Le génie pour l'industrie

RAPPORT TECHNIQUE
PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
DANS LE CADRE DU COURS
GTI792 PROJET DE FIN D'ÉTUDES EN GÉNIE DES TI

FIELDTEST

GUILLAUME PELLETIER
PELG09129105

DÉPARTEMENT DE GÉNIE LOGICIEL ET DES TI

Professeur-superviseur

Alain April

MONTRÉAL, 9 AOÛT 2016
ÉTÉ 2016

REMERCIEMENTS

Pour ce projet, je dois remercier l'équipe d'OPDAQ system pour m'avoir accompagné tout au long de ce projet et de m'avoir prêté du matériel afin de pouvoir poursuivre ce projet.

RAPPORT FINAL

GUILLAUME PELLETIER
PELG09129105

RÉSUMÉ

Ce rapport présente le travail réalisé sur le logiciel FieldTest qui permet la lecture simultanée de plusieurs appareils d'acquisition de données liée au monitorage de composantes de navire. Par exemple, l'acquisition de données liées aux mesures de torque, de RPM d'un arbre mécanique. Ce projet comporte le travail réalisé avec un équipement d'acquisition de données, le TPM2 dont le protocole de communication vient juste d'être établi et vient juste d'être mis sur le marché. Aussi, le travail sur l'amélioration de la performance de l'acquisition de donnée haute fréquence qui inclut la transformée de Fourier permettant de voir les pics d'amplitude des données acquises afin de bien déceler les problèmes. De plus, il comporte l'intégration d'un système permettant de faire des rapports liés aux données journalisées afin de garder une trace visuelle des analyses.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

Acronymes	Mot complet
FFT	Fast Fourier Transform
RPM	revolution per minute (tour par minute)
XML	Extensible Markup Language
MSI	Microsoft Installer
GPU	Graphic Processing Unit
TI	Technologie de l'information
USB	Universal Serial Bus
FTDI	Futur Technology Devices International
RDLC	Report Definition Language Client
TSV	Tab-separated values

LISTE DES SYMBOLES ET UNITÉS DE MESURE

Symboles	Unités de mesure
Torque	ft-lbs ou N-M
Shaft Speed	RPM
Power	hp ou watts

INTRODUCTION

Dans le milieu naval, il y a souvent de la maintenance à faire pour régler certains problèmes ou en prévenir d'autres. Pour faire cette maintenance, des appareils d'acquisition de données sont utilisés afin de mesurer de potentielles composantes de navire défaillantes. Par contre, pour se faire la personne faisant la maintenance doit être sur place afin de pouvoir voir les données qui sont affichées sur l'interface de l'appareil. De plus, il est impossible de visualiser un ensemble de données, car il y a seulement une donnée à la fois d'affichée sur l'interface de l'appareil. Ainsi, pour combler à ce problème un logiciel permettant de journaliser et d'analyser des données provenant d'appareils monitorant des composantes de navire en temps réel est la solution idéale. Donc, pour ce projet, le but premier est de continuer la refonte d'une nouvelle version plus à jour du logiciel FieldTest en ajoutant de nouvelles fonctionnalités.

Ainsi, ce présent document se veut un rapport technique montrant le divers aspect technique du projet. Ainsi, dans un premier temps la vision du projet sera présentée. Ensuite, suivront les exigences liées à ce projet. Après, une conception haut niveau de l'architecture sera présentée afin de montrer de façon générale le projet à réaliser. Ensuite, suivront les trois itérations du projet comportant chacune une analyse, une conception, une implémentation et finalement une revue et test de cette dernière. La première itération se rapporte à l'acquisition de donnée, la détection des appareils d'acquisition et l'ajout du nouvel appareil le TPM2. La deuxième itération, elle touchera à l'acquisition haute fréquence du logiciel plus particulièrement à l'amélioration de la transformée de Fourier. La dernière et troisième itération abordera l'aspect de l'analyse des différents types de données journalisées incluant la création de rapport, l'amélioration des interfaces utilisateurs et l'installateur du logiciel pour son déploiement. Pour ce rapport, un sommaire et une conclusion seront présentés afin de revenir sur l'ensemble du projet.

CHAPITRE 1

LA VISION ET LES OBJECTIFS DU PROJET

1.1 La mise en contexte du projet

Le logiciel FieldTest est un logiciel permettant d'analyser des données provenant d'appareils d'acquisition de données connecté à certaines composantes de navire afin de monitorer la performance et les problèmes pouvant survenir. Par exemple, l'utilisation du TPM2 sur un arbre mécanique afin d'y mesurer le torque, le RPM et les puissances générées. Ainsi, le logiciel journalise toutes les données durant toute la période où l'utilisateur lance l'acquisition de données, il peut à tout moment vérifier les données entrantes via un graphique affichant la courbe de données en temps réel. Le logiciel peut également faire fonctionner plusieurs appareils d'acquisition en même temps afin d'avoir une analyse avec plus d'un type de données à la fois qui seront également visibles dans le même graphique selon leur sélection. De plus, il peut aussi ouvrir des données journalisées pour vérifier à quel moment un problème est survenu en regardant les anomalies dans les données en cas où ce dernier n'était pas présent lors de l'acquisition en temps réel. Il peut également pour les appareils à haute fréquence d'acquisitions de données vérifier via une courbe les pics d'amplitude en fonction de la fréquence pour évaluer plus facilement les anomalies. Aussi, il permet la configuration des caractéristiques des appareils d'acquisition directement à partir de celui-ci. Le tout fonctionne via des câbles USB de l'ordinateur à l'appareil d'acquisition en question.

1.2 La problématique du projet

Le logiciel FieldTest antérieur permettait seulement de faire l'acquisition de données d'un nombre restreint d'appareils. De plus, ce dernier a été construit avec un langage de programmation qui est difficilement supportable avec Window 8 et 10. Ainsi, la nouvelle version du logiciel permettra de centraliser l'acquisition de données à un seul logiciel en permettant l'ajout de nouveaux appareils d'acquisition plus facilement. Aussi, le nouveau logiciel permettra entre autres de pouvoir faire l'acquisition de données du TPM2, un nouvel appareil dont aucun logiciel n'existe pour faire l'interprétation de données avec une interface utilisateur. De plus, ce dernier a eu un gain de popularité au cours des derniers mois ainsi la réalisation de ce logiciel est devenue très importante afin de fournir une manière d'analyser les données aux utilisateurs. En autre mot, ce logiciel permettra à un utilisateur voulant monitorer certaines caractéristiques d'un navire de pouvoir observer les données provenant de divers appareils de mesures au même endroit.

1.3 Les objectifs du projet

Pour le projet, le premier objectif était de revoir l'architecture pour qu'il soit plus facile de faire l'ajout de nouveaux appareils d'acquisition de données et pour améliorer la performance en général. Ensuite, l'implémentation du protocole du TPM2 au logiciel pour qu'il soit fonctionnel pour l'acquisition de données standard et celle liée à l'acquisition de données à haute fréquence utilisant la transformée de Fourier afin de voir les pics d'amplitudes pour mieux observer les anomalies. Une autre partie importante du projet était de retravailler l'acquisition des appareils, car la performance liée au « multithreading » lors de l'acquisition de données avait quelques problèmes de performance. Ce qui incluait également l'acquisition à haute fréquence où la performance de la FFT n'était pas très au adapté au contexte d'utilisation du logiciel. De plus, l'ajout d'une manière de bien monitorer les appareils d'acquisition de données en tout temps dans les modules d'acquisition de données du logiciel plus particulièrement la détection des appareils.

Un autre objectif de ce projet était de retravailler également les interfaces utilisateurs afin de les rendre d'une meilleure qualité. Aussi, l'ajout d'une fonctionnalité permettant de faire des analyses et des rapports à partir des données journalisées de l'acquisition standard, à haute fréquence, des erreurs survenues et de dépassement de seuils. Pour finir, le dernier objectif du projet était de faire un installateur pour l'installation du logiciel afin de déployer la première version bêta du logiciel.

CHAPITRE 2

LES EXIGENCES DU PROJET (SRS)

2.1 Les exigences fonctionnelles

Table 1 - Exigences fonctionnelles

ID	Exigence
EF01	Le logiciel doit permettre à l'utilisateur de faire de l'acquisition de données via divers appareils de toutes sortes, et ce, en temps réel.
EF02	Le logiciel doit permettre l'acquisition de données de plusieurs appareils en même temps.
EF03	Le logiciel doit via les ports USB détecter automatiquement les appareils.
EF04	Le logiciel doit afficher via des graphiques la courbe des données entrantes de deux appareils maximum à la fois par graphique.
EF05	Le logiciel doit afficher pour les appareils à haute fréquence la courbe de l'amplitude en fonction de la fréquence des données entrantes.
EF06	Le logiciel doit permettre de journaliser les données d'acquisition des différents appareils et les données d'amplitude en fonction de la fréquence pour les appareils à haute fréquence.
EF07	Le logiciel doit permettre de pouvoir démarrer, arrêter l'acquisition de données des appareils.
EF08	Le logiciel doit permettre la configuration des appareils et en garder une trace pour les utilisations futures en fichier XML.

EF09	Le logiciel doit après 24 h d'acquisition de données créer un niveau fichier de journalisation.
EF10	Le logiciel doit permettre de faire un zoom sur une certaine zone sur les graphes afin de voir plus en détail des données.
EF11	Le logiciel doit permettre pour l'acquisition haute fréquence de faire un autre type de journalisation lié aux données dépassant un certain seuil en gardant une trace d'un certain nombre de données avant et après la donnée dépassant le seuil.
EF12	Le logiciel doit également permettre la journalisation des erreurs provenant des appareils d'acquisition.
EF13	Le logiciel doit permettre de modifier les configurations des graphiques d'acquisition de données.
EF14	Le logiciel doit permettre pour l'acquisition de données standards, de faire plusieurs instances de graphique à la fois, mais contenant toujours deux appareils maximum par graphique pour respecter EF04.
EF15	Le logiciel doit permettre de définir les chemins où les fichiers de journalisations seront enregistrés.
EF16	Le logiciel doit permettre à l'utilisateur de voir l'état des appareils connectés avant de passer à l'acquisition de données.
EF17	Le logiciel doit permettre l'ouverture des différents fichiers journalisés afin d'en consulter les données via les mêmes graphiques que lors de l'acquisition de ces dernières.
EF18	Le logiciel doit permettre lors de la consultation des données journalisées d'afficher la moyenne de ces

	dernières.
EF19	Le logiciel doit permettre lors de la consultation des données journalisées de faire une autre sauvegarde de données, mais en contenant seulement la plage de données sélectionnées.
EF20	Le logiciel doit permettre de faire des rapports liés aux données journalisés.
EF21	Le logiciel doit permettre d'ouvrir les fichiers de données journalisés liés au dépassement de seuils et aux erreurs survenues des appareils.

2.2 Les attributs de qualité

Table 2 - Attributs de qualité

ID	Attribut de qualité	Sous-famille	Scénario	Priorité	Impact
AQ01	Maintenabilité	Modularité	Le système doit être bien séparé en modules afin qu'un ajout d'un nouvel appareil d'acquisition de données soit fait en moins de 3 heures lors de son implémentation.	Haute	Majeur
AQ02		Testabilité	Le système doit être en mesure d'effectuer des tests en détectant les fautes sans trop d'efforts.	Haute	Moyen
AQ03	Compatibilité	Interopérabilité	Le système de simulation doit supporter plusieurs types d'appareils et de communication à différente fréquence d'acquisition.	Haute	Majeur
AQ04	Fiabilité	Tolérance aux pannes	Le système doit limiter son taux de défaillance lié à l'acquisition de données à 5 %.	Moyen	Moyen

AQ05			Le système doit être en mesure de recevoir 90 % des valeurs provenant des appareils d'acquisitions.	Majeur	Majeur
AQ06		Recouvrement	En cas de défaillance mineure, le système doit continuer à faire l'acquisition de données.	Majeur	Majeur
AQ07			En cas de problèmes liés à un appareil de données, le système doit faire reprendre l'acquisition de données de ce dernier instantanément.	Moyen	Majeur

2.3 Contraintes

Table 3 - Contraintes

ID	Description
CON01	Le langage de programmation utilisé doit être c#.
CON02	Les appareils d'acquisition de données suivants doivent obligatoirement fonctionner avec le logiciel : TT10K, TPM2, GPS, ADAMs.

2.4 Les cas d'utilisations

Voir **Annexe 12** pour le document montrant les cas d'utilisation.

CHAPITRE 3
CONCEPTION DE L'ACHITECTURE HAUT NIVEAU

3.1 Vue modulaire

3.1.1 Objectif de la vue

Cette vue modulaire permet de bien séparer le travail à faire en module et il est également plus facile de voir les différentes relations que les modules ont entre eux.

3.1.2 Illustration de la vue

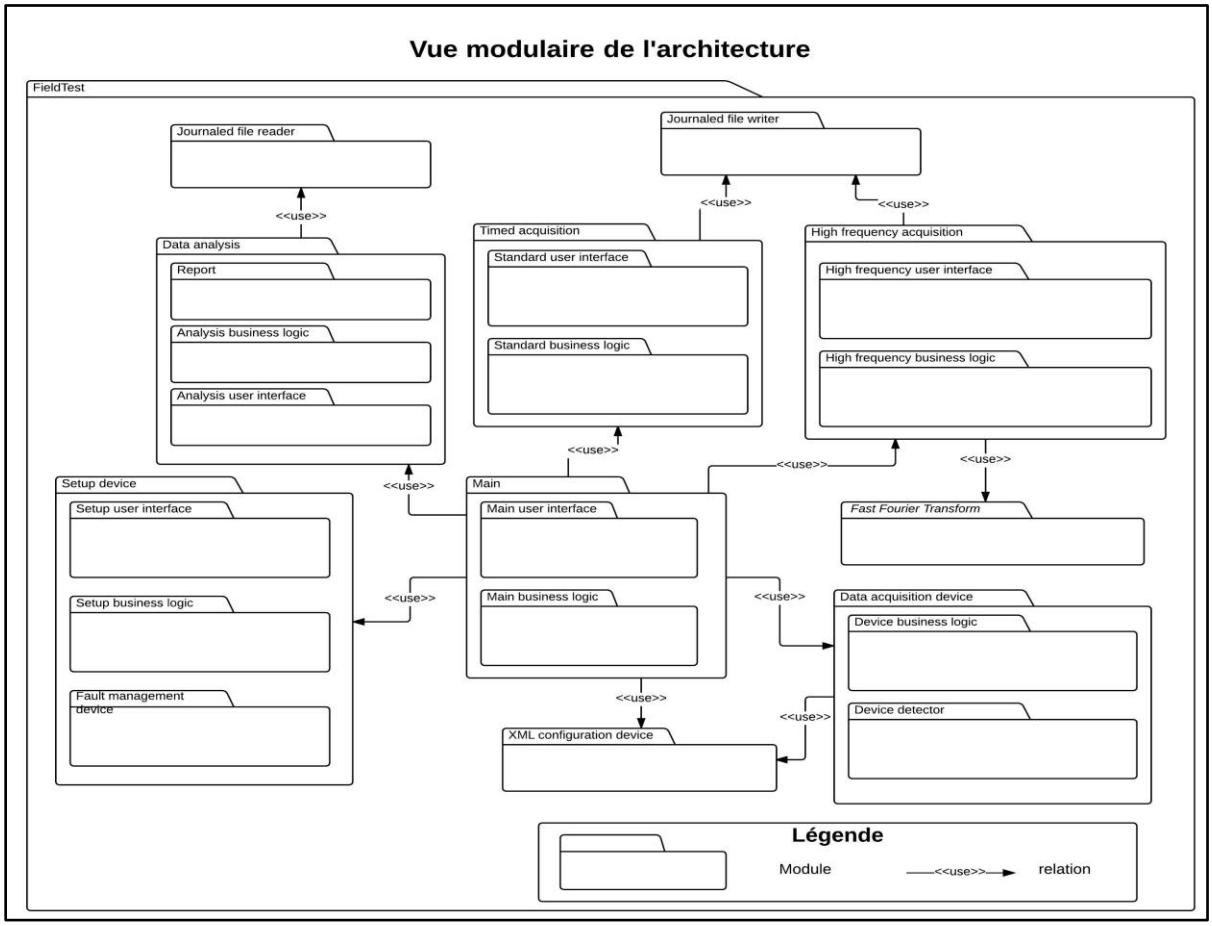

Figure 1 - Vue modulaire de l'architecture

Table 4 - Tableau de description de la vue modulaire

Élément(s)	Description(s)
FieldTest	Représente le module de l'ensemble du système.
Journaled filed reader	Représente le module qui va s'occuper de faire la lecture des fichiers journalisés de types standard, d'amplitude en fonction de la fréquence, d'erreur et dépassement de seuil.
Journaled filed writer	Représente le module qui va s'occuper de faire l'écriture des fichiers journalisés de types standard, d'amplitude en fonction de la fréquence, d'erreur et dépassement de seuil.
Data analysis	Représente le module s'occupant de faire l'analyse provenant de la journalisation et de générer les rapports.
Report	Représente le module s'occupant de la génération de rapport de types donnés standard, d'amplitude en fonction de la fréquence, d'erreur et dépassement de seuil.
Analysis business logic	Représente le module s'occupant de la logique d'affaire de l'analyse de données. Il s'occupe en autre des statistiques liées aux données.
Analysis user interface	Représente toutes les interfaces utilisateurs liées à l'analyse.
Timed acquisition	Représente le module lié à l'acquisition standard de données.

Standard user interface	Représente toutes les interfaces utilisateurs liées à l'acquisition de données standard.
Standard business logic	Représente la logique d'affaire liée à l'acquisition de données standard.
High frequency acquisition	Représente le module lié à l'acquisition des appareils à haute fréquence et exécutant la FFT pour avoir les données de l'amplitude en fonction de la fréquence.
High frequency user interface	Représente toutes les interfaces utilisateurs liées à l'acquisition de données d'appareil à haute fréquence.
High frequency business logic	Représente la logique d'affaire liée à l'acquisition de données d'appareil à haute fréquence.
Setup device	Représente le module qui va s'occuper de voir l'état des appareils et de faire la configuration des appareils.
Setup user interface	Représente toutes les interfaces utilisateurs liées à la configuration et à l'état des appareils.
Setup business logic	Représente la logique d'affaire liée à la configuration et à l'état des appareils.
Fault management device	Représente le module qui va s'occuper de gérer les fautes et la reprise liée à l'acquisition de données.
Data acquisition device	Représente le module qui va s'occuper de la détection des appareils et définir les différents protocoles d'acquisition de données.
Device business logic	Représente la logique d'affaire liée aux

	appareils d'acquisition de données. Le protocole de communication du TPM2 en autre.
Device detector	Représente le module qui va s'occuper de faire la détection des appareils connectés au PC.
XML configuration device	Représente le module s'occupant des fichiers XML de configuration des appareils.
Fast Fourrier Transform	Représente le module qui va s'occuper de faire la FFT sur les données standard afin d'avoir les données d'amplitude en fonction de la fréquence.
Main	Représente le module central permettant d'exécuter les différentes fonctionnalités du système.
Main user interface	Représente toutes les interfaces utilisateurs liées à l'interface usager principale.
Main business logic	Représente la logique d'affaire liée à l'utilisation de toutes les fonctionnalités du système.

3.1.3 Description de la vue

Dans cette vue, il y a un module central *Main* permettant d'exécuter toutes les fonctionnalités du système et les modules représentant chaque fonctionnalité du système. Le module *Timed acquisition* et *High frequency acquisition* pour la gestion des deux types d'acquisition soit l'acquisition de données standard et l'acquisition de données à haute fréquence utilisant le module *Fast Fourier Trasform* pour recevoir les données de l'amplitude en fonction de la

fréquence pour les évaluations des pics d'amplitudes. Ensuite, il y a le module *Setup device* pour la gestion de l'état des appareils en utilisant une tactique permettant de monitorer l'état des appareils en tout temps afin de détecter les erreurs et la configuration de ceux-ci. Le module *Data acquisition device* pour gérer la détection des appareils et la communication de chaque appareil.

De plus, il y a le module *Journaled filed reader* et *Journaled filed writer* pour gérer l'écriture et la lecture des fichiers de journalisation de données standard, d'amplitude en fonction de la fréquence, d'erreur et de dépassement de seuil. Le module *Data analysis* contient tout ce qui touche à la génération de rapport et à l'analyse statistique des données journalisées. Il y a également le module *XML configuration device* qui s'occupe de sauvegarder en format XML la configuration des appareils.

Ensuite, pour la fiabilité du système le module *Fault management device* permet de gérer les défaillances provenant des appareils avec une tactique qui va réessayer de toujours redemander un nombre de fois les données de l'appareil défaillant pour ne pas arrêter l'envoi de données au système et afin d'atteindre le 90 % de données reçues par le système. Également, ce module comporte une tactique de reprise de l'état initial des appareils afin de revenir à l'état de marche du système pour continuer l'acquisition de données dans le cas où la tactique de redemande de données a échoué.

3.2 Vue composants et connecteurs

3.2.1 Objectif de la vue

Pour la famille composants et connecteurs, le style architectural *Publisher-Subscriber* est utilisé afin de montrer l'interaction entre les diverses composantes de l'architecture. De plus, cette vue est la plus pertinente dans ce cas-ci en raison du fait que le système va contenir un nombre important d'appareils d'acquisition de données en plus d'avoir de nouveaux appareils pouvant se rajouter dans l'avenir.

3.2.2 Illustration de la vue

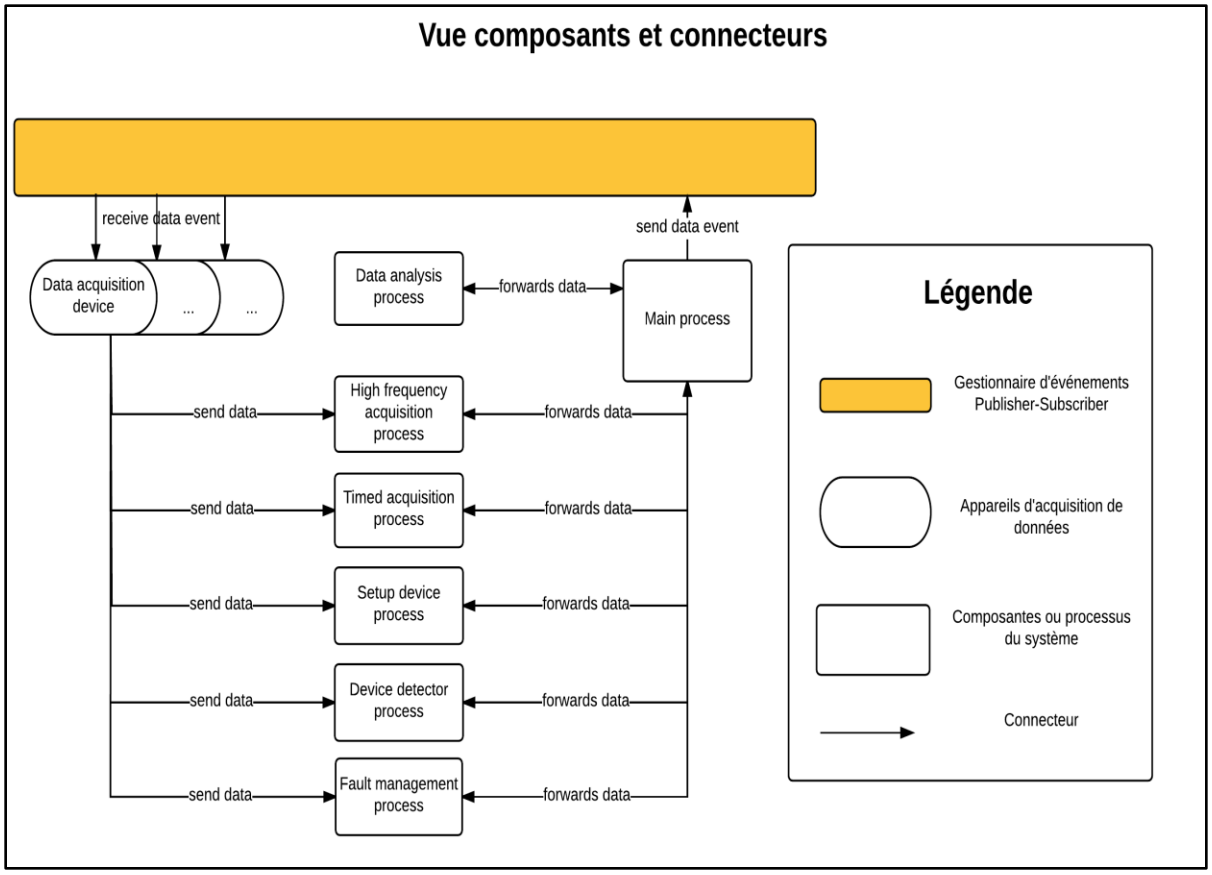

Figure 2 - Vue composants et connecteurs

Table 5 - Tableau de description de la vue composants et connecteurs

Élément(s)	Description(s)
Data acquisition device	Représente les appareils d'acquisition de données.
Main process	Représente le processus principal du système qui exécutera tous les autres processus et qui publiera des événements aux appareils d'acquisition pour l'envoi de données aux autres processus.

Data analysis process	Représente le processus regroupant tout ce qui touche à l'ouverture, l'analyse statistique et la génération de rapport des fichiers journalisés.
Device detector process	Représente le processus détectant et créant les appareils d'acquisition au niveau logiciel.
Fault management process	Représente le processus qui va s'occuper de gérer les fautes et la reprise liée à l'acquisition de données.
High frequency acquisition process	Représente le processus lié à l'acquisition des appareils à haute fréquence et exécutant la FFT pour avoir les données de l'amplitude en fonction de la fréquence.
Timed acquisition process	Représente le processus lié à l'acquisition standard de données.
Setup device process	Représente le processus qui va s'occuper de voir l'état des appareils, de faire la configuration des appareils et de gérer la configuration XML des appareils.
Publisher-Subscriber	Le gestionnaire d'événement qui va s'assurer que tous les appareils d'acquisition envoient des données quand le *Main process* l'aura demandé.

3.2.3 Description de la vue

Cette description de vue sert seulement à bien définir chaque composante et interaction entre elles du système sans plus de détails. L'analyse plus détaillée de ces éléments sera présentée dans les prochains chapitres du document.

La vue composants et connecteurs est principalement composée du style architectural *Publisher-Subscriber* étant donné que les appareils d'acquisition de données sont des composantes primordiales au fonctionnement du système. Cette vue démontre que le système peut gérer plusieurs appareils d'acquisition et puis facilite leur communication par le gestionnaire d'événements de *Publisher-Subscriber*. Ce changement de structure architecturale doit aider à découpler certaines classes de l'architecture actuelle en raison des appels implicites des évènements.

De plus, le style architectural *Publisher-Subscriber* implique l'utilisation d'événements transmis par le gestionnaire d'événements. Ceci peut avoir un impact négatif sur la testabilité du système, car la connexion implicite entre le *Main process* et les appareils d'acquisition peuvent compliquer le débogage du système. Ainsi, il pourrait être difficile de déterminer d'où vient un événement et qui l'a déclenché au niveau d'un appareil abonné. Par contre, étant donné qu'il y a un seul évènement par module d'acquisition de données, ce problème ne devrait plus en être un proprement dit.

Ainsi, le *Main process* est le processus principal du système qui exécute tous les autres processus et qui publie des événements aux appareils d'acquisition pour l'envoi de données aux autres processus. Ce processus communique avec les autres processus afin d'envoyer des requêtes de données aux appareils d'acquisition quand ceux-ci le demandent.

Aussi, il y a le *Timed acquisition process* et le *High acquisition frequency process* qui sont respectivement l'acquisition de données standard et l'acquisition de données haute fréquence utilisant la FFT pour obtenir les données d'amplitude en fonction de la fréquence. Ce sont aussi eux qui s'occupent de lancer la journalisation des données d'acquisition. De plus, l'ouverture de fichier journalisée, de l'analyse statistique de données et de la génération de rapport sont faites par le processus *Data analysis process*.

Ensuite, il y a *Setup device process* pour la gestion de l'état des appareils en utilisant une tactique permettant de monitorer l'état des appareils en tout temps afin de détecter les erreurs et la configuration de ceux-ci. Il permet également de gérer la configuration XML des fichiers de configuration. Aussi, la détection et la création au niveau logiciel des appareils d'acquisition sont faites par le *Device detector process*.

De plus, il y a le processus de gestion de défaillances *Fault management process* qui va gérer les défaillances provenant des appareils avec les mêmes tactiques mentionnées dans le module *Fault management device*.

Pour ce qui est de l'attribut de qualité de modificabilité, l'utilisation d'un style architectural tel que *Publisher-Subscriber,* facilite la modificabilité étant donné qu'il n'y a aucune dépendance directe entre les composantes qui publient et ceux qui s'abonnent au gestionnaire d'événements.

3.3 Vue de déploiement

3.3.1 Objectif de la vue

La vue de déploiement permet de bien voir quel artéfact du système est nécessaire lors du déploiement pour le fonctionnement du système. Ainsi, elle est nécessaire pour faire l'élaboration de l'installateur.

3.3.2 Illustration de la vue

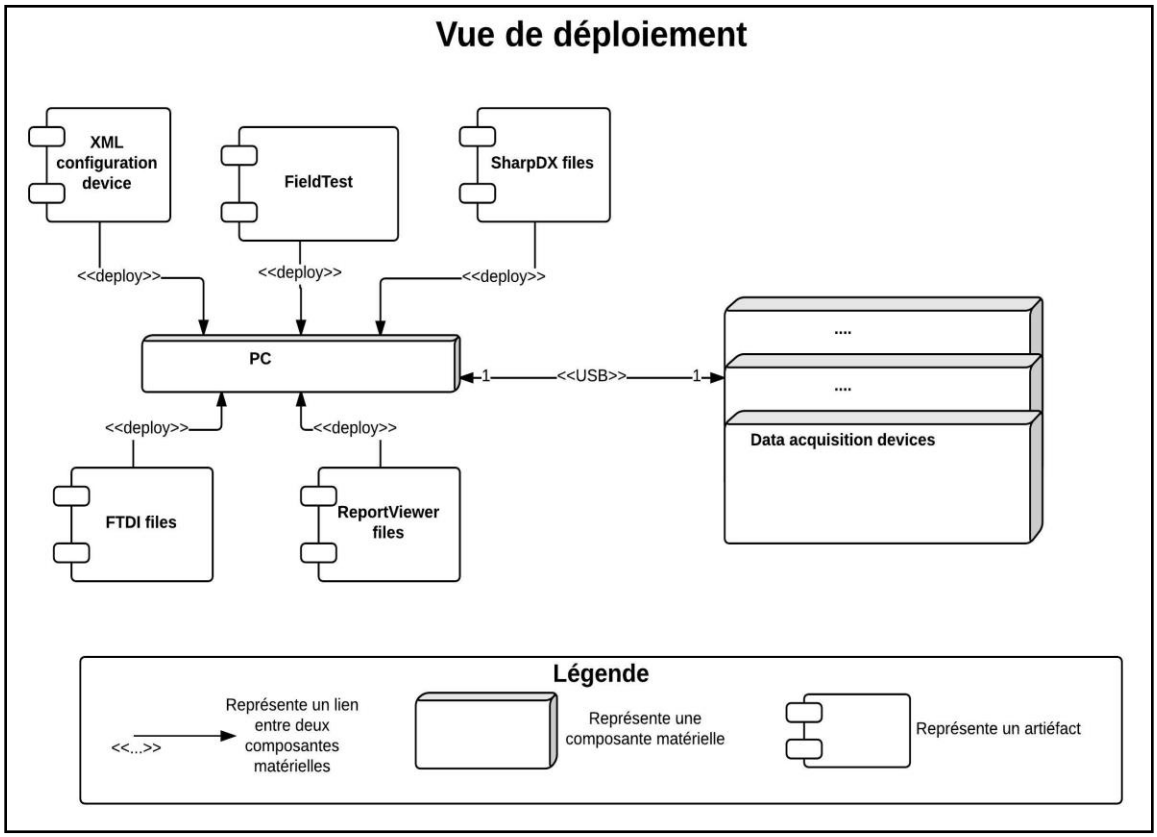

Figure 3 - Vue de déploiement

Table 6 - Tableau de description de la vue de déploiement

Élément(s)	Description(s)
PC	Représente un PC où seront déployés les artéfacts de FieldTest.
Data acquisition devices	Représente les appareils d'acquisition de données.
FieldTest	Représente le logiciel FieldTest contenant toutes les fonctionnalités lieés à l'acquisition de données.
XML configuration device	Représente les fichiers XML contenant la

	configuration des appareils d'acquisition.
SharpDX files	Représente les fichiers dll et les fichiers XML liés à l'utilisation de la carte graphique pour effectuer la FFT.
FTDI files	Représente les fichiers dll utilisés pour faire la détection et le mappage des ports USB avec les appareils d'acquisition.
ReportViewer files	Représente les fichiers dll utilisés pour tout ce qui touche à la génération de rapport.

3.3.3 Description de la vue

Cette vue montre le déploiement d'artefacts du système sur les différentes composantes matérielles du système. Ainsi, dans le système actuel, il y a seulement le PC et les appareils d'acquisition de données qui communiquent via le protocole USB.

Le déploiement comporte le logiciel FieldTest comportant toutes les fonctionnalités mentionnées dans les vues précédentes et le fichier de configuration de base des appareils d'acquisitions de données. Il y a également les dll de C# *FTDI* qui sont utilisés par FieldTest pour faire la détection et le mappage des ports USB avec les appareils d'acquisition au niveau logiciel.

De plus, afin d'améliorer la performance de calcul de la FFT, la carte graphique du PC est utilisée. Pour procéder à l'utilisation de la carte graphique, l'utilisation de SharpDX de C# est nécessaire. Une présentation de l'utilisation de SharpDX et la raison de son utilisation seront plus détaillées dans la deuxième itération.

Aussi, le déploiement comporte les dll liés à l'outil qui est intégré au logiciel soit *ReportView* qui permet de visualiser et ajuster les éléments du rapport avant l'impression.

CHAPITRE 4

PREMIÈRE ITÉRATION

4.1 Analyse

4.1.1 Analyse de l'acquisition de données

L'acquisition de données doit pouvoir se faire avec plusieurs appareils. Présentement, le logiciel permet d'acquérir des données provenant du TT10K, de GPS et du Adams. De plus, les données recueillies par ces appareils ne sont pas les mêmes et n'ont également pas toutes la même fréquence d'acquisition.

De plus, le logiciel doit être facilement maintenable afin qu'il soit facile d'ajouter des appareils d'acquisition et dans un temps raisonnablement court. Présentement, l'architecture pour l'acquisition de données comporte une classe *ADevicePortMapped* qui est une classe abstraite que chaque appareil d'acquisition doit implémenter afin de définir les caractéristiques de l'appareil d'acquisition, ces fonctions d'ouverture, de fermeture, de détection de port de communication et le protocole de lecture des données. Ensuite, cette classe contient une liste de *ADeviceReading* qui elle aussi est une classe abstraite qui sert à traiter les données reçues à l'aide de formules liées à chaque type de données lu par l'appareil. Par exemple, le GPS à 3 *ADeviceReading*, car il acquiert des données de latitudes, de longitudes et de vitesses.

Aussi, l'acquisition de données se fait par 3 modules principaux soient par le module *Setup Device*, *Timed acquisition*, *High frequency acquisition*. Le module *Setup Device* s'occupe de faire la détection de port et la création au niveau logiciel des appareils en utilisant le patron de conception *Factory*. Ainsi, la classe *PortFactory* crée les appareils selon l'identifiant unique défini par le câble USB de l'appareil détecté et la configuration du fichier XML. La

classe *PortScanner* s'occupe de faire le lien entre le port de l'appareil connecté et l'identification de l'appareil à l'aide de la configuration XML ayant un entête caractéristique pour chaque appareil. Par contre, le problème majeur de cette classe, c'est qu'elle est utilisée seulement durant l'état de la configuration et du visuellement de l'état des ports. Il n'y a donc aucune gestion pour savoir si un appareil est toujours connecté ou non connecté durant l'acquisition de données standard et à haute fréquence. Ainsi, il va falloir ajouter cette fonctionnalité de gestion aux deux autres modules afin de bien contrôler l'état des ports incluant la gestion des fautes.

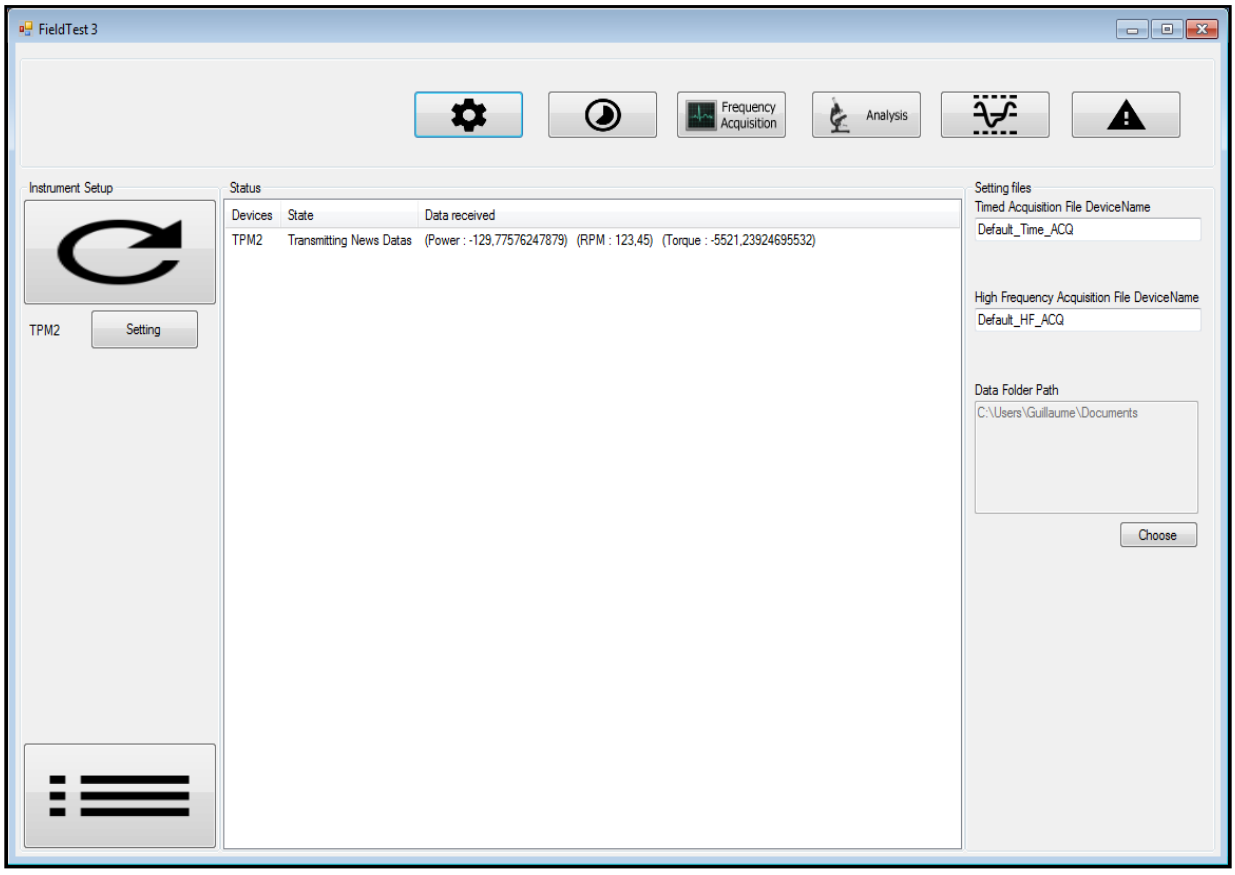

Figure 4 - Interface de monitorage de communication des ports

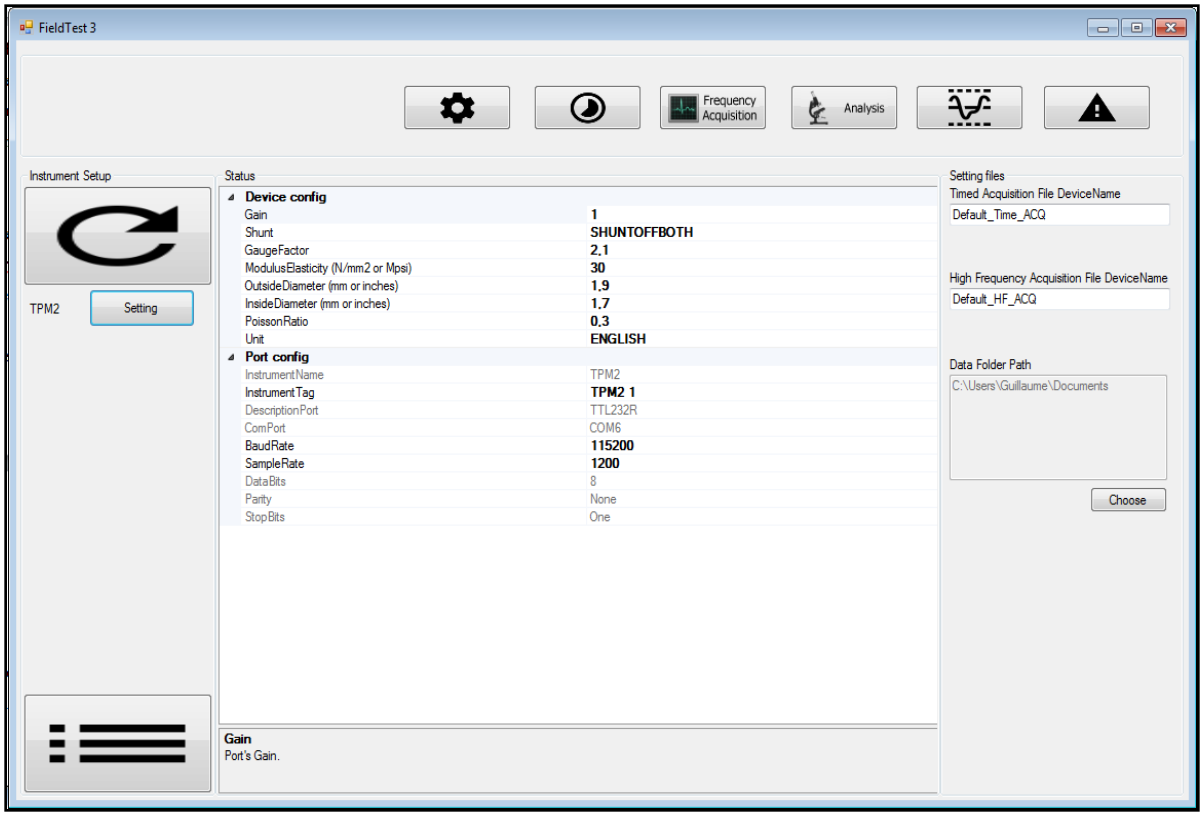

Figure 5 - Interface de gestion de la configuration d'un appareil d'acquisition

Les modules *Timed acquisition* et *High acquisition* eux s'occupent respectivement de l'acquisition standard et à haute fréquence. L'acquisition standard affiche les données brutes tout en combinant certaines données reçues pour en produire d'autres. Il est également possible de voir les données sur plusieurs graphiques à la fois en ayant deux appareils par graphique. L'acquisition à haute fréquence affiche les données brutes reçues comme l'acquisition standard sur un graphique, mais affiche également sur un autre graphique les données d'amplitude en fonction de la fréquence calculée à partir de la transformée de Fourier.

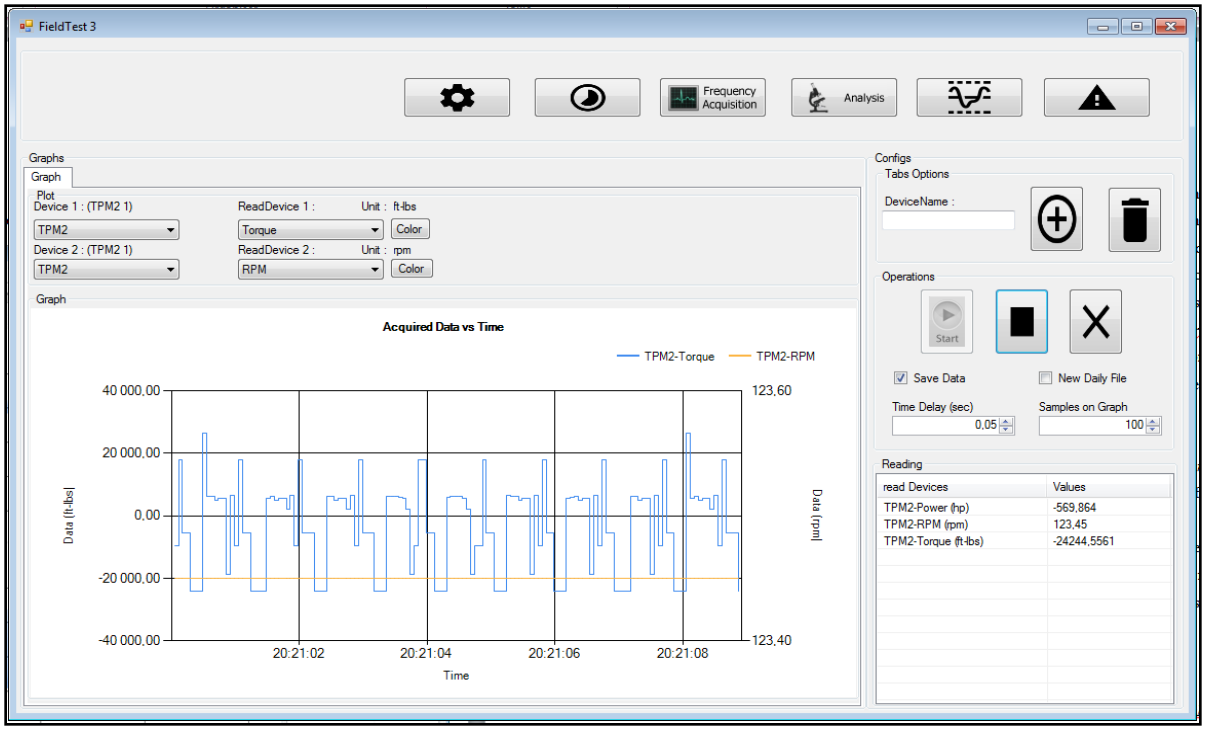

Figure 6 - Interface montrant l'acquisition standard

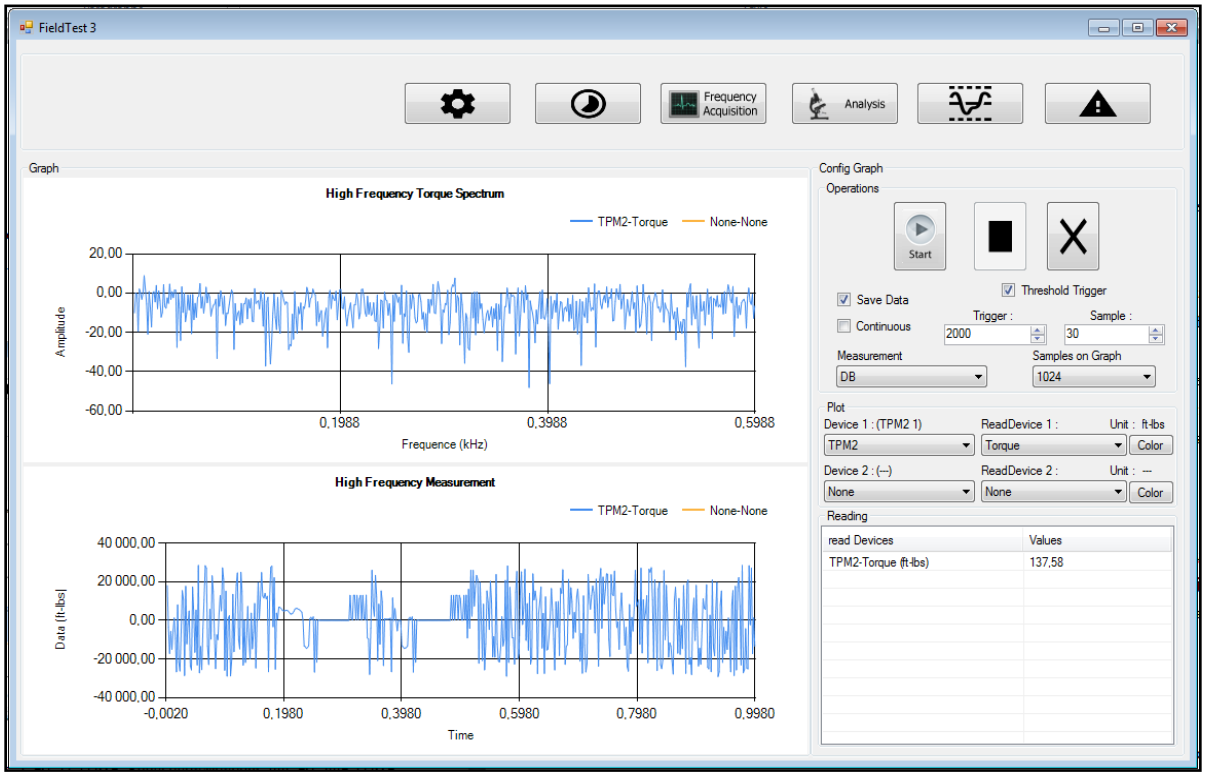

Figure 7 - Interface montrant l'acquisition à haute fréquence

Lors de l'acquisition de données, c'est la classe *ApplicationManager* qui est une classe statique qui va démarrer l'acquisition de données par l'entremise des interfaces utilisateurs de l'acquisition standard, de l'acquisition à haute fréquence et de la vérification de l'état des appareils. Ainsi, ces 3 modules vont seulement traiter les données reçues par la classe *ApplicationManager* qui aura accès à tous les appareils d'acquisition de données connectés et qui fera les demandes de données en « multithreading » aux appareils. Présentement, un semblant du patron observateur est utilisé par les trois modules *Setup Device*, *Timed acquisition*, *High frequency acquisition*. Ainsi, l'utilisation de méthodes explicites « Notify / Update » sont utilisées pour contrôler l'envoi et l'ajout des données des classes logiques d'affaires vers les contrôles soient les graphiques et les tableaux. Ainsi, cette manière de procéder crée un couplement cyclique entre les classes de logique d'affaires et les classes pour les contrôles de l'interface. Donc, la classe de logique d'affaires doit appeler explicitement chaque contrôle lui étant associé. Ainsi, le fait de modifier l'architecture pour le *Publisher-Subscriber* devrait mieux découpler l'architecture, mais aussi rendre plus performant l'envoie de données à chaque récepteur étant donné l'envoi implicite du publieur qui n'attend pas de retour du récepteur.

Pour résumé, pour ajouter un appareil au logiciel, il faut créer sa classe *ADevicePortMapped* et ces classes *ADeviceReading* pour chaque donnée lue par celui-ci. Ensuite, il faut ajouter la configuration de base au fichier XML pour l'appareil et ajouter le tout à la classe *PortFactory* pour la création lors de la détection.

4.1.2 Analyse du TPM2

Le TPM2 est un appareil d'acquisition de données permettant d'acquérir des données sur un arbre mécanique. Ainsi, il permet d'acquérir la vitesse de l'arbre « Shaft Speed », la puissance « Power » et la déformation de l'arbre « Strain Gage ». Le TPM2 de base fonctionne à une vitesse de transmission de 115000 bps et à une fréquence de 1200 Hz. Cependant, il est possible de changer le tout comme les autres appareils, mais la plupart du

temps c'est la configuration initiale qui est utilisée. Par contre, le TPM2 comporte également d'autres éléments de configuration liés à l'arbre mécanique. Ainsi, il y a le diamètre intérieur et extérieur du « shaft », les facteurs de « gain » et de « gage », d'élasticité du « shaft » et de ratio de ce dernier lié au matériel en lequel est fait le « shaft ».

Figure 8 – TPM 2

De plus, le TPM2 comporte une chaine de 8 octets pour la lecture et 5 pour l'écriture. L'écriture consiste à la configuration du TPM2. Ainsi, il faut remplir les 5 octets d'écriture avec les paramètres possibles comme mentionnés ci-haut pour envoyer une nouvelle configuration au TMP2. Après la demande de changement de configuration de l'appareil, il faut jusqu'à 10 millisecondes avant de recevoir l'octet de confirmation pour savoir si les changements ont été effectués et pour pouvoir commencer par la suite l'acquisition de données.

Ainsi, pour demander des données au TPM2, il faut attendre de recevoir 8 octets. Ensuite, il faut vérifier la validité des données avec le « checksum » qui correspond à la somme des 8 octets reçus en prenant seulement le dernier octet de cette somme le tout doit être égal au huitième octet reçu. Seulement 5 des 8 octets seront utilisés pour le logiciel. Ainsi, le premier octet comportant la donnée de « Strain Gage » et l'octet 2 pour la donnée de « Shaft Speed ». Les octets 5, 6, 7 sont les octets pour avertir d'erreur, de l'état d'acquisition de l'appareil et de la configuration actuelle de l'appareil. Les autres octets ne sont pas pertinents pour les données voulant être analysées par le logiciel.

Pour le traitement pour obtenir les données de « Shaft Speed », de « Power » et de « Strain Gage », le logiciel devra effectuer certaines formules afin de mettre les données au format souhaité sauf pour le « Shaft Speed » qui est obtenu directement de l'appareil. Ainsi, pour le « Power » et le « Strain Gage », les formules suivantes sont utilisées afin d'obtenir toutes les données nécessaires :

$$Strain\ Gage = (Strain\ gage\ provenant\ de\ l'appareil * 15729) / (GaugeFactor * GainFactor * 7864.32)$$

$$Torque = Strain * PI * élasticité * (diamètre\ extérieur\ du\ shaft - diamètre\ intérieur\ du\ shaft) / (192 * diamètre\ extérieur\ du\ shaft * (1 + ratio\ du\ shaft))$$

$$Power = \frac{Torque * 2 * PI * Shaft\ Speed}{kp}$$

kp étant de 60 ou 33000 si on utilise le système métrique ou anglais.

Voir **Annexe 1** pour la documentation du protocole de communication du TPM 2.

4.2 Conception et implémentation

4.2.1 Conception et implémentation pour le TPM2

Pour la conception, 4 classes ont été nécessaires pour bien définir l'appareil et les lectures possibles de ces derniers. Ainsi, une classe *TPM2PortMapped* définissant le protocole de communication de l'appareil et implémentant la classe abstraite *ADevicePortMapped* a été créée. Dans cette classe, 5 fonctions de la classe abstraite ont été implémentées soit la fonction *ReadThread*, *IsTransmittingData*, *ExtractData*, *UpdateDeviceSetting*, *OpenPort*.

La fonction *OpenPort* s'occupe d'ouvrir le port et dans le cas du TPM2, elle s'occupe de l'envoi du message de configuration et de la configuration de l'appareil. La fonction *IsTransmittingData* s'occupe tout simplement de vérifier s'il y a des octets à lire. La fonction *ReadThread* est la fonction définissant le protocole de communication. Ainsi, la fonction acquiert un octet à la fois jusqu'à atteindre un total de 8 octets. Ensuite, elle vérifie si le dernier octet correspond au bon « checksum » et reconstruit une chaine globale de caractère de données à la classe. Ensuite, c'est la fonction *ExtractData* qui va extraire les données de cette chaine globale en effectuant le traitement de chaque lecture possible à partir de ces classes *ADeviceReading*. Après, une *SortedList* contenant comme clé le type de donnée et la donnée est envoyée à la composante demandant ces données. La fonction *UpdateDeviceSetting* quant à elle s'occupe d'effectuer les changements de configuration du port de communication et de l'appareil. Elle utilise également deux fonctions soient *TrasmitterConfig* et *CommicationConfig* pour séparer l'envoi de configuration vers le TPM2.

Les 3 classes de lecture *ADeviceReading* contenues dans la classe *TPM2PortMapped* soient *PowerTPM2Reading*, *ShaftSpeedTMP2Reading* et *StrainGageTPM2Reading* implémentent toutes la fonction *ReadDataAsynchrone*. Cette fonction retourne la valeur de la donnée traitée. C'est ainsi que cette fonction est appelée par la fonction *ExtractData* dans la classe *TPM2PortMapped* pour traiter tous les types de données de l'appareil. Chaque classe de lecture utilise des sous-fonctions liées aux formules mentionnées dans l'analyse afin d'obtenir les données nécessaires avant de retourner la valeur finale.

Voir **Annexe 2** pour le diagramme de classes montrant la conception du TPM2.

Pour les tests du TPM2 et l'ensemble des fonctionnalités demandant des tests liés à l'acquisition de données un Arduino simulant le TPM 2 a été utilisé.

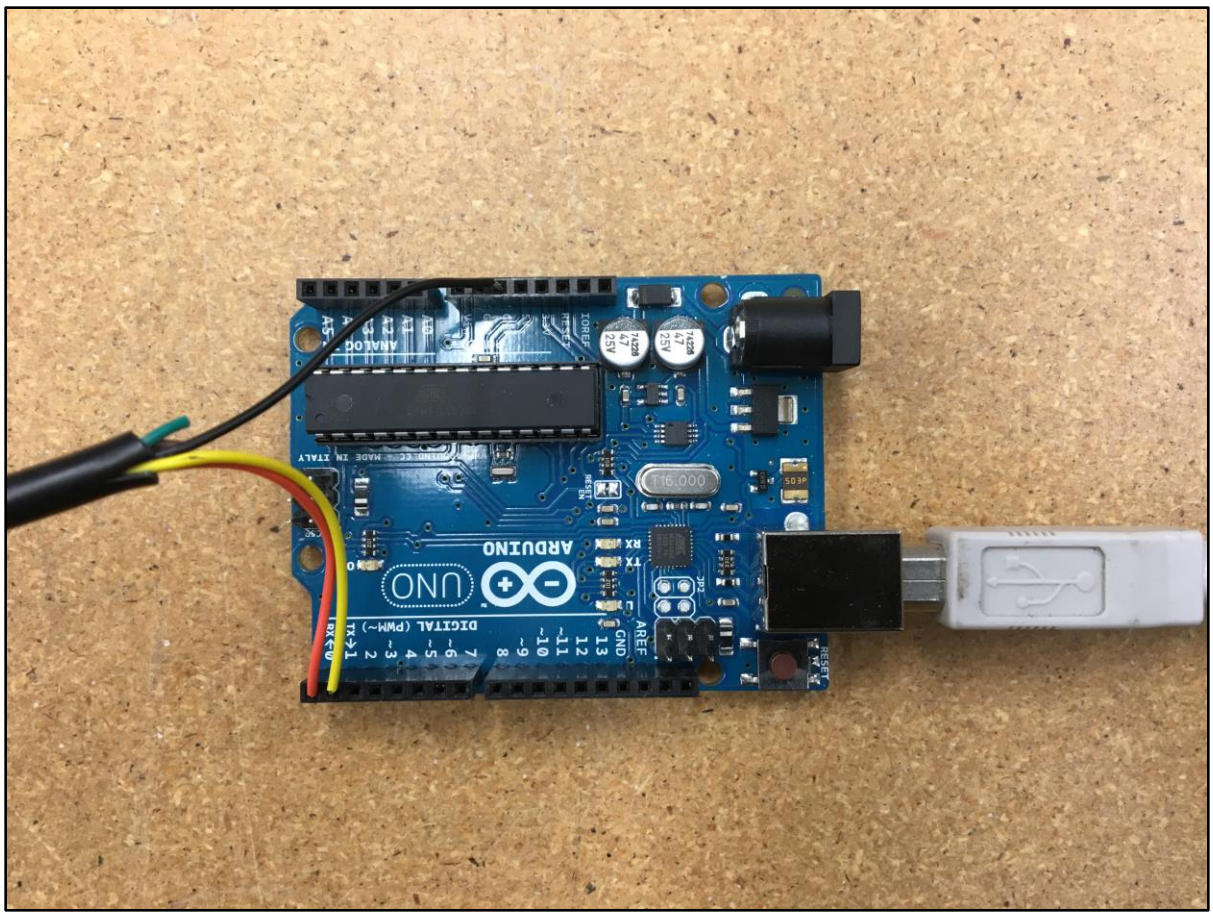

Figure 9 - Dispositif de test (Arduino)

4.2.2 Conception et implémentation du Publisher-Subscriber

Avant d'apporter le *Publisher-Subscriber* à l'architecture, les 3 modules *Setup Device, Timed acquisition, High frequency* acquisition utilisaient un semblant du patron observateur. Ainsi, il y avait une fonction « Notify » et « Update ». La fonction « Notify » était présente dans la classe s'occupant de la logique d'affaires et la fonction « Update » dans les contrôles afin de

remplir les graphiques et les tableaux de données. Ainsi, cette manière de procéder créait un couplement cyclique entre la classe de logique d'affaires et la classe pour les contrôles de l'interface. Donc, la classe de logique d'affaires devait appeler explicitement chaque contrôle lui étant associé.

La nouvelle implémentation a permis de régler ce problème. Ainsi, la nouvelle architecture a demandé l'ajout d'une classe pour l'événement et une classe pour le publieur de l'événement. La classe de logique d'affaires contient le publieur d'événement qui va lors de la réception de données publier à tous les contrôles contenant les mêmes événements définis que dans la fonction du publieur. Les contrôles eux définissent comment l'évènement sera traité.

Pour le module d'acquisition haute fréquence, il y a ainsi une classe *HighFrequencyDataEventArgs* pour l'évènement contenant 2 listes de données transformées par la FFT correspondant aux données de la série 1 et 2 affichées sur les graphiques et une liste contenant les moyennes calculées. La classe *HighFrequencyDataPublisher* crée cet évènement en passant ces 3 listes en paramètre dans la fonction *PublishData*. C'est cette fonction qui est appelée par la classe *HighFrequencyAcquistion* lorsque cette dernière reçoit des données. Ainsi, tous les « Notify » de la fonction *Read* étant la fonction « thread » de lecture de données en continu ont été enlevés de cette classe, seul l'appel à la fonction *PublishData* permet de distribuer les données à tous les contrôles correspondants. Les contrôles eux définissent les fonctions d'exécutions de l'évènement en utilisant les mêmes fonctions « Update » qu'avant, mais en passant l'événement en paramètre au lieu des listes de données. De plus, la définition des événements à exécuter par les publieurs est faite dans les constructeurs du contrôle.

Les modules d'acquisition standard et de configuration de port, quant à eux, ont le même publieur et le même évènement. Cependant, leur publieur et leur événement sont *DataPublisher* et *DataEventArg* et ils fonctionnent avec une liste contenant la valeur retournée de chaque appareil d'acquisition au lieu des 3 listes comme l'acquisition haute fréquence. La raison vient du fait que l'acquisition standard et la configuration de port

utilisent une liste de données ayant une valeur pour chaque lecture d'appareil à chaque appel tandis que l'acquisition à haute fréquence attend d'avoir un certain nombre de données avant de faire la FFT et d'afficher les données. Ainsi, elle a une liste complète de données à afficher pour l'appareil en question à chaque appel. Par contre, le reste de l'implémentation fonctionne de la même manière que mentionnée dans l'acquisition à haute fréquence.

Voir **Annexe 3** pour le diagramme de classes montrant la conception du *Publisher-Subscriber.*

Voici également un diagramme de séquence montrant comment s'effectue l'acquisition de données en général pour les 3 modules *Setup Device*, *Timed acquisition*, *High frequency acquisition* :

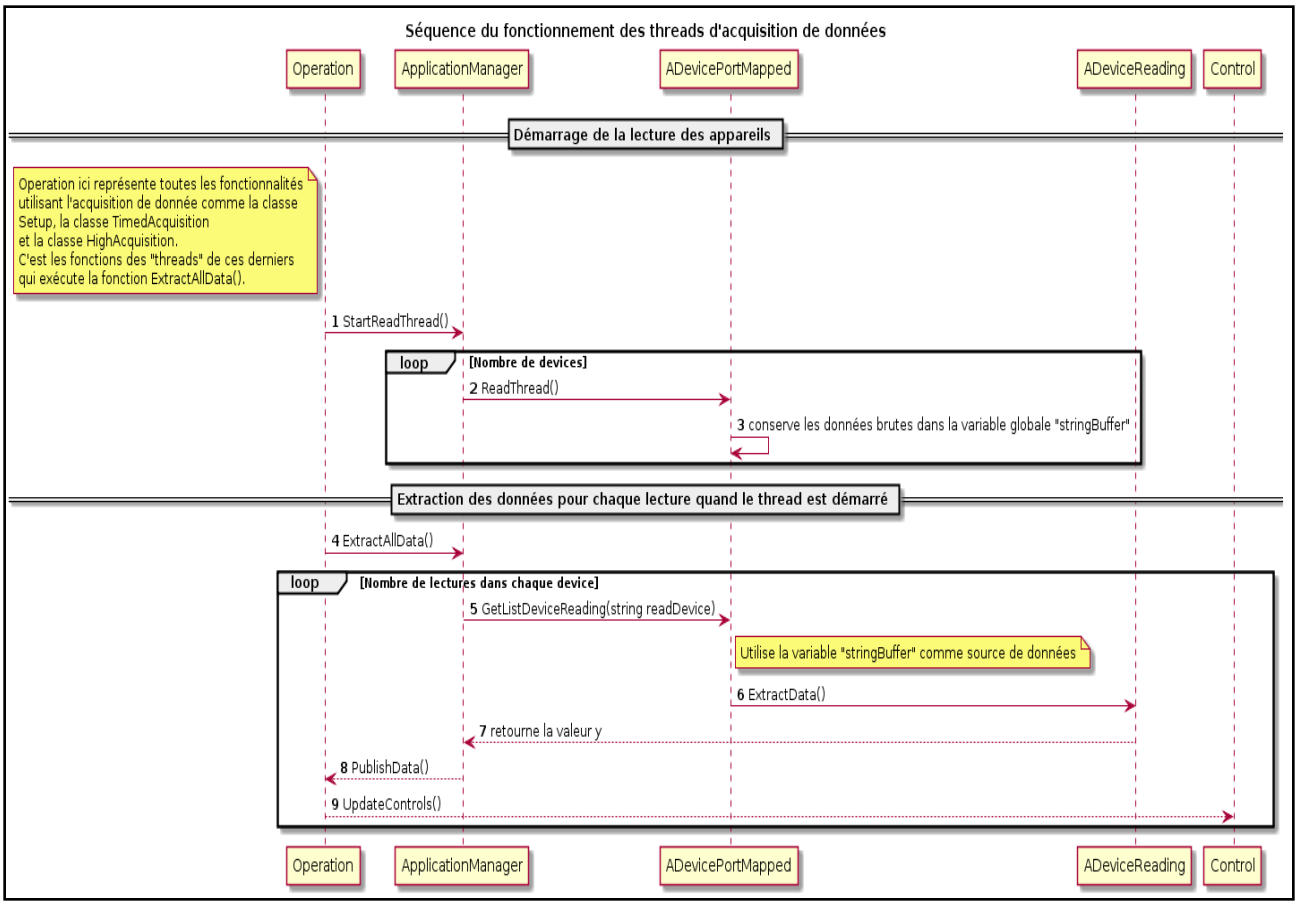

Figure 10 - Diagramme de séquence de l'acquisition de données vers les contrôles

4.2.3 Conception et implémentation pour la détection de port et monitorage

Afin de permettre une détection de port et du monitorage en continu des ports détectés par le logiciel durant l'acquisition normal et à haute fréquence, une fonction « thread » regardant l'état du port en même temps que l'acquisition de données a été nécessaire. Ainsi, dans le cas où un problème survient soit une déconnexion du port ou des erreurs de communication qui empêche l'acquisition de données de continuer, le logiciel va chercher à réinitialiser le port pour reprendre l'acquisition de données. Par contre, si le problème est au niveau de l'appareil, un bris ou une défaillance technique de l'appareil le logiciel ne sera pas capable de reprendre l'acquisition avant l'intervention d'une personne sur place.

Ainsi, l'implémentation d'une fonction pour vérifier l'état d'un port à la fois a été implémentée soit *FindFTDIDevise* dans la classe *PortScanner*. Cette fonction regarde si le port ne contient pas des erreurs ou des problèmes empêchant l'acquisition de données, si elle détecte que le port a des problèmes, cette dernière retournera un booléen négatif à la fonction de lecture et dans le cas contraire un booléen positif. Cette fonction de lecture *CheckPortConnected* est définie dans la classe abstraite *ADevicePortMapped* et est exécutée par le « thread » vérifiant l'état des ports *ThreadPortDectecting*. Ainsi, quand le « thread » est démarré cette fonction essaie en continu de réinitialiser le port jusqu'à ce que ce dernier redevienne dans un état où l'acquisition est possible. Ce « thread » est démarré à partir de la fonction *StartThreadPortDectection* et arrêté à partir de la fonction *StopThreadPortDectection*. Ce sont les classes concrètes de *ADevicePortMapped* qui dans leur fonction « thread » *ReadThread* d'acquisition de données démarreront ce « thread » de vérification lorsqu'une erreur sera attrapée par le « try catch ».

Voir **Annexe 4** pour le diagramme de classes montrant la conception liée à la détection et monitorage de port.

4.3 Revue et test

Le *Publisher-Subcriber* dans l'architecture a permis de découpler certaines classes. De plus, il a également permis d'enlever les boucles « For » lors de l'envoi de données et ainsi rendre beaucoup moins complexe le code. L'affichage de données est également légèrement plus rapide qu'avant. Pour ce qui est de la modifiabilité, il sera facile de définir de nouveaux événements dans chaque contrôle pour différents traitements avec les données reçues juste en définissant seulement le traitement à faire dans la classe du contrôle, ce qui aide à répondre à l'attribut de qualité **AQ01** lié à la facilité d'ajout de nouvelles composantes au logiciel.

Pour le TPM2 des tests unitaires ont été effectués sur les 3 fonctions des valeurs calculées retournées par les 3 lectures de données possibles soient « Shaft Speed », « Power » et « Strain Gage ». Ces tests ont démontré que les valeurs retournées étaient les bonnes selon les divers paramètres entrés. De plus, un Arduino contenant une simulation du TPM2 a été utilisée pour montrer que l'acquisition du TPM2 est fonctionnelle au niveau de l'acquisition standard et à haute fréquence. Par contre, les tests au niveau de la configuration du TPM2 restent encore à être effectués, car le dispositif de test ne permettait pas de tester tous les éléments de configuration et le comportement au complet du TPM2. Ainsi, présentement, seul des tests au niveau du changement de configuration liée à la communication du port en général ont pu être effectués, mais pas au niveau de la configuration des paramètres de mesure de la composante mesurée de l'appareil soit l'arbre mécanique. Ainsi, la prochaine étape serait de tester ces éléments sur un vrai appareil pour bien valider. Pour le moment, la version bêta est utilisée par certaines personnes et il ne semble pas avoir eu de problème à ce niveau, car il est possible de changer ces paramètres directement sur l'appareil.

Pour le monitorage et détection des ports, l'ensemble des éléments ajoutés a permis de bien gérer en tout temps l'état des ports dans le logiciel. Le fait que la fonctionnalité de monitorage vérifie en permanence que les liens sur les ports ne contiennent pas de problèmes empêchant l'acquisition de données et qu'en cas de problème, elle essaie en continu de rétablir l'acquisition de données à son état normal permet de bien répondre aux attributs de

qualité **AQ06** et **AQ07** qui sont liés à la reprise en cas de défaillance. Cette fonctionnalité aide également au niveau des attributs de qualités **AQ04**, **AQ05** en empêchant le logiciel de ne plus fonctionner en cas de problème sur un des ports où l'acquisition est en cours et en permettant de ressayer de rétablir l'acquisition sur le lien du port afin d'obtenir les données.

CHAPITRE 5

DEUXIÈME ITÉRATION

5.1 Analyse de la transformée de Fourier

Afin d'évaluer plus facilement les problèmes pouvant survenir lors de l'acquisition de données, la visualisation de l'amplitude en fonction de la fréquence permet beaucoup plus facilement d'observer les changements rapides dans les données acquises d'où le besoin d'avoir une transformée de Fourier capable de transformer les données. De plus, afin d'avoir une transformée de Fourier efficace, il faut que l'appareil d'acquisition fonctionne à une haute fréquence d'acquisition, car il faut attendre un certain nombre de données avant de procéder à la transformée. Aussi, étant donné que cette acquisition de données est en temps réel, il est impossible d'utiliser la transformée de Fourier standard ainsi la transformée de Fourier rapide (FFT) est alors démise pour répondre au besoin de rapidité de traitement.

Lors de mon dernier stage, une transformée de Fourier avait été implémenté dans une fonction en implémentant juste la formule mathématique telle quelle sans avoir regardé pour une optimisation. Les résultats étaient sensiblement bons, mais il fallait plus de 30 secondes de traitement avant l'affichage des données sur le graphique, ce qui ne pouvait pas vraiment être utilisé dans le contexte actuel. Ainsi, il y avait une classe statique *FourierTransform* demandant en paramètre la liste de données à transformer dans la fonction *PerformFFT* retournant la liste de données traitées.

Ainsi, afin d'améliorer cette performance, l'utilisation de la carte graphique (GPU) est une bonne solution afin de pouvoir accélérer les calculs liés à la FFT. La librairie *SharpDX* permet de faire des traitements sur la carte graphique, dont la FFT.

5.2 Conception et implémentation de la transformée de Fourier

Pour la conception, une analyse de la librairie *SharpDX* a été faite afin de repérer tous les éléments nécessaires à la réalisation de ce traitement. Ainsi, un objet *FastFourierTransform* allait être nécessaire afin de contenir l'information de l'exécution de la FFT et de permettre d'exécuter la FFT elle-même. Un objet *Device* pour définir quel type de performance et précision souhaitées lors du calcul avec la carte graphique. Un objet *Buffer* permettant de créer le tampon à partir du tableau de données qui sera traité par le GPU. Un objet *UnorderedAccessView* qui correspond au tampon converti en un autre tampon optimisé permettant d'être lu et écrit simultanément par plusieurs « threads » sans générer de conflits de mémoire. Cette conversion de tampon permet d'avoir une meilleure performance lors de la FFT.

Ensuite, pour l'implémentation, une fonction *CreateBufferUAV* a été implémentée servant à créer un objet *UnorderedAccessView*. Cette fonction prend ainsi en paramètre le tampon de données et un objet *Device*. Aussi, une fonction *CreateByteOrderBufferOnGPU* permettant de créer le tampon pouvant être traité par le GPU a été créée. Cette fonction a ainsi besoin en paramètre du nombre de données, le tableau de données d'entrée, l'objet *ResourceUsage* définissant comment le tampon sera traité par le GPU dans ce cas-ci les paramètres par défaut ont été utilisés.

Ensuite, une fonction initialisant les paramètres et les éléments nécessaires à la FFT, soit *setttingFFT* a également été créée. Cette fonction sert à définir les paramètres de la FFT soit la dimension du tableau traité, le type de valeur des éléments du tableau. De plus, cette dernière crée l'instance de l'objet *FastFourierTransform* pour ensuite faire une précomputation du tampon afin d'accélérer le processus lors du traitement de la transformée de Fourier.

Pour ce qui est de la fonction exécutant la FFT *PerformFTT*, elle prend en paramètre le tableau de données de valeur à transformer. Cette dernière crée le tampon à partir du tableau

de données en utilisant la fonction *CreateByteOrderBufferOnGPU* et convertit ce dernier par la suite en *UnorderedAccessView* avec la fonction *CreateBufferUAV*. Par la suite, la transformée est exécutée avec la fonction *ForwardTransform* sur le tampon optimisé et une série de transformations est faite sur le tampon de données transformées sortant pour remettre le tableau au même format d'entrée soit un tableau de nombre réel.

Voir **Annexe 5** pour le diagramme de classes montrant la conception liée à l'acquisition de données à haute fréquence.

5.3 Revue et test

Le changement apporté pour calculer la FFT via le GPU a augmenté considérablement la vitesse. Ainsi, pour 1024 données à transformer on passe de 30 secondes à 2 secondes de traitement. Dans l'ensemble, les valeurs retournées par la FFT sont exactement les bonnes afin de déceler les pics d'amplitude anormale. Ainsi, les résultats affichés sont très satisfaisants et répondent aux besoins demandés. Par contre, il serait possible de modifier encore les paramètres liés à la FFT pour que les pics d'amplitude soient mieux démarqués du reste des donnés afin de faciliter son repérage sur l'ensemble du graphique affiché.

CHAPITRE 6

TROISIÈME ITÉRATION

6.1 Analyse

6.1.1 Analyse pour la génération de rapport et du traitement des données journalisée

Afin de pouvoir analyser des données journalisées et de faire le même traitement que lors de l'acquisition en temps réel, il faut des interfaces capables d'ouvrir des fichiers journalisés, de les analyser et de faire des rapports au besoin afin de garder une trace de cette analyse.

Ainsi, pour l'acquisition standard et à haute fréquence, il faut être en mesure d'exporter d'autres fichiers selon un ensemble de données bien précis afin de garder seulement les données pertinentes pour une meilleure analyse.

Pour l'analyse de dépassement de seuils et d'erreur d'appareils, il faut être en mesure de pouvoir bien voir la fluctuation de données via la valeur dépassant le seuil ou l'erreur produite, et ce, de chaque côté de cette dernière afin de voir l'effet de cet événement sur l'ensemble des données.

De plus, les rapports doivent contenir toutes les données contenues dans les fichiers, ainsi que les graphiques générés afin de garder une trace similaire à ce qui est affiché à l'écran.
Les formats de fichier de journalisation ont déjà été réalisés auparavant. Les classes permettant d'écrire et de lire ces fichiers journalisés ont également déjà été réalisées. Ainsi, il y a *WriterError* pour les erreurs des appareils, *WriterTriggerError* pour le dépassement de seuil, *WriterNormalData* pour l'acquisition standard et *WriterHighData* pour l'acquisition haute fréquence. Pour la lecture, *ReaderData* pour l'acquisition standard et à haute fréquence, *ReaderError* pour les erreurs d'appareils et *ReaderTrigger* pour le dépassement de seuils.

Le format de chaque type de fichier de journalisation comporte un entête puis les données. Pour l'acquisition standard, l'entête est constitué d'une première ligne définissant le type d'acquisition, ensuite la seconde ligne présente tous les appareils d'acquisition présents dans la journalisation, la troisième ligne correspond à toutes les lectures possibles des appareils d'acquisition dans la journalisation. La dernière ligne de l'entête représente le type d'unité utilisé pour chaque appareil de mesure. Ensuite, la plage de données consiste en 3 données par ligne, soit le moment en temps où la donnée a été acquise, l'intervalle de temps entre la nouvelle et la dernière acquisition et la valeur de la donnée pour chaque lecture d'acquisition.

Voir **Annexe 6** pour le format de fichier de l'acquisition standard.

Pour l'acquisition haute fréquence, l'entête est le même sauf que une autre ligne est ajoutée à cette dernière pour stipuler la donnée standard en fonction du temps et la donnée d'amplitude en fonction de la fréquence. Ainsi, pour la plage de données, il y a en plus de la donnée standard la donnée d'amplitude.

Voir **Annexe 7** pour le format de fichier de l'acquisition à haute fréquence.

Pour le format de fichier du dépassement de seuil, le fichier est relativement similaire à l'acquisition standard, mais un fichier est créé pour chaque lecture d'appareil quand un dépassement de seuil survient pour une valeur de lecture et ce, pour toute la durée de l'acquisition. Ainsi, l'entête est similaire à celui de l'acquisition standard. Pour la plage de données, le fichier comporte le même nombre de données avant et après le dépassement de seuil obtenu. Le seuil et le nombre de valeurs retenues de chaque côté peuvent être modifiés à partir de l'interface utilisateur.

Voir **Annexe 8** pour le format de fichier de l'acquisition de dépassement de seuil.

Le format du fichier de journalisation d'erreurs est lui aussi similaire à celui de l'acquisition standard, mais un fichier est créé à chaque fois qu'une erreur se produit pour chaque appareil d'acquisition. De plus, comme pour le dépassement de seuils, un nombre de données avant et après l'erreur est gardé.

6.1.2 Conception et implémentation de la génération de rapport et du traitement des données journalisée

Dans un premier temps, une interface utilisateur comportant les mêmes composantes que les interfaces d'acquisition en temps réel ont été faites pour chaque type de journalisation possible.

Voici les interfaces s'occupant d'analyser et de générer des rapports à partir des données journalisées de manière standard et à haute fréquence :

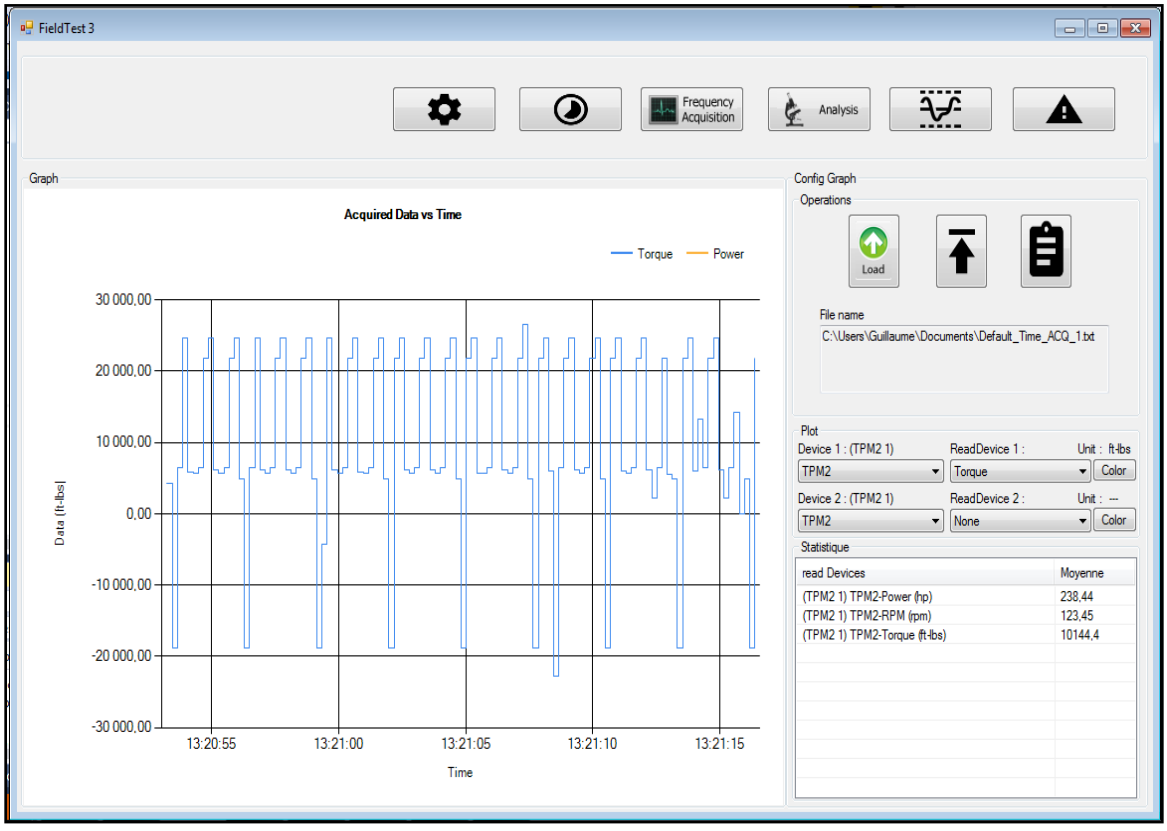

Figure 11 - Interface d'analyse de données standard

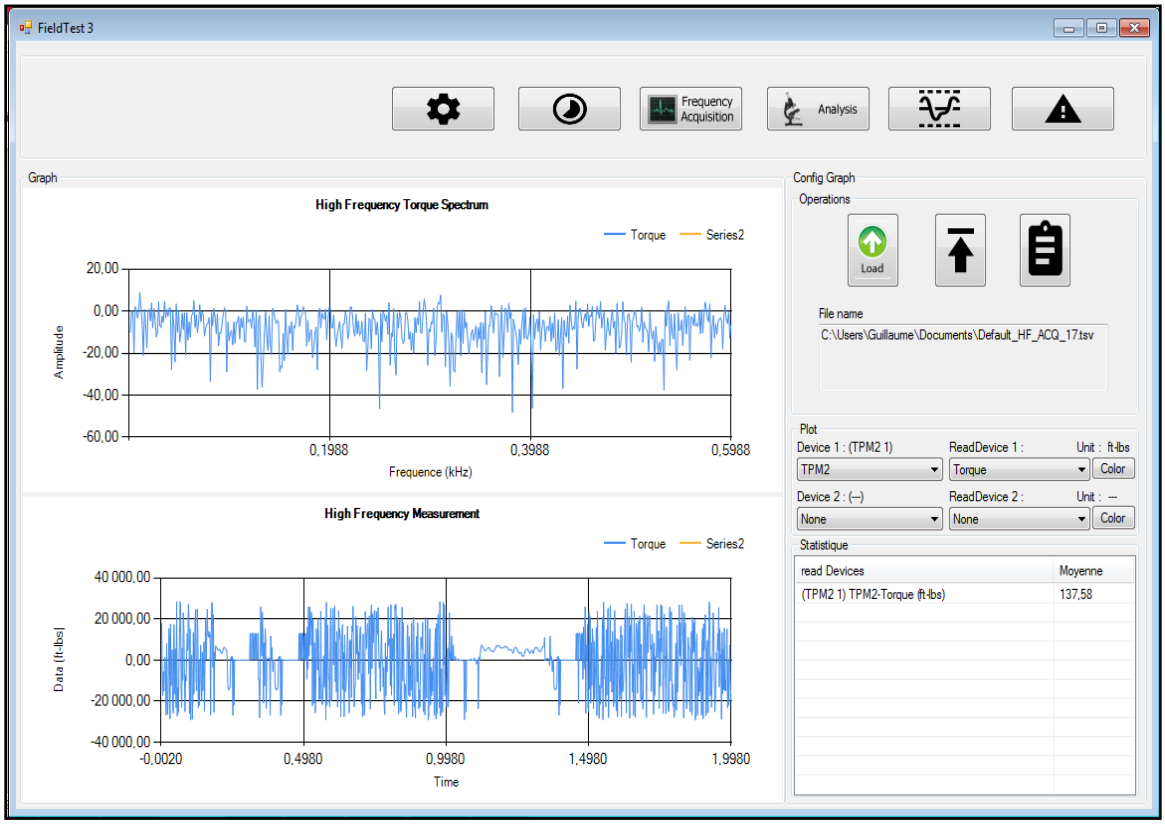

Figure 12 - Interface d'analyse de données à haute fréquence

Ainsi, le fait de garder le même genre d'interface que pour l'acquisition standard et l'acquisition à haute fréquence permet de faire les mêmes manipulations que l'acquisition en temps réel. Dans cette interface, le graphe change en fonction du type de fichier soit standard ou à haute fréquence. Le bouton « load » s'occupe de charger le fichier journalisé sélectionné et de remplir les « combobox » afin de contrôler l'affichage selon les appareils d'acquisition. La moyenne des données est également affichée pour chaque lecture présente dans le fichier de données. Le deuxième bouton, quant à lui, sert à exporter les données dans la zone sélectionnée par le zoom. Ainsi, il suffit de zoomer sur le graphe et ensuite, en cliquant sur ce bouton une fenêtre apparaitra pour sauvegarder ce nouveau fichier de données. Le troisième bouton, lui, sert à ouvrir le « rdlc » permettant de générer un rapport et de faire sa mise en page.

Voir **Annexe 9** et **Annexe 10** pour les exemples de rapports des deux types d'acquisition de données.

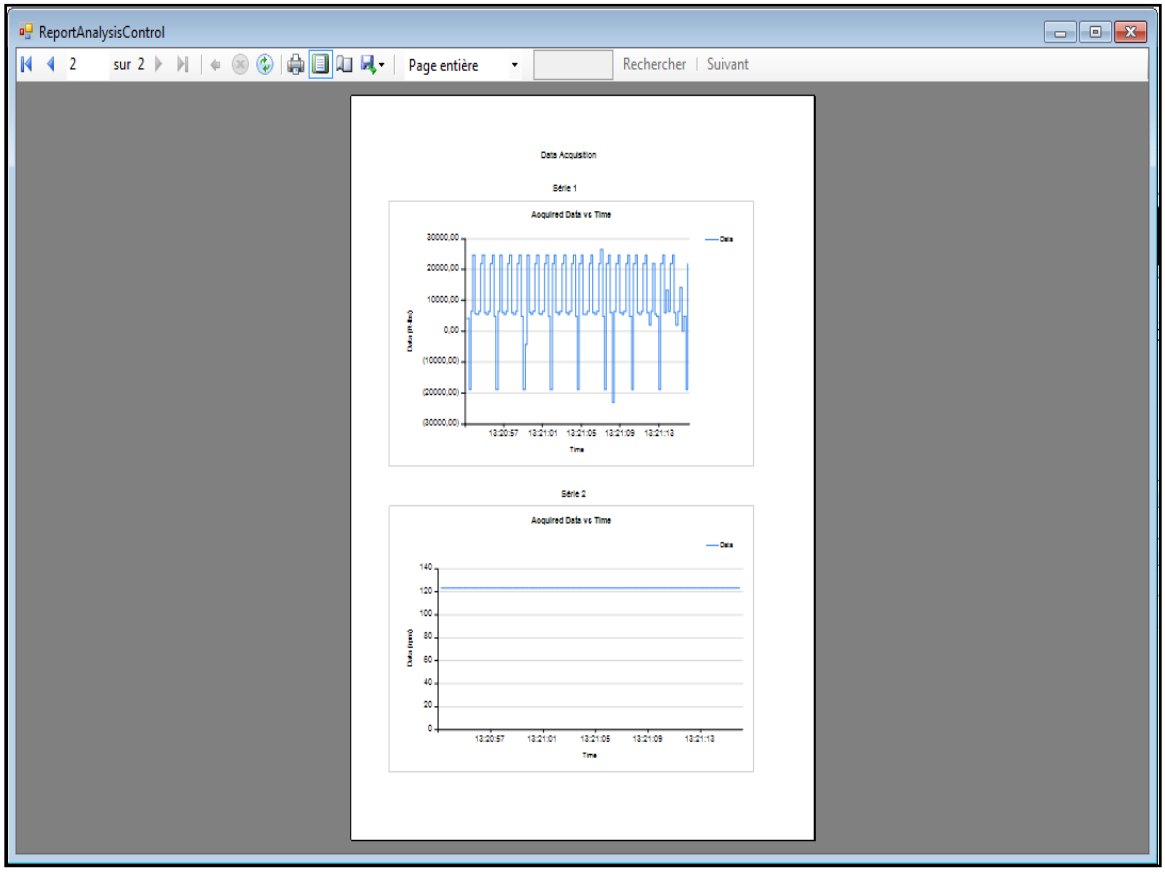

Figure 13 - Interface du générateur de rapport

Pour l'interface d'analyse de dépassement de seuil, il y a le même bouton que les interfaces précédentes pour ouvrir le fichier. Au chargement du fichier le contrôle *ListView* est rempli par les données. Chaque ligne est composée ainsi du moment où la donnée a été acquise et la valeur de cette dernière. De plus, les données dépassant le seuil sont soulignées en rouge orange. Par contre, il n'y a pas les fonctionnalités de génération de rapports et d'exportation d'une certaine zone de données. La raison pour laquelle ces fonctionnalités n'ont pas été implémentées est que le fichier en tant que tel ne contient pas assez d'éléments pour faire un rapport qui soit vraiment utile à l'utilisateur, même chose pour l'exportation de données où le fichier ne contient pas assez une grande quantité de données pour que la fonctionnalité soit

utile.

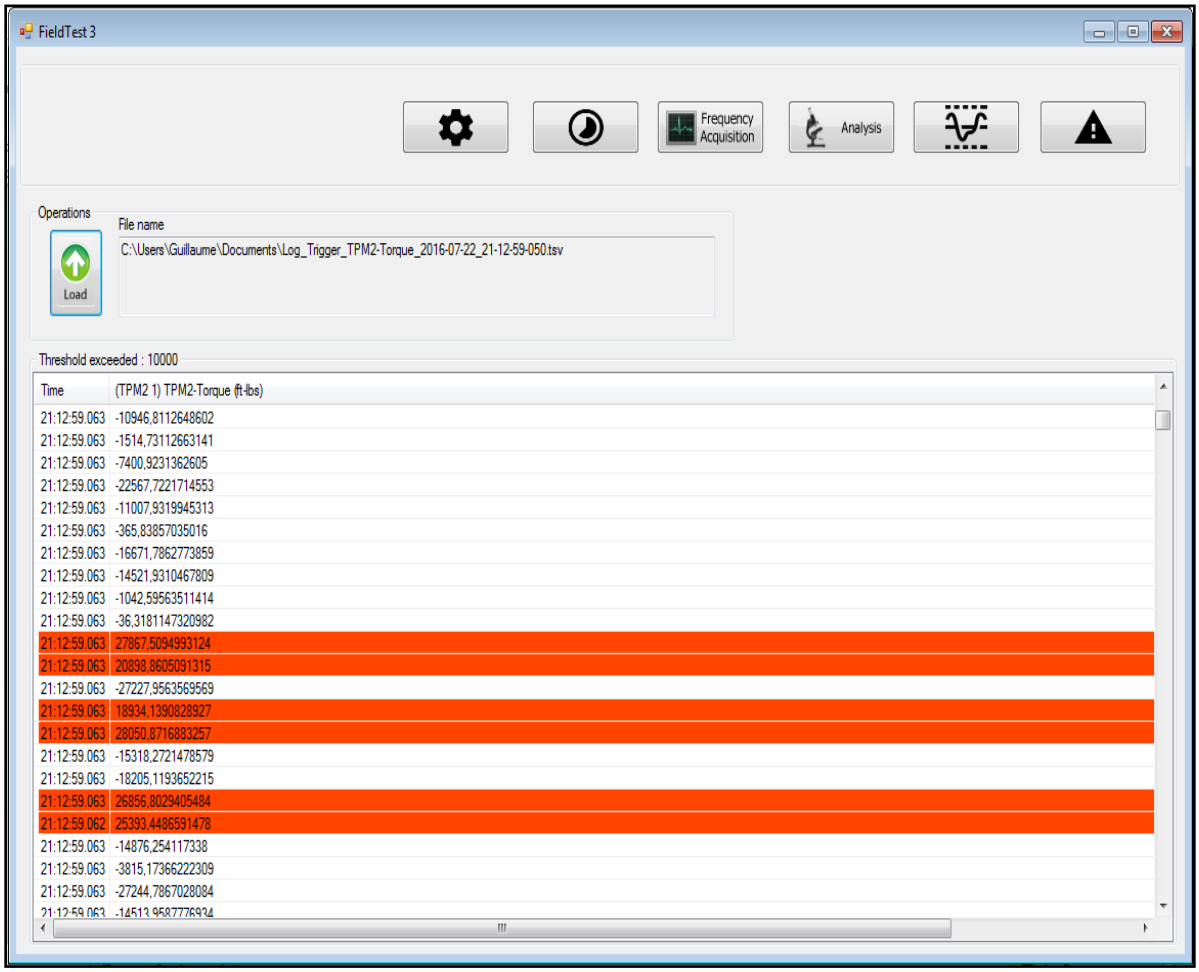

Figure 14 - Interface de l'analyse du dépassement de seuil

Pour l'interface analysant les données des erreurs, cette dernière est la même que celle du dépassement de seuil. La seule différence vient lors du chargement du fichier où le message d'erreur est également ajouté au contrôle *ListView*. Par contre, par manque de temps cette fonctionnalité n'a pas pu bien être testée et celle-ci n'est également pas présente dans la version bêta du logiciel utilisé présentement.

Voir **Annexe 11** pour plus de détails sur la conception liée à l'analyse et à la génération de rapport.

6.2 Installateur

6.2.1 Analyse des besoins de l'installateur

Pour l'installateur, il sera possible de le télécharger sur internet pour un client ayant acheté un appareil d'acquisition de données. Ainsi, cet installateur doit être très léger afin de faciliter son téléchargement sur internet.

Dans cet installateur, il faut les éléments de base soit l'exécutable pour l'installation et l'exécutable pour la désinstallation. Aussi, il faut le guide utilisateur et les fichiers expliquant la manière de fonctionner en général de l'application soit l'acquisition de données, les formules de calcul, etc...

De plus, il faut intégrer tous les dll nécessaires pour l'utilisation du logiciel. Il faut également s'assurer de faire installer les bons éléments du Framework de Microsoft afin que le client installe la version nécessaire avant l'installation du logiciel lors de l'exécution de l'installateur.

Étant donné qu'il faut un prototype rapidement et que le guide utilisateur et les divers documents à mettre dans l'installateur ne sont pas encore réalisés, un installateur MSI de base comportant seulement l'exécutable pour l'installation et la désinstallation sans la documentation liée au logiciel sera réalisé.

Ainsi, cet installateur comportera, le fichier de configuration de base des appareils d'acquisitions de données, les dll *FTDI* pour faire la détection et le mappage des ports USB avec les appareils d'acquisition, *SharpDX* pour l'exécution de la *FFT* à partir de la carte graphique et les dll de *ReportViewer* pour la génération de rapport. De plus, les bonnes versions de Framework de Microsoft seront également mises sur l'installateur afin de les installer si nécessaire.

6.2.2 Conception et implémentation de l'installateur

L'implémentation a été relativement rapide à faire, car Microsoft Visual Studio offre un outil capable de faire un installateur MSI intégré à la solution. Cet outil *Setup projet* peut être inclus dans la solution du projet et s'adapte au changement dans la solution entière. Ainsi, il n'est pas nécessaire de refaire un installateur à chaque fois qu'il y a un changement, il faut seulement recompiler la solution.

De base le *Setup projet* inclus dans la solution 3 dossiers représentant chacun d'eux un emplacement d'installation. Soit les dossiers *Application Folder* représentant le dossier du logiciel lui-même. C'est à cet endroit que tous les dll et fichiers liés à l'utilisation du logiciel doivent être déposés. C'est également à cet endroit que le logiciel à installer est défini.

Ensuite, il y a le dossier *User's Desktop* pour l'installation d'éléments sur le bureau. À cet emplacement, un raccourci pointant sur le logiciel défini précédemment a été créé.

Le dossier *User's Programs Menu* correspond à l'installation du menu « démarrer ». Ainsi, c'est à cet emplacement qu'un dossier Field Test 3 a été créé avec un raccourci pointant sur le logiciel et avec un désinstalleur de ce dernier. Ce désinstalleur a été créé en important le dossier *System Folder* contenant *msiexec.exe* qui permet de créer le désinstalleur.

Voir image ici-bas pour l'architecture de l'installateur MSI :

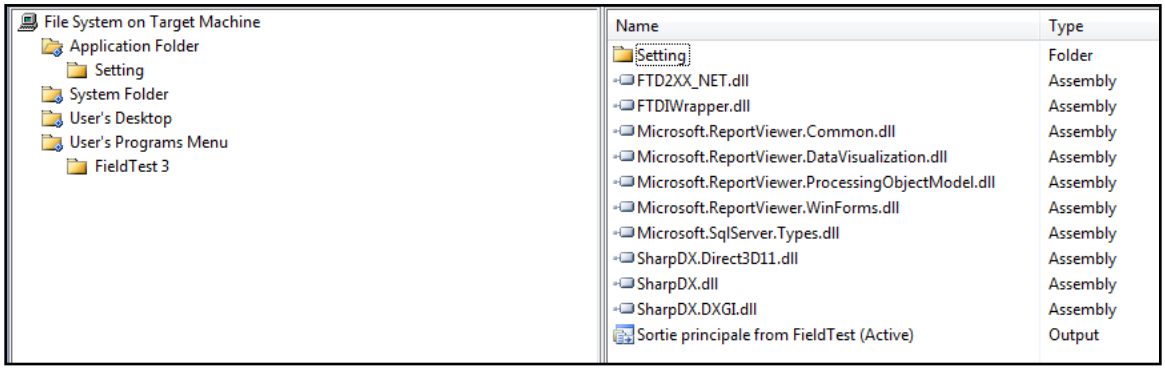

Figure 15 - Architecture de l'installateur MSI

Pour ce qui est des éléments au niveau du Framework nécessaire à l'utilisation du logiciel, un programme d'installation et de vérification de Framework a été créé afin de s'exécuter avant l'installation du logiciel lui-même. Les éléments du Framework sont ainsi inclus dans l'installateur MSI. Ainsi, la version 4.6 du Framework et la version 4.0 Client Profile sont nécessaires à l'exécution du logiciel.

6.3 Revue et test

Pour les fonctionnalités d'analyse de données et de génération de rapport pour les données journalisées standard, à haute fréquence et liés au dépassement de seuils, tous sont fonctionnels. Par contre, il serait intéressant de mettre un format de fichier standardisé comme *Tab-separated values* (TSV) ainsi le fichier serait également compatible sur Excel par exemple. De plus, il serait possible d'ajouter d'autres éléments statistiques à l'analyse et aux rapports générés comme les valeurs maximums et minimums obtenues par exemple. Pour ce qui est de l'analyse liée aux erreurs, il resterait à tester davantage pour voir si cette fonctionnalité est fonctionnelle à 100 %.

Pour l'installateur, ce dernier est fonctionnel et également utilisé en version bêta. Par contre, il resterait à ajouter les diverses documentations liées à l'utilisation du logiciel.

CONCLUSION

En règle générale, toutes les fonctionnalités ont été implémentées, mais pas nécessairement testées à 100 %. Ainsi, le *Publisher*-Subscriber a été implémenté afin d'améliorer légèrement la performance liée à l'acquisition des données avec son envoi de manière implicite des données aux contrôles. Également, le TPM2 a été ajouté au logiciel. Ainsi, son protocole de communication a été implémenté et des tests unitaires ont été réalisés sur les calculs liés à chaque lecture de l'appareil pour vérifier la validité des valeurs retournées. Le reste des tests sur le TPM2 ont été faits via Arduino. Par contre, il reste beaucoup d'éléments à tester liés à la configuration de l'appareil et au retour d'erreur. De plus, une manière de détection et de monitorage de port a été implémentée afin de pouvoir vérifier en tout temps l'état des ports lors de l'acquisition de données. Ensuite, une amélioration de l'acquisition à haute fréquence a été faite afin d'accélérer la vitesse d'exécution de la FFT. Ainsi, le GPU a été utilisé afin d'atteindre une meilleure performance avec cette dernière. Par la suite, une fonctionnalité d'analyse pour tous les types de données journalisés a été ajoutée. Ainsi, dans cette dernière, il est possible d'ouvrir les fichiers journalisés et de visualiser les données via des listes et des graphiques. Il est également possible de visualiser la moyenne des données pour chaque lecture d'appareils. Dans le cas de données journalisées standard et à haute fréquence, il est également possible de générer un rapport afin de garder une trace plus visuelle des données. Pour finir, un installateur MSI a été créé afin de pouvoir fournir une version bêta. Ainsi, cet installateur MSI comporte l'installateur, le désinstalleur et toutes les composantes pour faire fonctionner le logiciel et la configuration XML des appareils de base.

La suite serait de continuer en premier lieu les tests sur le TPM2 afin de tester son vrai comportement. Ensuite, changer les icônes et interfaces utilisateurs afin de les rendre d'une meilleure qualité à l'aide d'un graphiste. Il serait également intéressant d'ajuster davantage les paramètres de la FFT afin d'améliorer la démarcation des pics d'amplitude dans les données. Pour ce qui est des diverses fonctionnalités d'analyse, il serait intéressant d'ajouter plus de statistique que seulement la moyenne afin d'avoir plus d'information sur les données. Même chose pour les rapports où il serait intéressant d'avoir plus de détails. Pour ce qui est

de l’installateur, il resterait seulement à ajouter les divers documents à la compréhension du logiciel.

Il sera possible d’effectuer ces changements dans les temps avenir, car je vais travailler pour OPDAQ dans les prochaines semaines. Cette occasion me permettra également d’ajouter un nouvel appareil au logiciel soit le TT20K. Ainsi, il sera possible de voir si l’ajout d’un nouvel appareil se fait rapidement.

LISTE DE RÉFÉRENCES

Document PDF présentant le protocole de communication du TPM2, [En ligne] Disponible sur: http://www.binsfeld.com/userfiles/filemanager/375/

BIBLIOGRAPHIE

Site internet de l'entreprise Binsfeld, [En ligne] Disponible sur: http://www.binsfeld.com/

Document PDF présentant le protocole de communication du TPM2, [En ligne] Disponible sur: http://www.binsfeld.com/userfiles/filemanager/375/

Site internet de l'entreprise OPDAQ systèmes, [En ligne] Disponible sur: http://www.opdaq.com/

Site internet de la librairie SharpDX, [En ligne] Disponible sur: http://sharpdx.org/wiki/class-library-api/

Site internet MSDN de Microsoft pour Publisher-Subscriber c#, [En ligne] Disponible sur: https://msdn.microsoft.com/fr-ca/library/w369ty8x.aspx

Site internet MSDN de Microsoft pour RDLC, [En ligne] Disponible sur: https://msdn.microsoft.com/fr-fr/library/ms252067.aspx

Site internet de FTDIChip pour exemple d'utilisation, [En ligne] Disponible sur : http://www.ftdichip.com/Support/SoftwareExamples/CodeExamples/CSharp.htm

ANNEXE I

PROTOCOLE DE COMMUNICATION DU TPM2

Fichier PDF « **866600-C_A.pdf »** dans le dossier de remise.

ANNEXE II

DIAGRAMME DE CLASSES DU TPM2

Image « **Diagramme de classes du TPM2.png »** dans le dossier « Diagrammes de classes » du dossier de remise.

ANNEXE III

DIAGRAMME DE CLASSES DU PUSBLISHER-SUBSCRIBER

Image « **Diagramme de classes Publisher-Subscriber.png »** dans le dossier « Diagrammes de classes » du dossier de remise.

ANNEXE IV

DIAGRAMME DE CLASSES DE LA DÉTECTION ET MONITORAGE DE PORT

Image « **Diagramme de classes détection et monitorage de port.png »** dans le dossier « Diagrammes de classes » du dossier de remise.

ANNEXE V

DIAGRAMME DE CLASSES DE L'ACQUISTION HAUTE FRÉQUENCE

Image « **Diagramme de classes acquisition à hautefréquence.png** » dans le dossier « Diagrammes de classes » du dossier de remise.

ANNEXE VI

FICHIER DE JOURNALISATION DE DONNÉES STANDARD

Fichier « **Default_Time_ACQ_1.txt** » dans le dossier « Fichiers journalisés» du dossier de remise.

ANNEXE VII

FICHIER DE JOURNALISATION DE DONNÉES HAUTE FRÉQUENCE

Fichier « **Default_HF_ACQ_17.txt »** dans le dossier « Fichiers journalisés» du dossier de remise.

ANNEXE VIII

FICHIER DE JOURNALISATION DE DONNÉES DE DÉPASSEMENT DE SEUIL

Fichier « **Log_Trigger_TPM2-Torque_2016-07-22_21-12-59-050.txt »** dans le dossier « Fichiers journalisés » du dossier de remise.

ANNEXE IX

EXEMPLE DE RAPPORT POUR L'ACQUISITION STANDARD

Fichier pdf « **Report Normal**.**pdf »** dans le dossier « Exemples Rapports » du dossier de remise.

ANNEXE X

EXEMPLE DE RAPPORT POUR L'ACQUISITION À HAUTE FRÉQUENCE

Fichier pdf « **Report High.pdf »** dans le dossier « Exemples Rapports » du dossier de remise.

ANNEXE XI

DIAGRAMME DE CLASSES DE L'ANALYSE ET DE LA GÉNÉRATION DE RAPPORT

Image « **Diagramme de classes analyse et génération.png »** dans le dossier Diagramme de classes du dossier de remise.

ANNEXE XII

CAS D'UTILISATION

Fichier pdf « **Cas d'utilisation.pdf »** dans le dossier de remise.