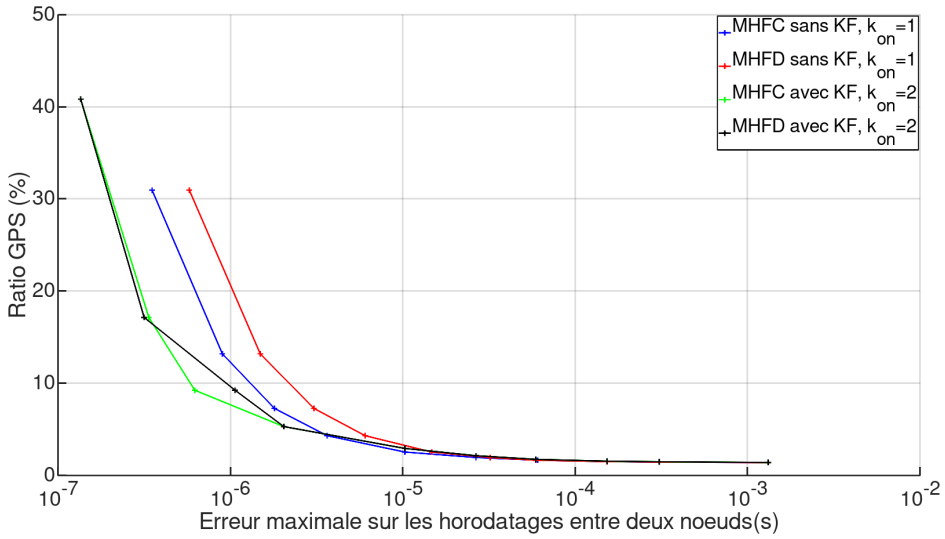


FIGURE 4.15 – Ratio GPS en fonction des erreurs maximales - temps réel

4.4.2 Cas particulier de la datation a posteriori

Pour certains cas d'application où les horodatages peuvent être reçus a posteriori des événements à dater une autre approche est possible. Dans ce cas de figure, le noeud de capteur peut calculer et transmettre les horodatages après le rallumage du récepteur GPS. Ce rallumage peut être périodique ou déclenché par la détection d'un évènement. Ici on s'intéresse au cas plus général ou la synchronisation est toujours périodique et les évènements sont enregistrés indépendamment de l'état du récepteur GPS mais horodatés au rallumage du cycle suivant. Les deux modèles peuvent êtres réécrits pour prendre en compte le déphasage et la fréquence avant l'apnée mais aussi le déphasage et la fréquence après l'apnée. Ainsi pour le MHFC le τ constant pendant le cycle allant de n à $n+k$ devient :

$$\tau = \frac{\theta[n+k] - \theta[n]}{k} + 1 \tag{4.31}$$

Le noeud de capteurs doit ensuite calculer le déphasage durant la dernière période d'apnée afin de calculer les horodatages :

$$\theta[i] = \theta[n] + i(\tau - 1) \quad \text{avec} \quad i \in [n+1; n+k] \tag{4.32}$$

De la même façon pour le MHFD, la raison α de la suite des $\tau[i]$ entre n et $n+k$ s'exprime par :

$$\alpha = 2\frac{\theta[n+k] - \theta[n] - k(\tau[n] - 1)}{(k-1)k} \tag{4.33}$$

Les paramètres τ et θ de la dernière apnée sont calculés à la fin du cycle :

$$\tau[i] = \tau[n] + i\alpha \tag{4.34}$$

$$\theta[i] = \theta[n] + i(\tau[n] - 1) + \frac{1}{2}\alpha(i-1)i \quad \text{avec} \quad i \in [n+1; n+k] \tag{4.35}$$

Comme dans la section 4.2.2, les deux modèles a posteriori ont été évaluées par rapport à des paramètres de références avec le GPS tout le temps ON. Les erreurs de prédiction de déphasage pour un cycle de 600 secondes avec les modèles "temps réel" et a posteriori sont illustrées en figure 4.16. Les méthodes d'horodatage précédentes sont qualifiées de "temps réel" dans le sens ou les horodatages sont calculés à chaque seconde du jeu de données, indépendamment du fait que ces horodatages soient calculés en post-traitement pour simplifier l'analyse. Les fréquences sont illustrées en figure 4.17. Les erreur de prédictions a posteriori sont naturellement moins grandes que les erreurs de prédictions "temps réel". Comme pour la version "temps réel", le MHFD a posteriori souffre des incertitudes dues au bruit sur le signal PPS.

En utilisant le filtre avec $k_{on} = 30s$ le jeu de donnée est soumis aux mêmes scénarios que précédemment (MHFC et MHFD a posteriori). Les erreurs de prédiction de déphasage pour un cycle de 600 secondes sont illustrées en figure 4.18. Ces erreurs correspondent à la différence entre le déphasage prédit par le modèle et le déphasage observé et filtré avec le GPS ON tout le temps. En figure 4.19, la fréquence observée et la fréquence prédite par le modèle sont illus-

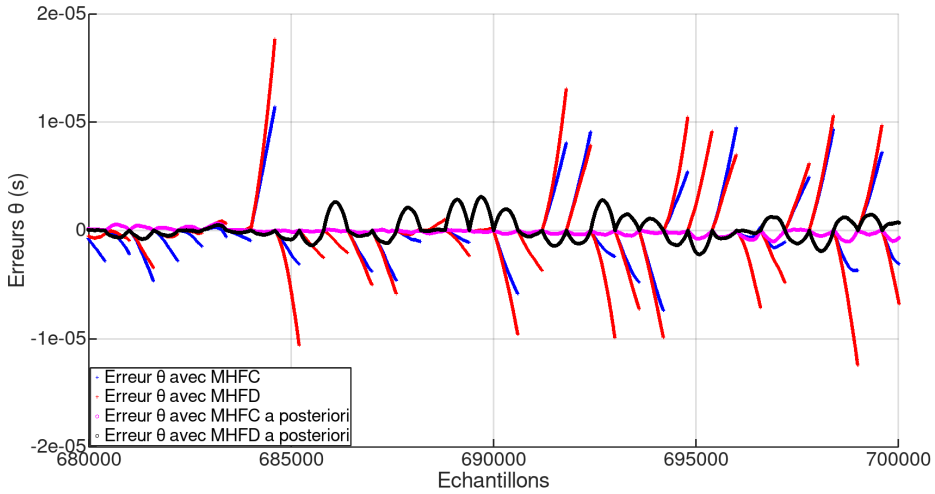


FIGURE 4.16 – Erreur d'estimaton de déphasage - a posteriori

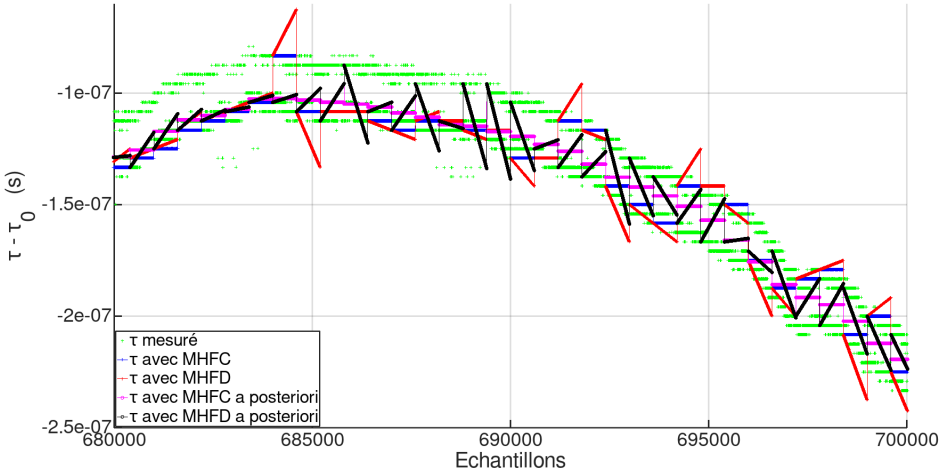


FIGURE 4.17 – Estimation de la fréquence - a posteriori

trées pour le même cycle de 600 secondes. Les erreurs de déphasages a posteriori sont naturellement moins grandes que les erreurs de déphasages "temps réel". Comme pour les version "temps réel", le filtrage permet d'améliorer les prédictions. Ces deux remarques sont également valables pour l'estimation de τ.

Afin de vérifier l'apport du filtrage de Kalman quand le ratio est pris en compte, les erreurs d'horodatage et leurs ratios respectifs sont calculés pour différentes durées de période "ON" GPS. Comme dans la section 4.4.1, les erreurs d'horodatages sont calculées en faisant la diffé-

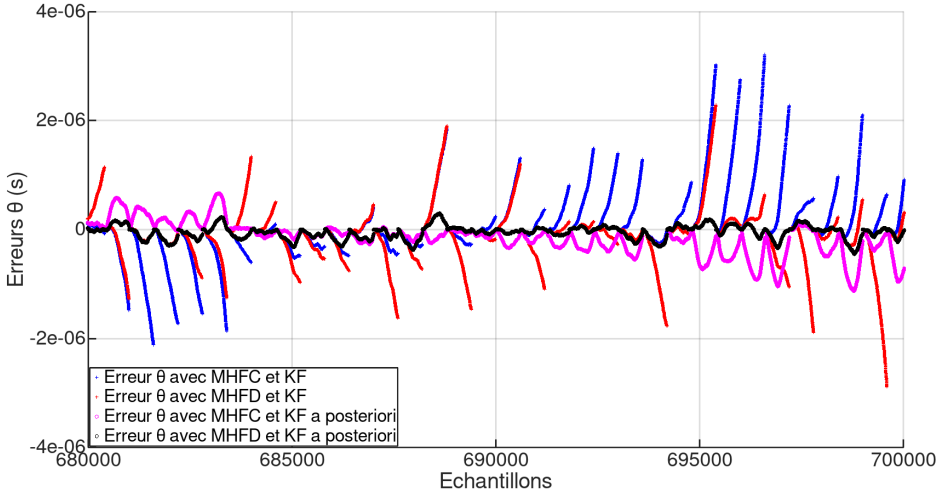


FIGURE 4.18 – Erreur d'estimation de déphasage avec KF - a posteriori

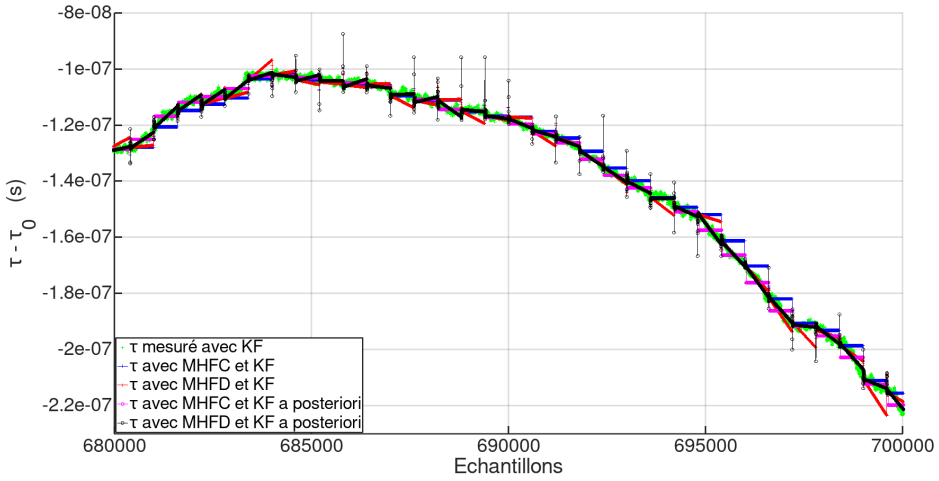


FIGURE 4.19 – Estimation de la fréquence avec KF - a posteriori

rence entre les horodatages des mêmes événements sur les deux noeuds. Les figures 4.20 et 4.21 représentent respectivement l'EAM en fonction du ratio GPS pour le MHFC et le MHFD a posteriori. Contrairement à la version "temps réel", l'utilisation du filtre de Kalman ne permet pas une meilleure EAM à un ratio donné dans le cas du MHFC. Cela signifie que même si le filtrage de Kalman permet d'améliorer la précision, cette amélioration ne permet pas de compenser le coût énergétique engendré par l'allongement de la durée "ON". Il est plus efficace de ne pas utiliser le filtrage et de réduire les temps d'apnée pour la synchronisation a posteriori avec le

MHFC. Dans le cas du MHFD, le filtrage avec une durée "ON" de 10 secondes permet d'obtenir les meilleurs ratios en fonction des EAM. Cependant on observe que le MHFC permet d'obtenir les meilleurs EAM. Le MHFD n'a donc pas réellement d'intérêt pour la synchronisation a posteriori.

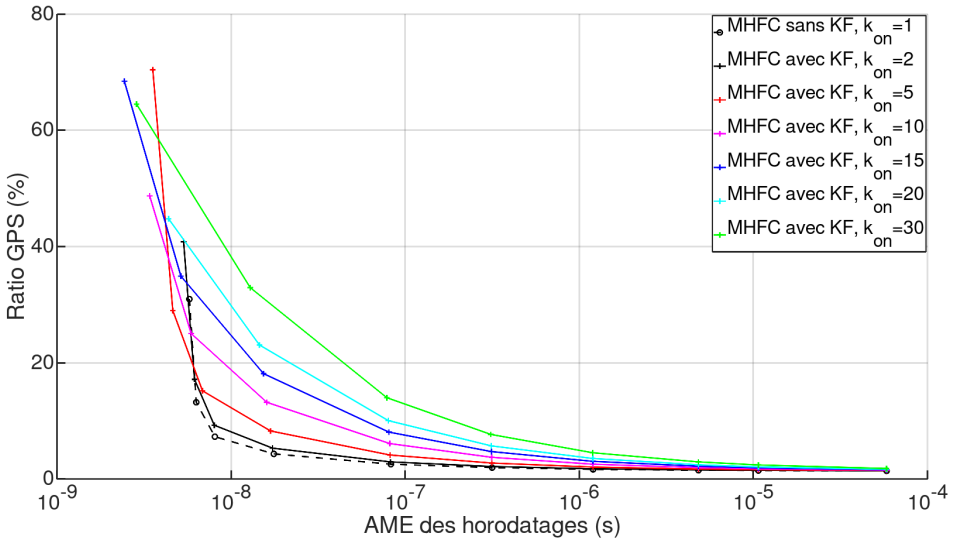


FIGURE 4.20 – Ratio GPS en fonction des EAM - MHFC a posteriori

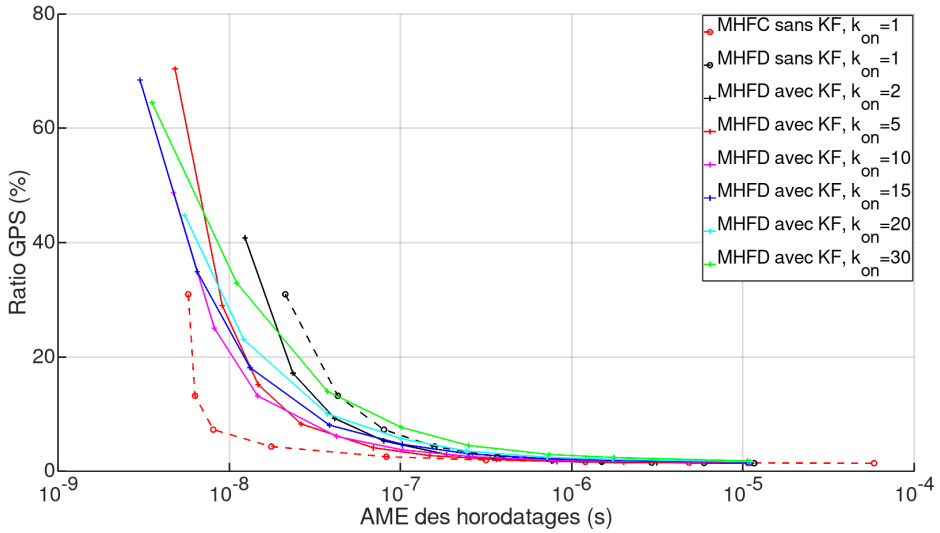


FIGURE 4.21 – Ratio GPS en fonction des EAM - MHFD a posteriori

Finalement, le ratio GPS est tracé en fonction des erreurs maximales sur les horodatages

entre les deux noeuds en figure 4.22. On observe toujours que le MHFC non filtré permet le meilleur ratio à une erreur maximale donnée. Les erreurs maximales ainsi que les EAM, le ratio et la puissance moyenne théorique pour le MHFC non filtré sont répertoriés dans le tableau 5.6. La puissance moyenne théorique nécessaire au récepteur GPS est calculée avec une puissance de 120 mW en fonctionnement nominal. Les erreurs maximales sont comprises entre 53 ns et 257 μs et les puissances entre 37 mW et 1.7mW, pour des cycles de 10 secondes à 7000 secondes. Comparativement aux résultats "temps réel", le "a posteriori" permet de diminuer les erreurs d'horodatages maximales d'un facteur 10.

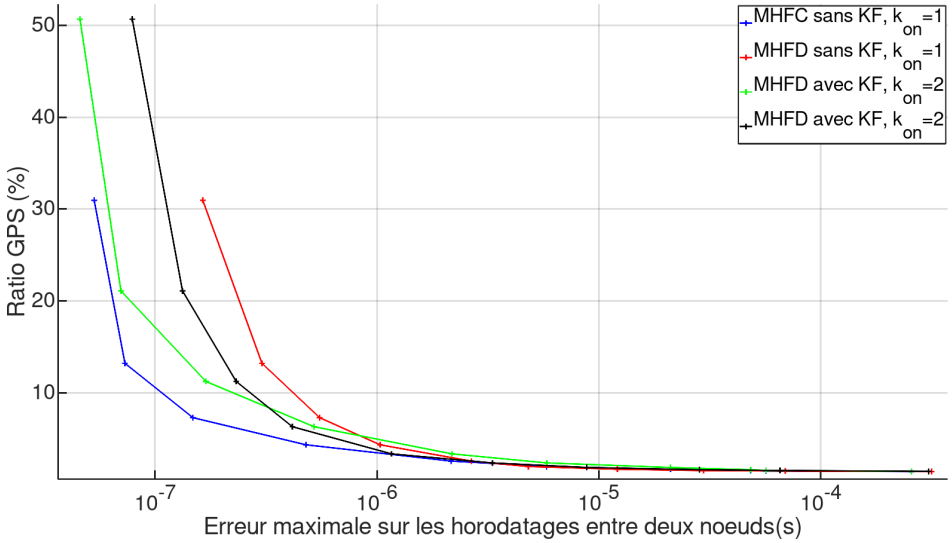


FIGURE 4.22 – Ratio GPS en fonction des erreurs maximales - a posteriori

	Erreurs absolues horodatages			Récepteur GPS	
k	EAM (s)	Écart-type (s)	Maximum (s)	ratio (%)	puissance (mW)
10	9.2603e-09	6.9947e-09	5.3085e-08	30.95	37.14
25	1.0106e-08	7.6326e-09	7.3051e-08	13.20	15.84
50	1.3325e-08	1.1190e-08	1.4791e-07	7.28	8.74
100	2.5579e-08	2.7013e-08	4.7846e-07	4.32	5.18
250	8.4719e-08	1.0907e-07	2.1590e-06	2.54	3.05
500	2.1915e-07	2.9840e-07	5.8131e-06	1.95	2.34
1000	6.2651e-07	9.3410e-07	2.1044e-05	1.66	1.99
2000	1.9922e-06	3.1651e-06	4.8324e-05	1.51	1.81
3000	3.9563e-06	5.6341e-06	5.6668e-05	1.46	1.75
7000	1.7936e-05	2.6857e-05	2.5729e-04	1.40	1.68

TABLE 4.8 – Erreurs d'horodatages entre deux noeuds et ratio en fonction de k - a posteriori

4.5 Conclusion

Dans ce chapitre, le principe de synchronisation adaptative par GPS a été décrit. Deux modèles d'horloges (MHFC et MHFD) et deux méthodes de datation ("temps réel" et "a posteriori") ont été développés. Un filtre de Kalman a été développé afin de filtrer le bruit GPS lors du rallumage de celui-ci. Il est démontré que le MHFD permet d'améliorer la précision de synchronisation uniquement pour des cycles longs mais que ces améliorations ne sont pas significatives sur le ratio durée "GPS allumé"/24 heures, car la majeure partie des gains sur le ratio apparaît pour des cycles plus courts. Il a été vu que le filtrage de Kalman lors du rallumage du récepteur GPS permet d'améliorer la précision de synchronisation mais engendre une durée "GPS allumé" plus longue. Cependant, il est aussi démontré que malgré cette durée plus longue, le filtrage est rentable d'un point de vue ratio, du fait que l'apnée peut être prolongée. Cet effet est uniquement observable pour la datation "temps réel" et pas pour la datation "a posteriori". Enfin, la datation "a posteriori" permet de diminuer les erreurs d'horodatages entre deux capteurs d'un facteur 10.

CHAPITRE 5

Évaluation de l'influence de la température

A partir d'une certaine durée d'apnée, les prédictions des modèles peuvent se dégrader si la température évolue trop brusquement. Afin de minimiser l'influence de la température, son intégration dans l'estimation des paramètres du modèle pendant les périodes d'apnée est étudiée ici. Dans un second temps, les gains apportés par l'utilisation d'un oscillateur compensé en température sont évalués.

Sommaire

5.1 Influence de la température ambiante sur les horloges des noeuds

Dans l'état de l'art (chapitre 2), il a été vu que le principal facteur d'influence sur la fréquence d'un oscillateur à quartz est la température [28]. C'est pourquoi les noeuds de capteurs décrit précédemment (chapitre 3) ont été équipés de capteurs de température. Ces capteurs de température sont des TMP112 [102], reliés à l'unité d'horodatage via une liaison série. Ces capteurs ont une précision annoncée à $\pm 0.5°C$ [102]. Les températures des noeuds ont été enregistrées dans le jeu de données pour chaque seconde. Cette première partie de chapitre est consacrée à l'étude de la relation entre la température ambiante et le fréquence des oscillateurs mesurées sur les noeuds de capteurs. En théorie, la relation entre la fréquence d'un oscillateur à quartz tel que celui présent sur le kit de développement Spartan 6 [97], et sa température est une fonction cubique. Cette fonction possède classiquement une partie affine entre $0°C$ et $40°C$ (cf chapitre 2). Le jeu de données ayant été acquis avec les noeuds à l'intérieur du laboratoire, la température ambiante évolue de 21.25 $°C$ minimum à 25.37 $°C$ maximum. Aussi, on utilise une fonction affine pour relier le paramètre τ (cf chapitre 4) mesuré à l'aide du signal PPS et la température ambiante observée au même moment, notée t_e :

$$\tau[n] = \alpha t_e[n] + \beta \tag{5.1}$$

Les paramètres α et β peuvent être évaluées à l'aide d'une régression linéaire après une période d'apprentissage avec le GPS ON. Si cette relation est juste et que les paramètres α et β demeurent stables dans le temps, le paramètre τ peut être estimé pendant la période d'apnée à l'aide des observations de température.

Pour estimer les paramètres α et β et leur stabilité dans le temps, le jeu de données à été séparé en 5 sous parties de 2×10^5 secondes (≈ 2.3 jours) chacune. Pour chaque sous partie, le coefficient de corrélation a été calculée ainsi que les paramètres α et β à l'aide de la méthode des moindres carrées ordinaire [103]. Le tableau 5.1 regroupe ces paramètres pour les deux noeuds de capteurs. On observe que la corrélation est inférieure à -0.91 pour toutes les sous parties du jeu de données et ce sur les deux noeuds. Les paramètres α et β sont très similaires sur les deux noeuds de capteurs et semblent peu varier au cours du temps.

TABLE 5.1 – Corrélation et OLS

	Noeud 1			Noeud 2		
Jeu	Corrélation	α	β	Corrélation	α	β
1	-0.97	7.2957e-7	0.99	-0.96	7.1185e-7	0.99
2	-0.91	6.3055e-7	0.99	-0.93	5.2580e-7	0.99
3	-0.97	6.4618e-7	0.99	-0.95	5.5033e-7	0.99
4	-0.98	6.3832e-7	0.99	-0.98	6.1768e-7	0.99
5	-0.99	6.4273e-7	0.99	-0.99	6.4525e-7	0.99

Afin d'évaluer l'erreur commise en calculant τ à partir de t_e, ce paramètre est calculé pour chaque sous partie avec toutes les paires $\{\alpha, \beta\}$ estimées. Ce τ est ensuite comparé au τ de référence, observé en présence du signal PPS. Les RMSE (Root-mean-square Error) de chaque évaluation de τ sont regroupés dans le tableau 5.2 pour le premier noeud et dans le tableau 5.3 pour le second. La RMSE moyenne est calculée en excluant la diagonale des tableaux qui

représente l'évaluation de τ sur une sous-partie avec la paire $\{\alpha, \beta\}$ estimée sur la même sous partie. Cette RMSE moyenne est semblable sur les deux noeuds avec 8.9e-8 s pour le premier et 8.2e-8 s pour le second.

TABLE 5.2 – RMSE Noeud 1

	α_1, β_1	α_2, β_2	α_3, β_3	α_4, β_4	α_5, β_5
Jeu 1	5.0255e-8	6.6020e-8	1.0455e-7	1.0943e-7	5.5830e-8
Jeu 2	1.3691e-7	7.4230e-8	8.6013e-8	9.2218e-8	8.3510e-8
Jeu 3	1.5051e-7	7.8577e-8	6.2458e-8	6.3221e-8	1.0331e-7
Jeu 4	9.4648e-8	7.6611e-8	4.4502e-8	4.4248e-8	9.8849e-8
Jeu 5	9.7525e-8	5.1918e-8	9.9871e-8	9.8172e-8	4.8570e-8

TABLE 5.3 – RMSE Noeud 2

	α_1, β_1	α_2, β_2	α_3, β_3	α_4, β_4	α_5, β_5
Jeu 1	5.9763e-8	1.1995e-7	8.4621e-8	7.0953e-8	7.1745e-8
Jeu 2	9.2173e-8	5.2502e-8	6.1202e-8	7.3404e-8	6.8236e-8
Jeu 3	9.8810e-8	7.4821e-8	6.4304e-8	7.5964e-8	7.6985e-8
Jeu 4	5.7790e-8	1.4879e-7	9.6529e-8	4.2574e-8	4.5825e-8
Jeu 5	6.5074e-8	2.4596e-7	1.7511e-7	6.1736e-8	5.1139e-8

Le paramètre τ réel et le τ estimé à l'aide de la régression linéaire pour le premier jeu du noeud 1 sont illustrés en figure 5.1. On observe bien que la mesure de la température ambiante permet d'estimer le paramètres τ.

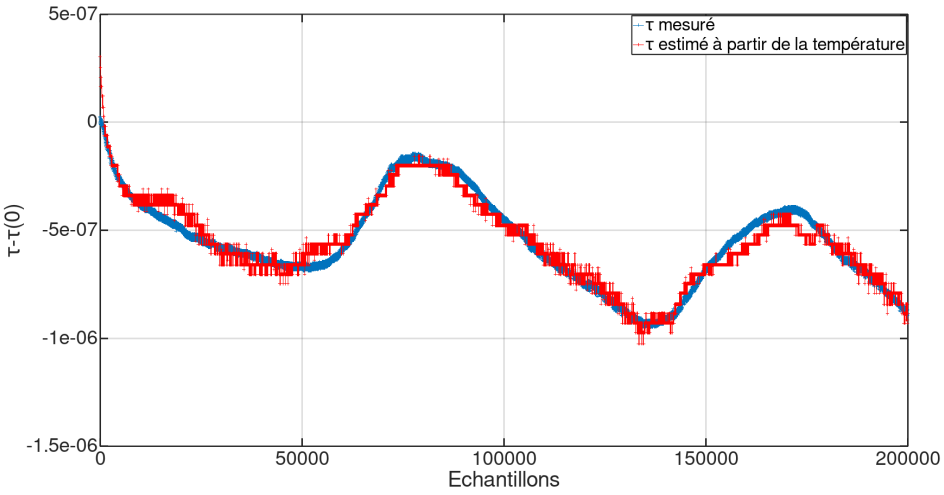


FIGURE 5.1 – Estimation du paramètre τ en fonction de la température

5.2 Prédictions du déphasage avec la température

Pour utiliser l'information de température pendant la période d'apnée, celle-ci est utilisée comme mesure dans le modèle d'état développé dans le chapitre 4. La matrice H devient $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$, les observations $z[n]$ sont calculées à partir de la température tel que :

$$z[n] = \alpha t_e[n] + \beta \quad \text{et} \quad x[n] = Hz[n] + w_{te} \tag{5.2}$$

$$\text{avec} \quad w_{te} \sim \mathcal{N}(0, \sigma_{te}^2) \tag{5.3}$$

Afin de tester l'utilisation de la température, la première sous-partie du jeu de données (le jeu 1) a été utilisée pour estimer les paramètres α et β. Sur tout le reste du jeu de données, des cycles de k points sont utilisés (cf chapitre 4) pour prédire les paramètres τ et θ. Au début de chaque cycle, le récepteur GPS est allumé pendant 30 secondes et les paramètres sont évalués à l'aide du filtre de Kalman décrit dans le chapitre 4. Ensuite, pendant la période d'apnée de $k - 30$ secondes, τ et θ sont prédits par le filtre basé sur le modèle d'état proposé et l'équation 5.2.

Le tableau 5.4 regroupe les statistiques des erreurs sur l'estimation du paramètre τ avec la température pour différentes valeurs de k. Ces erreurs sont obtenues en comparant les paramètre τ prédits et ceux de référence calculés avec le récepteur GPS tout le temps allumé et le filtrage du bruit sur le signal PPS. Les erreurs obtenues avec les méthodes MHFC et MHFD filtrées présentées au chapitre 4 sont aussi présentes dans le tableau (les valeurs sont légèrement différentes des valeurs précédentes car ici le jeu de données n'est pas utilisé au complet du fait de l'estimation des coefficients de température). On observe que la méthode utilisant la température ne donne pas de meilleurs résultats sauf pour des périodes d'apnées très longues (à partir de 10000 secondes). Ces périodes d'apnée très longues n'ont pas lieu d'être avec un rafraîchissement des éphémérides nécessaire toutes les deux heures (cf chapitre 4).

k	Moyenne erreur			Ecart-type erreur			Max abs erreur		
	MHFC	MHFD	Temp	MHFC	MHFD	Temp	MHFC	MHFD	Temp
1000	1.59e-10	1.08e-10	-2.55e-8	1.63e-8	1.31e-8	3.27e-8	2.85e-7	2.86e-7	2.08e-7
2000	1.33e-12	-7.76e-11	-4.71e-8	2.83e-8	2.26e-8	5.19e-8	2.84e-7	3.67e-7	2.88e-7
3000	3.86e-10	3.054e-10	-5.77e-8	4.31e-8	4.42e-8	5.87e-8	4.48e-7	7.15e-7	3.04e-7
7000	6.60e-11	5.34e-10	-7.09e-8	8.06e-8	6.67e-8	6.43e-8	4.47e-7	4.58e-7	3.04e-7
10000	-4.81e-9	-3.76e-9	7.45e-8	1.11e-7	1.09e-7	6.53e-8	5.26e-7	7.02e-7	2.88e-7
20000	-2.11e-8	-4.21e-10	-7.98e-8	2.22e-7	2.73e-7	6.51e-8	7.49e-5	1.31e-6	2.88e-7

TABLE 5.4 – Erreurs sur l'estimation de τ

Ces résultats sont illustrés en figure 5.2 pour un k de 10 000 secondes et en figure 5.2 pour un k de 3000 secondes. La température apparaît en effet être un indicateur moins précis que les modèles MHFC et MHFD filtrés pour la prédiction de τ pendant l'apnée sur les cycles de 3000 secondes. Ces résultats montrent que malgré la forte corrélation observée entre la température et la fréquence de l'oscillateur, la prise en compte de la température ne permet pas d'obtenir de meilleurs résultats que les modèles MHFC et MHFD pour des durées d'apnée de deux heures ou moins. Des variations de température plus rapides par rapport aux durées d'apnées pourraient cependant dégrader les performances des modèles MHFC et MHFD.

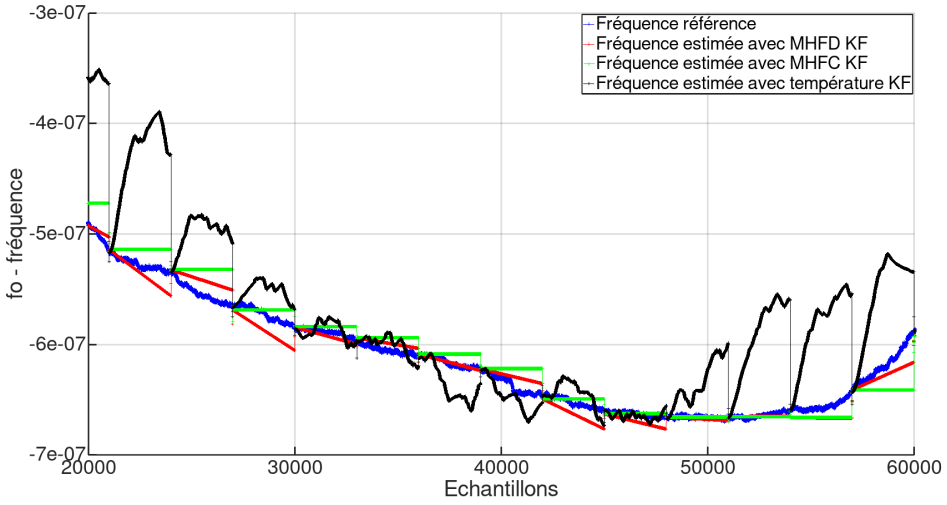


FIGURE 5.2 – Estimation de τ - MHFC et MHFD KF vs température KF pour k=3000

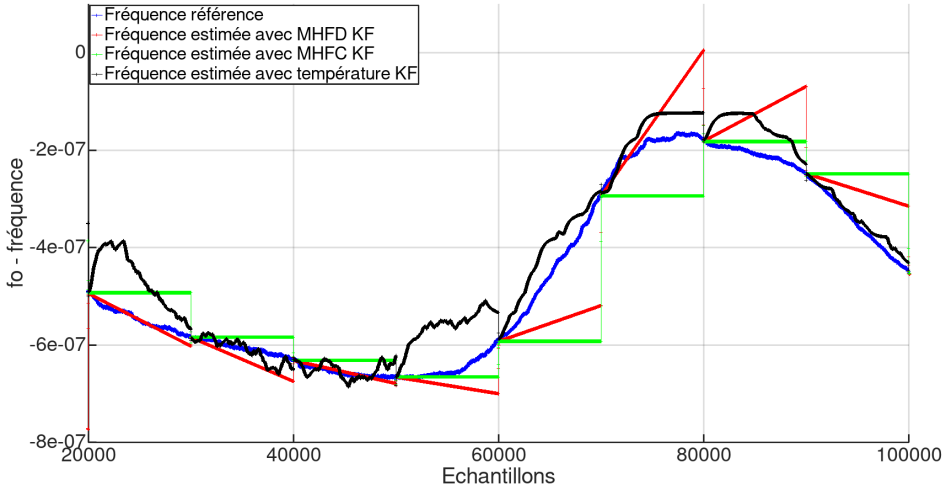
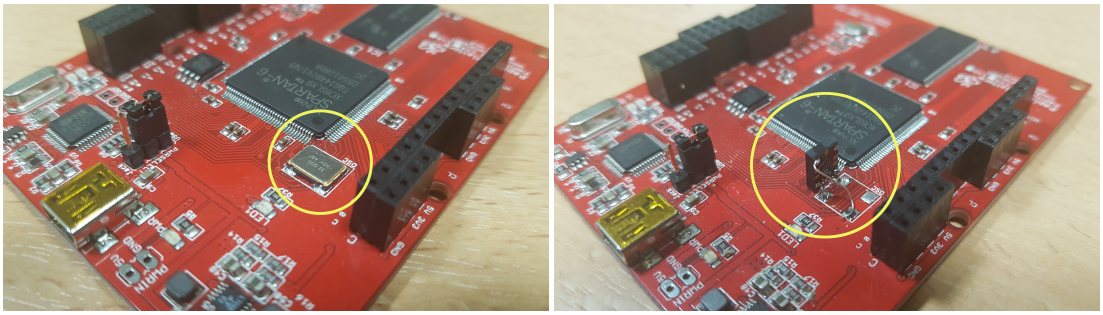


FIGURE 5.3 – Estimation de τ - MHFC et MHFD KF vs température KF pour k=10000

5.3 Utilisation d'un TCXO sur le noeud de capteurs

Comme vu dans l'état de l'art, il existe d'autres types d'oscillateurs qui embarquent directement un mécanisme de compensation en température (TCXO/DCXO) ou maintiennent le cristal de quartz à une température constante (OCXO). Comparés à la méthode de prise en compte de la température décrite précédemment les TCXO profitent mieux de l'information de température du fait d'une mesure de la température intégrée au composant et d'un étalonnage de la

fonction $freq = f(T_e)$ plus précis. Afin de tester nos algorithmes sur un autre type d'oscillateur, un noeud a été modifié pour être cadencé par un TCXO. L'oscillateur utilisé est le AST3TQ ([104]) oscillant à 40 MHz. Il remplace l'oscillateur originel sur un kit de développement Spartan 6 (Papilio Pro [97]). La photo en figure 5.4 illustre cette modification. Ce noeud modifié nous a permis de tester les mêmes algorithmes pour la prédiction des paramètres de l'horologe lors du ON/OFF GPS (MHFC, MHFD avec et sans filtre de Kalman, en "temps réel" ou pour la datation a posteriori cf chapitre 4) avec un oscillateur compensé en température.



(a) Carte non modifiée avec SPXO (b) Carte modifiée avec TCXO

FIGURE 5.4 – Montage d'un TCXO sur un dev kit Spartan 6

5.3.1 Prédiction "temps réel"

Un jeu de données TCXO a été obtenu à partir de l'enregistrement des valeurs de compteurs N_θ, N_f, N_Δ et de la date GPS pour chaque seconde (cf chapitre 3) pendant une durée de 7×10^5 secondes, soit un peu plus de 8 jours. Pendant l'expérimentation, l'entrée de capture de ce noeud de capteurs est connectée à un GBF simulant l'apparition d'évènements toutes les 2.34 secondes (intervalle choisi aléatoirement). L'acquisition des données s'est effectuée dans les mêmes conditions que pour le jeu de données SPXO : dans les mêmes locaux et avec la même antenne GPS positionnée sur le toit du bâtiment. Comme dans les chapitres précédents, les dates des événements peuvent être calculées en post-traitement sur le logiciel GNU Octave, à l'aide de l'équation 6.1.

Comme dans le chapitre 4, les deux modèles (MHFC et MHFD) sont utilisés pour prédire

les paramètres de l'horloge τ et θ pendant l'apnée, en post-traitement, dans les modes "temps réel" et "a posteriori", sur tout le jeu de données. Les horodatages sont ensuite calculés et comparés aux horodatages de référence. Les horodatages de référence sont toujours les horodatages calculés avec le récepteur GPS tout le temps ON et le filtre de Kalman (cf chapitre 3). Dans un premier temps, les versions non filtrées sont comparées pour le noeud de capteurs avec SPXO et le noeud avec TCXO. Le paramètre τ et les erreurs sur le paramètre θ (différences par rapport à la référence) sont tracés en figure 6.3b et 5.6 pour $k = 600s$. On observe que l'utilisation du TCXO ne semble pas apporter d'améliorations significatives sur les prédictions de θ ni de τ avec $k = 600s$, par contre la fréquence du TCXO parait beaucoup plus stable sur le long terme. Cette stabilité est illustré en figure 5.7 en comparant la dérive en fréquence d'un SPXO contre un TCXO.

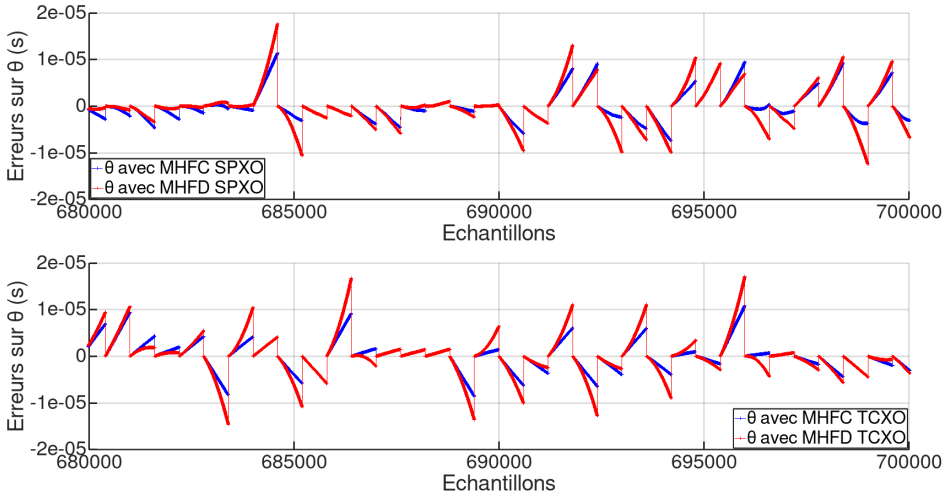


FIGURE 5.5 – Erreur sur l'estimation de θ - SPXO vs TCXO

Comme pour le jeu de données SPXO, les paramètres du filtre de Kalman sont réglés empiriquement à l'aide de tests sur l'innovation car les "vrais" paramètres ne sont pas connus. La distribution de l'innovation obtenue après ajustement des paramètres est représentée en figure 5.8, la distribution n'est pas tout à fait gaussienne mais s'en approche. La moyenne glissante du NIS est représentée en figure 5.9, le filtre ne présente pas de biais. L'autocorrélation de

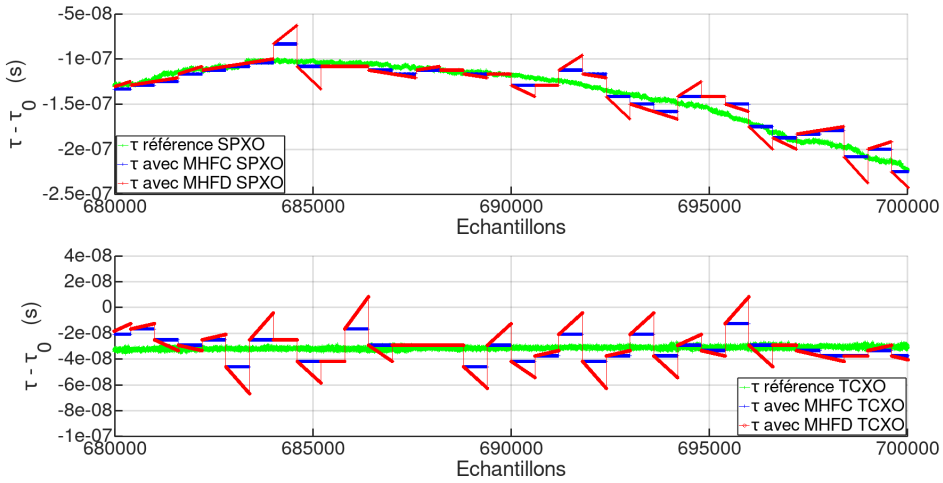


FIGURE 5.6 – Estimation de τ - SPXO vs TCXO

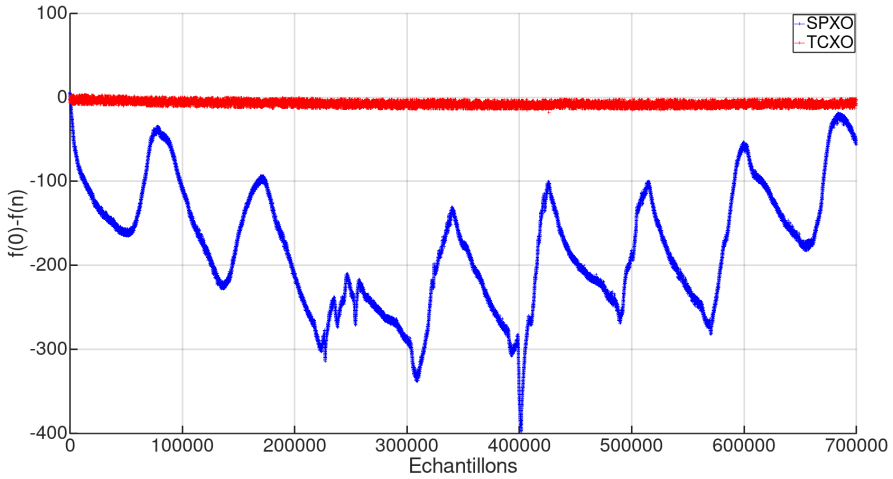


FIGURE 5.7 – Dérive en fréquence - SPXO vs TCXO

l'innovation est représentée en figure 5.10, les innovations semblent indépendantes. À partir de cette analyse des innovations, le filtre est considéré correctement réglé.

En utilisant le filtre avec $k_{on} = 30s$ le jeu de données TCXO est soumis aux mêmes scénarios que précédemment (MHFC et MHFD en temps réel). Les erreurs de prédiction de déphasage pour un cycle de 600 secondes sont illustrées en figure 5.11. En figure 5.12, la fréquence observée et la fréquence prédite par le modèle sont illustrées pour le même cycle de 600 secondes. Ici, les erreurs de déphasage avec le TCXO sont similaires aux erreurs de déphasages avec le

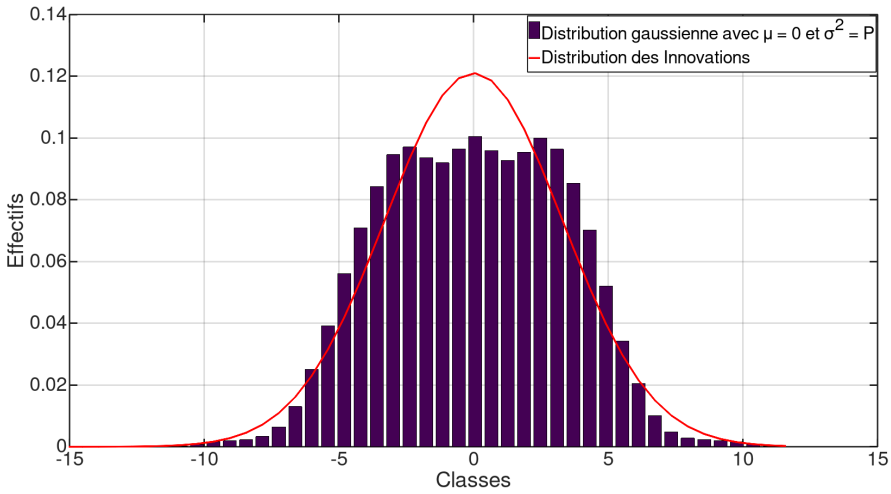


FIGURE 5.8 – Réglage du KF - Distribution des innovations

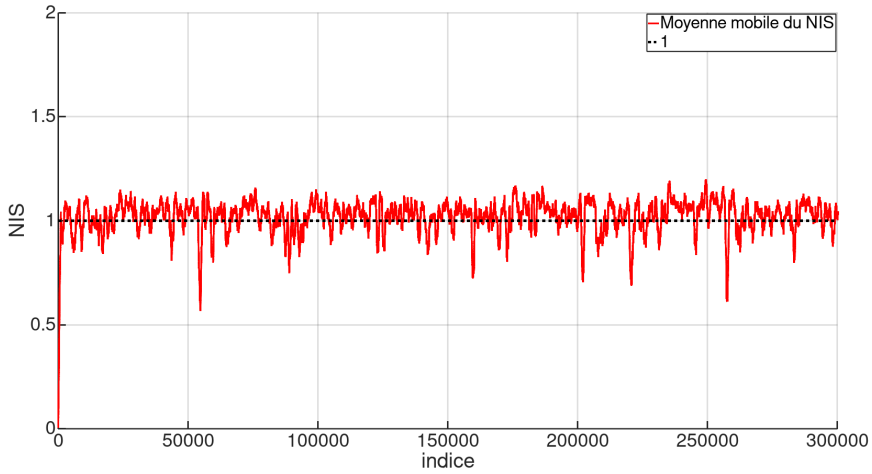


FIGURE 5.9 – Réglage du KF - NIS

SPXO pour le MHFD, elles semblent légèrement inférieures pour le MHFC. Comme pour les prédictions avec le SPXO, le filtrage sur le TCXO permet d'améliorer les prédictions avec les deux modèles.

La durée "ON" pour le filtrage ayant une influence sur la précision de synchronisation et sur le ratio ON/OFF, il faut trouver le bon compromis entre précision et ratio. Les figures 5.13 et 5.14 représentent le ratio en fonction des EAM (Erreur Absolue Moyenne) sur les horodatages des événements pour différentes durées de rallumage, avec le MHFC et le MHFD. Comme sur

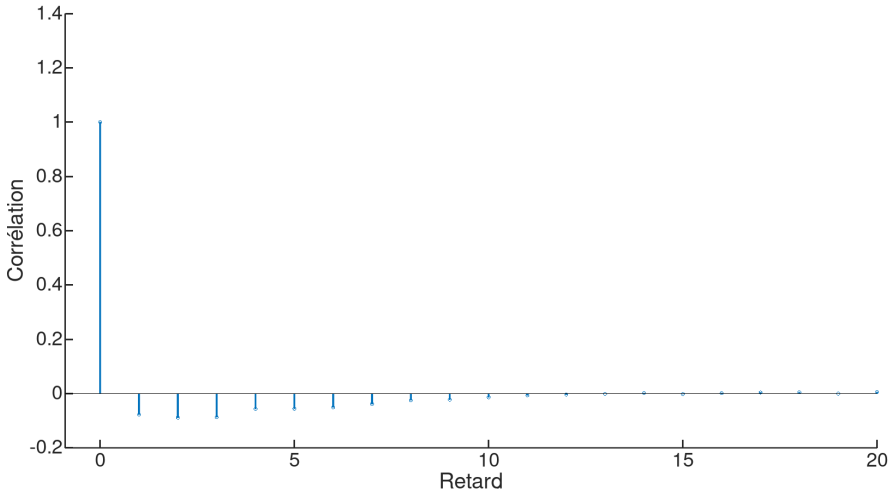


FIGURE 5.10 – Réglage du KF - Autocorrélation des innovations

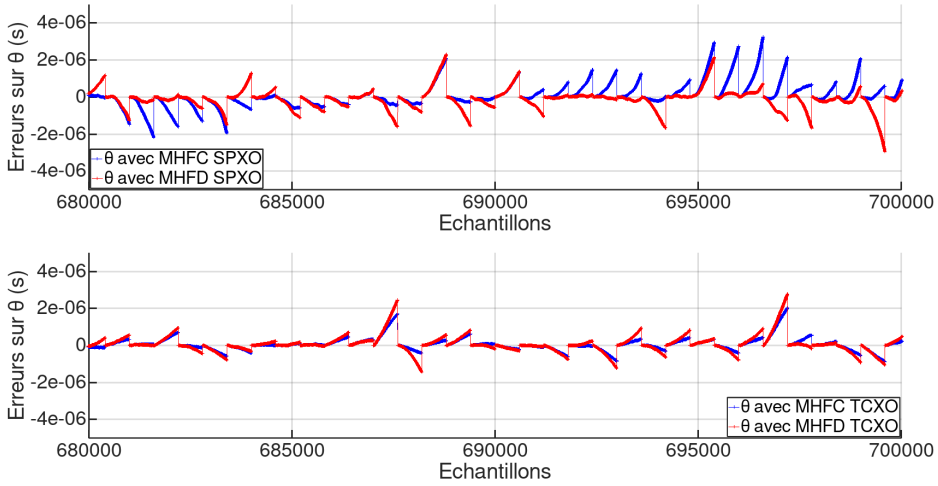


FIGURE 5.11 – Erreur sur l'estimation de θ avec KF - SPXO vs TCXO

les noeuds équipés de SPXO, les meilleurs résultats sont principalement obtenus avec un k_{on} de 2 secondes. Dans le cas du MHFC, le meilleur k_{on} est de 2 secondes pour une EAM supérieure à 11 ns, en dessous de ce seuil la durée "ON" optimale (celle qui permet de minimiser l'EAM) augmente progressivement à mesure qu'on se rapproche de la solution de référence avec le GPS tout le temps allumé. Pour le MHFD, la durée "ON" de deux secondes est optimale à partir d'une EAM de 13 ns. Dans les deux cas, les durées k_{on} supérieures à 5 secondes n'ont pas réellement

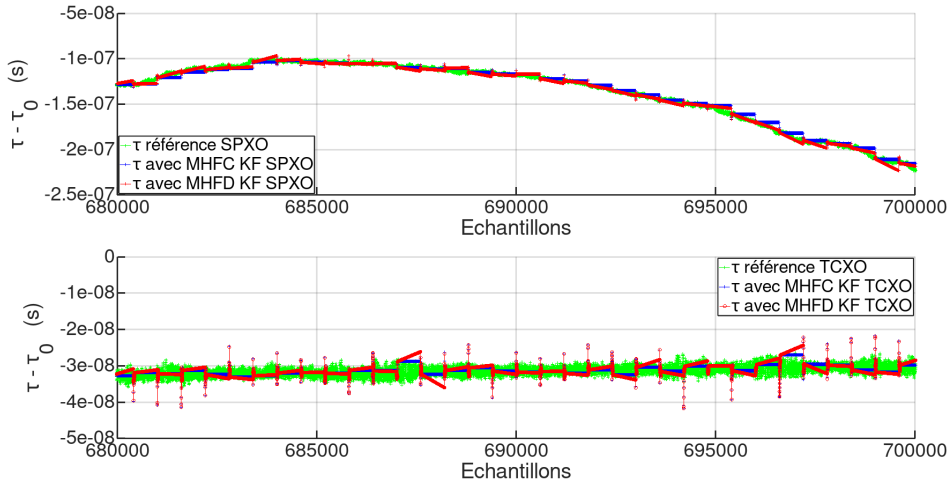


FIGURE 5.12 – Estimation de τ avec KF - SPXO vs TCXO

d'intérêt du point de vue du compromis entre le ratio et la précision.

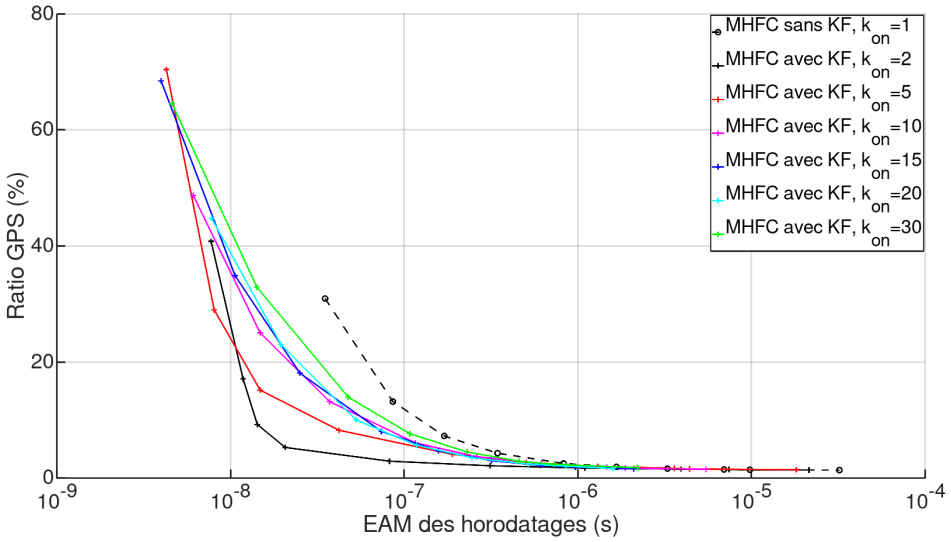


FIGURE 5.13 – Ratio GPS en fonction de l'EAM sur les horodatages - MHFC

Pour évaluer l'intérêt du modèle d'horologe MHFD avec un TCXO, la figure 5.15 illustre les ratios des deux modèles (MHFC et MHFD) en fonction de l'EAM sur les horodatages. On observe que le MHFD ne présente pas d'intérêt du point de vue du compromis entre le ratio et la précision. L'optimum est obtenu avec le MHFC filtré avec un k_{on} de 5 secondes pour une EAM inférieure à la dizaine de nanosecondes et un k_{on} de 2 secondes pour les EAM plus grandes.

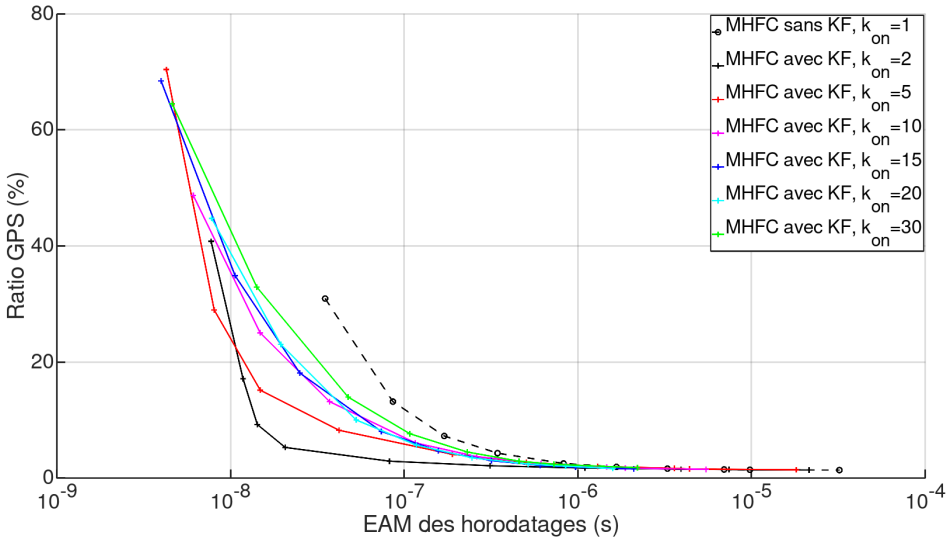


FIGURE 5.14 – Ratio GPS en fonction de l'EAM sur les horodatages - MHFD

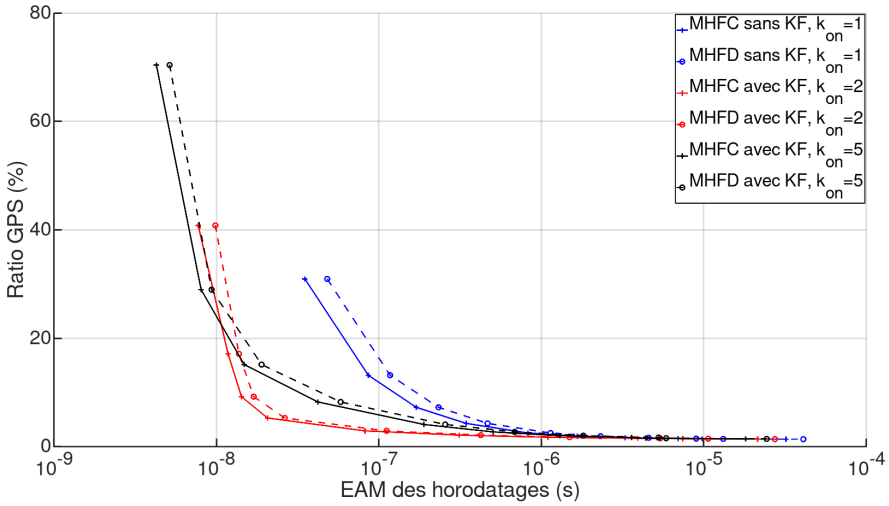


FIGURE 5.15 – Ratio GPS en fonction de l'EAM - MHFC vs MHFD avec et sans KF

Afin de comparer la synchronisation sur le noeud TCXO par rapport aux deux noeuds SPXO utilisés dans le chapitre précédent, les EAM et les erreurs absolues maximales sont calculées sur les trois noeuds de capteurs. Ces erreurs correspondent aux différences entre les horodatages de référence et les horodatages calculés avec l'estimation des paramètres θ et τ pendant les périodes d'apnées. Les EAM des modèles non filtrés sont représentés en figure 5.16. Les EAM des modèles filtrés sont représentés en figure 5.17. Les figures 5.18 et 5.19 illustrent respecti-

vement, les erreurs absolues maximales des modèles non filtrés et filtrés. Sans le filtrage, les résultats sont plutôt similaires entre les noeuds SPXO et le noeud TCXO. Avec le filtrage, le noeud TCXO permet un meilleur ratio à précision de synchronisation équivalente au-delà de la dizaine de nanoseconde d'EAM. Cette amélioration est surtout significative entre la dizaine de nanoseconde à la microseconde, avec une différence maximale d'un peu moins de 5% (8% par rapport au deuxième SPXO) sur le ratio aux alentours des 25 nanosecondes. Cette amélioration est aussi visible sur l'erreur maximale, même si elle est moins marquée.

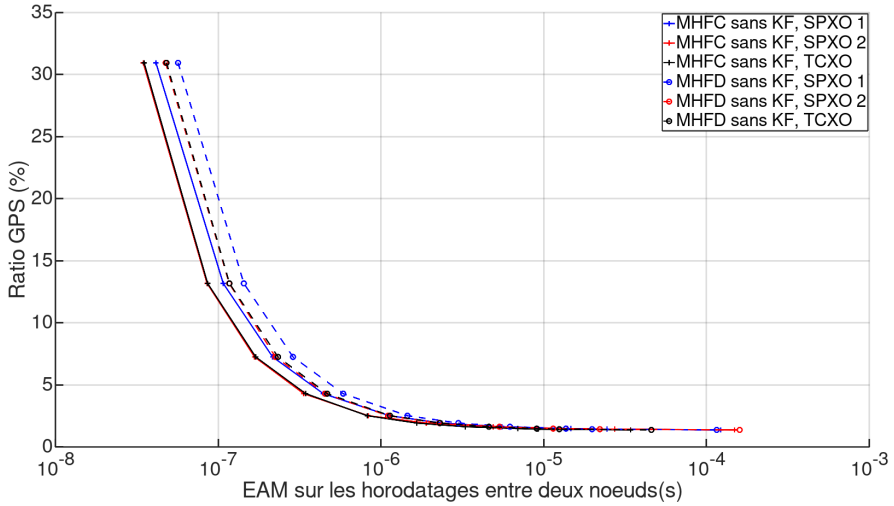


FIGURE 5.16 – Ratio GPS en fonction des EAM - SPXO vs TCXO

Les EAM, erreurs maximales et ratios en fonction de la durée du cycle k pour les MHFC filtrés sont regroupés dans le tableau 5.5. Le noeud TCXO est nommé N3 et les noeuds SPXO sont nommés N1 et N2. Ce tableau représente aussi la puissance moyenne théorique nécessaire au GPS pour assurer la synchronisation en se basant sur les 120 mW en fonctionnement tout le temps allumé [77] et l'équation 4.1 (cf chapitre 4). Si les performances des trois noeuds sont similaires pour les cycles courts, le TCXO se démarque à partir des cycles supérieurs à 50 secondes et il devance les SPXO d'un facteur 10 en matière de précision de synchronisation à 7000 secondes. Cette supériorité du TCXO sur des cycles plus longs est logique car ce sont sur ces cycles que les fréquences des SPXO dérivent tandis que la fréquence du TCXO reste stable.

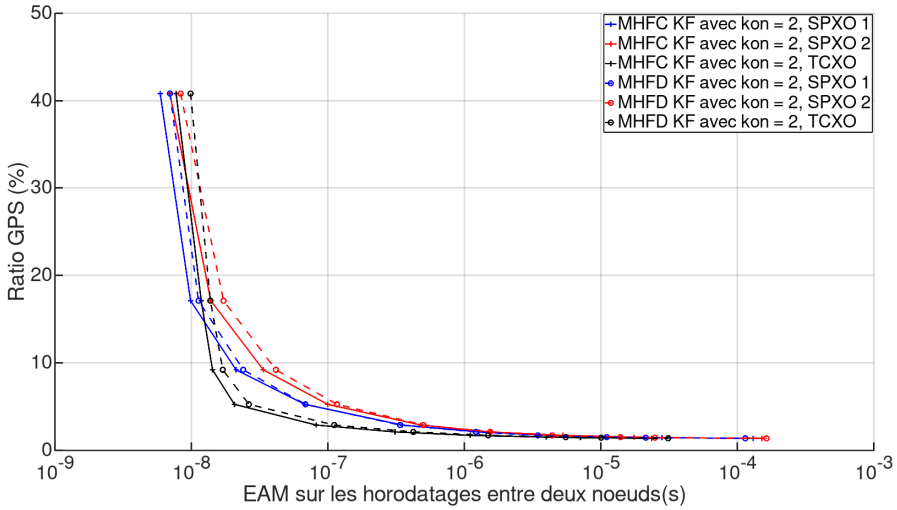


FIGURE 5.17 – Ratio GPS en fonction des EAM - SPXO vs TCXO avec KF

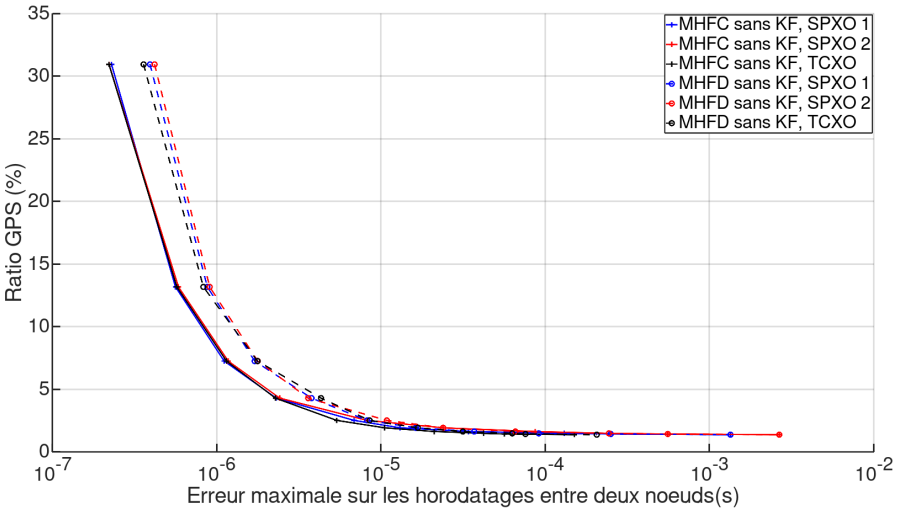


FIGURE 5.18 – Ratio GPS en fonction des Erreurs maximales - SPXO vs TCXO

Cependant, du fait de la croissance exponentielle des erreurs de synchronisation en fonction de la durée du cycle, les gains du TCXO par rapport au SPXO sont faibles au-delà d'une EAM de l'ordre de la microseconde. Le TCXO permet une réduction maximale de la puissance théorique moyenne requise par le récepteur GPS d'environ 6 mW (ou 9 mW par rapport au deuxième SPXO) vers les 25 ns d'EAM. Toutefois, il faut noter que les essais ont été effectués dans un environnement non contrôlé en température mais aussi non soumis à de brusques variations de

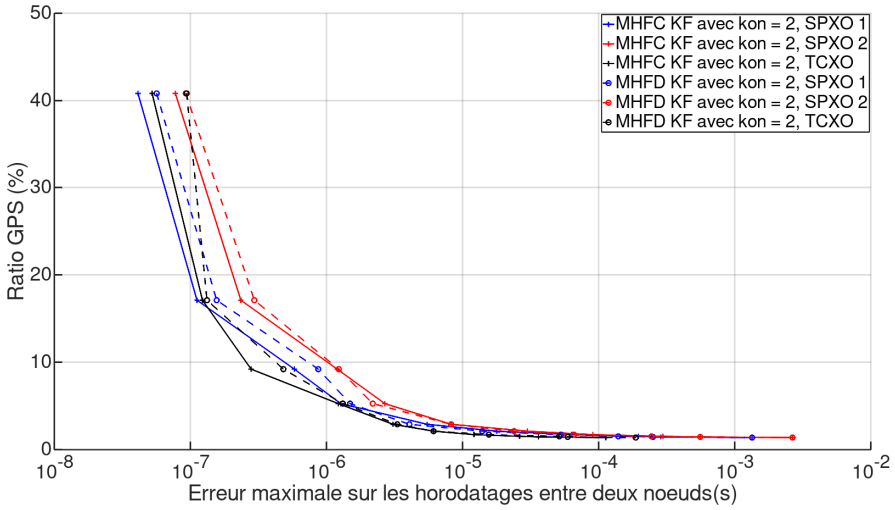


FIGURE 5.19 – Ratio GPS en fonction des Erreurs maximales - SPXO vs TCXO avec KF

température. Dans de telles conditions, le TCXO pourrait potentiellement apporter des gains supérieurs sur la précision de synchronisation.

k	EAM N1 (s)	EAM N2 (s)	EAM N3 (s)	Max N1 (s)	Max N2 (s)	Max N3 (s)	ratio (%)	puissance (mW)
10	5.90E-09	6.96E-09	7.72E-09	4.11E-08	7.75E-08	5.23E-08	40.80	48.96
25	9.85E-09	1.39E-08	1.18E-08	1.12E-07	2.35E-07	1.23E-07	17.12	20.54
50	2.12E-08	3.36E-08	1.43E-08	5.80E-07	1.20E-06	2.79E-07	9.23	11.07
100	6.76E-08	9.96E-08	2.06E-08	1.27E-06	2.68E-06	1.22E-06	5.28	6.34
250	3.50E-07	4.79E-07	8.21E-08	5.48E-06	8.24E-06	3.08E-06	2.91	3.49
500	1.28E-06	1.56E-06	3.10E-07	1.78E-05	3.00E-05	6.11E-06	2.12	2.55
1000	4.30E-06	5.26E-06	1.10E-06	6.34E-05	9.06E-05	1.21E-05	1.73	2.07
2000	1.40E-05	1.75E-05	3.98E-06	1.96E-04	2.97E-04	2.62E-05	1.53	1.84
3000	2.80E-05	3.19E-05	7.09E-06	2.86E-04	5.57E-04	3.61E-05	1.46	1.76
7000	1.30E-04	1.51E-04	2.36E-05	1.34E-03	2.66E-03	1.13E-04	1.39	1.67

TABLE 5.5 – Erreurs sur les horodatages - SPXO vs TCXO

5.3.2 Prédiction a posteriori

Dans la section précédente les améliorations apportées par l'utilisation d'un TCXO pour les prédictions pendant l'apnée en "temps réel" ont été quantifiées. Dans cette section les amé-

liorations sur les prédictions a posteriori de la période d'apnée (cf chapitre 4) sont évaluées. Comme dans la section précédente, les erreurs de synchronisations d'un noeud correspondent aux différences entre les horodatages calculés avec les estimations des paramètres τ et θ pendant les périodes d'apnées et les horodatages de référence. Afin de déterminer si l'utilisation du filtre de Kalman pendant les périodes "ON" est utile, le ratio et l'EAM des horodatages sont calculés pour plusieurs durées "ON". Ces ratios et EAM sont illustrés en figure 5.20 pour le MHFC et en figure 5.21 pour le MHFD.

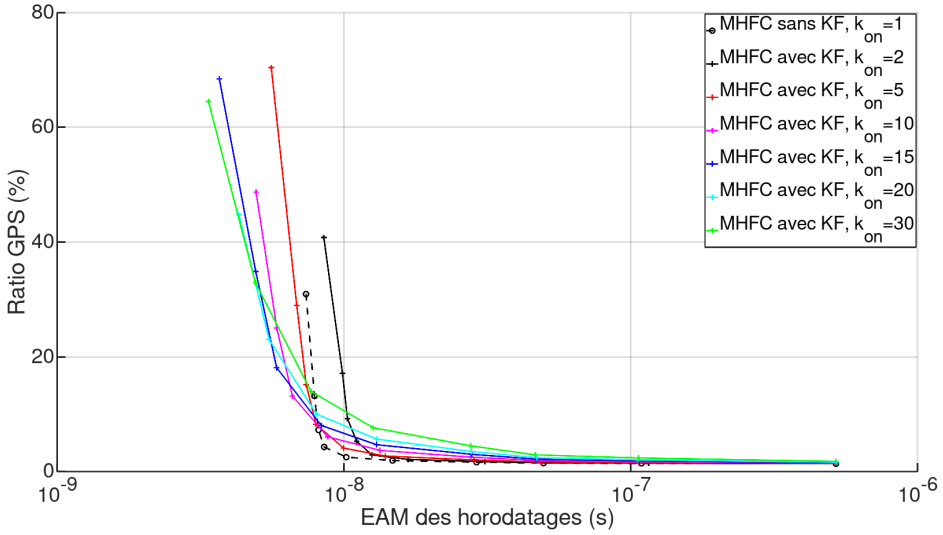


FIGURE 5.20 – Ratio GPS en fonction des EAM - MHFC avec KF a posteriori

De la même manière que pour les SPXO, le filtrage n'apporte pas vraiment d'améliorations du point de vue du compromis entre ratio et précision de synchronisation pour la datation a posteriori avec le MHFC. Le filtrage permet uniquement des gains pour des EAM inférieures à 8 nanosecondes. Dans le cas du MHFD par contre, le filtrage est utile jusqu'à une EAM de 3 microsecondes. La durée "ON" optimale est de 15 secondes. Pour pouvoir juger des caractéristiques du MHFD par rapport au MHFC quand on prend le ratio en compte, les EAM des deux modèles sont tracé en figure 5.22. Il apparaît que le MHFC offre le meilleur ratio en fonction des EAM.

Finalement, le ratio, l'EAM et l'erreur maximale absolue sont calculés pour les trois noeuds et respectivement tracés en figure 5.23 et 5.24. Sur tous les noeuds, le ratio décroît de façon

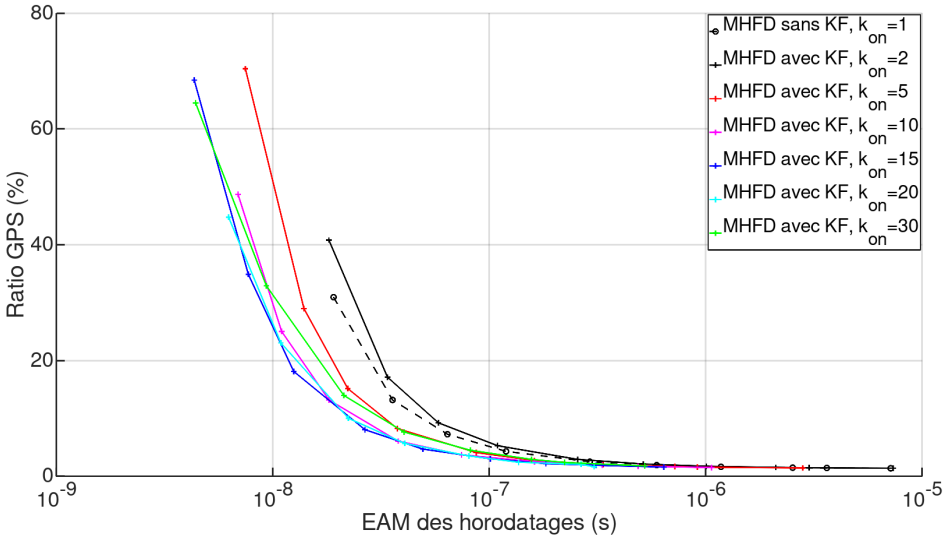


FIGURE 5.21 – Ratio GPS en fonction des EAM - MHFD avec KF a posteriori

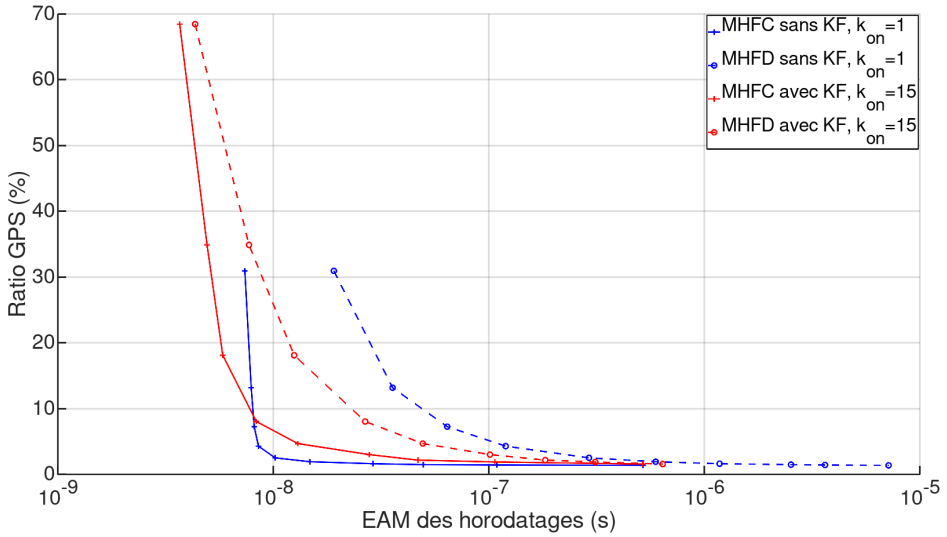


FIGURE 5.22 – Ratio GPS en fonction des EAM - MHFC vs MHFD a posteriori

exponentielle avec l'augmentation de l'AME pour tendre vers le ratio limite de 1.33% (cf chapitre 3). Le TCXO permet un meilleur ratio en fonction de l'AME ou des erreurs maximales de synchronisation. Comme dans le cas "temps réel", les améliorations sur la précision de synchronisation apportées par l'utilisation d'un TCXO sont surtout significatives sur une plage d'EAM allant de 8 nanosecondes à 0.5 microsecondes. Les EAM, valeurs maximales, ratios ainsi

que les puissances moyennes théoriques nécessaire au récepteur GPS sont répertoriés dans le tableau 5.6. En comparaison du SPXO, le TCXO offre une précision beaucoup plus importante, principalement pour la datation "a posteriori" avec une erreur maximale qui ne bouge pas jusqu'à un cycle de 250 secondes. Il faut par contre prendre en considération le fait que ces erreurs sont calculés uniquement pour un noeud de capteur, il faut donc les doubler pour obtenir une erreur maximale théorique entre deux noeuds.

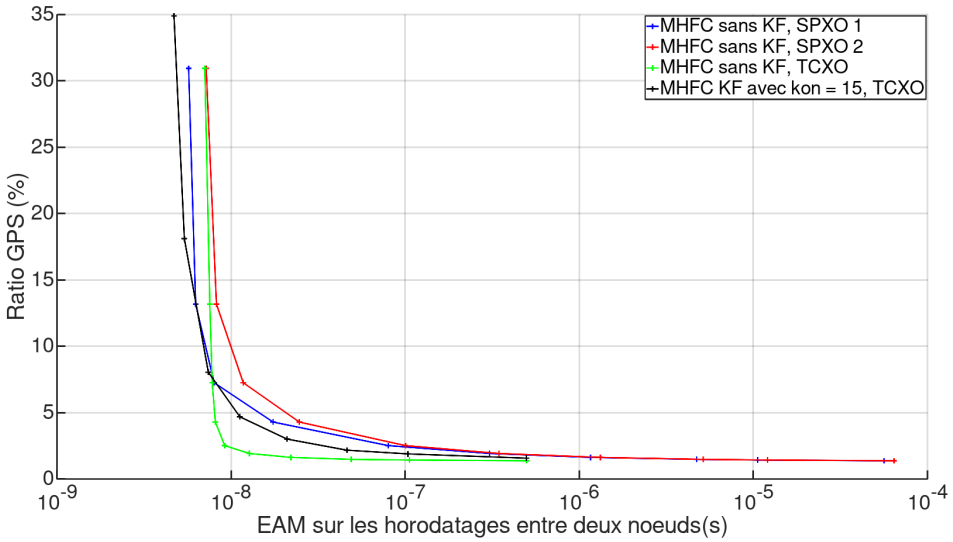


FIGURE 5.23 – Ratio GPS en fonction des EAM - SPXO vs TCXO a posteriori

k	EAM N1 (s)	EAM N2 (s)	EAM N3 (s)	Max N1 (s)	Max N2 (s)	Max N3 (s)	ratio (%)	puissance (mW)
10	5.70E-09	7.19E-09	7.06E-09	4.14E-08	3.72E-08	4.55E-08	30.93	37.12
25	6.26E-09	8.24E-09	7.55E-09	4.46E-08	5.63E-08	4.78E-08	13.17	15.81
50	7.99E-09	1.17E-08	7.79E-09	6.11E-08	1.66E-07	4.65E-08	7.25	8.70
100	1.74E-08	2.46E-08	8.12E-09	2.48E-07	3.24E-07	4.73E-08	4.29	5.15
250	7.99E-08	1.00E-07	9.19E-09	7.76E-07	2.35E-06	4.34E-08	2.52	3.02
500	3.05E-07	3.46E-07	1.27E-08	3.01E-06	5.31E-06	7.02E-08	1.92	2.31
1000	1.16E-06	1.32E-06	2.21E-08	1.12E-05	1.96E-05	1.26E-07	1.63	1.95
2000	4.72E-06	5.15E-06	4.90E-08	3.93E-05	5.85E-05	2.52E-07	1.48	1.78
3000	1.06E-05	1.21E-05	1.06E-07	8.73E-05	1.43E-04	3.56E-07	1.43	1.72
7000	5.63E-05	6.42E-05	4.98E-07	3.24E-04	5.81E-04	1.39E-06	1.38	1.65

TABLE 5.6 – Erreurs sur les horodatages - SPXO vs TCXO a posteriori

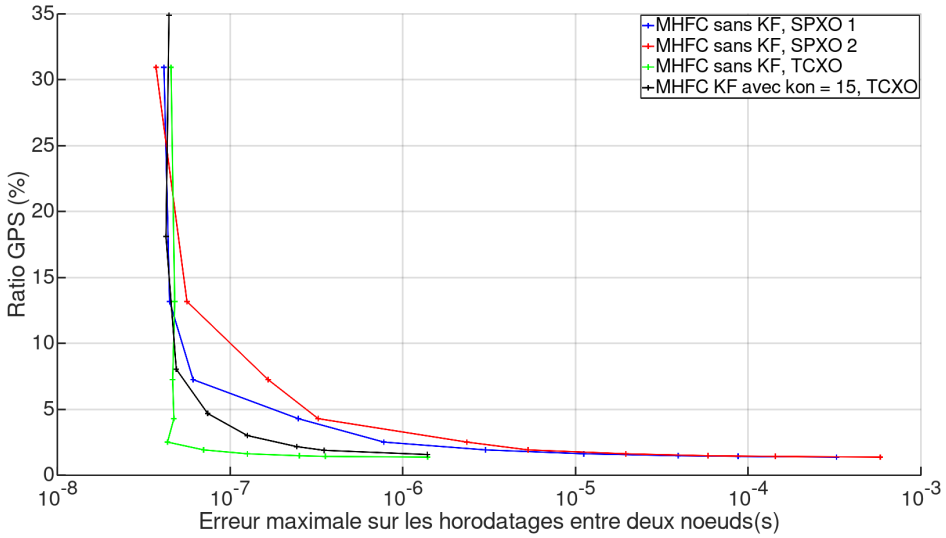


FIGURE 5.24 – Ratio GPS en fonction des Erreurs maximales - SPXO vs TCXO a posteriori

A partir de ces données, on observe que les bénéfices des oscillateurs TCXO en matière de ratio GPS, pour notre cas d'application, apparaissent pour des cycles plutôt courts (jusqu'à 250 secondes). Dans le mode "a posteriori", les erreurs de synchronisation maximales sur un noeud sont similaires entre les deux types d'oscillateurs ($\approx$ 50 ns) quand les apnées durent 10 fois plus longtemps avec le TCXO (25 s contre 250 s). Cette différence dans les durées d'apnée permet à la solution TCXO une consommation GPS théorique de 3.02 mW contre les 15.81 mW de la solution SPXO. Les consommations des deux types d'oscillateurs sont similaires d'après leurs spécifications [104], [105], la solution TCXO est donc, en théorie, toujours plus rentable que la solution SPXO en matière de consommation énergétique. Les expérimentations ont été effectuées en laboratoire avec une faible dynamique de la température. Même s'il a été démontré que cette dynamique est suffisante pour impacter la fréquence (cf section 5.1), les avantages des oscillateurs TCXO pourraient être plus importants si les noeuds de capteurs étaient soumis à des variations plus importantes de la température pendant les périodes d'apnée.

5.4 Conclusion

Dans ce chapitre le filtre de Kalman a été modifié pour estimer les paramètres du modèle d'horloge à partir de la mesure de la température pendant l'extinction du GPS. Cependant l'utilisation de la température n'a pas montré d'amélioration des estimations sur les cycles inférieurs à deux heures comparées aux méthodes d'estimations basées sur les MHFC et MHFD. Trois hypothèses peuvent expliquer celà : la mesure de la température n'était pas assez précise du fait de la résolution du capteur et de son emplacement sur le pcb et non sur le cristal de quartz; le modèle reliant la fréquence à la température n'est pas assez complet pour être utile; les noeuds n'ont pas été soumis à des variations de températures suffisamment rapides par rapport à la durée des cycles pour que la mesure de température ai un intérêt. Dans un second temps, un noeud a été modifié pour recevoir un oscillateur compensé en température. Il a été démontré que l'utilisation d'un oscillateur compensé en température permet une amélioration de la précision des horodatages à ratio équivalent, principalement pour la datation "a posteriori", ou la puissance théorique est divisée par 5 pour les erreurs maximales inférieures à 100 ns.

CHAPITRE 6

Implémentation d'un noeud à faible consommation

DANS les deux derniers chapitres, il a été démontré expérimentalement que le protocole de synchronisation adaptatif permet de diminuer la consommation du récepteur GPS. Cependant, ces résultats ont été obtenus avec un prototype basé sur un FPGA Spartan 6, qui consomme beaucoup d'énergie pour un noeud alimenté par batterie. De plus, l'implémentation n'utilise que 14 % des ressources (LUTs) du Spartan 6, une cible plus *petite*, au sens moins de blocs logiques, et plus économe en énergie peut donc être utilisée pour l'implémentation de l'unité d'horodatage. Une solution plus économe en énergie, basée sur un FPGA Lattice Ice 40 [106] à faible consommation est développée et évaluée dans ce chapitre.

Sommaire

6.1 Description de l'architecture

6.1.1 Architecture du noeud de capteurs

Comme dans le chapitre 3, le noeud est constitué de trois parties : l'unité de contrôle principale, l'unité d'horodatage et le récepteur GPS. Le récepteur GPS est toujours le Ublox NEO6T [77] et l'unité de contrôle principale est toujours implémentée sur une carte Raspberry Pi 3. L'unité d'horodatage est, à présent, implémentée sur un kit de développement TinyFPGA BX [107] basé sur le FPGA à faible consommation Ice40 de chez Lattice. Le but de cette architecture est d'appliquer des cycles ON/OFF au récepteur GPS et de la datation "en ligne" contrairement à l'architecture précédemment utilisée pour enregistrer des jeux de données et les exploiter hors ligne. Pour cela, le fonctionnement du noeud est différent, ici le noeud est asservi au signal PPS et le faux signal PPS est généré uniquement pendant les périodes d'apnée. Cette méthode permet de simplifier le calcul des horodatages mais présente l'inconvénient de ne plus permettre le filtrage du déphasage. Dans un premier temps, l'objectif est d'étudier les performances d'une architecture *simple*, l'architecture de l'unité d'horodatage est donc basée sur la datation a posteriori sans filtrage. Le principe est le suivant : l'unité de contrôle principale communique avec le récepteur GPS via une liaison série pour pouvoir le placer en mode "extinction" (tout est éteint sur le récepteur sauf la RTC et la mémoire) ou le rallumer ; l'unité de contrôle communique avec le FPGA via une liaison SPI pour pouvoir le placer en mode GPS

ou en mode apnée (i.e. GPS éteint). L'unité d'horodatage enregistre les horodatages "brut" des événements à l'aide de ces compteurs qui sont remis à zéro soit par le vrai signal PPS soit par le faux signal PPS suivant le mode. L'unité d'horodatage génère une interruption synchronisée sur le signal vrai/faux PPS pour que l'unité de contrôle principale récupère la fréquence et les événements. Ce principe est illustré sur le schéma en figure 6.1.

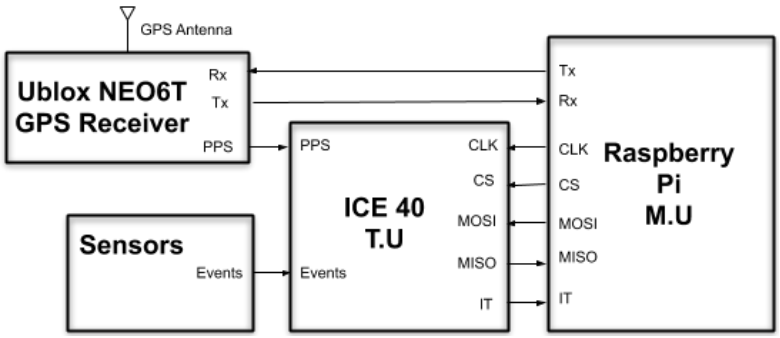


FIGURE 6.1 – Principe de fonctionnement du noeud ICE 40

Comme pour l'architecture précédente, la division entre l'unité d'horodatage et l'unité centrale présente plusieurs avantages. Premièrement, une implémentation matérielle permet de supprimer les overheads logiciels et de rendre déterministe le processus d'horodatage des signaux. Deuxièmement, la délocalisation des fonctions critiques, d'un point de vue temporel, à l'unité d'horodatage permet de relaxer les contraintes temps réel sur l'unité principale et donc simplifie la partie applicative. Enfin, l'implémentation matérielle permet de contrôler plus finement les aspects de l'horodatage tel que le nombre de voies d'acquisition (limité lors de l'implémentation par la taille de la cible) et l'erreur de granularité temporelle en ajustant la fréquence du FPGA.

Pour ce prototype, le fonctionnement de l'unité de contrôle principale, implémentée en python sur la carte Raspberry Pi, est relativement simple : une fonction de rappel est déclenchée par un signal d'interruption venant de l'unité d'horodatage après chaque front de PPS (qu'il soit vrai ou faux). Chaque fois que cette fonction est exécutée, les horodatages et la fréquence mesurée lors de la dernière seconde écoulée sont récupérés par l'unité d'horodatage

à l'aide de lectures sur le bus SPI. Une fois ces données récupérées, l'unité centrale décide de changer ou non de mode (GPS/apnée). Cette décision est rendue possible par une variable statique permettant de compter le nombre d'appels de la fonction. On définit un cycle de durée k secondes comme étant un mode GPS d'une durée de k_{on} secondes suivit d'un mode dit APNEE. Si le compteur dépasse k_{on}, le mode est changé de GPS à APNEE, la valeur de la fréquence à prendre en compte pendant la période d'apnée est transmise à l'unité d'horodatage par une écriture sur le bus SPI et le récepteur GPS est mis en mode *extinction* par l'unité centrale. Quand le compteur atteint k, il est remis à zéro, le mode change de APNEE à GPS et le récepteur GPS est rallumé par l'unité centrale.

6.1.2 Architecture de l'unité d'horodatage

Le coeur de l'unité d'horodatage reste le même que pour l'architecture précédente : des compteurs permettent de mesurer les délais entre le signal PPS et les événements tels qu'un front sur une entrée numérique ou l'arrivée d'un échantilllon issu d'un CAN. A la différence de l'architecture précédente, le PPS sortant du récepteur GPS est utilisé dans le mode GPS (comme *vrai* PPS) et le *faux* signal PPS est généré, en interne, de sorte à être dans la continuité du vrai signal PPS au basculement dans le mode APNEE. Ce faux signal PPS est généré à partir du dernier front du vrai signal PPS et de la fréquence transmise par l'unité de contrôle. De par ce mode de fonctionnement, le déphasage n'a plus besoin d'être mesuré, par contre la fréquence réelle est toujours mesurée en comptant le nombre de coups d'horloges entre deux fronts du signal PPS. Cette fréquence est lue toutes les secondes par l'unité de contrôle. L'oscillateur utilisé dans le cadre de cet essai, est un TCXO à 10 MHz et une PLL est utilisée pour pouvoir monter en fréquence et diminuer l'erreur de quantification.

Afin de communiquer avec l'unité de contrôle, l'unité d'horodatage lui présente trois registres adressables via la liaison SPI : le premier permet de passer en mode GPS ; le second permet de passer en mode APNEE et d'écrire, sur 4 octets, la fréquence à utiliser pendant ce mode ; une lecture sur la dernière adresse déclenche une transaction de 8 octets comprenant

la fréquence et l'horodatage brut (au sens valeur de compteur et non horodatage temporel) de l'événement. L'unité d'horodatage possède aussi une sortie d'interruption permettant de signaler à l'unité de contrôle que la seconde est écoulée et que les données peuvent être lues. Etant donné qu'il s'agit d'un prototype, cette architecture permet d'horodater seulement un événement par seconde et cette architecture n'offre qu'une seule voie d'acquisition. Pour avoir plus d'événements par seconde, il conviendrait d'ajouter des FIFOs permettant de bufferiser ces événements. Pour ajouter des voies d'acquisition, il conviendrait de dupliquer l'ensemble compteur/FIFO, d'ajouter un registre adressable via la liaison SPI pour pouvoir récupérer le nombre d'événements à lire par l'unité centrale et multiplexer les FIFOS pour la lecture des données. La figure 6.2 représente cette architecture.

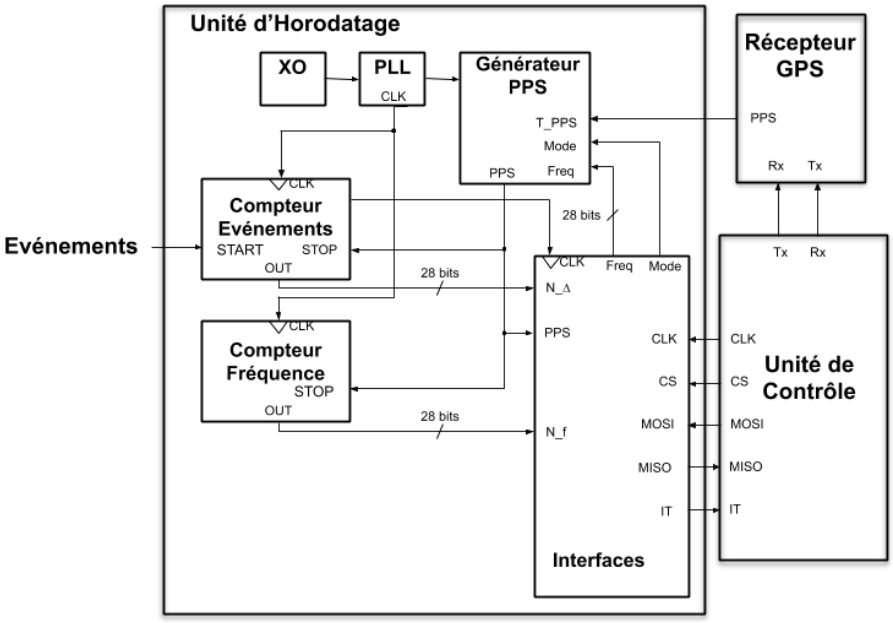


FIGURE 6.2 – Principe de fonctionnement de l'unité d'horodatage

Le calcul des horodatages est trivial dans le mode GPS du fait de l'utilisation du vrai signal PPS comme référence :

$$t[n] = t_{pps}[n] + \frac{\Delta[n]}{N_f[n]} \tag{6.1}$$

Dans le mode APNEE la même équation peut être utilisée pour calculer les horodatages, mais la fréquence n'est plus mesurée. Cette fréquence est fixée par l'unité de contrôle et correspond à la dernière fréquence mesurée à l'aide du vrai PPS issu du récepteur GPS. Étant donné que les erreurs sur la fréquence vont s'accumuler (cf chapitre 3) du fait de l'erreur de départ sur la mesure de fréquence et des variations de cette fréquence pendant la duré de l'apnée, la datation a posteriori est choisie avec un modèle d'horloge à fréquence constante (MHFC voir chapitre 4). Dans ce mode de datation, la fréquence du dernier cycle est évaluée lors du rallumage du récepteur GPS et ensuite les horodatages de tout le cycle sont recalculés. Les i horodatages entre le début du cycle à n et la fin du cycle à $n + k$ peuvent être calculés de la façon suivante :

$$t[n+i] = t_{pps}[n] + i + \frac{\Delta[n]}{N_f[n]} + i * u \quad \text{avec} \quad u = \frac{\theta[n+k]}{k} \tag{6.2}$$

Le déphasage au rallumage du récepteur GPS $\theta[k+n]$ peut être obtenu à l'aide de la dernière fréquence enregistrée dans le mode APNEE :

$$\theta[n+k] = \frac{N_f[n+k] - N_f[n+k-1]}{N_f[n+k+1]} \tag{6.3}$$

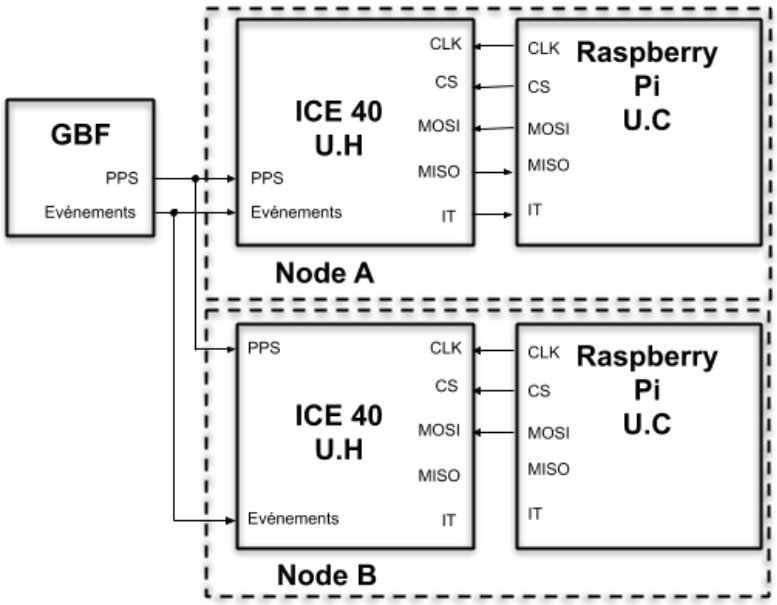
Lors du rallumage, il faut aussi prendre en compte le fait que N_f comprend le déphasage du fait du recalage sur le vrai signal PPS. Ainsi le calcul des horodatages au moment du rallumage est effectué à l'aide de l'équation suivante :

$$t[n+i] = t_{pps}[n] + i + \frac{\Delta[n]}{N_f[n]}(1 - \theta[n+k]) + i * u \tag{6.4}$$

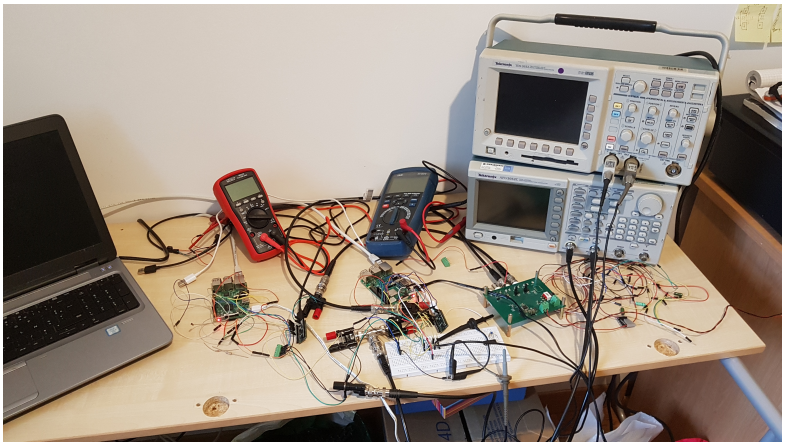
6.2 Essais et résultats

Afin de valider l'architecture, deux noeuds de capteurs prototypes sont utilisés. Les deux noeuds ont leur entrées connectées aux mêmes événements générés par un GBF. Les événements sont espacés de 2.34 secondes comme lors des expérimentations précédentes (cf chapitre 2). Pour cet essai, le signal PPS des deux noeuds est aussi simulé avec un GBF. Le banc d'essai est illustré en figure 6.3. Contrairement au chapitres précédents, seulement quatre périodes d'apnée sont testées et évaluées car chaque période nécessite sa propre acquisition de données : 10 secondes, 250 secondes, 1000 secondes et 7000 secondes. Pour chacune de ces périodes, les unités de contrôle des deux noeuds alternent les deux modes GPS et APNEE indépendemment l'une de l'autre. Les variables t_{pps}, Δ et N_f sont enregistrées sur les deux unités de contrôle pour ensuite pouvoir calculer et comparer les horodatages issus des deux noeuds sur le logiciel de calcul numérique GNU Octave. Même si cette architecture est conçue pour être utilisée en temps réel, un logiciel de calcul numérique est utilisé hors ligne pour simplifier l'analyse des résultats. Dans le cas d'une application réelle ces calculs peuvent être réalisés en temps réel directement par les unités de contrôle.

Comme dans les résultats illustrés précédemment, les erreurs d'horodatages correspondent aux différences entre les horodatages des deux noeuds pour un même événement. Ces erreurs sont représentées en figure 6.4 pour la durée d'apnée de 10 secondes. L'erreur sur les horodatages sans la prise en compte du déphasage, c'est à dire en utilisant uniquement l'équation 6.1, ainsi que le déphasage calculé a posteriori à l'aide de l'équation 6.3 sont aussi représentés sur cette figure. On observe que malgré la faible durée d'apnée, les erreurs sans la prise en compte



(a) Schéma bloc



(b) Photo du banc

FIGURE 6.3 – Banc de test

du déphasage sont jusqu'à 5 fois plus grandes. Comme vu dans la section précédente, cette erreur est due à deux facteurs : la fréquence évolue légèrement pendant l'apnée, en prenant en compte le déphasage (et donc la fréquence au rallumage), cette évolution est estimée ; La fréquence enregistrée est une approximation de la fréquence réelle du fait de la granularité des compteurs qui ne comptent que des valeurs entières. Pour une durée d'apnée faible de 10

secondes, on observe que la fréquence mesurée évolue peu entre le début et la fin de l'apnée, l'erreur est principalement due aux erreurs de granularité des compteurs.

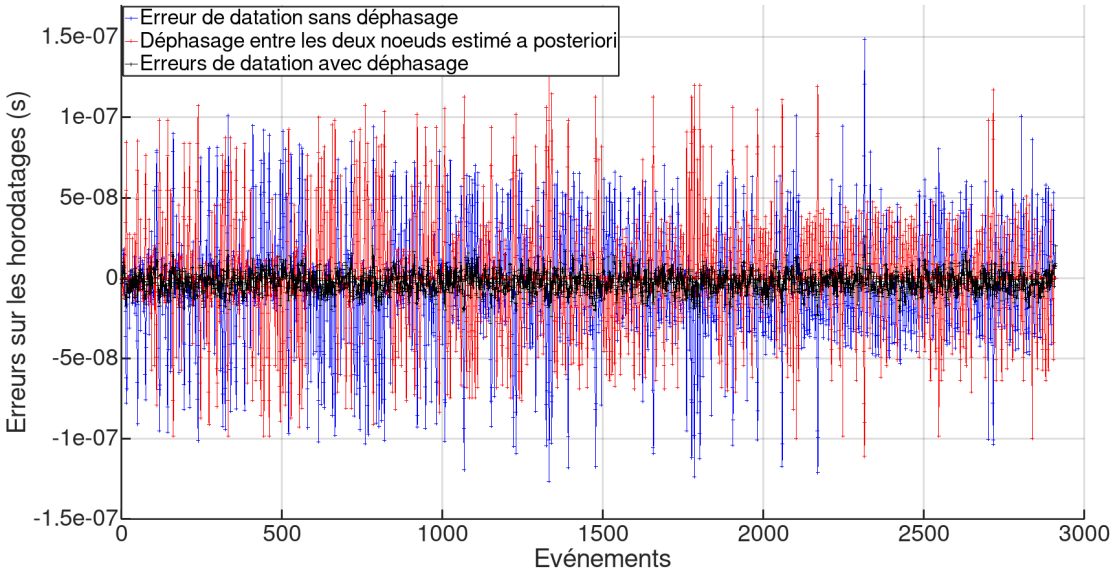


FIGURE 6.4 – Erreur sur les horodatages - Apnée de 10 secondes

Les mêmes données sont tracées pour des durées d'apnée de 250, 1000 et 7000 secondes, en figure 6.5, 6.6 et 6.7, respectivement. On observe des différences encore plus grandes entre les versions sans prise en compte du déphasage et les versions avec, du fait de l'évolution plus importante de la fréquence pendant les périodes d'apnée plus longues. A partir de durées d'apnées de 1000 secondes, les limites du modèle d'évolution linéaire du déphasage commencent à apparaître en comparant le déphasage estimé et les erreurs des datations non corrigées en déphasage. Les erreurs sont principalement dues au fait que la fréquence n'est pas exactement constante pendant la période d'apnée et donc que l'évolution du déphasage n'est pas exactement linéaire.

Les statistiques de ces erreurs sont regroupées dans le tableau 6.1. Malgré une erreur de quantification plus importante, Les erreurs sont du même ordre de grandeur que les erreurs obtenues avec le SPARTAN 6 (cadencé à 240MHz par un TCXO et avec une datation a posteriori) reportées dans le chapitre précédent : entre 91 ns et 2.78 μ s pour une durée d'apnée de 10

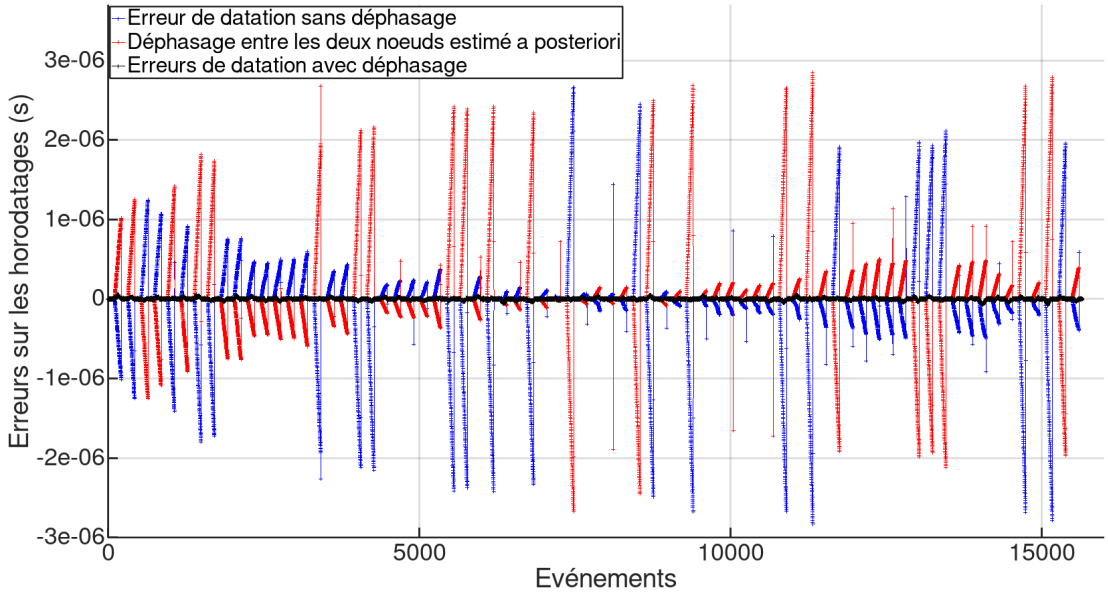


FIGURE 6.5 – Erreur sur les horodatages - Apnée de 250 secondes

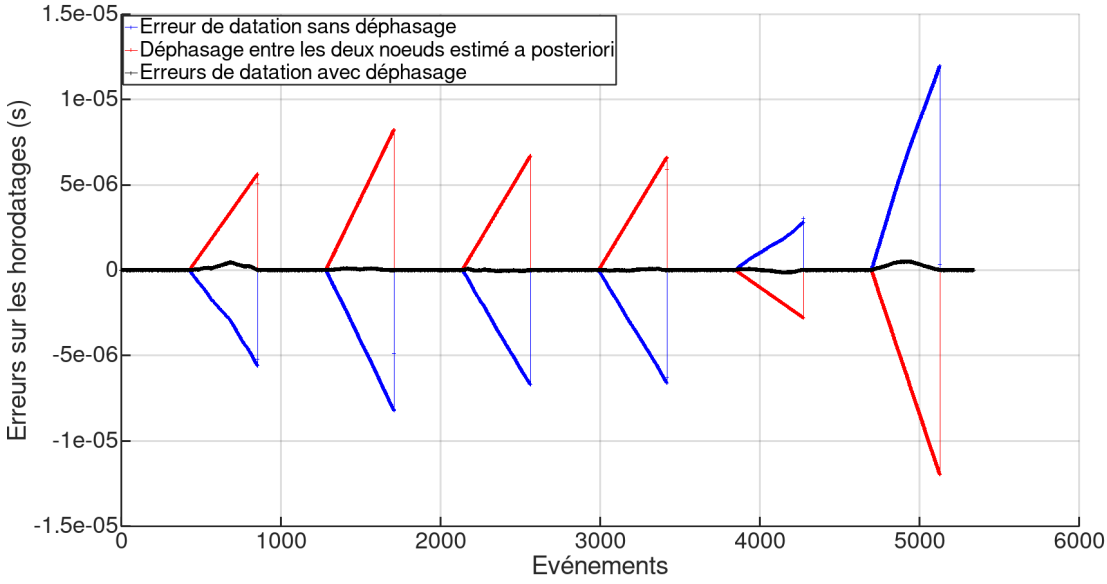


FIGURE 6.6 – Erreur sur les horodatages - Apnée de 1000 secondes

secondes à 7000 secondes. Cependant cette comparaison est approximative en raisons des différences entre les deux expérimentations : l'erreur due à la gigue sur le signal PPS n'est pas présente ici, car le PPS est généré par un GBF ; les erreurs mesurées sur le Spartan 6 TCXO ne

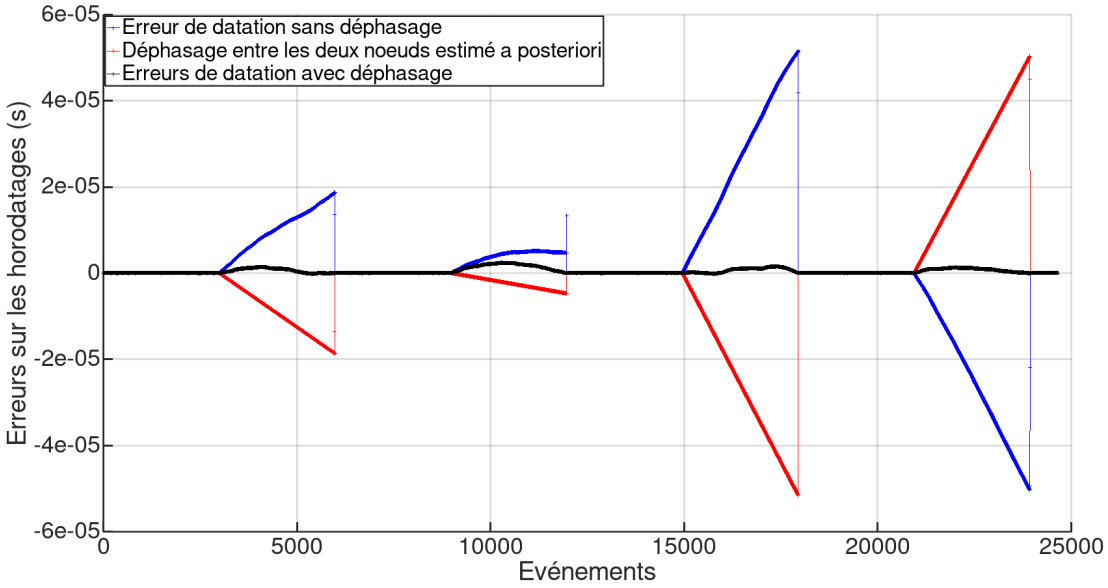


FIGURE 6.7 – Erreur sur les horodatages - Apnée de 7000 secondes

sont pas des erreurs entre deux noeuds mais sur un seul noeud, les erreurs maximales ont donc été doublées pour simuler des erreurs entre deux noeuds. Néanmoins, on constate que cette nouvelle architecture de d'unité d'horodatage est fonctionnelle et que les erreurs mesurées sont cohérentes avec les résultats précédents.

TABLE 6.1 – Erreurs sur les horodatages

k	Nombre d'événements	Moyenne	Ecart type	Moyenne abs	Maximum abs
10	2912	-2.61 ns	6.58 ns	5.66 ns	28.14 ns
250	15651	-2.17 ns	13.60 ns	9.46 ns	80.77 ns
1000	5341	37.26 ns	111.07 ns	54.23 ns	503.68 ns
7000	24658	415.18 ns	629.23 ns	426.34 ns	2.35 μ s

Des mesures de courant on été effectuées sur tous les kits de développement utilisés, ainsi que sur les TCXO, en branchant un multimètre Fluke 8840A [108] en série sur l'entrée d'alimentation (sous 5V). Ces mesures sont répertoriées dans le tableau 6.2. L'antenne 1 correspond aux antennes installées sur le toit du bâtiment (cf chapitres 3), l'antenne 2 correspond à une antenne active plus faible consommation, utilisée dans [7], [81].

TABLE 6.2 – Mesures de courant

NEO6T		Antenne GPS		FPGA		TCXO	
Allumé	Backup	1	2	Spartan 6	ICE 40	AST3TQ	170RC
50 mA	1.4 mA	60 mA	40 mA	51 mA	9 mA	2.5 mA	5.3 mA

Pour pouvoir comparer ces courants, il faut prendre en compte deux aspects : l'efficacité de l'étage d'alimentation et la différences de fréquence entre les deux implémentations (100 Mhz contre 240 MHz). Sur les kit de développement du ICE 40 et du NEO 6T, des régulateurs de tension linéaire convertissent le 5V en 3.3V. Sur le kit de développement ICE40 un deuxième régulateur linéaire est utilisé pour passer du 3.3V au 1.2V nécessaire pour le coeur du FPGA. Sur le kit de développement Spartan 6 un convertisseur Buck est utilisé pour générer le 3.3V des entrées/sorties ainsi que le 1.2V du coeur du FPGA. L'utilisation des d'un convertisseur Buck sur le kit de développement Spartan 6 diminue les pertes dans l'étage d'alimentation, le courant en entrée du kit serait plus important si un régulateur linéaire était utilisé comme sur le ICE 40. Malgré tout, l'implémentation sur le ICE 40 permet d'obtenir un courant d'entrée 5 fois moins élevé que l'implémentation sur Spartan 6 pour une fonctionnalité similaire : l'horodatage d'un événement par seconde sur une voie.

A partir de ces mesures de courant, le budget de courant total est calculé dans le tableau 6.3 pour les deux implémentations. Les courants totaux présentés dans ce tableau prennent en compte le courant de l'unité d'horodatage sur FPGA, le courant du récepteur GPS avec l'antenne active 1 et le courant pour le TCXO AST3TQ. Le tableau reprend aussi les erreurs maximales sur les horodatages pour les deux implémentation ; comme précédement, ces erreurs sont placées ici pour avoir un ordre d'idée des grandeurs et ne peuvent pas être strictement comparées du fait des différences entre les expérimentations. On constate que l'implémentation ICE 40 permet de diviser par deux le budget de courant pour une durée d'apnée de 10 secondes et par 3.7 pour une durée d'apnée de 250 secondes. Au delà de cette durée d'apnée, dans le cadre de la datation avec TCXO a posteriori, les gains sont moins important (inférieurs à 1% par pallier).

TABLE 6.3 – Budget de courant - ICE 40 vs Spartan 6

k	ratio (%)	Spartan 6 Erreur max	ICE 40 Erreur max	Courant GPS et Antenne 1	Courant total Spartan 6	Courant total ICE 40
10	30.93	91 ns	28 ns	35 mA	88.5 mA	46.5 mA
250	2.52	87 ns	81 ns	4.14 mA	57.6 mA	15.6 mA
1000	1.63	251 ns	504 ns	3.17 mA	56.7 mA	14.7 mA
7000	1.38	2.78 μ s	2.35 μ s	2.9 mA	56.4 mA	14.4 mA

6.3 Conclusion

Dans ce chapitre, une solution à faible consommation basée sur la datation a posteriori et cadencé par un TCXO est étudiée et implémentée sur un FPGA Lattice ICE 40. Il est montré que cette solution est fonctionnelle et permet des gains de 370% sur le courant consommé par l'ensemble unité d'horodatage et récepteur GPS. Même si l'erreur d'horodatage semble être du même ordre de grandeur que les erreurs précédemment observées, de nouveaux tests ont besoin d'être réalisés avec un signal PPS issu de récepteurs GPS pour quantifier les erreurs sur les horodatages de cette implémentation. Un autre aspect qui pourrait être étudié dans des travaux futurs est l'influence du passage en simple précision. Les horodatages ont été calculés en double précision dans ce chapitre. Pour une implémentation de l'unité de contrôle sur une architecture à microcontrôleur à faible consommation, celui-ci disposera probablement d'une Floating Point Unit (FPU) en simple précision.

D'autres pistes peuvent aussi être explorées pour diminuer encore la consommation de l'unité d'horodatage : Utiliser deux horloges dans l'implémentation, une haute fréquence pour les compteurs d'événements et une plus basse fréquence pour le reste ; diminuer les tensions d'alimentation des entrées/sorties pour toutes les faire fonctionner en 1.8V ; diminuer la fréquence en sortie de pll (50 MHz par exemple) et quantifier les erreurs d'horodatages engendrés par une erreur de quantification plus importante ; Utiliser les registres Double Data Rate (DDR) du ICE 40 pour garder la même erreur de quantification en divisant par deux la fréquence ; Passer sur une cible encore plus petite d'un point de vue ressources logiques : l'architecture

présentée dans ce chapitre tient sur 490 des 7680 *LogicCells* présentes sur le ICE40LP8K (soit un taux d'occupation de 6.38%).

CHAPITRE 7

Conclusion

Les travaux réalisés au cours de cette thèse ont tout d'abord permis de mettre en évidence, au travers d'une étude bibliographique, l'absence d'une solution de synchronisation par GPS, optimisée en consommation énergétique compatible pour les réseaux de capteurs sans fils. Dans le cadre du contrôle de santé des structures, l'utilisation de noeuds sans fil autonomes en énergie permet de simplifier l'installation des systèmes de surveillance et d'en réduire le coût. Sur ces noeuds, le GPS est rarement considéré comme une source de synchronisation viable du fait de son fort coût énergétique (typiquement de 100 à 200 mW). Pourtant, couplé à un système de remontée de données cellulaire (2G,3G,4G) ou LPWAN (LTE-M, NB-IoT, LoRaWAN, SigFox) la synchronisation GPS permet de déployer facilement des noeuds sur tous types de structures (même en *indoor* à l'aide de répéteurs GPS), sans contrainte de topologie de réseau. De surcroît, la synchronisation GPS permet de dater les échantillons avec une erreur inférieure à la centaine de nanosecondes. L'objectif de ces travaux est donc de proposer une solution de synchronisation basée sur le système GPS étant optimisée pour consommer moins d'énergie, tout en conservant une contrainte sur l'erreur de synchronisation maximale autorisée pour

une application donnée.

Un prototype de noeud de capteurs avec une unité d'horodatage matérielle a été conçu pour dater précisément des événements. L'implémentation matérielle permet d'obtenir de très bonne performance temporelle du fait de l'absence de surcout logiciel. L'implémentation sur FPGA Spartan 6 de l'architecture de noeud développée permet d'obtenir des erreurs d'horodatages entre deux nœuds de capteurs inférieures à 50 nanosecondes. Il a été démontré que cette erreur maximale peut être abaissée à 10 nanosecondes en utilisant un simple filtre moyenneur. Ce premier prototype a permis la constitution de plusieurs jeux de données permettant d'étudier différents scénarios de contrôle du récepteur GPS en post traitement.

Par la suite un mécanisme de synchronisation adaptative, basé sur l'extinction périodique du récepteur GPS, a été développé pour pouvoir diminuer la consommation du système tout en maintenant les paramètres cruciaux pour le démarrage à chaud du récepteur GPS. Deux modèles d'horloge ont été développés pour inférer son comportement pendant les périodes d'apnée ainsi que deux méthodes de calcul des horodatages : en "temps réel" et "a posteriori" où les horodatages sont calculés à l'issue de chaque période d'apnée. Lorsque la synchronisation adaptative est utilisée, il a été démontré que la datation "a posteriori" permet de diviser les erreurs par 10 par rapport à la datation "temps réel" et que l'ajout d'un filtre de Kalman permet de minimiser la consommation uniquement en datation "temps réel". Il a également été démontré qu'il est possible d'obtenir des gains de consommation de 59% à 99% pour des erreurs d'horodatages maximales de 140 ns à 1.4 ms en "temps réel". Ces consommations passent de 70% à 99% pour des erreurs d'horodatages maximales de 50 ns à 250 μs avec la datation "a posteriori".

L'étude de l'impact de la source de synchronisation (via l'utilisation d'un oscillateur à quartz) sur les erreurs de synchronisation a été menée. Un nœud prototype a été modifié pour y intégrer un oscillateur compensé en température (TCXO) afin d'évaluer l'impact de la qualité de l'oscillateur sur la solution de synchronisation. Il a été démontré que l'apport du TCXO est plus particulièrement visible en datation "a posteriori" avec des gains de consommation de 70% à

99% pour des erreurs d'horodatage de 100 ns à 2.8 μs.

Enfin, dans la dernière partie de ce manuscrit, le deuxième prototype de noeud, développé et implémenté sur une cible plus sobre en énergie (un FPGA ICE 40) est présenté. On y démontre que ce prototype de noeud est fonctionnel et permet un gain de 370% sur le courant consommé par l'ensemble unité d'horodatage et récepteur GPS, comparé au premier prototype. Cependant, même si les erreurs de synchronisation mesurées semblent êtres du même ordre de grandeur que les erreurs observée avec le premier prototype, de futures campagnes d'essais avec des récepteurs GPS demeurent nécessaires pour valider complètement ces résultats.

Limites

Ces travaux présentent certaines limites. Tout d'abord les travaux relatif à l'extinction des récepteurs GPS sont réalisés de manière "virtuelle" lors des différents tests de synchronisation sur les jeux de données du premier prototype : du fait de l'analyse en post traitement de jeux de données déjà enregistrés, les récepteurs GPS sont contrôlés virtuellement. Les gains de consommation sont donc théoriques et devront être validés physiquement par des mesures expérimentales de consommation. Ces travaux ont été initié et abordé dans le dernier chapitre mais doivent être poursuivis. Les essais à l'aide du deuxième prototype sont décrits dans le dernier chapitre, mais, du fait des particularités de l'année 2020, les noeuds non pas pu être qualifiés en laboratoire avec de vraies antennes GPS (le PPS était émulé par un un GBF). Bien que la gigue du PPS ne soit pas le facteur d'erreur prépondérant lors de la synchronisation adaptative (surtout lorsque les durées d'apnée sont plus longues), de nouvelles campagnes d'essais seront nécessaires pour qualifier complètement le bon fonctionnement des noeuds et quantifier précisément la consommation réelle des récepteurs GPS.

Par ailleurs, l'ensemble des campagnes de tests a été réalisé en laboratoire. Même si la température évolue, les noeuds ne sont pas soumis à de grandes variations de température tels qu'ils pourraient le subir en extérieur sur des structures. Il serait intéressant de soumettre

les noeuds à des plages de températures plus grandes et des variations de températures plus rapides, via des tests en étuves, pour quantifier leurs effets sur les prédictions des modèles d'horloges. Enfin, la visibilité des satellites GPS a toujours été optimale lors de nos campagnes de tests ce qui n'est pas non plus une garantie lors d'un déploiement réel in-situ. De nouvelles campagnes de mesure seront nécessaires pour analyser les estimations des modèles lorsque la visibilité des satellites est moins bonne ou lorsque les antennes GPS elles-même sont de moins bonne qualité (avec des antennes passives par exemple).

Perspectives

Ces travaux ouvrent plusieurs pistes de recherche, pour rendre plus robustes encore les mécanismes de synchronisation décrits et ainsi les rendre suffisamment matures pour être déployables sur le terrain. Ajouter un contrôle de la qualité des signaux GPS ainsi qu'une vérification du bon rafraîchissement des éphémérides toutes les deux heures s'avère primordial pour s'assurer que les solutions décrites dans ce manuscrit fonctionnent en toutes circonstances. Il serait aussi intéressant de développer un mécanisme de réévaluation de la durée du cycle en fonction de l'évolution des conditions environnementales et du vieillissement des composants pour pouvoir maintenir une consigne sur une erreur de synchronisation maximale autorisée. Toutes ces perspectives vont dans le sens de futures collaborations avec des acteurs industriels qui sont intéressés par le concept des noeuds synchronisés par GPS, à titre d'exemple, pour la surveillance d'équipements sur le réseau ferroviaire, la surveillance des lignes hautes tension du réseau électrique ou encore dans la surveillance des ouvrages de génie civil.

Un autre aspect qui pourrait être encore développé réside dans la diminution de la consommation énergétique des noeuds. Le deuxième prototype n'est, à ce stade, pas encore complètement optimisé, l'ajout d'une deuxième fréquence de fonctionnement plus basse au sein du FPGA pour contrôler les taches qui ne nécessitent pas de hautes fréquences pourrait diminuer la consommation de celui-ci. L'utilisation des registres Double Data Rate (DDR) ou l'ajout d'un

convertisseur temps-numérique (TDC) pourraient aussi permettre de diminuer la fréquence du FPGA tout en maintenant la même précision de synchronisation.

ANNEXE A

Architecture du premier prototype

L'implémentation de la première architecture d'unité d'horodatage sur un FPGA Spartan 6 a été développée à l'aide de la suite logicielle ISE de chez Xilinx. Ci-dessous le code VHDL correspondant à l'architecture. Le bloc *decodeur_de_trames* n'est pas décrit ici car il n'a pas été développé pendant ces travaux (de plus il est très long et comprend beaucoup de sous-blocs). Son rôle est de décoder les trames NMEA et UBX en provenance du récepteur GPS Ublox NEO6T. Les blocs *UART_RX* et *UART_TX* ne sont pas décrits ici, ils sont inspirées des descriptions VHDL suivantes : https://www.nandland.com/vhdl/modules/module-uart-serial-port-rs232.html

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

package EightChannels is
```

```
type canData is array (0 to 7) of std_logic_vector(23 downto 0);
type array_of_32stdlv is array(natural range <>) of std_logic_vector(31
   downto 0);
type array_of_96stdlv is array(natural range <>) of std_logic_vector(95
   downto 0);

function swap16(val: in std_logic_vector)
  return std_logic_vector is
begin
  assert val'length = 16 report "swap16 input in not 16bits" severity
  failure;
  if val'ascending then
    return val(val'low to val'low + 7) & val(val'low + 8 to val'low +
  15);
  else
    return val(7 + val'low downto val'low) & val(15 + val'low downto 8
  + val'low );
  end if;
end swap16;

function can_lsb(val: in std_logic_vector)
  return std_logic_vector is
begin
  assert val'length = 24 report "can_lsb input in not 24bits" severity
  failure;
  return swap16(val(val'low + 15 downto val'low));
end can_lsb;

function can_msb(val: in std_logic_vector)
  return std_logic_vector is
begin
  assert val'length = 24 report "can_msb input in not 24bits" severity
  failure;
```

```
  if val(val'high) = '1' then
    return val(val'high downto val'high - 7) & "11111111";
  else
    return val(val'high downto val'high - 7) & "00000000";
  end if;
end can_msb;

function f_hexToAscii(x : std_logic_vector(3 downto 0))
  return std_logic_vector is
begin
  if unsigned(x) > 9 then
    return std_logic_vector(unsigned(x) + x"37");
  else
    return std_logic_vector(unsigned(x) + x"30");
  end if;
end f_hexToAscii;

component UART_TX is
  port(
    i_Clk       : in  std_logic;
    i_rst     : in  std_logic;
      i_TX_DV     : in  std_logic;
      i_TX_Byte   : in  std_logic_vector(7 downto 0);
      o_TX_Active : out std_logic;
      o_TX_Serial : out std_logic;
      o_TX_Done   : out std_logic);
end component;

component UART_RX is
  port(
    i_Clk       : in  std_logic;
    i_rst       : in  std_logic;
      i_RX_Serial : in  std_logic;
```

```
      o_RX_DV     : out std_logic;
      o_RX_Byte   : out std_logic_vector(7 downto 0));
end component;

component myComponent_v2 is
  port(
    i_rst        : in std_logic;
    i_clk_hif    : in std_logic;
    i_clk_lof    : in std_logic;
    i_pps_true   : in std_logic;
    i_capture  : in std_logic;
    i_GPS_date   : in std_logic_vector(23 downto 0);
    i_GPS_qerr   : in std_logic_vector(31 downto 0);
    o_pps_fake   : out std_logic;
    o_data_ready : out std_logic;
    o_pps_len    : out std_logic_vector(29 downto 0);
    o_offset_len : out std_logic_vector(29 downto 0);
    o_count_evt  : out std_logic_vector(29 downto 0);
    o_GPS_date   : out std_logic_vector(23 downto 0);
    o_GPS_qerr   : out std_logic_vector(31 downto 0));
end component;

component capture is
  port(
    i_rst          : in std_logic;
    i_clk          : in std_logic;
    i_capture      : in std_logic;
    i_fake       : in std_logic;
    o_epoch      : out std_logic_vector(31 downto 0);
    o_count      : out std_logic_vector(31 downto 0));
end component;

component decodeur_de_trames is
```

```
port(
  clock : in STD_LOGIC;
  reset : in STD_LOGIC;
  UART : in STD_LOGIC;
  i_activation : in STD_LOGIC;

  --data trame GGA
  o_h1                  : out STD_LOGIC_VECTOR (3 downto 0);
  o_h2                  : out STD_LOGIC_VECTOR (3 downto 0);
  o_m1                  : out STD_LOGIC_VECTOR (3 downto 0);
  o_m2                  : out STD_LOGIC_VECTOR (3 downto 0);
  o_s1                  : out STD_LOGIC_VECTOR (3 downto 0);
  o_s2                  : out STD_LOGIC_VECTOR (3 downto 0);
  o_quality             : out STD_LOGIC_VECTOR (3 downto 0);
  o_nb_sat1             : out STD_LOGIC_VECTOR (3 downto 0);
  o_nb_sat2             : out STD_LOGIC_VECTOR (3 downto 0);
  o_trame_GGA_valide    : out STD_LOGIC;

  --data trame ZDA
  o_day1                : out STD_LOGIC_VECTOR (3 downto 0);
  o_day2                : out STD_LOGIC_VECTOR (3 downto 0);
  o_month1              : out STD_LOGIC_VECTOR (3 downto 0);
  o_month2              : out STD_LOGIC_VECTOR (3 downto 0);
  o_year1               : out STD_LOGIC_VECTOR (3 downto 0);
  o_year2               : out STD_LOGIC_VECTOR (3 downto 0);
  o_year3               : out STD_LOGIC_VECTOR (3 downto 0);
  o_year4               : out STD_LOGIC_VECTOR (3 downto 0);
  o_trame_ZDA_valide    : out STD_LOGIC;

  --data trame TIM_TP
  o_qerr                    : out STD_LOGIC_VECTOR (31 downto 0);
  o_weeks_TIM               : out STD_LOGIC_VECTOR (15 downto 0);
  o_milisec_of_week_TIM         : out STD_LOGIC_VECTOR (31 downto 0);
```

```
      o_trame_TIM_TP_valide    : out STD_LOGIC;

      --data trame NAV_TIMEGPS
      o_weeks_NAV                  : out STD_LOGIC_VECTOR (15 downto 0);
      o_milisec_of_week_NAV        : out STD_LOGIC_VECTOR (31 downto 0);
      o_leap_second                : out STD_LOGIC_VECTOR (7 downto 0);
      o_validity_flag              : out STD_LOGIC_VECTOR (7 downto 0);
      o_trame_NAV_TIMEGPS_valide   : out STD_LOGIC;

      --Debug
      o_debut_trame_GGA     : out STD_LOGIC;
      o_debut_trame_ZDA     : out STD_LOGIC;
      o_debut_trame_TIM_TP      : out STD_LOGIC;
      o_debut_trame_NAV_TIMEGPS : out STD_LOGIC);
  end component;
end EightChannels;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

library unisim;
use unisim.vcomponents.all;

use work.EightChannels.ALL;

entity top is
  generic (
    g_SIZE_CNT_CLK : integer := 30
   );
  Port (
    io_reset          : in std_logic;
    io_clk          : in std_logic;
```

```
    io_pps_true       : in std_logic;
    io_capture        : in std_logic;

    -- INTERFACE GPS
    i_rxUartGPS       : in std_logic;

    -- DEBUG
    io_debug1       : out std_logic;
    io_debug2       : out std_logic;
    io_debug3       : out std_logic;
    io_debug4       : out std_logic;
    io_debug5       : out std_logic;
    io_debug6       : out std_logic;
    io_debug7       : out std_logic;
    io_debug8       : out std_logic;

    -- INTERFACE PEGASE
    o_txUart          : out std_logic;

    -- INTERFACE TEMPERATURE
    i_rxUart          : in std_logic
  );
end top;

architecture Behavioral of top is
  -- Clocks
  signal clkb             : std_logic;
  signal clk_capture_unbuffered : std_logic;
  signal clk_capture        : std_logic;
  signal clk_xopps_unbuffered : std_logic;
  signal clk_xopps          : std_logic;
  signal clk_uart_unbuffered    : std_logic;
  signal clk_uart         : std_logic;
```

```
-- PLL Glue
signal pll1_clkfbout        : std_logic;
signal pll1_locked          : std_logic;
signal pll2_clkfbout        : std_logic;
signal pll2_locked          : std_logic;
signal internal_reset       : std_logic;

-- UART
signal r_dataTx           : std_logic_vector(7 downto 0);
signal r_validData          : std_logic;
signal r_txActive           : std_logic;
signal r_txDone           : std_logic;

-- PEGASE INTERFACE
type t_SM_interface is (s_waitDataReady, s_sendByte, s_waitTx);
type t_Buffer_interface is array(53 downto 0) of std_logic_vector(7
  downto 0);
signal r_SM_interface       : t_SM_interface; --:= s_waitDataReady;
signal r_dataReady          : std_logic;
signal r_buffer           : t_Buffer_interface;
signal r_indexBuff          : unsigned(5 downto 0);

-- BUFFERHANDLER
type t_SM_DataReady is (s_waitFreqOffset, s_loadBuffer, s_setDataReady)
  ;
signal s_SM_DataReady : t_SM_DataReady;

-- FREQUENCYMETER
signal r_ppsFake            : std_logic;
signal r_freqOffsetReady    : std_logic;
signal r_frequency          : std_logic_vector(31 downto 0);
signal r_offset           : std_logic_vector(31 downto 0);
```

```
  -- CAPTURE
  signal r_timestamp_epoch    : std_logic_vector(31 downto 0);
  signal r_timestamp_count    : std_logic_vector(31 downto 0);

  -- GPS INTERFACE
  signal r_GPS_qerr          : std_logic_vector(31 downto 0);
  signal r_GPS_date          : std_logic_vector(23 downto 0);
  signal r_GPS_qerr_valid     : std_logic_vector(31 downto 0);
  signal r_GPS_date_valid     : std_logic_vector(23 downto 0);


  -- TEMPERATURE INTERFACE
  signal r_validDataRx        : std_logic;
  signal r_tempByte          : std_logic_vector(7 downto 0);
  type t_temp_buffer is array(4 downto 0) of std_logic_vector(7 downto 0)
    ;
  signal r_temp           : t_temp_buffer;
  signal r_idx_temp          : unsigned(2 downto 0);


begin
  io_debug1 <= '0';--r_freqOffsetReady;
  io_debug2 <= '0';--r_ggabeg; --r_vtmtp; --'0';
  io_debug3 <= io_pps_true;
  io_debug4 <= r_ppsFake;
  io_debug5 <= '0';
  io_debug6 <= '0';
  io_debug7 <= '0';
  io_debug8 <= '0';

  -- Hold rest until PLL have started
  internal_reset <= io_reset or not pll1_locked or not pll2_locked;
```

```
-- Buffer input clock (32MHz)
Inst_CLKBUF_io_clk : IBUFG
port map (
  O => clkb,
  I => io_clk
);

Inst_PLL_1 : PLL_BASE
generic map (
  BANDWIDTH            => "OPTIMIZED",
  CLK_FEEDBACK         => "CLKFBOUT",
  COMPENSATION         => "INTERNAL",
  DIVCLK_DIVIDE        => 1,
  CLKFBOUT_MULT        => 15,
  CLKFBOUT_PHASE       => 0.000,
  CLKIN_PERIOD         => 31.250, -- 32MHz
  REF_JITTER           => 0.010
)
port map (
  -- Output clocks
  CLKFBOUT             => pll1_clkfbout,
  CLKOUT0              => open,
  CLKOUT1              => open,
  CLKOUT2              => open,
  CLKOUT3              => open,
  CLKOUT4              => open,
  CLKOUT5              => open,

  -- Status and control signals
  LOCKED               => pll1_locked,
  RST                  => io_reset,
```

```
  -- Input clock control
  CLKFBIN              => pll1_clkfbout,
  CLKIN                => clkb
);

Inst_PLL_2 : PLL_BASE
generic map (
  BANDWIDTH             => "OPTIMIZED",
  CLK_FEEDBACK          => "CLKFBOUT",
  COMPENSATION          => "INTERNAL",
  DIVCLK_DIVIDE         => 1,
  CLKFBOUT_PHASE        => 0.000,
  CLKFBOUT_MULT         => 15,
  CLKOUT0_DIVIDE        => 2,       -- 240MHz
  CLKOUT0_PHASE         => 0.000,
  CLKOUT0_DUTY_CYCLE    => 0.500,
  CLKOUT1_DIVIDE        => 128,     -- 3.750MHz
  CLKOUT1_PHASE         => 0.000,
  CLKOUT1_DUTY_CYCLE    => 0.500,
  CLKOUT2_DIVIDE        => 30,    -- 16MHz
  CLKOUT2_PHASE         => 0.000,
  CLKOUT2_DUTY_CYCLE    => 0.500,
  CLKIN_PERIOD          => 31.250,    -- 32MHz = 31.25ns
  REF_JITTER            => 0.010
)
port map (
  -- Output clocks
  CLKFBOUT              => pll2_clkfbout,
  CLKOUT0               => clk_capture_unbuffered,
  CLKOUT1               => clk_xopps_unbuffered,
  CLKOUT2               => clk_uart_unbuffered,
  CLKOUT3               => open,
  CLKOUT4               => open,
```

```
    CLKOUT5              => open,

    -- Status and control signals
    LOCKED               => pll2_locked,
    RST                  => io_reset,

    -- Input clock control
    CLKFBIN              => pll2_clkfbout,
    CLKIN                => clkb
  );

  Inst_CLKBUF_clk_capture : BUFG
  port map (
    O   => clk_capture,
    I   => clk_capture_unbuffered
  );

  Inst_CLKBUF_clk_xopps : BUFG
  port map (
    O   => clk_xopps,
    I   => clk_xopps_unbuffered
  );

  Inst_CLKBUF_clk_uart : BUFG
  port map (
    O   => clk_uart,
    I   => clk_uart_unbuffered
  );

--
--  End of Clock section
--
  r_frequency(31 downto g_SIZE_CNT_CLK) <= (others => '0');
```

```
  r_offset(31 downto g_SIZE_CNT_CLK) <= (others => '0');

  r_buffer(53) <= x"24";
  r_buffer(46) <= x"2C";
  r_buffer(37) <= x"2C";
  r_buffer(28) <= x"2C";
  r_buffer(19) <= x"2C";
  r_buffer(10) <= x"2C";
  r_buffer(1) <= x"0D";
  r_buffer(0) <= x"0A";

-- Interface SM
  process(internal_reset, clk_uart)
  begin
    if internal_reset = '1' then
      r_SM_interface <= s_waitDataReady;
      r_dataTx <= (others => '0');
      r_indexBuff <= to_unsigned(0, 6);
      r_validData <= '0';
    elsif clk_uart'event and clk_uart = '1' then
      case r_SM_interface is
        -- Wait for new data to Tx
        when s_waitDataReady =>
          if r_dataReady = '1' then
            r_indexBuff <= to_unsigned(53, 6);
            r_dataTx <= r_buffer(to_integer(r_indexBuff));
            r_SM_interface <= s_sendByte;
          end if;
        -- Send the Frequency
        when s_sendByte =>
          r_validData <= '1';
          if r_txActive = '1' then
            r_indexBuff <= r_indexBuff - 1;
```

```
            r_SM_interface <= s_waitTx;
          end if;
        -- Wait for TxDone
        when s_waitTx =>
          if r_txDone = '1' then
            r_validData <= '0';
            if r_indexBuff = 0 then
              r_SM_interface <= s_waitDataReady;
            else
              r_dataTx <= r_buffer(to_integer(r_indexBuff));
              r_SM_interface <= s_sendByte;
            end if;
          end if;
      end case;
    end if;
  end process;

-- Buffer Handler SM
  process(internal_reset, clk_uart)
  begin
    if internal_reset = '1' then
      r_dataReady <= '0';
      s_SM_DataReady <= s_waitFreqOffset;
    elsif clk_uart'event and clk_uart = '1' then
      case s_SM_DataReady is
        when s_waitFreqOffset =>
          if r_freqOffsetReady = '1' then
            r_buffer(52) <= f_hexToAscii(r_GPS_date_valid(23 downto 20));
            r_buffer(51) <= f_hexToAscii(r_GPS_date_valid(19 downto 16));
            r_buffer(50) <= f_hexToAscii(r_GPS_date_valid(15 downto 12));
            r_buffer(49) <= f_hexToAscii(r_GPS_date_valid(11 downto 8));
            r_buffer(48) <= f_hexToAscii(r_GPS_date_valid(7 downto 4));
            r_buffer(47) <= f_hexToAscii(r_GPS_date_valid(3 downto 0));
```

```
            r_buffer(45) <= f_hexToAscii(r_GPS_qerr_valid(31 downto 28));
            r_buffer(44) <= f_hexToAscii(r_GPS_qerr_valid(27 downto 24));
            r_buffer(43) <= f_hexToAscii(r_GPS_qerr_valid(23 downto 20));
            r_buffer(42) <= f_hexToAscii(r_GPS_qerr_valid(19 downto 16));
            r_buffer(41) <= f_hexToAscii(r_GPS_qerr_valid(15 downto 12));
            r_buffer(40) <= f_hexToAscii(r_GPS_qerr_valid(11 downto 8));
            r_buffer(39) <= f_hexToAscii(r_GPS_qerr_valid(7 downto 4));
            r_buffer(38) <= f_hexToAscii(r_GPS_qerr_valid(3 downto 0));
            r_buffer(36) <= f_hexToAscii(r_temp(4)(7 downto 4));
            r_buffer(35) <= f_hexToAscii(r_temp(4)(3 downto 0));
            r_buffer(34) <= f_hexToAscii(r_temp(3)(7 downto 4));
            r_buffer(33) <= f_hexToAscii(r_temp(3)(3 downto 0));
            r_buffer(32) <= f_hexToAscii(r_temp(1)(7 downto 4));
            r_buffer(31) <= f_hexToAscii(r_temp(1)(3 downto 0));
            r_buffer(30) <= f_hexToAscii(r_temp(0)(7 downto 4));
            r_buffer(29) <= f_hexToAscii(r_temp(0)(3 downto 0));
            r_buffer(27) <= f_hexToAscii(r_timestamp_count(31 downto 28))
    ;
            r_buffer(26) <= f_hexToAscii(r_timestamp_count(27 downto 24))
    ;
            r_buffer(25) <= f_hexToAscii(r_timestamp_count(23 downto 20))
    ;
            r_buffer(24) <= f_hexToAscii(r_timestamp_count(19 downto 16))
    ;
            r_buffer(23) <= f_hexToAscii(r_timestamp_count(15 downto 12))
    ;
            r_buffer(22) <= f_hexToAscii(r_timestamp_count(11 downto 8));
            r_buffer(21) <= f_hexToAscii(r_timestamp_count(7 downto 4));
            r_buffer(20) <= f_hexToAscii(r_timestamp_count(3 downto 0));
            r_buffer(18) <= f_hexToAscii(r_frequency(31 downto 28));
            r_buffer(17) <= f_hexToAscii(r_frequency(27 downto 24));
            r_buffer(16) <= f_hexToAscii(r_frequency(23 downto 20));
            r_buffer(15) <= f_hexToAscii(r_frequency(19 downto 16));
```

```
          r_buffer(14) <= f_hexToAscii(r_frequency(15 downto 12));
          r_buffer(13) <= f_hexToAscii(r_frequency(11 downto 8));
          r_buffer(12) <= f_hexToAscii(r_frequency(7 downto 4));
          r_buffer(11) <= f_hexToAscii(r_frequency(3 downto 0));
          r_buffer(9) <= f_hexToAscii(r_offset(31 downto 28));
          r_buffer(8) <= f_hexToAscii(r_offset(27 downto 24));
          r_buffer(7) <= f_hexToAscii(r_offset(23 downto 20));
          r_buffer(6) <= f_hexToAscii(r_offset(19 downto 16));
          r_buffer(5) <= f_hexToAscii(r_offset(15 downto 12));
          r_buffer(4) <= f_hexToAscii(r_offset(11 downto 8));
          r_buffer(3) <= f_hexToAscii(r_offset(7 downto 4));
          r_buffer(2) <= f_hexToAscii(r_offset(3 downto 0));
          s_SM_DataReady <= s_loadBuffer;
        end if;
      when s_loadBuffer =>
        r_dataReady <= '1';
        s_SM_DataReady <= s_setDataReady;
      when s_setDataReady =>
        r_dataReady <= '0';
        s_SM_DataReady <= s_waitFreqOffset;
    end case;
  end if;
end process;

-- Temperature decoder
process(internal_reset, clk_uart)
begin
  if internal_reset = '1' then
    r_idx_temp <= to_unsigned(4, 3);
    r_temp <= (others => "00000000");
  elsif clk_uart'event and clk_uart = '1' then
    if r_validDataRx = '1' then
      if r_tempByte = x"0A" then
```

```
          r_idx_temp <= to_unsigned(4, 3);
        elsif r_tempByte /= x"0D" then
          r_temp(to_integer(r_idx_temp)) <= r_tempByte;
          r_idx_temp <= r_idx_temp - 1;
        end if;
      end if;
    end if;
  end process;

-- UART Transmitter PEGASE
  Inst_UART_Transmitter: UART_TX
  port map(
    i_Clk       => clk_uart,
    i_rst     => internal_reset,
    i_TX_DV     => r_validData,
    i_TX_Byte   => r_dataTx,
    o_TX_Active => r_txActive,
    o_TX_Serial => o_txUart,
    o_TX_Done   => r_txDone
  );

-- UART Receiver TEMP
  Inst_UART_Receiver: UART_RX
  port map(
    i_Clk       => clk_uart,
    i_rst     => internal_reset,
    i_RX_Serial => i_rxUart,
    o_RX_DV     => r_validDataRx,
    o_RX_Byte => r_tempByte
  );

-- Frequencymeter
  Inst_ComponentDavid : myComponent_v2
```

```
  port map(
    i_rst         => internal_reset,
    i_clk_hif     => clk_capture,
    i_clk_lof     => clk_xopps,
    i_pps_true    => io_pps_true,
    i_capture     => io_capture,
    i_GPS_date    => r_GPS_date,
    i_GPS_qerr    => r_GPS_qerr,
    o_pps_fake    => r_ppsFake,
    o_data_ready  => r_freqOffsetReady,
    o_pps_len     => r_frequency(g_SIZE_CNT_CLK-1 downto 0),
    o_offset_len  => r_offset(g_SIZE_CNT_CLK-1 downto 0),
    o_count_evt   => r_timestamp_count(g_SIZE_CNT_CLK-1 downto 0),
    o_GPS_date    => r_GPS_date_valid,
    o_GPS_qerr    => r_GPS_qerr_valid
  );


-- Decodeur Trames
  Inst_DecodeurTrames : decodeur_de_trames
  port map(
    clock               => clk_uart,
    reset               => internal_reset,
    UART                => i_rxUartGPS,
    i_activation          => '1', --r_GPS_active,
    o_h1                => r_GPS_date(23 downto 20),
    o_h2                => r_GPS_date(19 downto 16),
    o_m1                => r_GPS_date(15 downto 12),
    o_m2                => r_GPS_date(11 downto 8),
    o_s1                => r_GPS_date(7 downto 4),
    o_s2                => r_GPS_date(3 downto 0),
    o_quality           => open,
    o_nb_sat1           => open,
    o_nb_sat2           => open,
```

```
    o_trame_GGA_valide      => open,
    o_day1              => open,
    o_day2              => open,
    o_month1              => open,
    o_month2              => open,
    o_year1             => open,
    o_year2             => open,
    o_year3             => open,
    o_year4             => open,
    o_trame_ZDA_valide      => open,
    o_qerr              => r_GPS_qerr,
    o_weeks_TIM           => open,
    o_milisec_of_week_TIM     => open,
    o_trame_TIM_TP_valide     => open,
    o_weeks_NAV           => open,
    o_milisec_of_week_NAV     => open,
    o_leap_second         => open,
    o_validity_flag       => open,
    o_trame_NAV_TIMEGPS_valide  => open,
    o_debut_trame_GGA       => open,
    o_debut_trame_ZDA       => open,
    o_debut_trame_TIM_TP      => open,
    o_debut_trame_NAV_TIMEGPS => open
  );

  r_timestamp_count(31 downto 30) <= (others => '0');
end Behavioral;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all; -- Unsigned

entity myComponent_v2 is
  port(
```

```
    i_rst        : in std_logic;
    i_clk_hif    : in std_logic;
    i_clk_lof    : in std_logic;
    i_pps_true   : in std_logic;
    i_capture    : in std_logic;
    i_GPS_date   : in std_logic_vector(23 downto 0);
    i_GPS_qerr   : in std_logic_vector(31 downto 0);
    o_pps_fake   : out std_logic;
    o_data_ready : out std_logic;
    o_pps_len    : out std_logic_vector(29 downto 0);
    o_offset_len : out std_logic_vector(29 downto 0);
    o_count_evt  : out std_logic_vector(29 downto 0);
    o_GPS_date   : out std_logic_vector(23 downto 0);
    o_GPS_qerr   : out std_logic_vector(31 downto 0));
end;

architecture behavioral of myComponent_v2 is
  -- Fake PPS generator
  signal r_pps_fake       : std_logic;
  signal r_cnt_fake     : unsigned(27 downto 0);

  -- Frequencymeter
  signal r_old_fake     : std_logic;
  signal r_new_fake     : std_logic;
  signal r_old_true     : std_logic;
  signal r_new_true     : std_logic;
  signal r_old_capture   : std_logic;
  signal r_new_capture    : std_logic;
  signal r_cnt_off      : unsigned(29 downto 0);
  signal r_cnt_freq     : unsigned(29 downto 0);
  signal r_last_freq    : unsigned(29 downto 0);
  signal r_cnt_evt      : unsigned(29 downto 0);
  signal r_last_evt     : unsigned(29 downto 0);
```

```
  -- Data ready
  type t_SM_data_ready is (s_wait_pps_fake_DR, s_start_DR, s_stop_DR);
  signal r_data_ready_state : t_SM_data_ready;

begin
  -- Fake PPS generator
  fake_pps_gen : process(i_rst, i_clk_hif) -- i_clk_lof
  begin
    if i_rst = '1' then
      r_pps_fake <= '0';
      r_cnt_fake <= to_unsigned(1, 28);
      elsif i_clk_hif'event and i_clk_hif = '1' then
          if r_cnt_fake = to_unsigned(240000000, 28) then --3750000
        r_cnt_fake <= to_unsigned(1, 28);
        r_pps_fake <= '1';
      elsif r_cnt_fake = to_unsigned(24000000, 28) then
        r_cnt_fake <= r_cnt_fake + 1;
        r_pps_fake <= '0';
          else
        r_cnt_fake <= r_cnt_fake + 1;
          end if;
      end if;
  end process fake_pps_gen;

-- Frequencymeter process on Fake
  frequencymeter : process(i_rst, i_clk_hif)
  begin
    if i_rst = '1' then
      r_old_fake <= '0';
      r_new_fake <= '0';
      r_old_true <= '0';
      r_new_true <= '0';
```

```
      r_old_capture <= '0';
      r_new_capture <= '0';
      r_cnt_off <= to_unsigned(1, 30);
      r_cnt_freq <= to_unsigned(1, 30);
      r_last_freq <= to_unsigned(1, 30);
      r_cnt_evt <= to_unsigned(1, 30);
      r_last_evt <= to_unsigned(1, 30);
      o_pps_len <= (others => '0');
      o_offset_len <= (others => '0');
      o_count_evt <= (others => '0');
    elsif i_clk_hif'event and i_clk_hif = '1' then
      r_new_fake <= r_pps_fake;
      r_new_true <= i_pps_true;
      r_new_capture <= i_capture;
      -- Front montant sur fake
      if r_new_fake = '1' and r_old_fake = '0' then
        r_cnt_freq <= r_cnt_freq + 1;
        r_cnt_evt <= to_unsigned(1, 30);
        o_offset_len <= std_logic_vector(r_cnt_off);
        o_pps_len <= std_logic_vector(r_last_freq);
        o_count_evt <= std_logic_vector(r_last_evt);
      -- Front montant sur true
      elsif r_new_true = '1' and r_old_true = '0' then
        r_cnt_evt <= r_cnt_evt + 1;
        r_cnt_off <= to_unsigned(1, 30);
        r_last_freq <= r_cnt_freq;
        r_cnt_freq <= to_unsigned(1, 30);
        o_GPS_date <= i_GPS_date;
        o_GPS_qerr <= i_GPS_qerr;
      -- Front montant sur capture
      elsif r_new_capture = '1' and r_old_capture = '0' then
        r_cnt_off <= r_cnt_off + 1;
        r_cnt_freq <= r_cnt_freq + 1;
```

```
        r_cnt_evt <= r_cnt_evt + 1;
        r_last_evt <= r_cnt_evt;
      else
        r_cnt_off <= r_cnt_off + 1;
        r_cnt_freq <= r_cnt_freq + 1;
        r_cnt_evt <= r_cnt_evt + 1;
      end if;
      r_old_fake <= r_new_fake;
      r_old_true <= r_new_true;
      r_old_capture <= r_new_capture;
    end if;
  end process frequencymeter;

-- Data Ready process Fake
  data_ready : process(i_rst, i_clk_lof)
  begin
    if i_rst = '1' then
      o_data_ready <= '0';
      r_data_ready_state <= s_wait_pps_fake_DR;
    elsif i_clk_lof'event and i_clk_lof = '1' then
      case r_data_ready_state is
        when s_wait_pps_fake_DR =>
          if r_pps_fake = '1' then
            r_data_ready_state <= s_start_DR;
          end if;
        when s_start_DR =>
          o_data_ready <= '1';
          r_data_ready_state <= s_stop_DR;
        when s_stop_DR =>
          o_data_ready <= '0';
          if r_pps_fake = '0' then
            r_data_ready_state <= s_wait_pps_fake_DR;
          end if;
```

```
      end case;
    end if;
  end process data_ready;

  o_pps_fake <= r_pps_fake;
end;
```

ANNEXE B

Architecture du deuxième prototype

L'implémentation de la deuxième architecture d'unité d'horodatage sur un FPGA ICE 40 a été développée à l'aide de la suite logicielle IceCube de chez Lattice. Ci-dessous le code Verilog correspondant à l'architecture.

```
module top(
  input CLK,
  input PIN_1,
  input PIN_2,
  input PIN_3,
  input PIN_7,
  input PIN_8,
  input PIN_23,
  output LED,
  output PIN_4,
  output PIN_5,
  output PIN_14,
  output PIN_15,
```

```
output PIN_16,
output PIN_17,
output PIN_18,
output PIN_19,
output PIN_20,
output PIN_21,
output USBPU);

parameter WAIT = 0, READ_REQ = 1, SWITCH = 2, SEND_DATA = 3,
  SEND_DATA_BYTE = 4, READ_FREQ = 5, READ_BYTE_FREQ = 6;
parameter LENGTH_DR = 100;

wire i_EVT;
wire clock_buffer;

// PULLUPS
SB_IO #(
  .PIN_TYPE(6'b 0000_01),
  .PULLUP(1'b1)
) my_input6(
  .PACKAGE_PIN(PIN_8),
  .D_IN_0(i_PPS)
);
SB_IO #(
  .PIN_TYPE(6'b 0000_01),
  .PULLUP(1'b1)
) my_input7(
  .PACKAGE_PIN(PIN_7),
  .D_IN_0(i_EVT)
);

// Clocks and pll
reg[7:0] cnt_clk_low = {8{1'b0}};
```

```
reg clk_low = 0; // CLK UART 10MHz

reg mode_GPS = 0;
reg[7:0] freq [3:0];

wire dataReady;
wire[31:0] frequency;
wire[31:0] timestamp;

wire clk_100;
wire clk_50;

pll pll_inst(
  .clock_in(PIN_23),
  .clock_out(clk_100),
  .clock_out_low(clk_50),
  .locked(locked)
);

mycomponent2 mycomponent_inst(
  clk_100,
  i_PPS,
  i_EVT,
  mode_GPS,
  {freq[0][3:0], freq[1], freq[2], freq[3]},
  dataReady,
  frequency[27:0],
  timestamp[27:0]);

reg[2:0] CK = 3'b111;
reg[2:0] CS = 3'b111;
reg[1:0] MOSI = 2'b00;
reg[3:0] DR = 4'b0000;
```

```
reg[7:0] byte = 8'h00;
reg[7:0] byte_send = 8'hE3;
reg[7:0] byte_received = 8'h00;
reg[2:0] cnt = 3'b000;
reg[3:0] cnt_byte = 4'b0000;
reg[7:0] buffer [7:0];

reg[2:0] state = WAIT;

// Sampling and delays
always @(posedge clk_100) begin //100
  CK <= {CK[1:0], PIN_1};
  CS <= {CS[1:0], PIN_2};
  MOSI <= {MOSI[0], PIN_3};
  DR <= {DR[2:0], dataReady};
end

wire CK_rising = (CK[2:1]==2'b01);
wire CK_falling = (CK[2:1]==2'b10);
wire CS_rising = (CS[2:1]==2'b01);
wire CS_falling = (CS[2:1]==2'b10);
wire DR_rising = (DR[3:2]==2'b01);
wire MOSI_data = MOSI[1];

always @(posedge clk_100) begin // 100
  case(state)
    WAIT:
      if(CS_falling) begin
        state <= READ_REQ;
        cnt <= 0;
        cnt_byte <= 0;
      end
    READ_REQ:
```

```
        if(CK_rising) begin
          cnt <= cnt + 1;
          byte <= {byte[6:0], MOSI_data};
          if(cnt == 7) begin
            state <= SWITCH;
          end
        end
      SWITCH:
        case(byte)
          8'h15:
            begin
              mode_GPS <= 0;
              state <= WAIT;
            end
          8'h16:
            begin
              mode_GPS <= 1;
              state <= READ_BYTE_FREQ;
            end
          8'h17:
            if(CK_falling) begin
              cnt <= cnt + 1;
              byte_send <= buffer[cnt_byte];
              cnt_byte <= cnt_byte + 1;
              state <= SEND_DATA;
            end
        endcase
      SEND_DATA:
        if(CK_falling) begin
          if(cnt==3'b000) begin
            cnt <= cnt + 1;
            byte_send <= buffer[cnt_byte];
            cnt_byte <= cnt_byte + 1;
```

```
            if(cnt_byte == 8) begin
              state <= WAIT;
            end
          end
          else begin
            cnt <= cnt + 1;
            byte_send <= {byte_send[6:0], 1'b0};
          end
        end
      READ_BYTE_FREQ:
        if(CK_rising) begin
          cnt <= cnt + 1;
          byte_received <= {byte_received[6:0], MOSI_data};
          if(cnt == 7) begin
            state <= READ_FREQ;
          end
        end
      READ_FREQ: begin
        cnt_byte <= cnt_byte + 1;
        freq[cnt_byte] <= byte_received;
        if(cnt_byte == 3) begin
          state <= WAIT;
        end
        else begin
          state <= READ_BYTE_FREQ;
        end
      end

      default: state <= WAIT;
    endcase
  end

  always @(posedge clk_100) begin // 100
```

```
  if(DR_rising) begin
    buffer[0] <= timestamp[31:24];
    buffer[1] <= timestamp[23:16];
    buffer[2] <= timestamp[15:8];
    buffer[3] <= timestamp[7:0];
    buffer[4] <= frequency[31:24];
    buffer[5] <= frequency[23:16];
    buffer[6] <= frequency[15:8];
    buffer[7] <= frequency[7:0];
  end
end

reg state_dr = 0;
reg dr = 0;
reg[7:0] cnt_dr = 0;

// Data Ready
always @(posedge clk_100) begin // 100
  case(state_dr)
    0:
      if(DR_rising) begin
        cnt_dr <= 1;
        dr <= 1;
        state_dr <= 1;
      end
    1:
      begin
        if(cnt_dr == LENGTH_DR) begin
          state_dr <= 0;
          dr <= 0;
        end
        else begin
          cnt_dr <= cnt_dr + 1;
```

```
          end
        end
      default: state_dr <= 0;
    endcase
  end

  reg test = 0;

  always @(posedge clk_100) begin // 100
    test <= i_PPS;
  end

  // Combinatorial ------------
  assign USBPU = 0;
  assign PIN_4 = byte_send[7];
  assign PIN_5 = dr;
  assign LED = test;
  assign PIN_14 = dataReady;
  assign PIN_15 = clk_100;
  assign PIN_16 = clk_50;
  assign PIN_17 = i_PPS;
  assign PIN_18 = i_EVT;
  assign PIN_19 = (state == SEND_DATA) ? 1'b1 : 1'b0; //freq[3][5];
  assign PIN_20 = freq[3][6];
  assign PIN_21 = freq[3][7];

  assign frequency[31:28] = {4{1'b0000}};
  assign timestamp[31:28] = {4{1'b0000}};

endmodule
```

```
module pll(
   input  clock_in,
   output clock_out,
```

```
    output clock_out_low,
    output locked
    );

    SB_PLL40_2F_CORE #(
          .FEEDBACK_PATH("SIMPLE"),
          .DIVR(4'b0000),       // DIVR =  0
          .DIVF(7'b1001111),    // DIVF = 49
          .DIVQ(3'b011),     // DIVQ =  3
          .FILTER_RANGE(3'b001),    // FILTER_RANGE = 1
          .FEEDBACK_PATH("SIMPLE"),
          .DELAY_ADJUSTMENT_MODE_FEEDBACK ("FIXED"),
          .FDA_FEEDBACK(4'b0000),
          .DELAY_ADJUSTMENT_MODE_RELATIVE("FIXED"),
          .FDA_RELATIVE(4'b0000),
          .SHIFTREG_DIV_MODE(2'b00),
          .PLLOUT_SELECT_PORTA("GENCLK"),
          .PLLOUT_SELECT_PORTB("GENCLK_HALF"),
          .ENABLE_ICEGATE_PORTA(1'b0),
          .ENABLE_ICEGATE_PORTB(1'b0)
       ) uut (
          .LOCK(locked),
          .RESETB(1'b1),
          .BYPASS(1'b0),
          .REFERENCECLK(clock_in),
          .PLLOUTCOREA(clock_out),
          .PLLOUTCOREB(clock_out_low)
          );
endmodule
```

```
module mycomponent2(
  input i_clk_hif,
  input i_pps_true,
  input i_capture,
```

```
input cnt_mode,
input[27:0] i_cntMax,
output o_dataReady,
output reg[27:0] o_pps_len,
output reg[27:0] o_count_evt
);

// Sampler
reg[2:0] pps_true_s = 3'b000;
reg[2:0] capture_s = 3'b000;
wire pps_true_rising;
wire capture_rising;

// Multiplexer
reg pps_r_cnt = 0;
reg[27:0] cnt_MAX = 100000000;

  // Frequency Counter
reg[27:0] cnt = 1;
reg PPSf = 0;
reg dr = 0;

// Frequency Counter
reg[27:0] cnt_evt = 1;

// Sampler
always @(posedge i_clk_hif) begin
  pps_true_s <= {pps_true_s[1:0], i_pps_true};
  capture_s <= {capture_s[1:0], i_capture};
end

assign pps_true_rising = (pps_true_s[2:1]==2'b01);
assign capture_rising = (capture_s[2:1]==2'b01);
```

```
// Multiplexer
always @(posedge i_clk_hif) begin
    pps_r_cnt <= cnt_mode ? 1'b0 : pps_true_rising;
    cnt_MAX <= cnt_mode ? i_cntMax : 28'b1111111111111111111111111111;
  end

// Frequency Counter
always @(posedge i_clk_hif) begin
  if(pps_r_cnt) begin
    o_pps_len <= cnt;
    PPSf <= 1;
    cnt <= 1;
    dr <= 1;
  end
  else if(cnt == cnt_MAX) begin
    o_pps_len <= cnt;
    PPSf <= 1;
    cnt <= 1;
    dr <= 1;
  end
  else if(cnt == 50000000) begin
    cnt <= cnt+1;
    dr <= 0;
  end
  else begin
    cnt <= cnt+1;
    PPSf <= 0;
  end
end

// Event Counter
always @(posedge i_clk_hif) begin
```

```
    if(PPSf) begin
      if(capture_rising) begin
        o_count_evt <= cnt_evt;
      end
      cnt_evt <= 1;
    end
    else begin
      if(capture_rising) begin
        o_count_evt <= cnt_evt;
      end
      cnt_evt <= cnt_evt+1;
    end

  end

  // Data Ready
  assign o_dataReady = dr;
endmodule
```

ANNEXE C

Algorithme de l'unité de contrôle

L'algorithme de l'unité de contrôle du deuxième prototype de noeud a été implémenté en python. Ce script python est exécuté sur un Raspberry Pi 3 dans nos expérimentations. Ci-dessous le code python.

```
#!/usr/bin/env python

import spidev
import RPi.GPIO as GPIO
import time
import signal
import sys
from serial import *
from binascii import unhexlify



def signal_handler(sig, frame):
  spi.close()
```

```
  ps.close()
  print('Exiting...')
  sys.exit(0)

signal.signal(signal.SIGINT, signal_handler)

def setGPS_ON():
  r = spi.writebytes([0x15])
  print('Switch ON')

def setGPS_OFF(freq_arg):
  freq = '%0*X'%(2,freq_arg)
  if(len(freq) == 7):
    r = spi.xfer2([0x16, int(freq[-7], 16), int(freq[-6]+freq[-5], 16),
    int(freq[-4]+freq[-3], 16), int(freq[-2]+freq[-1], 16)])
  elif(len(freq) == 6):
    r = spi.xfer2([0x16, 0x00, int(freq[-6]+freq[-5], 16), int(freq[-4]+
    freq[-3], 16), int(freq[-2]+freq[-1], 16)])

  print('Switch OFF with freq : ' + str(freq_arg))

def my_cb(channel):
  global cnt, modeGPS, last_freq

  [data, last_freq] = getData()
  print(data)

  cnt += 1
  if(cnt == 5):
    cnt = 0;
    modeGPS = ~modeGPS
    setGPS_ON() if modeGPS else setGPS_OFF(last_freq)
```

```
def getData():
  r = spi.xfer2([0x17, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])
  timestamp = ['%0*X' % (2,i) for i in r[1:5]]
  frequency = ['%0*X' % (2,i) for i in r[5:9]]
  return ['$'+str(int(''.join(timestamp),16))+','+str(int(''.join(
    frequency),16))+','+str(time), int(''.join(frequency),16)]

print("Init...")

cnt = 0
modeGPS = 0
last_freq = 100000000;

spi = spidev.SpiDev()
spi.open(0,0)
spi.max_speed_hz = 7800000

setGPS_OFF(last_freq)
#setGPS_ON();

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.IN)
GPIO.add_event_detect(17, GPIO.RISING, callback=my_cb, bouncetime=1)

print("Waiting...")
#time.sleep(600)
time = 0
with Serial(port="/dev/ttyAMA0", baudrate=9600) as ps:
    if ps.isOpen():
      while True:
        line = ps.readline()
        fields = line.split(',')
        if fields[0] == "$GPRMC":
```

```
            time = int(float(fields[1]))


spi.close()
GPIO.cleanup()
print("Done!")
```

Les fonctions suivantes sont utilisées pour contrôler le récepteur GPS, ils permettent d'éteindre le GPS (le mettre en *backup mode*), le rallumer et sauvegarder sa configuration dans la mémoire flash.

```
def stopGPS(ps):
    try:
      packet = "b5620241080000000000020000004d3b"
      ps.write(unhexlify(packet))
    except(KeyboardInterrupt, SystemExit):
      print "error"



def startGPS(ps):
    try:
      packet = "ffffff"
      ps.write(unhexlify(packet))
    except(KeyboardInterrupt, SystemExit):
      print "error"



def saveGPS(ps):
    try:
      packet = "b56206090d0000000000ffff000000000000031dab"
      ps.write(unhexlify(packet))
    except(KeyboardInterrupt, SystemExit):
      print "error"
```

Liste des publications et communications

Publications à venir

- D. Pallier, S. Pillement, V. Le Cam, "A new node architecture for low power time synchronization with GPS", Journal of Systems Architecture - Elsevier, 2021, **pas encore soumis**

Publications dans des revues internationales avec comité de lecture

- D. Pallier, V. Le Cam, S. Pillement, "Energy-efficient GPS synchronization for wireless nodes", IEEE Sensors Journal, 2020

Publications dans des conférences internationales avec actes et comité de lecture

- D. Pallier, V. Le Cam, S. Pillement, Q. Zhang, L. Mevel, Offset Tracking of sensor clock using Kalman filter for wireless network synchronization, Proc. IWSHM 2019 - 12th International Workshop on Structural Health Monitoring, 2019
- V. Le Cam, A. Bouche, D.Pallier, “Wireless Sensors Synchronization : an accurateand deterministic GPS-based algorithm”, Proc. IWSHM 2017 - 11th International Workshop on Structural Health Monitoring, 2017, **co-auteur**

Présentations en workshop ou séminaires sans actes

- Doctoriales du Département Cosys de l'IFSTTAR, présentation des travaux réalisés à ce jour, Le Pouliguen, 2019.
- Doctoriales de l'Ecole doctorale MathSTIC, présentation des travaux réalisés à ce jour, Nantes, 2019
- Doctoriales du Département Cosys de l'IFSTTAR, présentation des travaux réalisés à ce jour, Marne La Vallée, 2018.
- Doctoriales du Département Cosys de l'IFSTTAR, présentation des travaux réalisés à ce jour, Marne La Vallée, 2018
- Doctoriales de l'IETR, présentation des travaux réalisés à ce jour, Nantes, 2018

Bibliographie

[1] J. Elson, L. Girod et D. Estrin, “Fine-grained network time synchronization using reference broadcasts”, *ACM SIGOPS Operating Systems Review*, t. 36, p. 147-163, SI 31 déc. 2002. doi : 10.1145/844128.844143. (visité le 27/08/2017) (cf. p. 3, 19, 21, 27, 29-31, 65).

[2] S. Ganeriwal, R. Kumar et M. B. Srivastava, “Timing-sync protocol for sensor networks”, in *Proc. SenSys*, Los Angeles, CA, USA, Los Angeles, CA, USA, Nov. 2003 2003, p. 138-149. doi : 10.1145/958491.958508. (visité le 27/08/2017) (cf. p. 3, 19-21, 27, 30, 31).

[3] M. Maróti, B. Kusy, G. Simon et A. Lédeczi, “The Flooding Time Synchronization Protocol”, in *Proc. SenSys*, New York, NY, USA, New York, NY, USA, Nov. 2004 2004, p. 39-49. doi : 10.1145/1031495.1031501. (visité le 02/11/2017) (cf. p. 3, 19, 21, 25, 27, 29-31, 65).

[4] R. Exel, “Clock synchronization in IEEE 802.11 wireless LANs using physical layer timestamps”, in *Proc. IEEE Int. Symp. on Precision Clock Synchronization*, San Francisco, CA, USA, Sep. 2012, p. 1-6 (cf. p. 3, 24, 31).

[5] T. Beluch, D. Dragomirescu et R. Plana, “A sub-nanosecond Synchronized MAC–PHY cross-layer design for Wireless Sensor Networks”, *Elsevier Ad hoc networks*, t. 11, n° 3, p. 833-845, oct. 2012. doi : https://doi.org/10.1016/j.adhoc.2012.09.010 (cf. p. 3, 26, 31).

[6] J. Serrano, M. Lipinski, T. Wlostowski, E. Gousiou, E. van der Bij, M. Cattin et G. Daniluk, “The white rabbit project”, 2013 (cf. p. 3, 18).

[7] V. Le Cam, M. Döhler, M. Le Pen, I. Guéguen et L. Mevel, “Embedded subspace-based modal analysis and uncertainty quantification on wireless sensor platform PEGASE”, in *Proc. EWSHM*, p. 1705-1715 (cf. p. 3, 27, 123).

[8] J. Ko et Y. Q. Ni, “Technology developments in structural health monitoring of large-scale bridges”, *Engineering structures*, t. 27, n° 12, p. 1715-1725, 2005 (cf. p. 7).

[9] T. Harms, S. Sedigh et F. Bastianini, "Structural health monitoring of bridges using wireless sensor networks", *IEEE Instrumentation & Measurement Magazine*, t. 13, n° 6, p. 14-18, 2010 (cf. p. 7).

[10] D. Barke et W. K. Chiu, "Structural health monitoring in the railway industry : a review", *Structural Health Monitoring*, t. 4, n° 1, p. 81-93, 2005 (cf. p. 7).

[11] F. Flammini, A. Gaglione, F. Ottello, A. Pappalardo, C. Pragliola et A. Tedesco, "Towards wireless sensor networks for railway infrastructure monitoring", in *Electrical Systems for Aircraft, Railway and Ship Propulsion*, IEEE, 2010, p. 1-6 (cf. p. 7).

[12] C. C. Ciang, J.-R. Lee et H.-J. Bang, "Structural health monitoring for a wind turbine system : a review of damage detection methods", *Measurement science and technology*, t. 19, n° 12, p. 122 001, 2008 (cf. p. 7).

[13] A. Ghoshal, M. J. Sundaresan, M. J. Schulz et P. F. Pai, "Structural health monitoring techniques for wind turbine blades", *Journal of Wind Engineering and Industrial Aerodynamics*, t. 85, n° 3, p. 309-324, 2000 (cf. p. 7).

[14] K. Diamanti et C. Soutis, "Structural health monitoring techniques for aircraft composite structures", *Progress in Aerospace Sciences*, t. 46, n° 8, p. 342-352, 2010 (cf. p. 7).

[15] I. Garcia, J. Zubia, G. Durana, G. Aldabaldetreku, M. A. Illarramendi et J. Villatoro, "Optical fiber sensors for aircraft structural health monitoring", *Sensors*, t. 15, n° 7, p. 15 494-15 519, 2015 (cf. p. 7).

[16] T.-H. Yi et H.-N. Li, "Methodology developments in sensor placement for health monitoring of civil infrastructures", *International Journal of Distributed Sensor Networks*, t. 8, n° 8, p. 612 726, 2012 (cf. p. 8).

[17] P. Swindell, J. Doyle et D. Roach, "Integration of structural health monitoring solutions onto commercial aircraft via the Federal Aviation Administration structural health monitoring research program", in *AIP conference proceedings*, AIP Publishing LLC, t. 1806, 2017, p. 070 001 (cf. p. 9).

[18] F. CEGLA et J. ALLIN, "Ultrasonics, Corrosion and SHM-the Story of Permasense Ltd.", *Structural Health Monitoring 2017*, n° shm, 2017 (cf. p. 9).

[19] P. Nie et Z. Jin, "Requirements, challenges and opportunities of wireless sensor networks in structural health monitoring", in *2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, IEEE, 2010, p. 1052-1057 (cf. p. 11).

[20] P. Wang, Y. Yan, G. Y. Tian, O. Bouzid et Z. Ding, "Investigation of wireless sensor networks for structural health monitoring", *Journal of Sensors*, t. 2012, 2012 (cf. p. 11).

[21] J. Cao et X. Liu, "Requirements, challenges, and summary of hardware and software design for a WSN-Based SHM system", in *Wireless sensor networks for structural health monitoring*, Springer, 2016, p. 7-11 (cf. p. 11).

[22] K. Dragos et K. Smarsly, "A comparative review of wireless sensor nodes for structural health monitoring", in *Proc. of the 7th International Conference on Structural Health Monitoring of Intelligent Infrastructure. Turin, Italy*, t. 1, 2015, p. 2015 (cf. p. 11).

[23] M. Abdulkarem, K. Samsudin, F. Z. Rokhani et M. F. A Rasid, "Wireless sensor network for structural health monitoring : A contemporary review of technologies, challenges, and future direction", *Structural Health Monitoring*, t. 19, n° 3, p. 693-735, 2020 (cf. p. 11).

[24] "Thirteenth General Conference on Weights and Measures", *Comptes rendus de la 13ème CGPM (1967/68)*, p. 103, 1969 (cf. p. 12).

[25] "Fourteenth General Conference on Weights and Measures", *Comptes rendus de la 14ème CGPM (1971)*, p. 77, 1972 (cf. p. 12).

[26] M. J. Dunn, "GLOBAL POSITIONING SYSTEMS DIRECTORATESYSTEMS ENGINEERING & INTEGRATIONINTERFACE SPECIFICATIONIS-GPS-200", 2019 (cf. p. 13, 61-63).

[27] B. A. Renfro, J. Rosenquest, A. Terry et N. Boeker, "An Analysis of Global Positioning System (GPS) Standard Positioning System (SPS) Performance for 2015", 2017. adresse : https://www.gps.gov/systems/gps/performance/2015-GPS-SPS-performance-analysis.pdf (cf. p. 13).

[28] J. R. Vig, "Introduction to quartz frequency standards", U.S. Army Electronics Technology et Devices Lab., NJ, USA, SLCET-TR-92-1, Tech. Rep. 1992 (cf. p. 14, 16, 17, 92).

[29] F. L. Walls et J. R. Vig, "Fundamental limits on the frequency stabilities of crystal oscillators", *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, t. 42, n° 4, p. 576-589, juil. 1995. doi : 10.1109/58.393101 (cf. p. 14).

[30] D. B. Leeson, "Oscillator Phase Noise : A 50-Year Review", *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, t. 63, n° 8, p. 1208-1225, août 2016, issn : 0885-3010. doi : 10.1109/TUFFC.2016.2562663 (cf. p. 14).

[31] *ASV10 Datasheet*, Abracon. adresse : https://abracon.com/Oscillators/ASV.pdf (cf. p. 15, 17).

[32] *T100F Datasheet*, Connor Winfield. adresse : http://www.conwin.com/datasheets/tx/tx350.pdf (cf. p. 15, 17).

[33] *AST3TQ Datasheet*, Abracon. adresse : https://www.mouser.fr/datasheet/2/3/AST3TQ-28-461101.pdf (cf. p. 15, 17).

[34] *AOCJYR Datasheet*, Abracon. adresse : https://abracon.com/Precisiontiming/AOCJYR-10.000MHz-M5625LF.pdf (cf. p. 15, 17).

[35] L. Lamport, "Time, clocks, and the ordering of events in a distributed system", *Commun. ACM*, t. 21, n° 7, p. 558-565, juil. 1978. doi : 10.1145/359545.359563 (cf. p. 18).

[36] D. L. Mills, "Internet time synchronization : the network time protocol", *IEEE Trans. Commun.*, t. 39, n° 10, p. 1482-1493, oct. 1991. doi : 10.1109/26.103043 (cf. p. 18).

[37] ——, "Improves algorithms for synchronizing computer network clocks", *IEEE/ACM Trans. Networking*, t. 3, n° 3, p. 245-254, juin 1995. doi : 10.1109/90.392384 (cf. p. 18).

[38] D. Mills, J. Martin, J. Burbank et W. Kasch, "Network time protocol version 4 : Protocol and algorithms specification", 2010 (cf. p. 18).

[39] IEEE, *IEEE 1588-2019 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std. 2012 (cf. p. 18).

[40] E. F. Dierikx, A. E. Wallin, T. Fordell, J. Myyry, P. Koponen, M. Merimaa, T. J. Pinkert, J. C. Koelemeij, H. Z. Peek et R. Smets, "White rabbit precision time protocol on long-distance fiber links", *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, t. 63, n° 7, p. 945-952, 2016 (cf. p. 18).

[41] J. Elson et K. Römer, "Wireless sensor networks : A new regime for time synchronization", *ACM SIGCOMM Comp. Com.*, t. 33, n° 1, p. 149-154, jan. 2003. doi : 10.1145/774763.774787 (cf. p. 19).

[42] H. Kopetz et W. Ochsenreiter, "Clock synchronization in distributed real-time systems", *IEEE Trans. Comput.*, t. C-36, n° 8, p. 933-940, août 1987. doi : 10.1109/TC.1987.5009516 (cf. p. 19).

[43] A. Woo et D. E. Culler, "A transmission control scheme for media access in sensor networks", in *Proceedings of the 7th annual international conference on Mobile computing and networking*, 2001, p. 221-235 (cf. p. 21).

[44] A. R. Swain et R. Hansdah, "A model for the classification and survey of clock synchronization protocols in WSNs", *Ad Hoc Networks*, t. 27, p. 219-241, avr. 2015. doi : 10.1016/j.adhoc.2014.11.021 (cf. p. 21).

[45] D. Djenouri et M. Bagaa, "Synchronization protocols and implementation issues in wireless sensor networks : A review", *IEEE Syst. J.*, t. 10, n° 2, p. 617-627, juin 2016. doi : 10.1109/JSYST.2014.2360460 (cf. p. 21).

[46] S. Chen, A. Dunkels, F. Osterlind, T. Voigt et M. Johansson, "Time synchronization for predictable and secure data collection in wireless sensor networks", in *The Sixth Annual Mediterranean Ad Hoc Networking WorkShop*, 2007, p. 165-172 (cf. p. 22).

[47] D. Djenouri, "R^{4}Syn : Relative referenceless receiver/receiver time synchronization in wireless sensor networks", *IEEE Signal Process. Lett.*, t. 19, n° 4, p. 175-178, avr. 2012 (cf. p. 22, 29).

[48] C. Lenzen, P. Sommer et R. Wattenhofer, "Optimal clock synchronization in networks", in *Proc. SenSys*, p. 225-238 (cf. p. 22, 29).

[49] P. Sommer et R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks", in *2009 International Conference on Information Processing in Sensor Networks*, IEEE, 2009, p. 37-48 (cf. p. 22, 29).

[50] K. Apicharttrisorn, S. Choochaisri et C. Intanagonwiwat, "Energy-Efficient Gradient Time Synchronization for Wireless Sensor Networks", in *2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks*, 2010, p. 124-129. doi : 10.1109/CICSyN.2010.14 (cf. p. 22).

[51] L. Schenato et F. Fiorentin, "Average TimeSynch : A consensus-based protocol for clock synchronization in wireless sensor networks", *Automatica*, t. 47, n° 9, p. 1878-1886, 2011 (cf. p. 22, 29).

[52] J. Wu, L. Zhang, Y. Bai et Y. Sun, "Cluster-Based Consensus Time Synchronization for Wireless Sensor Networks", *IEEE Sensors J.*, t. 15, n° 3, p. 1404-1413, mar. 2015. doi : 10.1109/JSEN.2014.2363471 (cf. p. 23).

[53] J. Koo, R. K. Panta, S. Bagchi et L. Montestruque, “A tale of two synchronizing clocks”, in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, 2009, p. 239-252 (cf. p. 23, 30).

[54] S. Gao, X. Zhang, C. Du et Q. Ji, “A multichannel low-power wide-area network with high-accuracy synchronization ability for machine vibration monitoring”, *IEEE Internet Things J.*, t. 6, n° 3, p. 5040-5047, juin 2019 (cf. p. 24, 30, 31).

[55] IEEE, *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks-Specific Requirements - Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std. 2012 (cf. p. 24, 31).

[56] ——, *IEEE Standard for Low-Rate Wireless Networks*, IEEE Std. 2020 (cf. p. 24).

[57] A. Mahmood, R. Exel, H. Trsek et T. Sauter, “Clock Synchronization Over IEEE 802.11—A Survey of Methodologies and Protocols”, *IEEE Trans. Ind. Informat.*, t. 13, n° 2, p. 907-922, avr. 2017. doi : 10.1109/TII.2016.2629669 (cf. p. 24).

[58] A. Mahmood, R. Exel et T. Sauter, “Performance of IEEE 802.11’s Timing Advertisement Against SyncTSF for Wireless Clock Synchronization”, *IEEE Trans. Ind. Informat.*, t. 13, n° 1, p. 370-379, fév. 2017. doi : 10.1109/TII.2016.2521619 (cf. p. 24).

[59] J. Kannisto et al., “Precision Time Protocol Prototype on Wireless LAN”, in *Telecommunications and Networking - ICT 2004*, Berlin, Heidelberg : Springer Berlin Heidelberg, 2004, p. 1236-1245 (cf. p. 24).

[60] J. Kannisto, T. Vanhatupa, M. Hannikainen et T. D. Hamalainen, “Software and hardware prototypes of the IEEE 1588 precision time protocol on wireless LAN”, in *2005 14th IEEE Workshop on Local Metropolitan Area Networks*, 2005, 6 pp.-6. doi : 10.1109/LANMAN.2005.1541513 (cf. p. 24).

[61] T. Cooklev, J. C. Eidson et A. Pakdaman, “An implementation of IEEE 1588 over IEEE 802.11 b for synchronization of wireless local area network nodes”, *IEEE Transactions on Instrumentation and Measurement*, t. 56, n° 5, p. 1632-1639, 2007 (cf. p. 24).

[62] A. Mahmood, G. Gaderer, H. Trsek, S. Schwalowsky et N. Kerö, “Towards high accuracy in IEEE 802.11 based clock synchronization using PTP”, in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, IEEE, 2011, p. 13-18 (cf. p. 24).

[63] A. Mahmood, R. Exel et T. Sauter, “Delay and jitter characterization for software-based clock synchronization over WLAN using PTP”, *IEEE Transactions on industrial informatics*, t. 10, n° 2, p. 1198-1206, 2014 (cf. p. 25).

[64] D. Anand, D. Sharma, Y. Li-Baboud et J. Moyne, “EDA performance and clock synchronization over a wireless network : Analysis, experimentation and application to semiconductor manufacturing”, in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, IEEE, 2009, p. 1-6 (cf. p. 25).

[65] A. Mahmood, G. Gaderer et P. Loschmidt, “Clock synchronization in wireless LANs without hardware support”, in *2010 IEEE International Workshop on Factory Communication Systems Proceedings*, IEEE, 2010, p. 75-78 (cf. p. 25).

[66] G. Cena, S. Scanzio, A. Valenzano et C. Zunino, “Implementation and evaluation of the reference broadcast infrastructure synchronization protocol”, *IEEE Transactions on Industrial Informatics*, t. 11, n° 3, p. 801-811, 2015 (cf. p. 25).

[67] F. Ferrari, M. Zimmerling, L. Thiele et O. Saukh, “Efficient network flooding and time synchronization with glossy”, in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IEEE, 2011, p. 73-84 (cf. p. 25).

[68] C. Lenzen, P. Sommer et R. Wattenhofer, “PulseSync : An Efficient and Scalable Clock Synchronization Protocol”, *IEEE/ACM Trans. Netw*, t. 23, n° 3, p. 717-727, juin 2015. doi : 10.1109/TNET.2014.2309805 (cf. p. 25).

[69] D. Cox, E. Jovanov et A. Milenkovic, “Time synchronization for ZigBee networks”, in *Proc. South. Symp. Sys. Theory*, Tuskegee, AL, USA, Mar. 2005, p. 135-138 (cf. p. 25).

[70] V.-H. Nguyen et J.-Y. Pyun, “Location detection and tracking of moving targets by a 2D IR-UWB radar system”, *Sensors*, t. 15, n° 3, p. 6740-6762, 2015 (cf. p. 26).

[71] B. Silva, Z. Pang, J. Åkerberg, J. Neander et G. Hancke, “Experimental study of UWB-based high precision localization for industrial applications”, in *2014 IEEE International Conference on Ultra-WideBand (ICUWB)*, IEEE, 2014, p. 280-285 (cf. p. 26).

[72] B. Silva et G. P. Hancke, “IR-UWB-based non-line-of-sight identification in harsh environments : Principles and challenges”, *IEEE Transactions on Industrial Informatics*, t. 12, n° 3, p. 1188-1195, 2016 (cf. p. 26).

[73] V. Srivastava et M. Motani, “Cross-layer design : a survey and the road ahead”, *IEEE communications magazine*, t. 43, n° 12, p. 112-119, 2005 (cf. p. 26).

[74] S. Niranjayan et A. F. Molisch, “Ultra-wide bandwidth timing networks”, in *2012 IEEE International Conference on Ultra-Wideband*, IEEE, 2012, p. 51-56 (cf. p. 26).

[75] M. Segura, S. Niranjayan, H. Hashemi et A. F. Molisch, “Experimental demonstration of nanosecond-accuracy wireless network synchronization”, in *2015 IEEE International Conference on Communications (ICC)*, 2015, p. 6205-6210. doi : 10.1109/ICC.2015.7249312 (cf. p. 26, 31).

[76] E. D. Kaplan, éd., *Understanding GPS : principles and applications*, Boston : Artech House, 1996, 554 p., isbn : 978-0-89006-793-2 (cf. p. 26).

[77] *NEO6T Specifications*, 13003351, ublox. adresse : https://www.u-blox.com/sites/default/files/products/documents/LEA-NEO-6T_ProductSummary%5C_(UBX-13003351).pdf (cf. p. 27, 39, 44, 47, 48, 60-62, 105, 114).

[78] R. Scopigno et H. A. Cozzetti, “GNSS synchronization in vanets”, in *2009 3rd International Conference on New Technologies, Mobility and Security*, IEEE, 2009, p. 1-5 (cf. p. 27).

[79] K. F. Hasan, Y. Feng et Y.-C. Tian, “GNSS time synchronization in vehicular ad-hoc networks : benefits and feasibility”, *IEEE Transactions on Intelligent Transportation Systems*, t. 19, n° 12, p. 3915-3924, 2018 (cf. p. 27).

[80] K. F. Hasan, C. Wang, Y. Feng et Y.-C. Tian, “Time synchronization in vehicular ad-hoc networks : A survey on theory and practice”, *Vehicular communications*, t. 14, p. 39-51, 2018 (cf. p. 27).

[81] V. Le Cam, A. Bouche et D. Pallier, “Wireless Sensors Synchronization : an accurate and deterministic GPS-based algorithm”, in *Proc. IWSHM* (cf. p. 27, 31, 123).

[82] E. Sazonov, V. Krishnamurthy et R. Schilling, “Wireless intelligent sensor and actuator network-a scalable platform for time-synchronous applications of structural health monitoring”, *Structural Health Monitoring*, t. 9, n° 5, p. 465-476, 2010 (cf. p. 27, 31).

[83] R. E. Kim, J. Li, B. F. Spencer, T. Nagayama, K. A. Mechitov et al., “Synchronized sensing for wireless monitoring of large structures”, *Smart Struct. Syst*, t. 18, n° 5, p. 885-909, 2016 (cf. p. 28, 31).

[84] Y.-C. Wu, Q. Chaudhari et E. Serpedin, “Clock synchronization of wireless sensor networks”, *IEEE Signal Process. Mag.*, t. 28, n° 1, p. 124-138, jan. 2011. doi : 10.1109/MSP.2010.938757 (cf. p. 28).

[85] S. Ganeriwal, I. Tsigkogiannis, H. Shim, V. Tsiatsis, M. B. Srivastava et D. Ganesan, “Estimating clock uncertainty for efficient duty-cycling in sensor networks”, *IEEE/ACM transactions on Networking*, t. 17, n° 3, p. 843-856, 2008 (cf. p. 29, 30).

[86] K. S. Kim, S. Lee et E. G. Lim, “Energy-efficient time synchronization based on asynchronous source clock frequency recovery and reverse two-way message exchanges in wireless sensor networks”, *IEEE Transactions on Communications*, t. 65, n° 1, p. 347-359, 2016 (cf. p. 29).

[87] T. Schmid, Z. Charbiwala, R. Shea et M. B. Srivastava, “Temperature compensated time synchronization”, *IEEE Embedded Systems Letters*, t. 1, n° 2, p. 37-41, 2009 (cf. p. 29).

[88] Z. Yang, L. Cai, Y. Liu et J. Pan, “Environment-aware clock skew estimation and synchronization for wireless sensor networks”, in *Proc. IEEE INFOCOM*, p. 1017-1025 (cf. p. 29).

[89] Z. Yang, L. He, L. Cai et J. Pan, “Temperature-assisted clock synchronization and self-calibration for sensor networks”, *EEE Trans. Wireless Commun.*, t. 13, n° 6, p. 3419-3429, juin 2014. doi : 10.1109/TWC.2014.051414.130270 (cf. p. 29).

[90] M. Jin, D. Fang, X. Chen, Z. Yang, C. Liu et X. Yin, “Voltage-aware time synchronization for wireless sensor networks”, *International Journal of Distributed Sensor Networks*, t. 10, n° 7, p. 285 265, 1er juil. 2014, issn : 1550-1477. doi : 10.1155/2014/285265. adresse : http://dx.doi.org/10.1155/2014/285265 (visité le 09/08/2017) (cf. p. 29).

[91] A. Rowe, V. Gupta et R. Rajkumar, “Low-power clock synchronization using electromagnetic energy radiating from ac power lines”, in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, 2009, p. 211-224 (cf. p. 29).

[92] A. Jain et E. Y. Chang, “Adaptive sampling for sensor networks”, in *Proc. 1st Int. Workshop on Data Management for Sensor Networks (DMSN'04)*, p. 10-16 (cf. p. 29).

[93] B. R. Hamilton, X. Ma, Q. Zhao et J. Xu, “ACES : adaptive clock estimation and synchronization using Kalman filtering”, in *Proc. ACM MobiCom*, p. 152-162 (cf. p. 30).

[94] J. Shannon, H. Melvin et A. G. Ruzzelli, “Dynamic Flooding Time Synchronisation Protocol for WSNs”, in *Proc. IEEE GLOBECOM*, Anaheim, CA, USA, Mar. 2012, p. 365-371 (cf. p. 30).

[95] M. Ruiz-Sandoval, T. Nagayama et B. Spencer Jr, “Sensor development using Berkeley Mote platform”, *J. Earthquake Engineering*, t. 10, n° 02, p. 289-309, mar. 2006. doi : 10.1142/S136324690600261X (cf. p. 31).

[96] *Spartan 6 Datasheet*, DS160, Xilinx. adresse : https://www.xilinx.com/support/documentation/data_sheets/ds160.pdf (cf. p. 38).

[97] *PapilioPro Specifications*, Gadget Factory. adresse : https://papilio.cc/index.php?n=Papilio.PapilioPro (cf. p. 41, 92, 98).

[98] *GPS antennas - RF design considerations for u-blox GNSS receivers*, 15030289, ublox. adresse : https://www.u-blox.com/sites/default/files/products/documents/GNSS-Antennas_AppNote_%28UBX-15030289%29.pdf (cf. p. 60).

[99] J. A. Klobuchar, “Ionospheric time-delay algorithm for single-frequency GPS users”, *IEEE Transactions on aerospace and electronic systems*, n° 3, p. 325-331, 1987 (cf. p. 62).

[100] G. Welch, G. Bishop et al., *An introduction to the Kalman filter*, 1995 (cf. p. 74).

[101] I. Reid. (2001). “Estimation ii”, adresse : http://www.robots.ox.ac.uk/~ian/Teaching/Estimation/Lec%20tureNotes2.pdf (cf. p. 74, 75).

[102] *TMP112 Datasheet*, Texas Instruments. adresse : https://www.ti.com/lit/ds/symlink/tmp112.pdf?ts=1605406700481 (cf. p. 92).

[103] G. D. Hutcheson, “Ordinary least-squares regression”, *L. Moutinho and GD Hutcheson, The SAGE dictionary of quantitative management research*, p. 224-228, 2011 (cf. p. 93).

[104] *AST3TQ Datasheet*, Abracon. adresse : https://eu.mouser.com/datasheet/2/3/AST3TQ-50-553285.pdf (cf. p. 98, 111).

[105] *ASV32 Datasheet*, Abracon. adresse : https://www.mouser.fr/datasheet/2/3/ASV-24925.pdf (cf. p. 111).

[106] (). “Lattice ICE40 datasheet”, adresse : http://www.latticesemi.com/Products/FPGAandCPLD/iC%20E40?ActiveTab=Data+Sheet%5C#%5C_21E33C7EC0BD48AA80%20FE384ED73CC895 (cf. p. 113).

[107] (). “TinyFPGA Bx Specifications”, adresse : https://github.com/tinyfpga/TinyFPGA-BX (cf. p. 114).

[108] *Fluke 8840A Datasheet*, Fluke. adresse : https://www.valuetronics.com/pub/media/vti/datasheets/Fluke%208840%20Series.pdf (cf. p. 123).

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE NANTES

Titre : SENTAUR

Mots-clés : Réseaux de capteurs sans fil (WSN), Synchronisation d'horloge, Géolocalisation et Navigation par un Système de Satellites (GNSS), Efficience énergétique, Surveillance de l'état de santé des structures (SHM)

Résumé : Les WSNs sont de plus en plus utilisés pour le contrôle de santé des structures car c'est une solution non-invasive plus simple à mettre en œuvre que les solutions filaires. La majorité de ces applications utilisent la fusion de données récoltées par des capteurs répartis sur la structures à surveiller/-diagnostiquer. Pour pouvoir comparer ou fusionner des données dépendantes du temps provenant de différents capteurs ceux-ci ont besoin d'être synchronisés. La précision de synchronisation nécessaire dépend de l'application, pour des applications comme la surveillance des lignes à haute tension elle peut atteindre la dizaine de nanosecondes. Les topologies des réseaux sont aussi très variées suivant les applications, la taille de ces réseaux atteint souvent l'ordre du kilomètre pour les grandes infrastructures. Pour ces cas d'applications, cette thèse propose une architecture de noeud spécifique couplée à un protocole de synchronisation basé sur le GPS. Ce protocole est facilement déployable sur toutes les structures extérieures car indépendant de la topologie réseau, robuste à la perte de noeud et précis à +/- 50 nanosecondes. Une solution adaptative est ensuite évaluée afin de minimiser la consommation du récepteur GPS en fonction de la précision de synchronisation attendue. L'évaluation expérimentale de cette solution démontre une diminution de la consommation de 70% à 99% pour des précisions de synchronisation de 50 ns à 1.4 μs.

Title: SENTAUR

Keywords: Wireless Sensors Network (WSN), Clock synchronization, Satellite Navigation (GNSS), Energy efficiency, Structural Health Monitoring (SHM)

Abstract: The usage of WSN over wired solutions for SHM is increasing due to their ease of setup and non-invasive nature. Most SHM applications require data fusion from multiple nodes scattered across the structure under evaluation. In order to use this time-dependent data, the nodes need to be synchronized. The synchronization accuracy is determined by the application needs, it can go as low as tens of nanoseconds for specific use cases like power lines monitoring. These applications also present a wide variety of network topologies. For large infrastructures, the size of the network is substantial, as it can cover several kilometers. A specific node architecture paired with a GPS synchronization scheme is presented in this thesis for the aforementioned use cases. This synchronization solution allows for a +/- 50 nanoseconds accuracy, is easy to set up, and is robust to node failure since it does not depend on the network topology. An adaptable synchronization solution is then evaluated to minimize the GPS receiver power consumption under a synchronization accuracy goal. Experimental evaluation shows a 70% to 99% decrease of the energy consumption for an accuracy of 50 ns to 1.4 μs.