

UNIVERSITÉ DE TOULON

IUT DE TOULON

Département Génie Électrique et Informatique Industrielle (GEII)

# Rapport de Stage

## Conception d'un outil de décodage de trames CAN

*Problématique : Résolution de problèmes de communication d'un véhicule  
d'exploration sous-marine*

**Réalisé par :** Lucas COQUARD - BUT3 GEII : Parcours Électronique et Systèmes  
Embarqués (ESE)

**Entreprise d'accueil :** EXAIL ROBOTICS, La Garde

**Tuteur entreprise :** Philippe Weingertner

**Tuteur enseignant :** Philippe Arlotto

**Date de rendu :** 26 juin 2025

# Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réussite de mes projets et à mon parcours académique.

Tout d'abord, je souhaite remercier Philippe Weingertner, mon maître de stage et Responsable Recherche & Développement chez Exail, pour son accompagnement tout au long de mon stage. Son soutien et ses conseils ont été très précieux.

Un grand merci également à Romain Deur, Ingénieur Développement chez Exail, avec qui j'ai collaboré pour mettre en place les tests et bien d'autres aspects du projet. Travailler à ses côtés a été une expérience très enrichissante.

Pour le second projet sur lequel j'ai travaillé, je remercie Pascale Meltzheim, Ingénieure Études, pour sa collaboration et son aide précieuse dans l'exploration des détails du projet. Je suis également reconnaissant envers Philippe Chiarlone, Responsable Ingénierie MCO, qui m'a confié ce sujet en accord avec mon maître de stage. Philippe a également été mon maître de stage l'année dernière et je lui suis très reconnaissant pour son soutien continu.

Je tiens aussi à remercier Philippe Arlotto, professeur à l'IUT de Toulon et mon tuteur de stage, pour sa visite et son intérêt pour le bon déroulement de ce dernier.

Enfin, en tant que dernier travail que je rends dans mes études, je souhaite exprimer toute ma gratitude à l'IUT de Toulon et à toute l'équipe enseignante du BUT GEII pour la formation et les opportunités qu'ils m'ont offertes tout au long de mon parcours académique.

# Table des matières

|                                                                           |           |
|---------------------------------------------------------------------------|-----------|
| <b>Introduction</b>                                                       | <b>4</b>  |
| <b>1 L'entreprise et le secteur d'activité</b>                            | <b>5</b>  |
| 1.1 Présentation du secteur d'activité général . . . . .                  | 5         |
| 1.1.1 Exail en chiffres . . . . .                                         | 7         |
| 1.1.2 Activités . . . . .                                                 | 7         |
| 1.1.3 Les marchés . . . . .                                               | 10        |
| 1.2 Les spécificités de l'entreprise . . . . .                            | 11        |
| <b>2 Mon activité de stagiaire</b>                                        | <b>13</b> |
| 2.1 Environnement technique et contraintes liées . . . . .                | 13        |
| 2.2 Présentation des missions principales . . . . .                       | 14        |
| 2.2.1 Objectifs fixés . . . . .                                           | 14        |
| 2.2.2 Présentation du véhicule . . . . .                                  | 15        |
| 2.2.3 Identification du problème de communication . . . . .               | 20        |
| 2.2.4 Fonctionnalités principales de l'outil . . . . .                    | 23        |
| 2.2.5 Utilisation de l'outil . . . . .                                    | 29        |
| 2.2.6 Phase de test . . . . .                                             | 34        |
| 2.3 Projet annexe . . . . .                                               | 41        |
| 2.3.1 Identification du besoin . . . . .                                  | 41        |
| 2.3.2 Réalisation du logiciel de création de fichier Word . . . . .       | 41        |
| 2.3.3 Réalisation de l'interface de sélection de fichiers . . . . .       | 47        |
| <b>3 Bilan de mon activité de stagiaire</b>                               | <b>49</b> |
| 3.1 Présentation des résultats . . . . .                                  | 49        |
| 3.1.1 Sujet principal du stage . . . . .                                  | 49        |
| 3.1.2 Sujet secondaire du stage . . . . .                                 | 49        |
| 3.2 Prolongements possibles de la mission réalisée . . . . .              | 50        |
| 3.2.1 Ajout de l'analyse en temps réel . . . . .                          | 50        |
| 3.2.2 Amélioration de la réactivité de la page HTML . . . . .             | 50        |
| 3.3 Apports en termes d'expérience personnelle et disciplinaire . . . . . | 51        |
| <b>Conclusion</b>                                                         | <b>53</b> |
| <b>Lexique</b>                                                            | <b>54</b> |
| <b>Annexes</b>                                                            | <b>i</b>  |

# Table des figures

|      |                                                                                                                                  |    |
|------|----------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1  | Présence de l'entreprise à travers le monde . . . . .                                                                            | 5  |
| 1.2  | Spécificités des sites Exail en France et en Belgique . . . . .                                                                  | 6  |
| 1.3  | Évolution des prises de commandes sur 5 ans . . . . .                                                                            | 7  |
| 1.4  | Fonctionnement du systèmes UMIS . . . . .                                                                                        | 8  |
| 1.5  | Schéma du système INS . . . . .                                                                                                  | 9  |
|      |                                                                                                                                  |    |
| 2.1  | Photo du véhicule sous-marin Ulyx . . . . .                                                                                      | 15 |
| 2.2  | Synoptique général d'un POD avec STM32 . . . . .                                                                                 | 16 |
| 2.3  | Architecture générale d'un système à double niveau . . . . .                                                                     | 18 |
| 2.4  | Structure d'une trame CAN avec identifiant étendu . . . . .                                                                      | 18 |
| 2.5  | Mise en évidence des problèmes de communication avec des données<br>récupérées depuis la partie haut niveau du système . . . . . | 21 |
| 2.6  | Exemple de la structure d'un fichier CSV généré par le logger IXXAT . . .                                                        | 24 |
| 2.7  | Diagramme de séquence de la fonction de parsing du logiciel . . . . .                                                            | 24 |
| 2.8  | Exemple de la structure d'un fichier TXT généré par le logger CL2000 . . .                                                       | 25 |
| 2.9  | Schéma explicatif de la redistribution des messages . . . . .                                                                    | 26 |
| 2.10 | Page de sélection de la configuration de l'interface . . . . .                                                                   | 29 |
| 2.11 | Structure du dossier <b>configs</b> . . . . .                                                                                    | 30 |
| 2.12 | Page de sélection de l'interface à visualiser . . . . .                                                                          | 31 |
| 2.13 | Page de visualisation des trames brutes . . . . .                                                                                | 32 |
| 2.14 | Structure du dossier <b>decoded_logs</b> . . . . .                                                                               | 33 |
| 2.15 | Page de visualisation du rapport de distribution des trames . . . . .                                                            | 33 |
| 2.16 | Synoptique d'un POD NRJ . . . . .                                                                                                | 35 |
| 2.17 | Montage à réaliser pour la récupération de logs . . . . .                                                                        | 36 |
| 2.18 | Brochage du connecteur DB9 de la sonde IXXAT . . . . .                                                                           | 37 |
| 2.19 | Visualisation des trames décodées dans le fichier HTML . . . . .                                                                 | 37 |
| 2.20 | Visualisation des graphiques dans le fichier HTML . . . . .                                                                      | 38 |
| 2.21 | Visualisation des trames décodées dans le fichier Excel . . . . .                                                                | 38 |
| 2.22 | Brochage du connecteur DB9 du CL2000 . . . . .                                                                                   | 40 |
| 2.23 | Montage de la manipulation avec le CL2000 . . . . .                                                                              | 40 |
| 2.24 | Modèle de référence pour les tableaux des fichiers de type CRE . . . . .                                                         | 42 |
| 2.25 | Tableau de test rempli avec les données extraites du fichier XML . . . . .                                                       | 44 |
| 2.26 | Interface de sélection des fichiers et du dossier de sortie . . . . .                                                            | 47 |

# Liste des tableaux

|     |                                                                               |    |
|-----|-------------------------------------------------------------------------------|----|
| 1.1 | Les principaux secteurs d'activité et domaines d'application chez Exail . . . | 6  |
| 1.2 | Les principaux domaines d'application par secteurs . . . . .                  | 10 |
| 2.1 | Fonctions principales des différents PODs (hors fonctionnalités récurrentes)  | 17 |
| 2.2 | Répartition des fonctions des PODs ACT bâbord et tribord . . . . .            | 17 |
| 2.3 | Structure d'un identifiant étendu (29 bits) NMEA 2000 . . . . .               | 18 |
| 2.4 | Exemple de filtres générés par l'outil . . . . .                              | 20 |
| 2.5 | Informations extraites par les méthodes de parsing selon le type de fichier . | 25 |

# Introduction

Étudiant en troisième année de BUT Génie Électrique et Informatique Industrielle, spécialité Électronique et Systèmes Embarqués, ma formation comprend un stage de quatorze semaines obligatoires.

Pour ma part, j'effectue mon stage au sein de l'entreprise Exail Robotics à La Garde. Exail est une entreprise industrielle spécialisée dans les hautes technologies dans le domaine de la robotique autonome avec une intégration verticale des métiers. Le groupe propose des systèmes complexes de drones, de navigation, ainsi que des produits pour l'aérospatial et la photonique.

Durant ces quatorze semaines, ma mission principale a été la conception d'un logiciel de visualisation, d'analyse et de décodage de trames CAN dans le cadre de la résolution d'un problème de communication sur un véhicule sous-marin livré à l'IFREMER (Institut Français de Recherche pour l'Exploitation de la MER). Ce document a pour but de présenter mon travail.

Dans un premier temps, pour mieux cerner les activités de l'entreprise dans laquelle j'opère, je vous la présenterai, elle, ainsi que son secteur d'activités. Ensuite, je détaillerai avec précision quelles ont été les tâches que j'ai réalisées pour compléter ma mission. Enfin, nous verrons ce que ce stage m'a apporté ainsi que les évolutions possibles de mon projet.

# 1. L'entreprise et le secteur d'activité

## 1.1 Présentation du secteur d'activité général

Exail est une entreprise née en 2022 de la fusion entre iXblue et le Groupe ECA. Fondée en 1936, ECA s'est d'abord illustrée dans le secteur aéronautique avant de réorienter ses activités vers les systèmes maritimes, notamment au travers de la robotique sous-marine. Le groupe devient filiale du Groupe Gorgé en 1992, avant une fusion complète en 2020. Deux ans plus tard, le Groupe Gorgé rachète iXblue, entreprise spécialisée dans les technologies de navigation, la photonique et l'autonomie maritime, pour donner naissance à Exail.

Cette fusion stratégique a permis de mutualiser les compétences complémentaires des deux entités. ECA Group, sous-traitant de renom pour des groupes européens comme Airbus, Naval Group ou Thales, proposait des systèmes robotiques de mission vendus dans plus de 80 pays. iXblue, de son côté, apportait son expertise technologique dans des solutions de navigation inertielle, de positionnement, et des applications scientifiques de pointe.

Aujourd'hui, Exail possède 21 sites industriels en France, deux en Belgique, ainsi que deux laboratoires de recherche communs. À cela s'ajoute un réseau international avec des branches implantées sur plusieurs continents :



FIGURE 1.1 – Présence de l'entreprise à travers le monde

Les sites en dehors de la France et de la Belgique sont principalement dédiés aux activités commerciales et de support aux opérations, tandis que les sites industriels français se

distinguent par des spécialités techniques héritées des sociétés fondatrices :

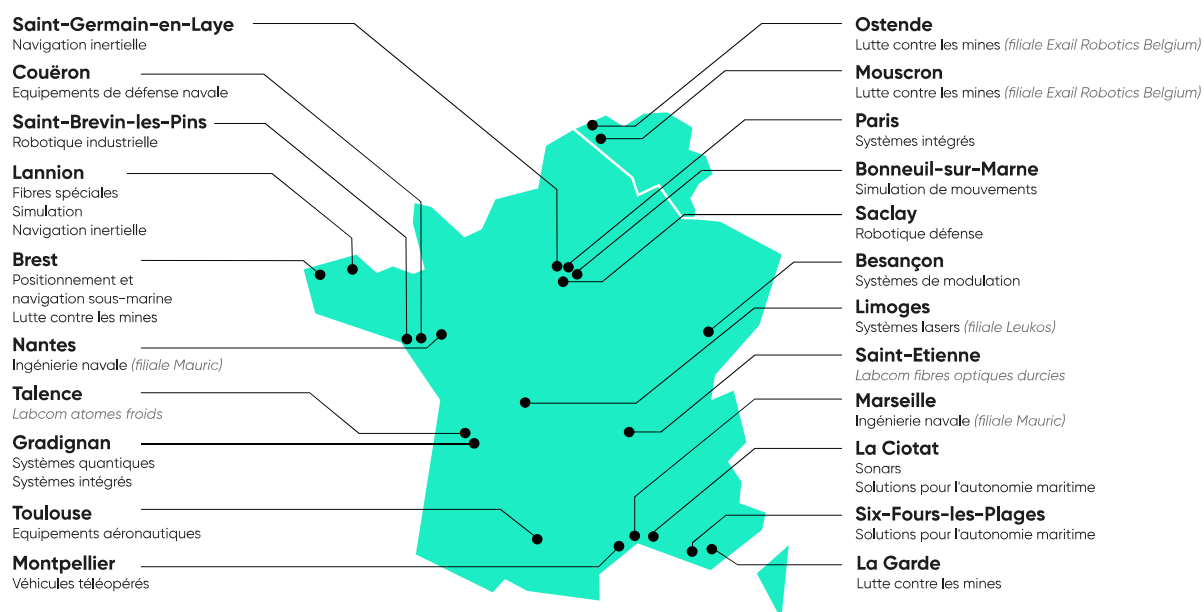


FIGURE 1.2 – Spécificités des sites Exail en France et en Belgique

Grâce à cette implantation stratégique, Exail est présent et compétitif dans six grands secteurs :

| Secteurs                | Domaines d'application                                                                                       |
|-------------------------|--------------------------------------------------------------------------------------------------------------|
| Défense et sécurité     | Navigation navale et terrestre, protection de navires, MCM, MDA, sécurité intérieure, simulation             |
| Maritime                | Énergies, hydrographie, océanographie, géosciences, pêche                                                    |
| Spatial                 | Navigation, communications, ballons stratosphériques                                                         |
| Aéronautique            | Maintenance au sol, moyens de test, lignes d'assemblage, simulation d'entraînement                           |
| Industrie et logistique | Ferroviaire, tunneliers, traitement des matériaux, cartographie mobile, nucléaire, véhicules autonomes       |
| Recherche               | Technologies quantiques, océanographie, photonique, surveillance terrestre, grands instruments scientifiques |

TABLE 1.1 – Les principaux secteurs d'activité et domaines d'application chez Exail

Les activités d'Exail dans la recherche sont plutôt importantes puisque le groupe réinvestit 20 % de ses revenus dans le domaine.

### 1.1.1 Exail en chiffres

Au cours de l'année 2025, le groupe Exail a enregistré 373 millions d'euros de recettes, soit une augmentation de 16 % par rapport à l'année 2023. Il est également intéressant de regarder l'EBITDA (Bénéfices avant intérêts, impôts, dépréciation et amortissement) de cette même année pour mieux visualiser les gains réels de l'entreprise. Toujours en 2024, l'entreprise a réalisé une marge de 20 %, soit 74 millions d'euros. Ces chiffres illustrent la présence de l'entreprise sur le marché.

Au cours des dernières années, Exail a connu une forte croissance. En 2025, l'entreprise compte plus de 2 000 collaborateurs à travers le monde, ce qui témoigne de son expansion continue et de son engagement envers l'innovation et l'excellence opérationnelle.

Comme l'illustre le graphique ci-dessous, Exail ne cesse de croître et d'occuper une place de plus en plus importante au fil des ans. Sur quatre ans, les prises de commandes, en euros, ont augmenté de plus de 105 % pour atteindre les 451 millions d'euros sur l'année 2024.

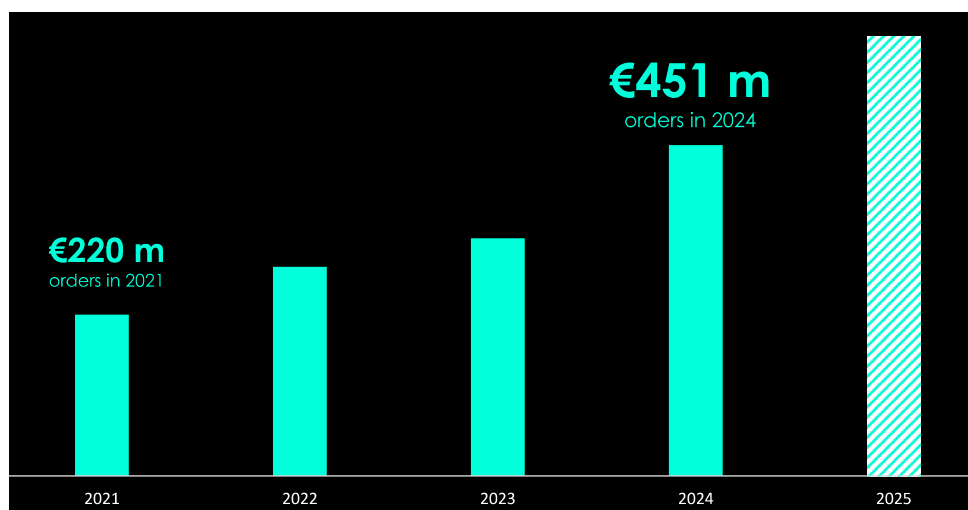


FIGURE 1.3 – Évolution des prises de commandes sur 5 ans

### 1.1.2 Activités

Exail fournit essentiellement des systèmes complexes pour des missions critiques. Les applications principales de ces systèmes sont les systèmes de drones autonomes, notamment utilisés pour des opérations de déminage sous-marin ou d'hydrographie et de cartographie des fonds marins. L'autre application principale est les systèmes de navigation et de positionnement avec des applications de défense navale et des opérations maritimes civiles ou encore sur des véhicules de défense terrestre et de positionnement par satellite.

## Les solutions phares générant la majeure partie des ventes

Le système maritime avancé de contre-mesures antimines (MCM) multi-drones d'Exail de la troisième génération, UMIS, est le premier système de lutte antimines de troisième génération à garantir la sécurité et l'efficacité des opérations de guerre des mines. Le système UMIS intègre des véhicules sous-marins autonomes (AUV), des véhicules de surface sans pilote (USV), des véhicules aériens sans pilote (UAV) et des systèmes d'identification et de destruction des mines (MIDS) pour détecter, classer, identifier et neutraliser les mines marines. L'UMIS se distingue par le fait qu'il peut être mis en place à partir d'un navire mère, d'un navire d'opportunité ou depuis la côte.

UMIS optimise l'efficacité et la rapidité des opérations de déminage grâce à la parallélisation des missions en lançant et en gérant plusieurs drones en parallèle pour couvrir simultanément une grande zone.

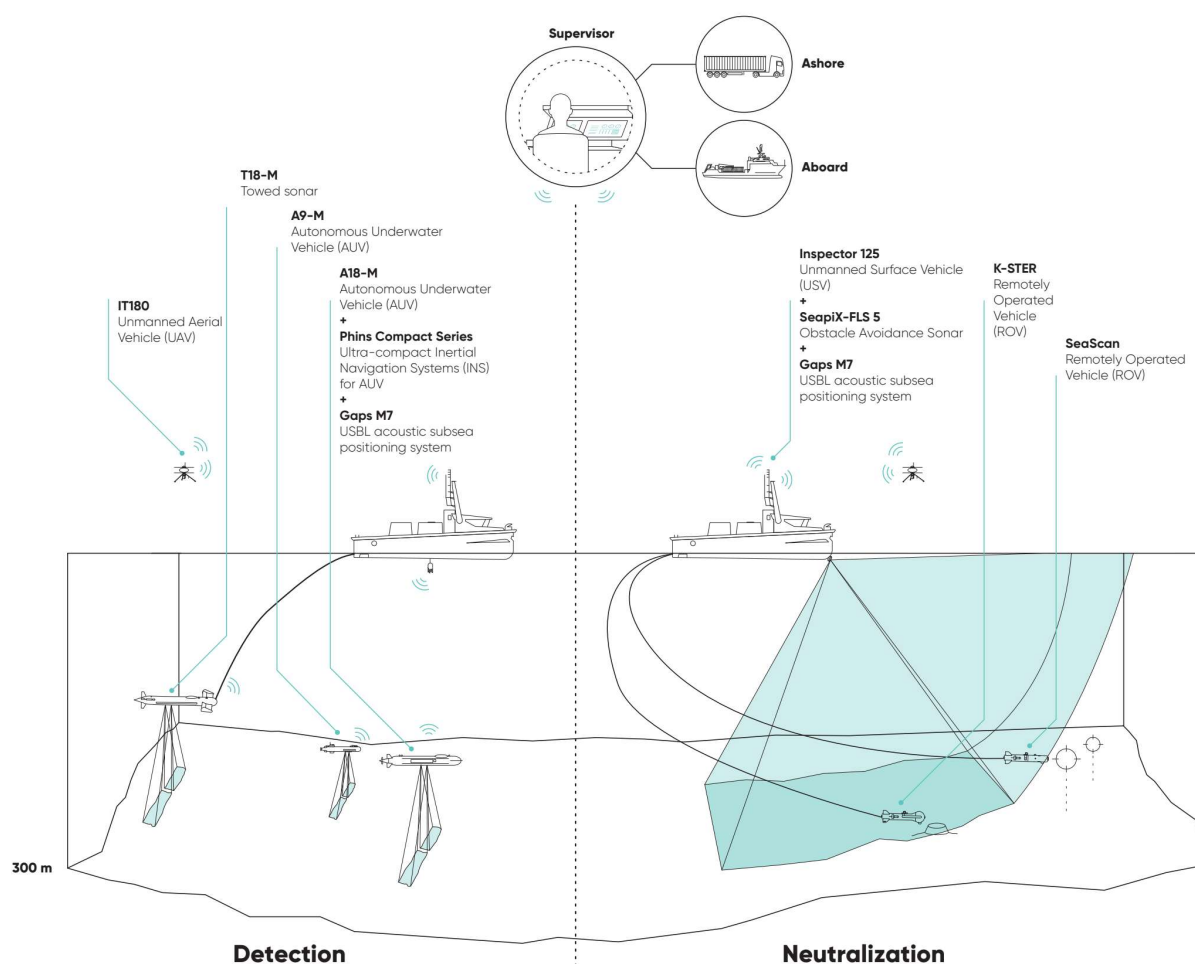


FIGURE 1.4 – Fonctionnement du systèmes UMIS

Exail propose également le système INS (Inertial Navigation Systems), une technologie de haute performance basée sur un gyroscope à fibre optique et des accéléromètres à quartz. Ces systèmes, compacts et légers, équipent plus de 80 % des véhicules sous-marins dans les

industries de l'énergie et des géosciences. Ils offrent une précision de navigation inégalée pour les ROVs et AUVs dans des environnements exigeants, avec une précision de cap exceptionnelle.

La gamme de produits INS d'Exail, idéale pour des applications variées, permet une navigation optimale jusqu'à 6 000 mètres de profondeur. Ces systèmes s'intègrent parfaitement avec les DVL de Nortek et Teledyne RD Instruments, améliorant ainsi la précision et la stabilité. Leur conception modulaire et plug-and-play permet une intégration facile et flexible.

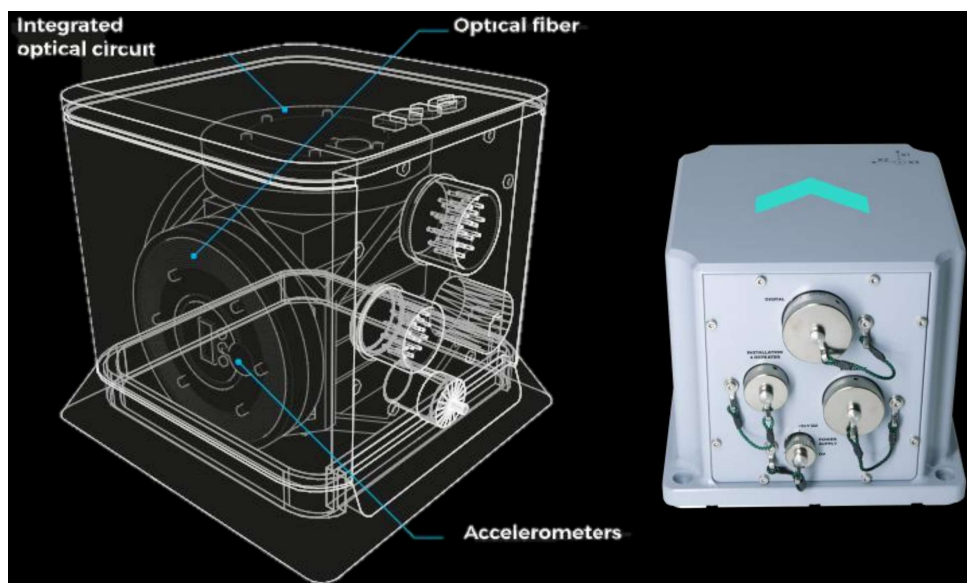


FIGURE 1.5 – Schéma du système INS

### Deux segments intégrés verticalement et en croissance

Les activités de l'entreprise sont essentiellement divisées en deux segments.

Le premier, le segment navigation et robotique maritime, représente 364 millions d'euros de commandes en 2024. Cette partie des activités est alimentée par de nombreuses commandes de systèmes robotiques maritimes, avec des nouveaux clients, mais aussi des clients déjà existants. La dynamique de la navigation se poursuit, stimulée par les applications civiles et de défense terrestre civile.

Les 87 millions d'euros de commandes restants sur l'année 2024 correspondent au segment technologies avancées. Au cours des dernières années, il est possible de remarquer une demande croissante de sources laser haut de gamme, en particulier pour les applications spatiales et les télécommunications. L'activité a également augmenté en ce qui concerne la demande interne de composants photoniques.

### 1.1.3 Les marchés

Dans les domaines du civil et de la défense, le volume annuel des marchés adressables s'élève à environ 8 milliards d'euros. Le domaine civil en concentre 4,5 milliards, tandis que celui de la défense en regroupe 3,5 milliards.

| Secteur | Domaines d'application                                                                                                                                           |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Civil   | Opérations maritimes et hydrographie, Transport maritime, Positionnement par satellite et communication, Industries laser et quantiques                          |
| Défense | Chasse aux mines et levés militaires, Systèmes de navigation des navires de surface et des sous-marins, Guerre sous-marine, Positionnement des forces terrestres |

TABLE 1.2 – Les principaux domaines d'application par secteurs

Ces dernières années, les dépenses en matière de défense augmentent fortement en Europe, avec 42 % d'augmentation des investissements dans le domaine. Ces dépenses correspondent aux achats d'équipements, mais aussi à la R&D (Recherche & Développement). En 2024, cet investissement représentait 102 milliards d'euros à l'échelle de l'Europe et pourrait doubler d'ici à 2030, dépassant les 200 milliards d'euros de dépenses annuelles.

Cette croissance des dépenses ouvre de nombreuses opportunités à Exail. En effet, la stratégie européenne de défense à l'horizon 2030 s'articule autour de quatre axes principaux :

- **Les drones parmi les 7 domaines de défense prioritaires** : « Les drones sont une capacité indispensable » ;
- **Préférence européenne** : « D'ici à 2030, au moins 50 % du budget de défense des États membres devrait être consacré à des achats auprès de fournisseurs européens » ;
- **Davantage de soutien pour les entreprises de taille intermédiaire et les PME** : « Les instruments au niveau de l'UE devraient donner la priorité aux PME et augmenter massivement le soutien qui leur est apporté » ;
- **Soutien accru à la R&D et à l'innovation** : « Dans le contexte d'une augmentation substantielle des dépenses de défense, une part plus importante doit être investie dans la recherche, le développement et la technologie en matière de défense ».

Ces quatre axes sont une opportunité de croissance pour Exail, puisque l'entreprise rentre parfaitement dans la stratégie déployée par l'Union européenne.

## **Perspectives de l'entreprise**

Exail est bien positionnée pour remporter les futurs programmes de chasse aux mines sur les prochaines années. En effet, Exail est le seul acteur à disposer d'une gamme complète de drones pour répondre aux missions dites de "guerre des mines" (détection, identification et neutralisation). De plus, Exail possède une solide expérience en matière d'appels d'offres avec environ 90 % d'entre eux remportés depuis 2019. Enfin, Exail propose l'offre la plus compétitive. La solution Exail a été classée première en termes de performance, mais aussi en termes de prix par le client.

## **1.2 Les spécificités de l'entreprise**

Le site de la Garde se spécialise dans le domaine de la guerre des mines et des grands fonds, en se concentrant principalement sur des projets stratégiques européens. Il collabore avec des partenaires internationaux dans le cadre de programmes financés ou cofinancés par des institutions européennes ou des organismes de défense nationaux. Son activité est intrinsèquement liée à d'autres sites de l'entreprise et collabore fortement avec ces derniers.

En raison de sa taille, avec plus de 500 collaborateurs, le site a abandonné une organisation traditionnelle par métiers au profit d'une structure organisée par domaines. En effet, au milieu de l'année 2024, une réorganisation a été mise en place pour optimiser la gestion du site. Dans l'ancienne organisation par métiers, les collaborateurs travaillaient sur plusieurs projets simultanément, ce qui compliquait la gestion et la coordination. De plus, ces derniers n'étaient pas pleinement intégrés aux projets, ce qui pouvait entraîner un manque d'engagement et un suivi insuffisant des travaux réalisés.

La nouvelle organisation, illustrée en Annexe 1, favorise une intégration plus poussée des collaborateurs dans les projets, renforçant ainsi leur engagement et leur intérêt pour les missions auxquelles ils participent. Dans cet organigramme, chaque domaine, représenté en violet, regroupe plusieurs projets nécessitant des ressources techniques similaires. Cette structure permet non seulement une meilleure répartition des ressources, mais aussi la réutilisation de certains travaux à travers différents projets, optimisant ainsi l'efficacité et la cohérence des efforts déployés.

L'ensemble de cette organisation est supervisé par un groupe de direction composé de David Barre, François Falcou, Olivier Viné, Steven Luys et Yann Sambarino. Chacun de ces membres est responsable d'au plus deux domaines représentés dans l'organigramme,

assurant ainsi une supervision ciblée et une structuration claire de l'ensemble des activités. Pour faciliter davantage l'organisation et assurer une gestion fluide des projets, des responsables techniques sont affectés à chaque branche. Ces responsables jouent un rôle clé dans la coordination des efforts techniques et la communication entre les différentes équipes, garantissant que les objectifs des projets sont atteints de manière efficace et dans les délais impartis.

Cette structure organisationnelle permet également une plus grande flexibilité et adaptabilité face aux changements et aux défis techniques, tout en maintenant une vision stratégique cohérente au sein de l'entreprise.

La branche Projets Spéciaux (PSX), où s'effectue mon stage, est un département dédié à la réussite des projets sur cahier des charges spécifique. Elle compte un total de 45 personnes réparties sur deux sites : 8 collaborateurs à Mouscron et 37 à La Garde. La branche PSX gère deux types de projets : ceux destinés à des clients spécifiques et ceux menés en consortium. Actuellement, six projets sont en cours au sein de cette branche.

Les rôles et responsabilités des Projets Spéciaux incluent le pilotage et la réalisation de projets répondant à des besoins clients hors des lignes de produits standards. Cela comprend les évolutions de systèmes spéciaux toujours en usage chez les clients, ainsi que le développement de nouveaux systèmes adaptés à des usages clients spécifiques. La branche vise également à conserver la capacité de répondre aux besoins des clients historiques ou des acteurs majeurs, parallèlement aux produits Toolbox, tout en maintenant les standards de qualité, de coût et de délai des projets industriels d'Exail Robotics.

En outre, la branche PSX pilote des projets de développement en consortiums français et européens, faisant rayonner la marque Exail et son savoir-faire au sein de l'écosystème MCM et grands fonds. Elle identifie les fonctions et produits à faible niveau de maturité technologique (TRL) à développer dans le cadre de ces projets et suit le développement des fonctions et produits créés par d'autres départements dans le cadre de projets aidés. La branche travaille également à identifier des partenaires universitaires ou industriels majeurs pour soutenir ces développements et compléter les capacités de financement de R&D interne par des financements externes, tout en conservant les capacités de transfert de produits et de technologies vers les gammes UMIS. Des contrats d'engagement sont mis en place pour garantir la compatibilité avec la gamme UMIS sur les produits et fonctions applicables, et la branche est partie prenante des cercles métiers.

L'organisation de cette branche (Annexe 2) est divisée en six projets. Pour ma part, je suis intégré dans la branche projet A6K en tant que stagiaire systèmes bas niveau.

## 2. Mon activité de stagiaire

### 2.1 Environnement technique et contraintes liées

Avant de détailler les aspects techniques de mon stage, il est important de noter que celui-ci s'est déroulé en deux phases principales. La première phase était axée sur le développement d'applications logicielles, tandis que la seconde concernait les tests matériels. Cette seconde phase incluait des tâches d'électronique telles que la validation des cartes électroniques, la validation du logiciel embarqué sur ces cartes, ainsi que la fabrication de câbles d'outillage.

Dans le cadre de ce stage, l'environnement de travail était très normé, notamment en raison des sujets traités qui sont souvent liés à des activités militaires.

Pour la partie développement d'applications, le choix s'est porté sur Python avec la librairie CustomTkinter pour l'interface graphique. Cette solution légère, entièrement en Python, a permis une intégration simple dans notre environnement de travail, tout en assurant une bonne réactivité de l'interface et une personnalisation avancée des composants graphiques. Le développement a été effectué sous Visual Studio Code, avec un environnement dédié configuré pour le projet.

En raison des contraintes sécuritaires liées à l'environnement militaire, il est crucial d'éviter de laisser des fichiers .py en clair, car ceux-ci pourraient être modifiés. La génération d'exécutables permet non seulement de sécuriser le code, mais aussi de simplifier son utilisation, car il n'est pas nécessaire d'installer Python et les librairies associées sur chaque machine. Pour ce faire, la librairie PyInstaller est utilisée pour transformer l'ensemble du projet en un binaire exécutable sur n'importe quelle machine.

Bien que d'autres moyens existent pour générer un exécutable, ceux générés par PyInstaller sont considérés comme sûrs et peuvent être exécutés sur les ordinateurs sécurisés contenant des informations sensibles. Pour vérifier ce point, la majorité des bâtiments de l'entreprise sont équipés d'une "station blanche" qui scanne tout le contenu d'une clé USB pour s'assurer qu'aucun fichier n'est porteur de virus. Toute personne voulant insérer une clé USB dans un des ordinateurs de travail doit impérativement la passer dans une "station blanche".

Cela dit, cette méthode n'est pas totalement hermétique : il reste possible de faire du reverse engineering sur les exécutables générés avec PyInstaller. Si le déploiement avait été destiné à l'extérieur ou s'il avait inclus des éléments de propriété intellectuelle critiques, l'utilisation de Nuitka aurait été plus appropriée, car celui-ci compile le code

en C avant de produire un exécutable, rendant l'analyse inverse plus complexe. Dans mon cas, l'exécutable est uniquement destiné à un usage interne et contrôlé, ce qui rend l'utilisation de PyInstaller suffisante et acceptable.

Concernant la partie électronique du stage, notamment le test de cartes et la réalisation de câbles d'outillage, l'utilisation d'un certain nombre d'équipements a été requise. Pour la réalisation des câbles d'outillage, l'utilisation d'un fer à souder a été nécessaire. Pour ce qui est des tests électroniques, une alimentation de laboratoire (60 V - 2 A) a été utilisée pour alimenter les cartes électroniques. Afin d'effectuer des mesures de tension et de courant ainsi que des tests de continuité, un multimètre a été utilisé avec des sondes et des pointes de test.

Pour tester le logiciel, notamment pour récupérer des logs transitant sur le bus CAN (Controller Area Network), deux loggers différents ont été utilisés :

- Sonde IXXAT USB to CAN V2 : Plutôt destiné aux développeurs, permet de se relier à un bus CAN et de visualiser en temps réel les données ou d'enregistrer ces logs dans un fichier `.csv` ;
- Logger CL2000 : Plutôt destiné aux usages opérationnels du véhicule, similaire à la sonde IXXAT mais ayant la particularité de pouvoir enregistrer les logs sur une carte SD et ainsi d'avoir des logs en fonctionnement sans nécessairement avoir besoin d'un ordinateur.

## 2.2 Présentation des missions principales

### 2.2.1 Objectifs fixés

Comme mentionné précédemment dans ce document, le but de ce stage est de développer un logiciel de visualisation, d'analyse et de décodage de trames CAN dans le cadre de la résolution de problèmes de communication entre le haut niveau (IFREMER) et le bas niveau (EXAIL) d'un véhicule sous-marin. Bien que certains outils existent déjà sur le marché, tels que BUS Master ou Can Analyser, ils ne permettent pas le décodage des messages internes spécifiques à l'entreprise, rendant nécessaire le développement d'un outil sur mesure.

En effet, bien que ces outils existants offraient des fonctionnalités intéressantes, ils ne les regroupaient pas toutes au même endroit. Par exemple, BUS Master est excellent pour la simulation et l'envoi de messages CAN, mais il manque de fonctionnalités avancées d'analyse et de visualisation des données. D'un autre côté, Can Analyser offre des capacités de décodage robustes, mais son interface utilisateur est moins intuitive pour une visualisation rapide et efficace des trames CAN.

Ainsi, au terme de ce stage, l'entreprise devait avoir en sa possession un outil capable de parser des logs CAN, de les décoder et de pouvoir générer des rapports comme l'occupation du bus CAN ou l'évolution de certaines données retournées par le véhicule.

### 2.2.2 Présentation du véhicule

Pour rappel, le véhicule sur lequel porte ce stage est nommé ULyX et est réalisé en collaboration avec l'IFREMER. Les développements effectués sont également conçus pour être réutilisables sur l'A6K-M, une version du véhicule destinée à la Marine française.

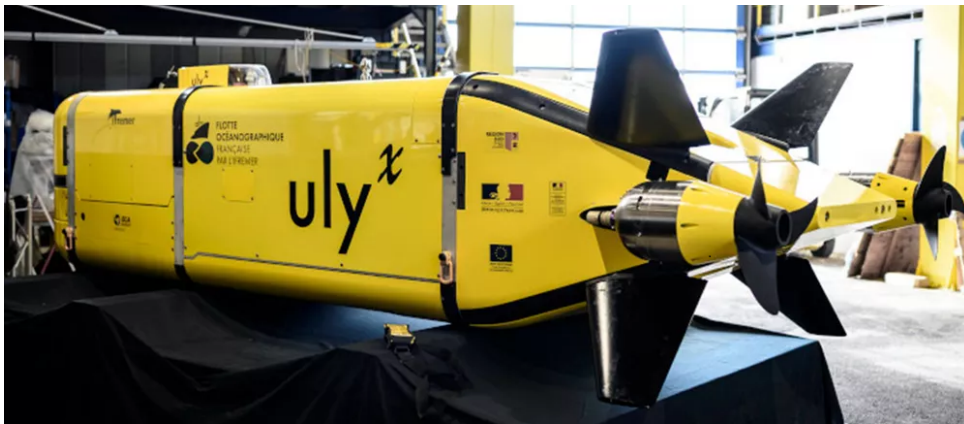


FIGURE 2.1 – Photo du véhicule sous-marin Ulyx

IFREMER est un organisme public français spécialisé dans la recherche et l'exploration des milieux marins profonds. En 2024, il a déployé le drone autonome ULyX, développé avec ECA Group, pour une mission visant à localiser d'anciens fûts de déchets radioactifs immergés dans l'Atlantique dans les années 1950-60. Conçu pour atteindre 6000 mètres de profondeur, ULyX embarque des capteurs avancés et fonctionne sans câble, offrant une grande autonomie et une capacité d'exploration précise des zones difficiles d'accès.

Toutes les communications entre le logiciel et le matériel passent par le bus CAN. Le contrôle à distance implique un chaînon supplémentaire avec la communication acoustique typiquement. Dans ce projet, c'est l'IFREMER qui gère toute la partie haut niveau et Exail se charge de la partie bas niveau.

#### Structure interne du véhicule

À l'intérieur de ce drone sous-marin se trouvent quatre parties principales, appelées PODs. Ces différents PODs sont structurés autour d'un microcontrôleur avec un cœur CORTEX M3 orienté communications (5 ports UART, 2 ports CAN, ...). Chaque POD est relié à un bus CAN principal assurant l'interconnexion avec le système de supervision. Ce dernier transmet les consignes et collecte les états via des trames PGN. Les capteurs et actionneurs sont gérés par le microcontrôleur via des bus locaux (I/O, SPI, RS485, CAN, ...), et leur

contrôle est effectué avec des messages PGN via un bus CAN externe. Il permet aux organes de décision haut niveau d'accéder aux informations et au contrôle/commande bas niveau du véhicule.

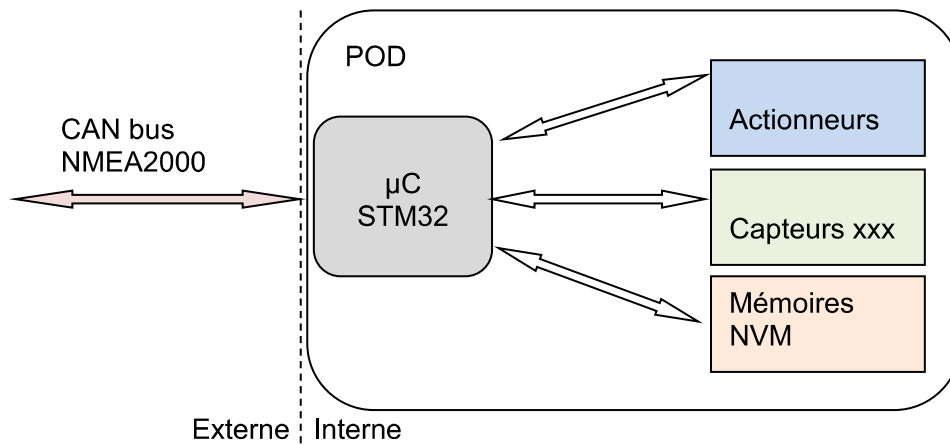


FIGURE 2.2 – Synoptique général d'un POD avec STM32

Tous les PODs ont leurs spécificités, mais possèdent aussi des fonctionnalités récurrentes comme des mesures environnementales (température, pression, humidité) et la diffusion de l'état de fonctionnement (Cbits). Leurs spécificités sont décrites ci-dessous :

| Type de POD                   | Fonctions du POD               | Fonctions principales                                                                                                                                                                                                                                                                                    |
|-------------------------------|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>POD NRJ</b><br>(NRJ1/NRJ2) | Gestion de l'énergie           | <ul style="list-style-type: none"> <li>- Batterie équipée de BMS via bus CAN</li> <li>- Lignes d'alimentation pour PODs VCC, PCC, ACT</li> <li>- Mesure d'isolement électrique</li> <li>- Fonction ON/OFF véhicule</li> </ul>                                                                            |
| <b>POD ACT</b><br>(ACT1/ACT2) | Gestion moteurs et actionneurs | <ul style="list-style-type: none"> <li>- Moteur horizontal</li> <li>- Moteur vertical</li> <li>- 4 gouvernes</li> <li>- Orienteur (de 0 à 45 degrés) pour la propulsion longitudinale, qui permet de créer une force latérale et augmenter la capacité de giration du véhicule</li> <li>- Mât</li> </ul> |
| <b>POD VCC</b>                | Gestion capteurs/navigation    | <ul style="list-style-type: none"> <li>- Synchronisation PPS (Pulse Per Second) in/out</li> <li>- Gestion PPS du GPS</li> <li>- Motorisation d'un mât rétractable</li> </ul>                                                                                                                             |

| Type de POD | Fonctions du POD       | Fonctions principales                                                                                                                      |
|-------------|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| POD PCC     | Gestion charges utiles | <ul style="list-style-type: none"> <li>- Synchronisation PPS in/out</li> <li>- Lignes d'alimentation pour équipements embarqués</li> </ul> |

TABLE 2.1 – Fonctions principales des différents PODs (hors fonctionnalités récurrentes)

De plus, il est nécessaire de préciser que les PODs NRJ et ACT existent chacun en deux exemplaires : NRJ1 et ACT1, NRJ2 et ACT2. Chaque couple NRJ/ACT est physiquement installé d'un côté du véhicule : le couple NRJ1/ACT1 côté bâbord, le couple NRJ2/ACT2 côté tribord. Cette disposition garantit une répartition bilatérale stricte des masses actives et passives, réduisant les moments d'inertie parasites autour de l'axe longitudinal. Chaque côté dispose ainsi de ses propres modules d'alimentation et d'actionnement, ce qui permet de répartir uniformément les masses et d'optimiser la stabilité dynamique du véhicule.

Les PODs ACT intègrent l'électronique de gestion des actionneurs et commandent les éléments suivants :

| Bâbord                                                                                                                                                                                                                       | Tribord                                                                                                                                                                                             |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>- Propulseur longitudinal tribord</li> <li>- Propulseur vertical arrière</li> <li>- Aileron bâbord haut</li> <li>- Aileron tribord haut</li> <li>- Orienteur de propulsion</li> </ul> | <ul style="list-style-type: none"> <li>- Propulseur longitudinal bâbord</li> <li>- Propulseur vertical avant</li> <li>- Aileron bâbord bas</li> <li>- Aileron tribord bas</li> <li>- Mât</li> </ul> |

TABLE 2.2 – Répartition des fonctions des PODs ACT bâbord et tribord

Dans la suite du document, les PODs doubles (NRJ1/NRJ2 et ACT1/ACT2) seront regroupés sous la bannière NRJ ou ACT pour plus de clarté.

## Principe de communication du véhicule

Ces différents PODs communiquent entre eux, mais aussi avec la partie haut niveau. L'identifiant des messages échangés répond à des critères définis par la norme NMEA 2000.

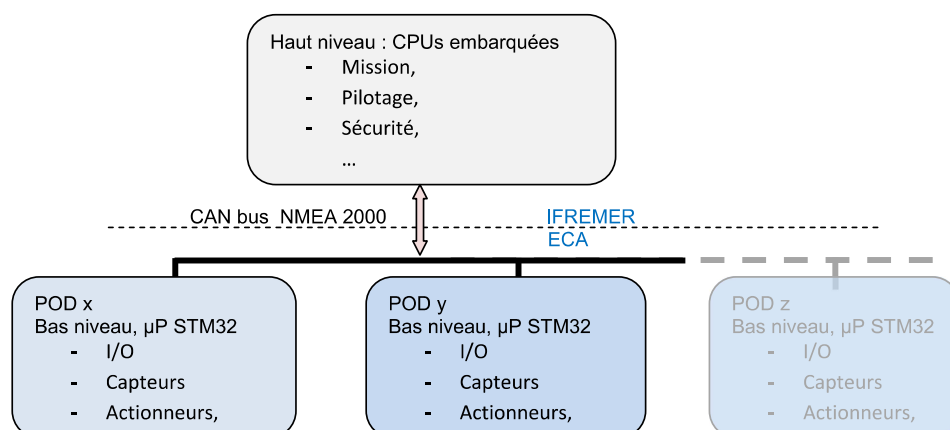


FIGURE 2.3 – Architecture générale d'un système à double niveau

La norme NMEA 2000, pour National Marine Electronics Association, est un protocole de communication basé sur le bus CAN, conçu pour les systèmes électroniques marins. Elle permet l'échange de données entre divers équipements embarqués (capteurs, moteurs, systèmes de navigation) via un réseau unique et robuste, capable de supporter les conditions exigeantes du milieu marin. Utilisant une vitesse de 250 kbps, NMEA 2000 transmet des messages structurés appelés PGNs (Parameter Group Numbers), assurant l'interopérabilité entre appareils de fabricants différents. Dans les véhicules sous-marins comme les AUV ou les ROV, NMEA 2000 est utilisé pour intégrer des systèmes de navigation, de surveillance moteur, ou des capteurs environnementaux.

Afin de respecter cette norme, les messages sont envoyés avec un identifiant étendu. La structure de la trame CAN dans ce cas précis est rappelée ci-dessous.



FIGURE 2.4 – Structure d'une trame CAN avec identifiant étendu

Cet identifiant étendu est structuré comme suit :

| Priorité     | Libre  | ID Message (PGN) | Adresse de la source |
|--------------|--------|------------------|----------------------|
| Bits 28 à 26 | Bit 25 | Bits 24 à 8      | Bits 7 à 0           |

TABLE 2.3 – Structure d'un identifiant étendu (29 bits) NMEA 2000

En ayant un identifiant construit de telle sorte, il est possible, pour chaque message, de

connaître rapidement la source et le type de message.

Concernant le moyen de communication, le protocole CAN est utilisé, plus précisément le CAN2.0B, qui permet d'utiliser des identifiants étendus. Ce protocole est dit *half-duplex*, ce qui signifie que chaque microcontrôleur ne peut pas recevoir et envoyer un message simultanément. Conçu pour les systèmes embarqués, notamment dans le domaine de l'automobile, ce protocole permet à plusieurs équipements d'échanger des messages sur un seul réseau physique. Chaque équipement connecté correspond à un nœud, mais un seul nœud peut émettre à la fois. L'émission des messages dépend de la priorité définie dans leur identifiant : le message ayant la priorité la plus élevée sera transmis en premier.

Compte tenu du mode de réception, interrogation cyclique (polling) via la pile logicielle NMEA utilisée pour la réception des trames, et des capacités du STM32 (FIFO de 3 PGN), et ce, valable pour chaque firmware, les émissions des messages CAN issus du haut niveau (Linux) doivent respecter un timing minimum entre chaque message PGN émis, soit 5 ms.

### Filtrage des messages dans les STM32

Bien que tous les microcontrôleurs parlent sur le même bus, ils n'ont pas nécessairement besoin de recevoir tous les messages qui passent. C'est pourquoi un principe de filtre est implémenté afin d'éviter que les microcontrôleurs se retrouvent surchargés.

Pour générer ces filtres, l'entreprise utilise un outil interne. L'outil CanFilterGenerator génère automatiquement les filtres à partir d'une base de données décrivant l'ensemble des PGN utilisés (issue de BusMaster V3.2.2) et d'une liste spécifiant les PGN à autoriser. Cette base de données sera extrêmement utile par la suite pour reproduire ce filtrage. L'outil permet d'émettre une table de PGN représentant l'ensemble des messages CAN circulant sur le bus. Une fois les PGN attendus renseignés, l'outil compare le comportement du filtre et signale :

1. Les PGN correctement autorisés ;
2. Ceux attendus mais absents ;
3. Ceux autorisés à tort.

Pour le POD PCC, l'outil donne les filtres suivants :

| idx | mask       | value      | coverage |
|-----|------------|------------|----------|
| 1   | 0xeaf0e608 | 0x8f000008 | 15       |
| 2   | 0xffffe208 | 0x18ff0008 | 15       |
| 3   | 0xfffffe08 | 0x18ff00c8 | 11       |
| 4   | 0xeaf06c3c | 0x8f0003c  | 10       |

| idx | mask       | value      | coverage |
|-----|------------|------------|----------|
| 5   | 0xffffc83c | 0x18ff003c | 9        |
| 6   | 0xf3ffc9c8 | 0x10ff01c8 | 6        |
| 7   | 0xfffffd08 | 0x18f0808  | 5        |
| 8   | 0xffffccc8 | 0x18ff04c8 | 5        |
| 9   | 0xeaf2f93c | 0x8f2083c  | 3        |
| 10  | 0xffffc1f  | 0x18ff0408 | 3        |
| 11  | 0xeaf2f9c8 | 0x8f208c8  | 3        |

TABLE 2.4 – Exemple de filtres générés par l'outil

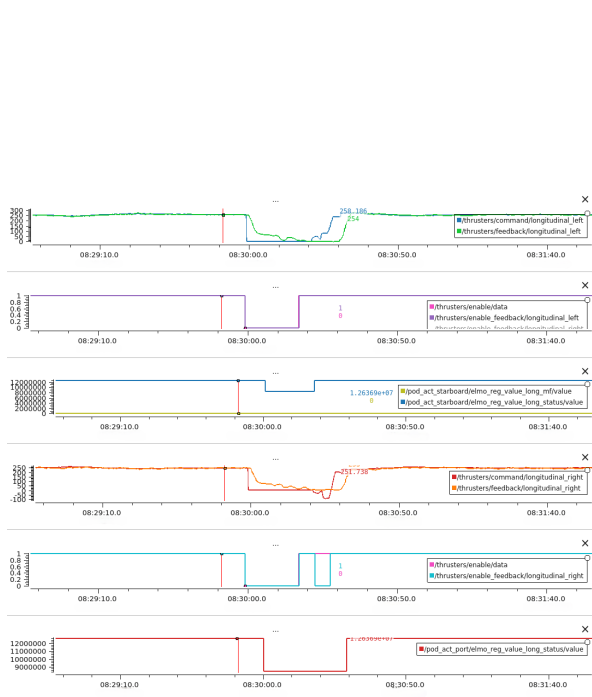
Le test des filtres est ensuite effectué avec un autre outil interne, assurant leur bonne définition.

### 2.2.3 Identification du problème de communication

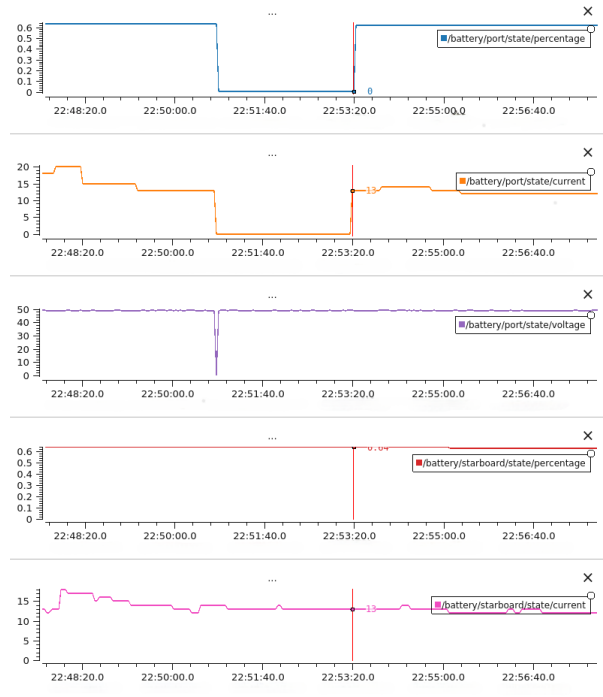
#### Mise en évidence des problèmes

Avant de s'intéresser à la réalisation de l'outil, il est important d'identifier correctement le problème, car la conception de ce dernier en a largement dépendu.

La problématique initiale indiquait que des messages étaient "perdus" et que certaines données n'étaient pas correctement reçues. Pour mettre en évidence ce problème, des essais ont été menés en amont de ce stage et deux problèmes majeurs ont été mis en évidence :



(a) Bug de perte de contrôle du moteur



(b) Bug de pertes des informations de la batterie (charge et courant)

FIGURE 2.5 – Mise en évidence des problèmes de communication avec des données récupérées depuis la partie haut niveau du système

Le premier problème, en Figure 2.5a, montre que le moteur Tribord reste désactivé alors que la commande d’activation a été envoyée.

Le second problème, en Figure 2.5b, montre que certaines informations de la batterie (la charge et le courant) sont perdues pendant 3 minutes, alors que d’autres informations, pourtant contenues dans le même message, ne sont pas perdues.

## Causes possibles

En ayant ces informations, il est possible de commencer à faire des suppositions sur l’origine du problème. Bien qu’une erreur d’implémentation des filtres ait pu être envisagée, l’utilisation de CanFilterGenerator permet de d’écarter cette hypothèse. Ainsi, deux pistes restent envisageables :

- Une surcharge du bus CAN qui entraînerait la perte de messages et la corruption de certains autres ;
- Une taille de FIFO trop faible dans les microcontrôleurs qui là encore pourrait causer une perte de données.

Concernant la première option, il est important de savoir que deux types de messages transitent dans le véhicule : des messages périodiques et des messages événementiels. En prenant en compte la période des messages, le nombre total de messages émis par seconde

sur le bus peut être déduit. Par exemple, un message ayant une période d'émission de 100 ms sera comptabilisé 10 fois pour le total. En procédant ainsi, en analysant les messages émis par chaque PODs, un total de 172 messages par seconde a été identifié. Sachant que, pour chacun d'entre eux, les 64 bits de données sont toujours utilisés, tous les messages ont une longueur totale de 126 bits. Ainsi :

$$172 \times 126 = 21672 \text{ bit/s} \rightarrow \approx 21 \text{ kbit/s}$$

La bande passante restante est calculée comme suit :

$$250\,000 \text{ bits/s} - 21\,672 \text{ bits/s} = 228\,328 \text{ bits/s}$$

Ensuite, pour déterminer combien de messages de 126 bits peuvent être envoyés par seconde avec cette bande passante restante :

$$\frac{228\,328 \text{ bits/s}}{126 \text{ bits/message}} \approx 1812.127$$

Donc, environ 1812 messages de 126 bits peuvent être envoyés en plus des messages périodiques existants. Au vu du nombre de messages transitant sur le bus, il est assez peu probable qu'il y ait 1812 messages événementiels qui transitent par seconde. Cependant, sans une analyse approfondie des logs, il n'est pas complètement possible de l'écarter.

Concernant la seconde option, la taille de la FIFO pourrait, elle aussi, entraîner une perte de messages. En effet, si un message est reçu alors qu'elle est pleine, il existe deux cas de figure :

- Rejet : Dans ce cas, le message n'est pas ajouté à la file et est définitivement perdu ;
- Écrasement : Le message nouvellement reçu va écraser le message le plus ancien dans la file, entraînant à nouveau une perte de données.

Bien que la manière dont la FIFO est implantée dans le microcontrôleur ne soit pas connue, les deux cas entraînent une perte de données. De plus, le temps de traitement des messages étant, lui aussi, inconnu, il est difficile d'évaluer si la file est surchargée ou non.

C'est à ce stade qu'intervient le sujet du stage. En se plaçant sur le bus CAN, tous les messages qui transitent seront reçus. De cette manière, il sera possible de connaître la charge du bus CAN à tout instant et donc de savoir si ce dernier est surchargé ou non. Pour ce qui est du second cas, les messages sont reçus indépendamment de la vitesse de traitement et de la taille de la FIFO, il sera donc possible de comparer les données des logs avec celles du haut niveau pour savoir rapidement où se situe le problème.

## 2.2.4 Fonctionnalités principales de l’outil

### Intégration logicielle des filtres et parsing des fichiers de logs

Compte tenu du problème, il est important que l’outil de visualisation des trames reflète le comportement réel de chacun des quatre PODs. Pour ce faire, il était important de reproduire le principe de filtrage actif dans les STM32 sur l’interface afin de voir uniquement les messages que chacun d’entre eux a pu recevoir.

Pour ce faire, la liste des messages échangés sur chaque POD a pu être récupérée sur le document de conception et a été transposée dans un fichier TOML appelé `A6K_pgns.toml`. L’avantage des fichiers TOML est qu’ils offrent une syntaxe simple et lisible, conçue pour être facile à comprendre et à modifier. Le langage TOML est particulièrement adapté pour les fichiers de configuration grâce à sa structure claire et sa capacité à représenter des données hiérarchiques de manière intuitive. Dans ce fichier, pour chaque POD, deux listes distinctes sont définies :

- `pgns_rx` : liste pour tous les pgns reçus ainsi que la source associée. Dans le cas de provenances multiples, un message est défini pour chaque source ;
- `pgns_tx` : liste des pgns émis par le POD concerné.

Ainsi dans le fichier de configuration, pour chaque POD, les trames sont définies de cette manière :

```
[VCC]
pgns_rx = [{pgn = 65280, sources = ["CPU", "PC"]}, ...]
pgns_tx = [{pgn = 65282, sources = ["VCC"] }, ...]
```

Différencier les trames émises des trames reçues permet de rapidement connaître le sens des transmissions et donc de mieux comprendre l’origine du problème, ce qui sera très utile par la suite.

Comme évoqué précédemment, la source apparaît dans l’identifiant de chaque message sous forme numérique. Cependant, afin de faciliter l’écriture du fichier de configuration, les sources y sont écrites en toutes lettres. Il est donc nécessaire d’indiquer dans un autre fichier, nommé `A6K_adresses.toml`, la valeur décimale associée à chaque mot (*VCC*, *PCC*, *ACT* ou *NRJ*) :

```
[Sources]
PCC = 30
VCC = 40
```

De cette manière, le programme possède toutes les informations permettant de trier les trames à travers les PODs, du moment que leur identifiant est connu (voir Tableau 2.3).

Le filtrage logiciel des trames suppose de les avoir préalablement récupérées. En effet, pour l’instant, ces dernières sont toutes stockées dans des fichiers de logs, mais ne sont

pas encore intégrées dans le programme. Afin de trier ces fichiers, il est nécessaire de définir une méthode de parsing qui va permettre de parcourir ces fichiers et d'en extraire les trames. Deux méthodes distinctes ont dû être définies pour lire et interpréter des fichiers de logs CAN provenant de deux types de loggers différents : IXXAT et CL2000.

La méthode `parse_ixxat_log_file` est utilisée pour analyser un fichier de log au format IXXAT. Elle commence par définir le chemin du fichier de log et imprime un message indiquant le début de l'analyse. Le fichier est ouvert et un échantillon de 1024 octets est lu pour détecter le format CSV.

| Bus                         | No | Time         | State | ID (hex) | DLC | Data (hex)              |
|-----------------------------|----|--------------|-------|----------|-----|-------------------------|
| USB-to-CAN V2 compact CAN-1 | 1  | 15:29:26.416 | E     | 18FF070B | 8   | 40 98 02 48 80 00 00 00 |
| USB-to-CAN V2 compact CAN-1 | 2  | 15:29:26.421 | E     | 18FF070B | 8   | 40 98 02 48 80 00 00 00 |
| USB-to-CAN V2 compact CAN-1 | 3  | 15:29:26.422 | E     | 14FF1A0B | 8   | 12 0B 0C 00 00 00 00 FF |
| USB-to-CAN V2 compact CAN-1 | 4  | 15:29:26.422 | E     | DF20D0B  | 8   | 0B 40 0D 00 40 FF FF FF |

FIGURE 2.6 – Exemple de la structure d'un fichier CSV généré par le logger IXXAT

Ce fichier est structuré, avec une ligne d'en-tête contenant les noms des colonnes (`Time`, `ID (hex)`, `Data Length`, `Data Bytes`, `Bus`, `State`). Ces champs sont accessibles par nom via le dictionnaire retourné par `csv.DictReader` qui est utilisé pour lire le fichier ligne par ligne. Chaque ligne est analysée par la méthode `_parse_ixxat_frame`, qui extrait les informations pertinentes telles que l'identifiant de trame, les données, le timestamp, la longueur, l'état et le bus. Les trames sont stockées dans `self.all_frames` et un compteur de trames est incrémenté. En cas d'erreur, un message est affiché.

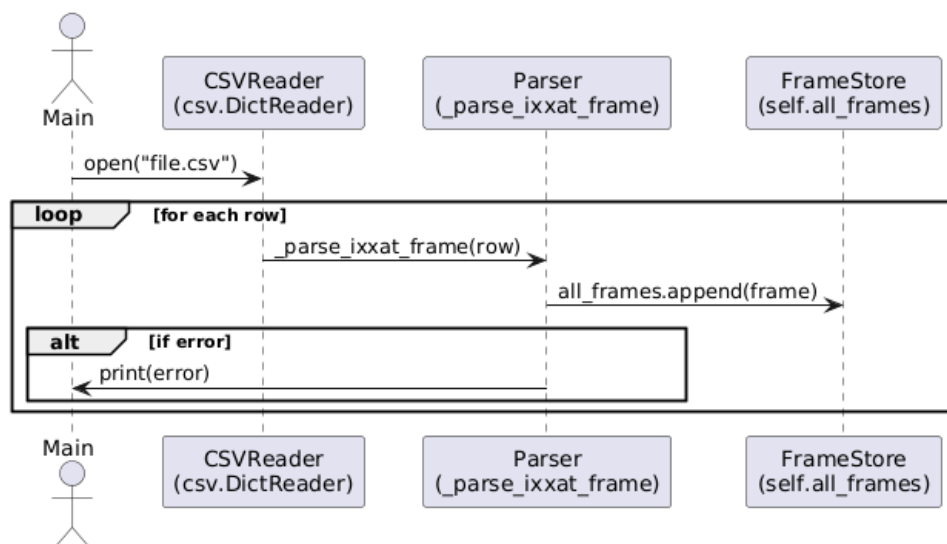


FIGURE 2.7 – Diagramme de séquence de la fonction de parsing du logiciel

La méthode `parse_cl2000_log_file` est destinée aux fichiers au format CL2000. Elle suit un schéma similaire : définition du chemin, affichage d'un message de début, puis

ouverture du fichier. Les fichiers CL2000 sont des fichiers texte brut où chaque ligne représente une trame, avec des champs séparés par des points-virgules (;).

```
Timestamp;Type;ID;Length;Data
20T083328273;1;14ff1a0b;8;030b0c00000000ff
20T083328318;1;18ff070b;8;4098024080000000
20T083328478;1;14ff1a0b;8;040b0a00000000ff
20T083328568;1;18ff070b;8;4098014009000000
```

FIGURE 2.8 – Exemple de la structure d'un fichier TXT généré par le logger CL2000

Les commentaires (lignes commençant par #) sont ignorés. Les données sont extraites par index après un `split(";")`. Les lignes valides sont analysées une à une par la méthode `_parse_cl2000_line`, qui en extrait les informations essentielles : identifiant, données, timestamp, longueur. Les trames sont enregistrées dans `self.all_frames` et un compteur est mis à jour. Les erreurs rencontrées sont également signalées.

| Information extraite      | Méthode<br><code>parse_ixxat_log_file</code>                                   | Méthode<br><code>parse_cl2000_log_file</code>                        |
|---------------------------|--------------------------------------------------------------------------------|----------------------------------------------------------------------|
| Format du fichier         | CSV avec en-tête, colonnes accessibles par nom via <code>csv.DictReader</code> | Texte brut, lignes séparées par ;                                    |
| Identifiant de trame (ID) | Champ "ID (hex)" du CSV                                                        | Champ à une position fixe après <code>split(";")</code>              |
| Données de la trame       | Champ "Data Bytes", converti en liste d'octets                                 | Champ contenant les données en hexadécimal, séparées par des espaces |
| Timestamp                 | Champ "Time" en secondes ou millisecondes                                      | Champ à une position fixe, converti en format standard               |
| Longueur des données      | Champ "Data Length"                                                            | Déduite du nombre d'octets extraits ou d'un champ spécifique         |
| Bus                       | Champ "Bus"                                                                    | Non présent                                                          |
| État (State)              | Champ "State" (erreurs, type de trame)                                         | Non présent                                                          |

TABLE 2.5 – Informations extraites par les méthodes de parsing selon le type de fichier

## Tri des trames

Une fois les trames récupérées depuis le fichier de log (IXXAT ou CL2000), elles sont stockées dans une liste centrale. L'étape suivante consiste à les redistribuer vers les

interfaces pertinentes, en fonction de leur origine et de leur type. Pour ce faire, pour chaque interface définie dans la configuration, la paire (**Source**, **PGN**) de chaque trame est comparée avec les entrées correspondantes dans le fichier **A6K\_pgns**. Si une correspondance est trouvée, une copie de la trame est ajoutée à une liste spécifique à cette interface. Ainsi, à l'issue de cette étape, chaque interface dispose d'un ensemble de trames qui lui sont associées, c'est-à-dire qui ont transité par le microcontrôleur à travers cette interface. Si un message n'est associé à aucune interface, il est inséré dans une liste **Unknown** avec tous les messages n'ayant trouvé aucune correspondance.

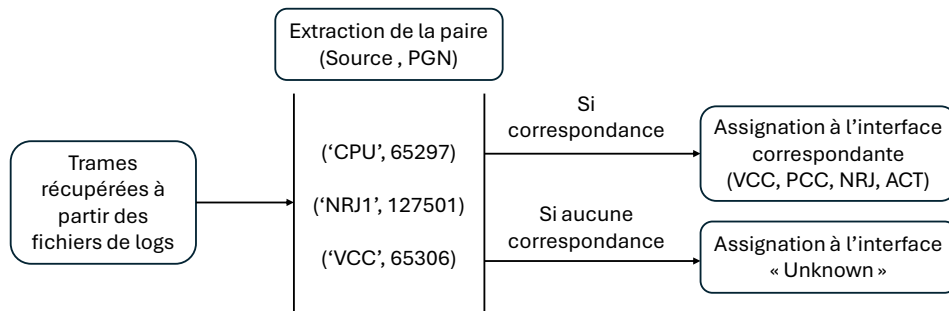


FIGURE 2.9 – Schéma explicatif de la redistribution des messages

Cependant, l'appartenance d'une trame à une interface ne suffit pas : il faut encore déterminer son sens relatif, c'est-à-dire s'il s'agit d'une trame reçue (**Rx**), transmise (**Tx**) ou les deux (**Rx/Tx**). Pour cela, chaque trame est comparée aux listes **pgns\_rx** et **pgns\_tx** de l'interface considérée. Si la paire (**Source**, **PGN**) figure dans la liste **pgns\_rx**, la direction est marquée **Rx**. Si elle figure dans **pgns\_tx**, elle est marquée **Tx**. Dans le cas où la trame est présente dans les deux listes, ce qui peut arriver pour des modules communicants en double (par exemple **NRJ1/NRJ2** ou **ACT1/ACT2**), la direction est notée **Rx/Tx**. Cela permet une annotation fine du rôle de chaque trame au sein de l'architecture multipoint.

La possibilité qu'une trame soit associée à plusieurs interfaces a posé un problème de programmation lors de l'assignation de la direction. Initialement, sans duplication explicite des objets, plusieurs interfaces partageaient en mémoire la même instance de trame. Cela signifiait que la modification de la direction dans une interface affectait toutes les autres, un comportement indésirable qui n'était pas immédiatement évident.

Ce n'est qu'en approfondissant le problème qu'il est apparu nécessaire de trouver une solution plus robuste pour garantir l'indépendance des instances de trame. Pour résoudre ce problème, l'utilisation de la fonction **deepcopy** s'est avérée nécessaire. Cette fonction permet de créer une copie complète et indépendante de l'objet original, assurant ainsi que chaque interface dispose de sa propre instance de trame avec un identifiant mémoire distinct. Ainsi, les modifications apportées à une trame dans une interface n'affectent pas

les autres, résolvant efficacement le problème de partage involontaire de données.

Ce traitement utilise une logique de correspondance hiérarchique : une même valeur de PGN peut être liée à plusieurs sources dont les identifiants partagent un préfixe commun suivi d'un suffixe numérique. Le code valide les correspondances en considérant ce préfixe comme identifiant principal. Par exemple, les sources ACT1 et ACT2 sont regroupées sous le préfixe ACT, ce qui permet d'associer les trames de PGN à toutes les interfaces correspondantes tant que le suffixe est un numéro.

## Décodage et export

Chaque interface ayant désormais une liste de trames associées, l'étape suivante consiste à les décoder. Pour cela, un nouveau fichier de configuration est ajouté : `A6K_database.txt`. Pour chaque message, ce fichier décrit les signaux qu'il contient en fonction de sa source. La définition des signaux y est structurée de la manière suivante :

```
[START_MSG] ACT1_PGN_EnvironmentalParameters2,368903957
[START_SIGNALS] SID,8,1,0
[START_SIGNALS] TemperatureInstance,6,2,0
[START_SIGNALS] HumidityInstance,2,2,6
[START_SIGNALS] Temperature,16,3,0
[START_SIGNALS] Humidity,16,5,0
[START_SIGNALS] AtmosphericPressure,16,7,0
[END_MSG]
```

Chaque bloc correspond à un message et commence par une balise `[START_MSG]` suivie de deux éléments :

- Le nom symbolique du message, ici `ACT1_PGN_EnvironmentalParameters2`, qui permettra par la suite d'associer un nom aux trames décodées pour trouver plus facilement les informations recherchées ;
- L'identifiant associé à ce message (voir Tableau 2.3) qui permet de rapidement trouver le message associé à l'identifiant récupéré dans les logs.

À l'intérieur de ce bloc figurent plusieurs lignes `[START_SIGNALS]`, chacune définissant une donnée de la trame. La syntaxe de chaque ligne est la suivante :

```
[START_SIGNALS] <Nom_du_signal>,<Longueur>,<Octet_de_départ>,<Bit_de_départ>
```

Le paramètre `<Bit_de_départ>` correspond au bit de départ au sein de l'octet, plusieurs signaux étant parfois imbriqués dans le même octet.

Afin de décoder les trames, une classe Python dédiée, `DatabaseDecoder`, charge puis parse le fichier `A6K_database.txt` décrit ci-dessus pour construire une base de données interne des messages définis.

Les trames associées à l'interface sélectionnée sont ensuite transmises une par une à la méthode `decode_frame()`. L'identifiant brut est utilisé pour récupérer la description du message et les données contenues dans la trame sont converties en octets, puis chaque signal est extrait bit à bit selon les paramètres définis (longueur, octet de départ, offset). L'extraction s'appuie sur l'application de masques binaires pour isoler les bits d'intérêt et sur des opérations de décalage à droite (shift) afin d'aligner correctement la valeur brute. Les valeurs sont ensuite reconstituées en nombres décimaux via un parcours des octets concernés, en tenant compte des éventuels chevauchements de bits sur plusieurs octets.

Une fois toutes les trames décodées, elles peuvent être exportées vers deux formats :

- Excel ;
- HTML.

L'export vers le fichier Excel s'effectue en utilisant la bibliothèque **pandas**, qui permet de structurer les données sous forme de `DataFrame`. Chaque trame décodée est représentée comme une ligne contenant les champs suivants : identifiant, timestamp, nom du message et valeurs des signaux extraits. L'ensemble est ensuite écrit dans un fichier `.xlsx` à l'aide de **ExcelWriter**, avec une feuille principale regroupant toutes les trames, ainsi que des feuilles supplémentaires classées par type de message pour faciliter l'analyse.

Le format HTML repose sur une conversion similaire, générant une table interactive via la méthode `to_html()` de **pandas**, pouvant être visualisée directement dans un navigateur. Ce mode d'export est principalement utilisé pour une consultation rapide ou pour intégrer les résultats dans une interface web.

L'export HTML repose sur une classe `HTMLExporter` conçue pour structurer les trames CAN décodées dans un format interactif destiné à une consultation humaine via navigateur. Lors de la génération via `generate_html()`, les données des trames sont sérialisées en JSON et injectées dans une page HTML contenant des composants dynamiques.

Chaque trame est affichée sous forme d'un conteneur extensible, contenant un en-tête avec les métadonnées essentielles (numéro, timestamp, nom du message, source, direction, DLC, données brutes) et une section de détails affichant les signaux décodés sous forme de tableau. Les trames peuvent être filtrées dynamiquement par PGN via des cases à cocher générées automatiquement.

L'interface est divisée en deux onglets :

- Frames View : consultation brute des trames décodées ;
- Visualization : génération de graphiques via Plotly à partir de signaux sélectionnés.

Le HTML est autonome et contient :

- du CSS embarqué pour la présentation et l'ergonomie (onglets, filtres, animations) ;

- du JavaScript embarqué (sans dépendance externe autre que Plotly) pour l'interactivité;
- l'intégration directe des données via `json.dumps(frame_data)` pour une visualisation client-side sans appel serveur.

L'intérêt de l'onglet **Visualization** est de voir l'évolution d'un ou plusieurs signaux en fonction du temps. Pour générer un graphique, l'utilisateur commence par choisir le message concerné puis sélectionne le signal à tracer. De cette manière, il est possible de comparer l'évolution d'un signal de commande avec le signal censé être impacté par la commande, afin de s'assurer que la commande est bien prise en compte notamment.

## 2.2.5 Utilisation de l'outil

Concernant l'utilisation de l'outil, une interface graphique a été réalisée. Comme mentionné précédemment, toute la partie graphique a été réalisée avec la librairie **CustomTkinter**. Cette interface est composée de quatre fenêtres principales. Les spécificités de chacune d'entre elles sont détaillées dans les parties ci-dessous.

### Sélection de la configuration

En lançant l'application, l'utilisateur arrive sur la première page qui permet de configurer l'outil pour s'adapter au contexte.

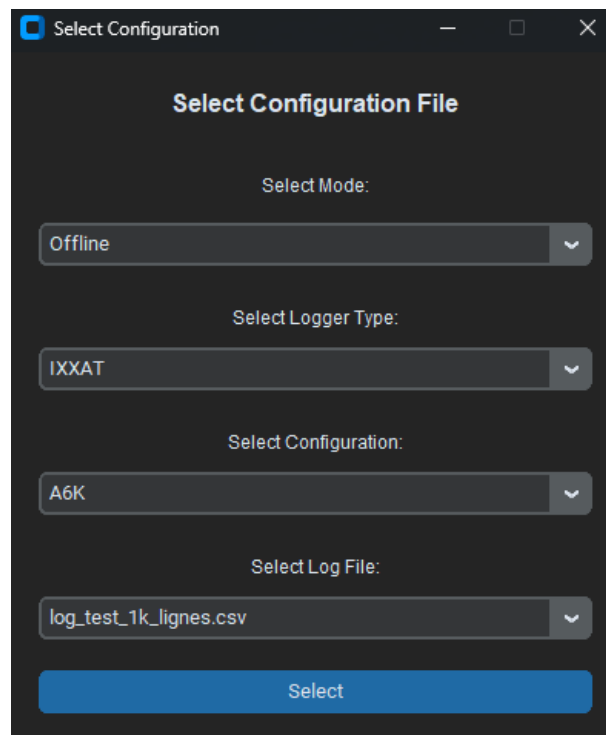


FIGURE 2.10 – Page de sélection de la configuration de l'interface

Sur cette première page, l'utilisateur doit sélectionner 4 paramètres :

- Le mode d'utilisation ;
- Le logger utilisé ;
- La configuration ;
- Le fichier de logs.

Concernant le mode d'utilisation, seul un mode **Offline** a été implémenté, il n'y donc pour le moment pas de choix à faire. L'autre mode qui aurait pu être ajouté aurait été un mode temps réel qui, en utilisant la sonde IXXAT ou une autre sonde permettant de s'interfacer sur un bus CAN, aurait permis de récupérer des logs en temps réel. Cependant, pour les besoins de ce stage, son développement n'était pas nécessaire, car le travail se base essentiellement sur des fichiers de logs. Son ajout sera considéré plus tard au paragraphe 3.2.1.

Ensuite, l'utilisateur doit sélectionner le logger utilisé. Cette étape est cruciale puisque, comme vu au paragraphe 2.2.4, la méthode de parsing est dépendante de l'outil utilisé.

Une fois le mode d'utilisation et l'outil sélectionnés, il est nécessaire de définir la configuration. Il a été évoqué plus tôt que pour fonctionner correctement, le logiciel a besoin de trois fichiers de configuration :

- `A6K_adresses.toml` ;
- `A6K_database.txt` ;
- `A6K_pgns.toml`.

Bien que ce ne soit pas un des attendus du stage, il a été décidé de ranger ces trois fichiers dans un sous-dossier du dossier `configs`.

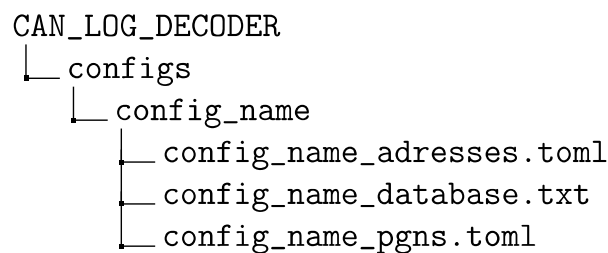


FIGURE 2.11 – Structure du dossier `configs`

En procédant ainsi, si tant est que ces fichiers aient été réalisés, l'outil de décodage de trame devient utilisable pour tous les véhicules de l'entreprise, améliorant grandement ses perspectives d'utilisation. D'un point de vue logiciel, le contenu du dossier `configs` est scanné pour permettre à l'utilisateur de choisir parmi les configurations définies en amont.

Pour finir, il reste à sélectionner le fichier de logs à utiliser lors de la visualisation et du décodage des trames. Là encore, c'est le logiciel qui scanne un dossier, le dossier `logs`, et propose à l'utilisateur de choisir parmi les fichiers présents. Cependant, pour éviter que

des fichiers de logs dans un format non reconnu par le logiciel soient sélectionnés, seuls les fichiers avec les extensions suivantes sont affichés : `.csv` et `.TXT/.txt`.

## Sélection de l'interface

Une fois la configuration du logiciel définie, si le chargement de tous les fichiers s'est passé correctement, l'utilisateur arrive sur une nouvelle page.

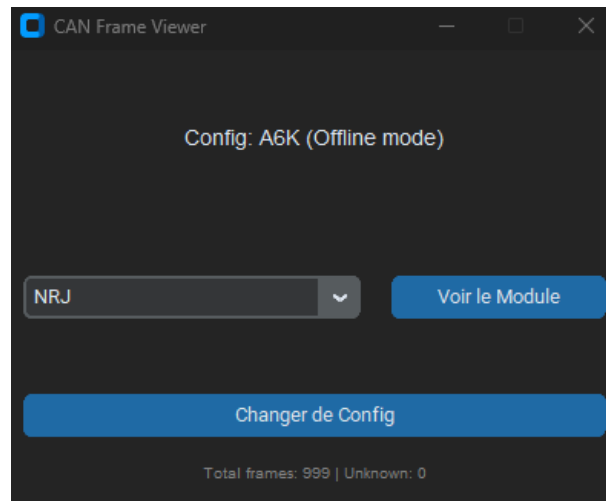


FIGURE 2.12 – Page de sélection de l'interface à visualiser

Ici, l'utilisateur peut choisir le POD à visualiser pour connaître les trames qui ont, ou qui auraient dû, transiter par ce dernier. Toujours dans un souci de portabilité du logiciel, le nombre et le nom des interfaces est dépendant du fichier `config_name_pgns.toml` dans lequel elles sont définies. Cette page permet également de connaître le rapport de parsing du fichier. Sous le bouton permettant de revenir à la page précédente (bouton "Changer de config") apparaissent deux nombres :

- Total frames : indique le nombre de trames ayant été trouvées dans le fichier de logs ;
- Unknown : indique le nombre de trames qui n'ont trouvé aucune correspondance (Source + PGN) dans tout le fichier `config_pgns`.

Le but de ce projet étant d'avoir uniquement un exécutable au format `.exe` l'apparition de ce rapport permet de pallier l'absence de console de debug.

## Visualisation des trames brutes et export

Une fois l'interface à visualiser sélectionnée, la page suivante permet de visualiser les trames **non décodées**.

CAN Frame Viewer

NRJ - Trames CAN (999 / 999) Passer à la Visualisation

| No | Timestamp    | ID (hex) | PGN    | Source | Direction | DLC | Data (hex)         |
|----|--------------|----------|--------|--------|-----------|-----|--------------------|
| 1  | 15:29:26.416 | 18FF070B | 65287  | NRJ1   | Rx/Tx     | 8   | 0x4098024880000000 |
| 2  | 15:29:26.421 | 18FF070B | 65287  | NRJ1   | Rx/Tx     | 8   | 0x4098024880000000 |
| 3  | 15:29:26.422 | 14FF1A0B | 65306  | NRJ1   | Tx        | 8   | 0x120B0C00000000FF |
| 4  | 15:29:26.422 | DF20D0B  | 127501 | NRJ1   | Tx        | 8   | 0x0B400D0040FFFFFF |
| 5  | 15:29:26.426 | 19F2120B | 127506 | NRJ1   | Rx/Tx     | 8   | 0x7E0B000000000000 |
| 6  | 15:29:26.427 | 19F2140B | 127508 | NRJ1   | Tx        | 8   | 0x0B00000000B26A3D |
| 7  | 15:29:26.427 | 15FD070B | 130311 | NRJ1   | Tx        | 8   | 0x4002F8759D030000 |
| 8  | 15:29:26.431 | 18FF2B0B | 65323  | NRJ1   | Tx        | 8   | 0x40980B0000FFFFFF |
| 9  | 15:29:26.626 | 14FF1A0B | 65306  | NRJ1   | Tx        | 8   | 0x130B0A00000000FF |
| 10 | 15:29:26.671 | 18FF070B | 65287  | NRJ1   | Rx/Tx     | 8   | 0x4098014809000001 |
| 11 | 15:29:26.831 | 14FF1A0B | 65306  | NRJ1   | Tx        | 8   | 0x140B0B00000000FF |
| 12 | 15:29:26.921 | 18FF070B | 65287  | NRJ1   | Rx/Tx     | 8   | 0x4098024880000000 |
| 13 | 15:29:26.936 | 18FF2B0B | 65323  | NRJ1   | Tx        | 8   | 0x40980B0000FFFFFF |
| 14 | 15:29:26.941 | DF20D0B  | 127501 | NRJ1   | Tx        | 8   | 0x0B400D0040FFFFFF |
| 15 | 15:29:27.036 | 14FF1A0B | 65306  | NRJ1   | Tx        | 8   | 0x150B0C00000000FF |
| 16 | 15:29:27.171 | 18FF070B | 65287  | NRJ1   | Rx/Tx     | 8   | 0x4098014809000001 |
| 17 | 15:29:27.241 | 14FF1A0B | 65306  | NRJ1   | Tx        | 8   | 0x160B0A00000000FF |
| 18 | 15:29:27.421 | 18FF070B | 65287  | NRJ1   | Rx/Tx     | 8   | 0x4098024880000000 |
| 19 | 15:29:27.426 | 15FD070B | 130311 | NRJ1   | Tx        | 8   | 0x4102F8759D030000 |
| 20 | 15:29:27.431 | 19F2140B | 127508 | NRJ1   | Tx        | 8   | 0x0B00000000B26A3E |
| 21 | 15:29:27.441 | 18FF2B0B | 65323  | NRJ1   | Tx        | 8   | 0x40980B0000FFFFFF |
| 22 | 15:29:27.446 | 14FF1A0B | 65306  | NRJ1   | Tx        | 8   | 0x170B0B00000000FF |

☒ 65287 (NRJ1)  
☒ 65306 (NRJ1)  
☒ 65323 (NRJ1)  
☒ 127501 (NRJ1)  
☒ 127506 (NRJ1)  
☒ 127508 (NRJ1)  
☒ 130311 (NRJ1)

Tout sélectionner Tout désélectionner  
Basculer Rx Basculer Tx

Exporter vers Excel Exporter vers HTML Retour à l'interface principale

FIGURE 2.13 – Page de visualisation des trames brutes

Ici plusieurs fonctionnalités ont été intégrées. Dans un premier temps, toute la partie gauche de l'interface comprend un tableau dans lequel toutes les trames assignées à l'interface apparaissent. Toutes les données utilisables sans prétraitement sont affichées (numéro de trame, timestamp, id, DLC, data) auxquels sont ajoutées des données extraites de l'identifiant de la trame comme le numéro de PGN ou le nom de la source. Dans ce tableau apparaît également la direction de la trame par rapport à l'interface choisie qui est déduite de ces deux informations.

Sur la droite de l'interface se trouve un menu déroulant permettant à l'utilisateur de filtrer les trames de ce tableau. Toutes les checkboxes sont générées en fonction des PGN présents dans les fichiers de logs. Ainsi, si un PGN n'a jamais été reçu, aucune checkbox ne sera créée. Il est également possible de filtrer les trames du tableau non pas par PGN mais par direction avec les boutons "Toggle Rx" et "Toggle Tx".

Comme évoqué précédemment lors de la présentation du véhicule, une certaine partie des trames reçues et envoyées sont périodiques. Cette caractéristique permet de connaître le nombre de trames exactes supposées transiter dans un intervalle de temps. C'est pourquoi, à côté du titre de la page, apparaît un compteur indiquant combien de trames sont actuellement visibles dans le tableau. De cette manière, en utilisant les options de filtrage pour sélectionner seulement un seul PGN associé à une trame périodique, il est possible de comparer rapidement la valeur pratique à la valeur théorique pour identifier un éventuel problème d'émission de trame au sein du véhicule en cas de différence trop importante.

La dernière fonctionnalité intégrée sur cette page est l'export et le décodage des données évoqué au paragraphe 2.2.4. Un bouton a été créé pour chaque format d'export (Excel ou HTML). Les fichiers associés sont tous créés dans un dossier nommé `decoded_logs`, construit comme ceci :

```

CAN_LOG_DECODER
├── decoded_logs
│   ├── config_name
│   │   ├── logs_file_name
│   │   │   ├── interface_name
│   │   │   │   ├── interface_name_decoded.html
│   │   │   │   └── interface_name_decoded.xlsx

```

FIGURE 2.14 – Structure du dossier `decoded_logs`

Tous les sous-dossiers sont créés automatiquement, ce qui permet un tri efficace des fichiers générés. Cela permet également à l'utilisateur de décoder les trames de plusieurs interfaces issues de différents fichiers de logs, sans craindre que les autres fichiers ne soient écrasés.

Enfin, à l'aide de deux boutons ("Passer à la visualisation" et "Retour à l'interface principale"), l'utilisateur peut soit revenir à la sélection de l'interface à observer, soit basculer vers la dernière fenêtre de l'application.

## Visualisation du rapport de distribution

Sur cette dernière fenêtre, un rapport de distribution est dessiné à l'aide de `matplotlib`.

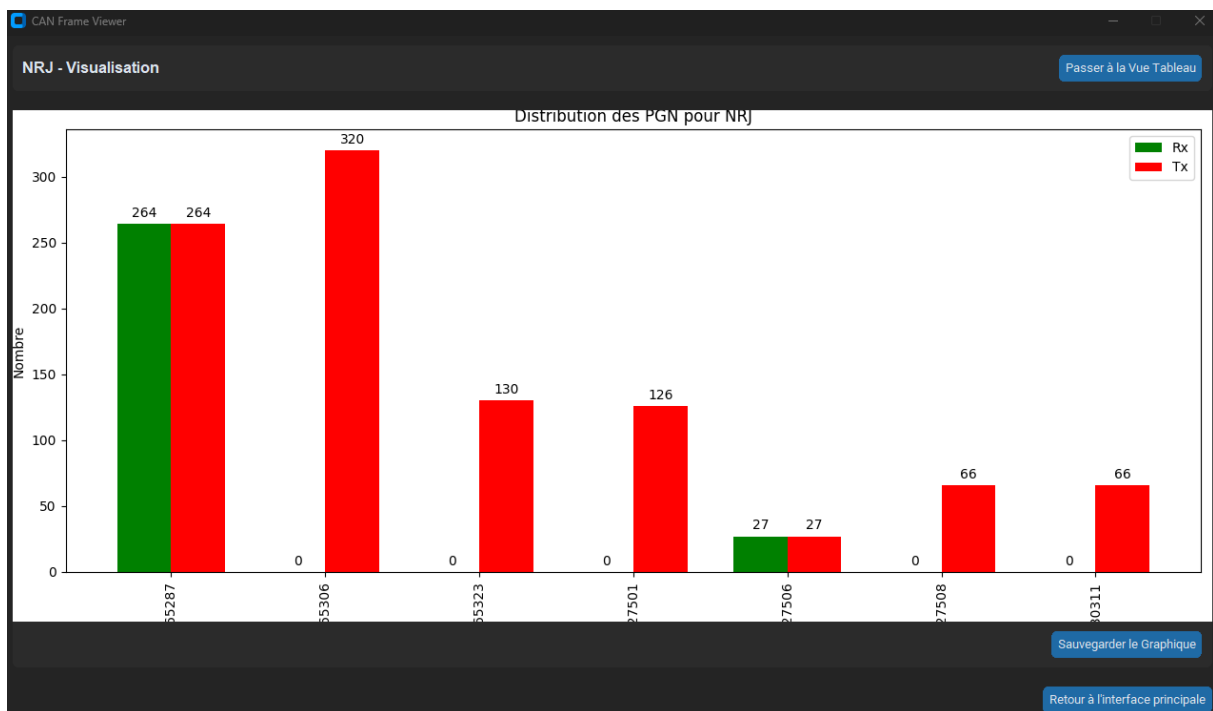


FIGURE 2.15 – Page de visualisation du rapport de distribution des trames

Ce diagramme permet de connaître le nombre de trames émises (en rouge) et reçues (en vert) pour chaque PGN, indépendamment de la source et ainsi d'identifier des différences majeures entre les deux directions. Cette visualisation est particulièrement utile pour identifier les problèmes sur certaines trames qui sont émises uniquement après réception d'une trame de commande. Dans ce cas précis, les deux colonnes associées à chaque direction sont supposées être égales. Dans le cas contraire, cela peut signifier deux choses :

- Soit la partie haut niveau envoie trop fréquemment la trame d'instruction, ne respectant pas les consignes d'envoi et entraînant par la même occasion une surcharge sur le bus CAN. Ce cas s'illustrerait par un nombre de trames reçues grandement supérieur ;
- Soit la différence entre les deux colonnes est moindre, ce qui indiquerait que, même sans que la consigne ait été envoyée de manière répétitive, le microcontrôleur était surchargé et n'a pas pu répondre.

Un bouton a également été ajouté pour sauvegarder rapidement l'image dans le même dossier que les fichiers `.xlsx` et `.html`. Le format de l'image sauvegardé n'est pas le même que dans l'interface pour permettre une meilleure visualisation. En effet, en Figure 2.15, le titre du graphique ainsi que les PGNs sont coupés, ce qui n'est pas idéal, surtout si l'image est utilisée dans un document par la suite. Aucune solution logicielle n'a été trouvée pour en modifier le format directement dans l'application afin d'assurer une visualisation optimale.

## 2.2.6 Phase de test

Maintenant que l'outil a été construit, il est nécessaire de le tester pour valider son fonctionnement. Pour ce faire, il est nécessaire d'avoir un fichier de log correspondant à un des deux formats acceptés pour l'outil. Dans l'idéal, ce fichier aurait dû être envoyé par l'IFREMER afin d'avoir des logs illustrant un des deux problèmes définis dans le paragraphe 2.2.3. Ces logs ont bien été reçus, mais les données à l'intérieur étaient corrompues, empêchant leur utilisation.

Afin de récupérer des logs et de ne pas avoir à les construire à la main, il a été nécessaire de trouver les cartes électroniques associées à un des PODs pour obtenir des logs corrects. La PFI (Plateforme d'Intégration) pour le projet A6K n'étant pas encore disponible dans l'entreprise au moment du stage, la récupération des logs ne sera possible que sur une seule carte à la fois, limitant grandement la quantité et la diversité des messages à recevoir. Cependant, cette phase de test étant uniquement destinée à valider le fonctionnement de l'outil, ce manque de diversité n'est pas réellement impactant.

Au vu des problèmes remontés par l'IFREMER, les cartes à trouver en priorité étaient les cartes associées au POD NRJ (problème sur la batterie) et au POD ACT (problème

sur l'activation du moteur).

L'ensemble de cartes qui a été trouvé est l'ensemble correspondant au POD NRJ. En effet, les cartes fonctionnent par paire avec, dans ce cas précis :

- Une carte PIE GESTION NRJ ;
- Une carte PIE DISTRIBUTION NRJ.

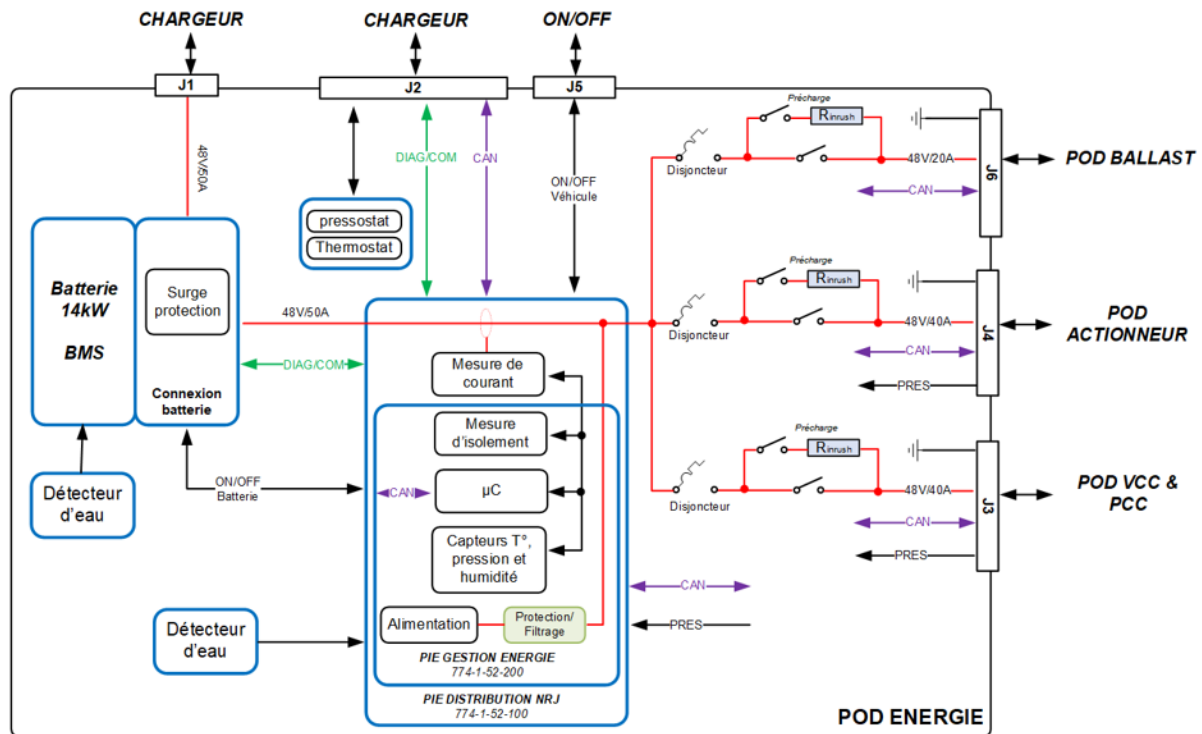


FIGURE 2.16 – Synoptique d'un POD NRJ

Comme le montre le synoptique du POD NRJ (Figure 2.16), la carte PIE GESTION ENERGIE est chargée de superviser l'ensemble du système électrique : elle mesure le courant, surveille l'isolement, collecte les données environnementales (température, pression, humidité), et assure la communication via le bus CAN. Elle agit comme une unité intelligente centralisant les données critiques. La carte PIE DISTRIBUTION ENERGIE, quant à elle, est responsable de la distribution de la puissance vers les différents sous-systèmes (PODs BALLAST, ACT, VCC et PCC) à travers des disjoncteurs protégés par des résistances de précharge. Elle joue donc un rôle clé dans la sécurisation et le découplage des charges.

## Test de cartes électroniques

Ces cartes ayant été trouvées sous forme de prototype, qui de fait n'ont pas été utilisées depuis un certain temps, il était nécessaire de les tester pour valider leur fonctionnement. Pour ce faire, à chaque carte est associé un document appelé NCR (Notice de Contrôles

et Réglages) qui liste tous les tests devant être effectués pour valider le fonctionnement de la carte.

Ainsi, ce document a été déroulé pour les deux cartes afin d'assurer leur intégrité et d'être sûr qu'il n'y ait aucun problème lors de leur utilisation. Tous les tests ayant été validés, ces cartes peuvent désormais être utilisées.

## Récupération de logs

Le fonctionnement des deux cartes ayant été validé, les premiers logs peuvent être récupérés. Pour ce faire, le montage suivant a été réalisé.

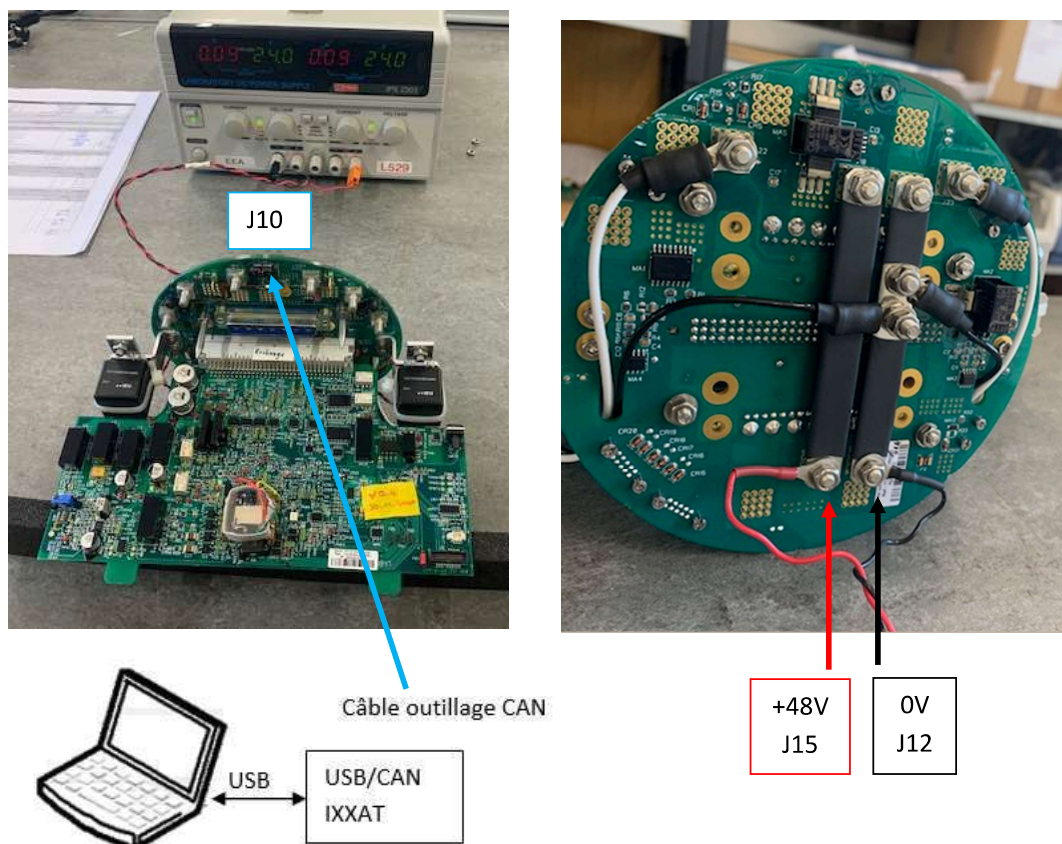
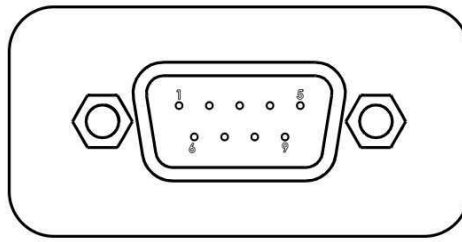


FIGURE 2.17 – Montage à réaliser pour la récupération de logs

Le logger utilisé pour cette manipulation était la sonde IXXAT USB-to-CAN V2. Le câble d'outillage n'étant pas présent avec la carte, il a fallu le réaliser. Il permet de relier le connecteur 2x2 de la carte au connecteur DB9 de la sonde pour s'interfacer sur le bus CAN. Le brochage du connecteur DB9 est le suivant :



| Pin    | Description   |
|--------|---------------|
| 2      | CAN L         |
| 3      | GND CAN       |
| 7      | CAN H         |
| Autres | Non connectés |

FIGURE 2.18 – Brochage du connecteur DB9 de la sonde IXXAT

## Test de l'application

Une fois le câble réalisé, le montage de la Figure 2.17 peut être mis en place. En connectant la sonde à un ordinateur en utilisant le logiciel **canAnalyser3 Mini**, un fichier de logs est créé au format .csv. En appliquant la procédure décrite dans le paragraphe 2.2.5 en utilisant le fichier de logs tout juste créé, le fichier HTML et le fichier Excel sont générés.

CAN Frames - NRJ

Frames View

Visualization

Expand All

Collapse All

Filters

☒ PGN 65287
 ☒ PGN 65306
 ☒ PGN 127501
 ☒ PGN 127506
 ☒ PGN 127508
 ☒ PGN 130311
 ☒ PGN 65323

No: 1 | Timestamp: 15:29:26.416 | Message: NRJ1\_PGN\_\_ContinuousBuiltInTest | Source: NRJ1 | Direction: Rx/Tx | DLC: 8 | Data: 0x4098024880000000

No: 2 | Timestamp: 15:29:26.421 | Message: NRJ1\_PGN\_\_ContinuousBuiltInTest | Source: NRJ1 | Direction: Rx/Tx | DLC: 8 | Data: 0x4098024880000000

Decoded Message: NRJ1\_PGN\_\_ContinuousBuiltInTest

| Signal           | Value      |
|------------------|------------|
| ManufacturerCode | 512        |
| Reserved1        | 3          |
| IndustryGroup    | 4          |
| CBitType         | 2          |
| Temperature      | 72         |
| CBitValue        | 2147483648 |

FIGURE 2.19 – Visualisation des trames décodées dans le fichier HTML

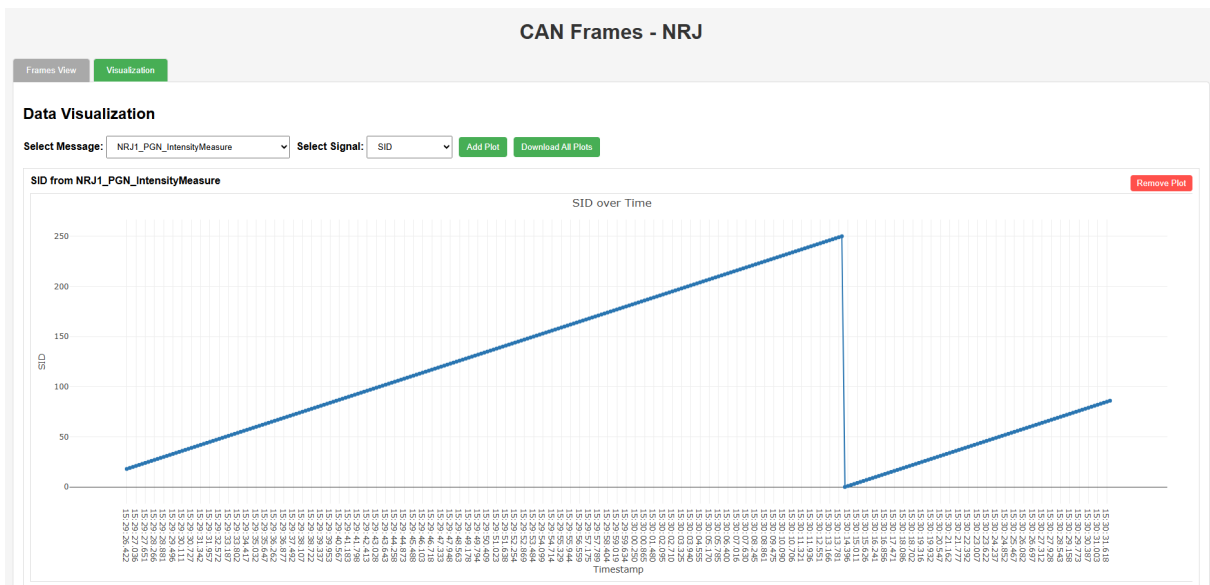


FIGURE 2.20 – Visualisation des graphiques dans le fichier HTML

|    | A                              | B        | C     | D      | E            | F         | G                    | H         | I     | J                | K         | L             | M        | N           | O          | P | Q | R | S | T | U |
|----|--------------------------------|----------|-------|--------|--------------|-----------|----------------------|-----------|-------|------------------|-----------|---------------|----------|-------------|------------|---|---|---|---|---|---|
|    | Message                        | ID       | PGN   | Source | Timestamp    | Direction | RawData              | DatLength | Frame | ManufacturerCode | Reserved1 | IndustryGroup | CBITType | Temperature | CBIRValue  |   |   |   |   |   |   |
| 1  | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:26.416 | Rx/Tx     | 0x4098024880000000   | 8         | 1     | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 2  | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:26.421 | Rx/Tx     | 0x4098024880000000   | 8         | 2     | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 3  | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:26.671 | Rx/Tx     | 0x409802488090000001 | 8         | 10    | 512              | 3         | 4             | 1        | 72          | 150994945  |   |   |   |   |   |   |
| 4  | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:26.921 | Rx/Tx     | 0x4098024880000000   | 8         | 12    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 5  | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:27.171 | Rx/Tx     | 0x409802488090000001 | 8         | 16    | 512              | 3         | 4             | 1        | 72          | 150994945  |   |   |   |   |   |   |
| 6  | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:27.421 | Rx/Tx     | 0x4098024880000000   | 8         | 18    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 7  | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:27.671 | Rx/Tx     | 0x409802488090000000 | 8         | 25    | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 8  | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:27.921 | Rx/Tx     | 0x4098024880000000   | 8         | 27    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 9  | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:28.171 | Rx/Tx     | 0x409802488090000000 | 8         | 31    | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 10 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:28.421 | Rx/Tx     | 0x4098024880000000   | 8         | 33    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 11 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:28.671 | Rx/Tx     | 0x409802488090000000 | 8         | 39    | 512              | 3         | 4             | 1        | 72          | 150994945  |   |   |   |   |   |   |
| 12 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:28.921 | Rx/Tx     | 0x4098024880000000   | 8         | 42    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 13 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:29.171 | Rx/Tx     | 0x409802488090000000 | 8         | 47    | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 14 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:29.421 | Rx/Tx     | 0x4098024880000000   | 8         | 49    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 15 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:29.671 | Rx/Tx     | 0x409802488090000000 | 8         | 55    | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 16 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:29.921 | Rx/Tx     | 0x4098024880000000   | 8         | 58    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 17 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:30.172 | Rx/Tx     | 0x409802488090000000 | 8         | 62    | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 18 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:30.421 | Rx/Tx     | 0x4098024880000000   | 8         | 64    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 19 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:30.672 | Rx/Tx     | 0x409802488090000000 | 8         | 70    | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 20 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:30.921 | Rx/Tx     | 0x4098024880000000   | 8         | 72    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 21 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:31.172 | Rx/Tx     | 0x409802488090000000 | 8         | 77    | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 22 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:31.422 | Rx/Tx     | 0x4098024880000000   | 8         | 79    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 23 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:31.672 | Rx/Tx     | 0x409802488090000000 | 8         | 86    | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 24 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:31.921 | Rx/Tx     | 0x4098024880000000   | 8         | 88    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 25 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:32.172 | Rx/Tx     | 0x409802488090000000 | 8         | 93    | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 26 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:32.422 | Rx/Tx     | 0x4098024880000000   | 8         | 95    | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 27 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:32.672 | Rx/Tx     | 0x409802488090000000 | 8         | 101   | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 28 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:32.922 | Rx/Tx     | 0x4098024880000000   | 8         | 103   | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 29 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:33.172 | Rx/Tx     | 0x409802488090000000 | 8         | 106   | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 30 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:33.422 | Rx/Tx     | 0x4098024880000000   | 8         | 110   | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 31 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:33.672 | Rx/Tx     | 0x409802488090000000 | 8         | 115   | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 32 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:33.922 | Rx/Tx     | 0x4098024880000000   | 8         | 118   | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 33 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:34.172 | Rx/Tx     | 0x409802488090000000 | 8         | 122   | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 34 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:34.422 | Rx/Tx     | 0x4098024880000000   | 8         | 126   | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 35 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:34.672 | Rx/Tx     | 0x409802488090000000 | 8         | 131   | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |
| 36 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:34.922 | Rx/Tx     | 0x4098024880000000   | 8         | 134   | 512              | 3         | 4             | 2        | 72          | 2147483648 |   |   |   |   |   |   |
| 37 | NRJ1_PGN_ContinuousBuiltInTest | 41936667 | 65287 | NRJ1   | 15:29:35.172 | Rx/Tx     | 0x409802488090000000 | 8         | 138   | 512              | 3         | 4             | 1        | 72          | 150994944  |   |   |   |   |   |   |

FIGURE 2.21 – Visualisation des trames décodées dans le fichier Excel

Comme le montrent les trois figures ci-dessus, le test de l'application a été concluant. En effet, le logiciel a pu répartir correctement les trames à travers les quatre interfaces. De plus, le décodage des trames est complètement fonctionnel et permet de connaître les données contenues dans toutes les trames. Enfin, les fonctionnalités du fichier HTML, visualisation détaillée des trames à la demande et tracé des signaux, fonctionnent correctement.

Bien que ce test ait été concluant, il ne suffit pas à valider totalement le fonctionnement de cette application. En effet, le fichier de test ne contient que 14 000 lignes, ce qui reste minime comparé à la taille que peuvent atteindre les fichiers de logs en conditions réelles. La robustesse de la fonction de tri du logiciel n'a pas pu être testée, car les logs

sont récupérés d'une seule source, la carte du POD NRJ, avec seulement une partie des messages puisque les messages événementiels ne sont pas envoyés.

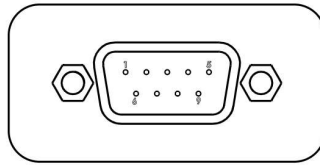
### **Mise en place de la manipulation pour récupérer un fichier de logs authentique**

Pour valider pleinement le fonctionnement de l'application, il est indispensable de disposer d'un fichier de logs authentique. Pour obtenir ce fichier et résoudre les problèmes de communication du véhicule, il est nécessaire de le récupérer directement depuis celui-ci. Comme la récupération du fichier de logs doit être possible sans avoir un ordinateur à proximité, le CL2000 a été envoyé à l'IFREMER. Comme mentionné précédemment, le CL2000 a la particularité d'enregistrer les données sur une carte SD. Cette fonctionnalité présente un léger inconvénient : comme l'équipement n'est pas connecté directement à un ordinateur, il n'est pas alimenté. Cela nécessite donc une source d'alimentation si les logs sont récupérés lors d'essais en piscine, ou bien de tirer des câbles à l'intérieur du véhicule et trouver un emplacement pour l'équipement au sein d'un POD étanche lors de tests en mer. Dans le cas présent, les problèmes de communication mis en évidence en Figure 2.5 ont été observés lors d'essais en piscine, ce qui rend la manipulation plus facile.

Pour réaliser la manipulation, les équipements suivants doivent être utilisés :

- Câble outillage DB9 vers fils ;
- Câble outillage du connecteur J2 du POD NRJ ;
- Logger CL2000 ;
- Câble mini-USB vers USB ;
- Alimentation (+7.0 V à +32 V DC).

L'IFREMER ayant déjà en sa possession le câble outillage du connecteur J2 (connecteur permettant de récupérer le bus CAN sur un bornier à vis), seul le câble outillage DB9 vers fils doit être réalisé. Ce dernier est quelque peu différent de celui qui avait été réalisé pour la sonde IXXAT (voir Figure 2.18) car cette dernière n'avait pas besoin d'être alimentée. Sur le site du fournisseur du CL2000, le brochage du connecteur DB9 de l'outil est fourni.



| Pin # | DB9 connector   |
|-------|-----------------|
| 1     | NC              |
| 2     | CAN L           |
| 3     | GND             |
| 4     | NC              |
| 5     | NC              |
| 6     | GND (optional)  |
| 7     | CAN H           |
| 8     | NC              |
| 9     | Supply (7-32 V) |

FIGURE 2.22 – Brochage du connecteur DB9 du CL2000

La broche 3 est utilisée pour le GND\_CAN, tandis que la broche 6 est utilisée pour le GND\_VCC. Une fois le câble outillage réalisé et testé, la manipulation peut être mise en place. Le synoptique de cette manipulation est le suivant :

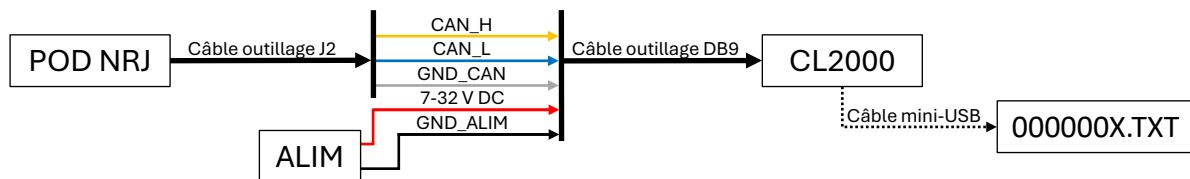


FIGURE 2.23 – Montage de la manipulation avec le CL2000

Une fois tous les équipements nécessaires à la manipulation collectés et préparés, ils sont prêts à être envoyés à l'IFREMER pour récupération des logs. Pour faciliter la mise en place de la manipulation, un document répertoriant toutes les informations ci-dessus a également été réalisé et envoyé avec les équipements.

## 2.3 Projet annexe

Le sujet principal de ce stage ayant été réalisé en moins de temps que prévu, un deuxième sujet de stage a été traité. Cette partie mettra en évidence le travail réalisé dans le cadre de ce second sujet.

### 2.3.1 Identification du besoin

Cette fois, le travail à réaliser est la conception d'un outil de génération automatique de documents Word, devant impérativement suivre une charte graphique stricte. Lors de la procédure de test d'un équipement, un fichier XML est généré à la fin des tests et contient les résultats des tests de non-régression. Un autre fichier de type `.comment` est également généré et contient les éventuels commentaires de la personne ayant effectué les tests.

L'outil repose sur une combinaison de bibliothèques Python pour la manipulation de fichiers Word (`python-docx`), de données XML (`lxml`), et sur l'automatisation de Microsoft Word via le modèle COM (`win32com.client`). Cette dernière librairie impose l'utilisation de l'outil dans un environnement Windows avec Microsoft Word installé localement uniquement.

Lors de la génération du document, un maximum d'informations doivent être préremplies en plus des données récupérées du fichier XML, pour avoir un document qui nécessite le minimum d'actions humaines. Parmi ces informations se trouvent, le nom de l'auteur, le titre du document, le vérificateur, et d'autres informations qui seront explicitées plus bas.

### 2.3.2 Réalisation du logiciel de création de fichier Word

Le document à générer à la fin devant suivre une présentation stricte et difficilement reproductible en utilisant Python, le modèle fourni par l'entreprise est utilisé en tant que modèle dans le code. En utilisant la librairie `python-docx`, il est possible d'ouvrir un document existant et de directement le modifier. De cette manière, pour réaliser ce document, il suffit d'intégrer les informations du fichier XML et quelques informations supplémentaires.

#### Création des tableaux de tests

Comme dit précédemment, la partie la plus importante du document Word à générer est l'intégration des résultats de tests. Dans le modèle associé à ce type de document, qui est une CRE (Compte Rendu d'Essais) de non-régression, les tableaux dans lesquels insérer les résultats des tests sont normés.

| N° Item   | DESIGNATION / DEROULEMENT | MOYEN DE CONTROLE | RESULTATS ATTENDUS | RESULTATS OBTENUS | Commentaires |
|-----------|---------------------------|-------------------|--------------------|-------------------|--------------|
| 1 . 0 . 0 | Titre 1.1                 |                   |                    |                   |              |
| 1 . 1 . 0 | Titre 1.1.1               |                   |                    |                   |              |
| 1 . 1 . 1 |                           |                   |                    |                   |              |
| 1 . 1 . 2 |                           |                   |                    |                   |              |
| 1 . 2 . 0 | Titre 1.1.2               |                   |                    |                   |              |
| 1 . 2 . 1 |                           |                   |                    |                   |              |
| 2 . 0 . 0 | Titre 1.2                 |                   |                    |                   |              |
| 2 . 1 . 0 | Titre 1.2.1               |                   |                    |                   |              |
| 2 . 1 . 1 |                           |                   |                    |                   |              |
| 6 . 0 . 0 | Fin des essais            |                   |                    |                   |              |

FIGURE 2.24 – Modèle de référence pour les tableaux des fichiers de type CRE

Toujours en utilisant la librairie `python-docx`, il est facile de créer une copie de ce tableau en connaissant la largeur de chaque colonne. Cependant, l'insertion de ce tableau dans le document est plus compliquée. En effet, le modèle utilisé est composé de sept sections et ce tableau est supposé être inséré au sein de la cinquième section.

Pour insérer des informations au milieu d'un document Word en utilisant Python, il existe deux options. La première option consiste à définir un signet directement dans le modèle utilisé. Les signets dans Word sont des emplacements marqués qu'il est possible d'utiliser pour insérer du texte ou d'autres éléments en utilisant la librairie évoquée plus haut. Pour insérer du texte à un signet, il faut parcourir les paragraphes et les segments de texte pour trouver le signet, puis utiliser la méthode `add_text` pour insérer le texte souhaité. La seconde méthode consiste à utiliser des balises uniques dans le modèle pour indiquer où le texte doit être inséré. Par exemple, il est possible d'utiliser une balise comme `{{INSERER_TEXTE_ICI}}`. Comme pour l'utilisation des signets, il suffit de rechercher la balise et de la remplacer par le texte souhaité.

Bien que ces deux méthodes répondent, en théorie, au problème d'insertion des tableaux dans le texte, cela ne fonctionne pas en réalité. En effet, l'utilisation de n'importe laquelle de ces méthodes ne permet d'insérer qu'un seul élément, en une seule fois, à l'emplacement souhaité. Les autres éléments seront insérés tout en bas du document par défaut, ne répondant pas au besoin d'insertion au milieu du document. Pour résoudre ce problème, la solution a été de séparer le modèle en deux parties :

- `model.docx` : première partie du modèle contenant les parties 1 à 5 ainsi que la page de garde ;
- `model_end.docx` : seconde partie du modèle contenant les parties 6 et 7 du document.

En séparant le modèle de cette manière, il suffit d'insérer les éléments à la fin du

premier modèle, qui correspond au paragraphe 5 du document puis de fusionner les deux documents en copiant les éléments du second modèle à l'intérieur du premier.

## Extraction des données du fichier XML

Maintenant que les tableaux sont correctement créés, il est nécessaire de les remplir avec les informations extraites du fichier XML. Le processus d'extraction des informations commence par le chargement du fichier XML qui est ensuite analysé. Le fichier XML généré lors de la phase de tests est structuré avec des balises qui définissent différents éléments ainsi que leurs attributs. La méthode de parsing utilisée, identifie et utilise ces balises pour extraire les informations. Deux types de balises distincts sont utilisés dans le fichier.

Le premier type de balise correspond aux balises d'éléments principaux, telles que `<MissionName>` et `<StartDate>`, qui représentent les informations générales du document. Par exemple, comme son nom l'indique, la balise `<MissionName>` contient le nom de la mission qui sera utilisée pour définir le nom du document Word. La balise `<StartDate>` indique la date et l'heure de début de la mission, information aussi utilisée pour définir le nom du document.

Les balises de blocs de tests, comme `<Start-PUC>`, `<STATUT>` et `<CARTO>`, encapsulent des groupes de tests individuels. Ces balises contiennent plusieurs attributs, mais seul l'attribut `desc` est intéressant puisqu'il contient le titre du bloc associé. Dans le fichier Word, chaque bloc représentera une sous-partie de la cinquième partie afin de mieux trier les différents tests et permettre une lecture plus facile du document. Chacun de ces blocs de tests contiennent plusieurs balises `<UnitTest>` qui détaillent les tests effectués dans cette partie. Toutes ces balises contiennent plusieurs attributs, essentiels pour la réalisation du document Word :

- **Type** : Cet attribut peut prendre deux valeurs. Un test de type "TestLigne" indique que cette ligne contient le titre d'une sous-partie du bloc de tests. Un test de type "Standard" indique que la ligne contient les résultats d'un test ;
- **Libelle** : Contient le nom du test ou du titre ;
- **Status** : Dans le cas d'une ligne avec `Type="Standard"`, indique si ce test est validé ou non. Si `Type="TestLigne"`, prend la valeur "Non teste" ;
- **CodePanne** : Peut prendre plusieurs valeurs, chaque valeur est associée à une action à effectuer par la suite.

Pour accéder aux éléments et attributs spécifiques dans le fichier XML, la bibliothèque `lxml` est utilisée pour analyser et interpréter la structure du fichier. Le fichier XML est chargé puis analysé en utilisant `lxml.etree`, ce qui permet de créer une représentation arborescente du document XML, facilitant grandement la navigation dans le fichier et ainsi l'accès aux différents éléments.

```
tree = ET.parse(xml_file_path)
root = tree.getroot()
```

Une fois l'arborescence du fichier extraite, le code peut facilement accéder aux éléments spécifiques en utilisant les méthodes `find` et `findall` pour localiser les balises. Par exemple, pour extraire le nom de la mission, il suffit de rechercher la balise `<MissionName>` et de récupérer le texte associé :

```
mission_name = root.find("MissionName").text
```

De la même manière, il devient facile d'accéder aux attributs des différentes balises du document pour en extraire la valeur. Par exemple, pour connaître la valeur de l'attribut `desc` de la balise `<Start-PUC>` :

```
desc = root.find("Start-PUC").get("desc")
```

En utilisant ces méthodes, il devient facile de peupler les différents tableaux du fichier Word avec les données des tests. En combinant l'extraction des données à la création des tableaux, le résultat est le suivant :

## 5. PROCÉDURE

### 5.1. MISE SOUS TENSION DU VEHICULE PAR LE PUC

| N° Item                                                           | Désignation / Déroulement | Moyen de contrôle | Résultats attendus | Résultats obtenus                                                                                      | Commentaires |
|-------------------------------------------------------------------|---------------------------|-------------------|--------------------|--------------------------------------------------------------------------------------------------------|--------------|
| <b>5.1.1. Demarrage</b>                                           |                           |                   |                    |                                                                                                        |              |
| 5.1.1.1.1                                                         | Demarrage sur START       | Visuel            |                    | <input checked="" type="checkbox"/> OK<br><input type="checkbox"/> POK<br><input type="checkbox"/> NOK | 0000         |
| 5.1.1.2.1                                                         | Le voyant clignote        | Visuel            |                    | <input checked="" type="checkbox"/> OK<br><input type="checkbox"/> POK<br><input type="checkbox"/> NOK | 3010         |
| <b>Fin des essais de Mise sous tension du vehicule par le PUC</b> |                           |                   |                    |                                                                                                        |              |

### 5.2. CONTROLE D'ABSENCE DE DEFAULT - MODE EXPERT

| N° Item                     | Désignation / Déroulement           | Moyen de contrôle | Résultats attendus | Résultats obtenus                                                                                      | Commentaires |
|-----------------------------|-------------------------------------|-------------------|--------------------|--------------------------------------------------------------------------------------------------------|--------------|
| <b>5.2.1. Navigation</b>    |                                     |                   |                    |                                                                                                        |              |
| <b>5.2.1.1. Vue globale</b> |                                     |                   |                    |                                                                                                        |              |
| 5.2.1.1.1                   | Centrale inertielle - Jaune ou vert | Visuel            |                    | <input checked="" type="checkbox"/> OK<br><input type="checkbox"/> POK<br><input type="checkbox"/> NOK | 2010         |

FIGURE 2.25 – Tableau de test rempli avec les données extraites du fichier XML

La cinquième partie du document étant correctement remplie, les deux modèles peuvent maintenant être fusionnés pour avoir le document complet.

## Mise à jour des propriétés du document

Comme dit plus tôt, le document Word attendu en sortie n'est pas seulement composé du tableau réalisé ci-dessus, d'autres informations doivent être remplies. Ces informations sont les propriétés du document. Tout document Word possède des

propriétés telles que le titre, l’auteur, le sujet, et d’autres métadonnées. Pour les modifier, c’est à nouveau la librairie `python-docx` qui est utilisée. Ces propriétés sont accessibles via l’objet `core_properties` du document qui permet de définir et de mettre à jour ces métadonnées. Par exemple, dans le code, le titre du document est défini avec :

```
doc.core_properties.title = f"{formatted\_date}\_{mission_name}"
```

Où `formatted_date` et `mission_name` sont des valeurs dynamiques, issues du fichier XML précédemment analysé.

Cependant, ces propriétés ne sont pas celles qui nous intéressent le plus. En effet, dans le modèle défini par l’entreprise, certaines propriétés sont personnalisées. Elles apparaissent sous forme de balises dans le texte et sont automatiquement modifiées en définissant leur valeur via une interface dédiée, associée à l’entreprise, dans le fichier Word. En utilisant seulement les librairies disponibles sur Python, il n’est pas possible de les modifier, c’est pourquoi il a fallu utiliser une autre méthode.

## Structure d’un document Word

En réalité, un fichier `.docx` est une archive au format ZIP. Lorsque l’extension du fichier Word est modifiée pour `.zip` et que le fichier compressé est extrait, sa structure interne, organisée selon la norme Open XML définie par Microsoft, est révélée :

- `/word/document.xml` : corps principal du document (texte, tableaux, paragraphes...);
- `/word/styles.xml` : définitions de style (titres, tableaux, marges, interlignes...);
- `/word/settings.xml` : options globales du document;
- `/docProps/core.xml` : métadonnées classiques (titre, auteur, date de création...);
- `/docProps/app.xml` : métadonnées additionnelles telles que la propriété Manager;
- `/customXml/itemX.xml` : fichiers contenant des propriétés personnalisées, le "X" représente une valeur de 1 à 6;
- `/[Content_Types].xml`, `_rels/`, etc. : fichiers de définition et de relation entre les composants du document.

En connaissant cette organisation modulaire, il devient possible de cibler précisément les éléments du document à modifier, sans passer par l’interface graphique. Pour modifier les propriétés personnalisées du document, il est nécessaire de modifier directement les fichiers `.xml` de l’archive ZIP associée au modèle du document.

## Mise à jour des propriétés personnalisées du document

La mise à jour des propriétés personnalisées d’un document Word nécessite une manipulation directe des fichiers XML contenus dans l’archive ZIP du fichier `.docx`. Le

processus commence par la transformation du fichier `.docx` en une archive ZIP. Cette opération est réalisée en utilisant le module `zipfile` de Python, qui permet de décompresser le contenu du fichier dans un répertoire temporaire.

L'utilisation de fichiers temporaires est cruciale pour assurer l'intégrité du fichier final. En effet, toutes les modifications sont effectuées sur une copie temporaire du document, ce qui évite toute corruption du fichier original. Une fois les modifications terminées, un nouveau fichier `.docx` est créé à partir des fichiers temporaires modifiés. Cette approche garantit que le fichier original reste intact en cas d'erreur lors du processus de mise à jour.

La modification des propriétés personnalisées se fait principalement dans un des fichiers `itemX.xml`, situé dans le répertoire `customXml` de l'archive. Ce fichier contient les propriétés personnalisées définies par l'utilisateur. Le numéro du fichier `itemX.xml` n'étant pas toujours le même et pouvant changer si une modification est faite sur le modèle, le code recherche une balise spécifique pour identifier le bon fichier. Cette recherche est effectuée en parcourant les fichiers XML présents dans le répertoire `customXml` et en vérifiant la présence de la balise `Custom_Properties`, qui indique que le fichier contient les propriétés personnalisées recherchées.

Le code utilise la bibliothèque `lxml` pour analyser et modifier le contenu XML. Les propriétés sont mises à jour en parcourant les éléments XML et en modifiant les valeurs des balises correspondantes.

Un cas particulier est celui de la propriété `Manager`, qui se trouve dans le fichier `app.xml` situé dans le répertoire `docProps`. Le code ouvre et modifie directement ce fichier pour mettre à jour le nom de l'approbateur du document.

Toutes les valeurs devant être associées aux propriétés personnalisées sont contenues dans un fichier `config.toml` qui est lu par le programme. Les valeurs sont ensuite ajoutées aux balises XML correspondantes. Il est donc impératif que le fichier de configuration soit mis à jour en amont de la génération du fichier Word.

### **Automatisation Word via `win32com.client`**

Afin que le fichier n'ait besoin que d'un minimum d'interventions de la part de l'opérateur, il reste encore quelques ajouts à faire dans le programme. Certaines manipulations Word ne sont pas réalisables avec `python-docx`, notamment :

- Insérer du texte dans un signet (`bookmark.Range.InsertAfter()`);
- Mettre à jour la table des matières (`doc.TablesOfContents(1).Update()`).

La bibliothèque `win32com.client`, basée sur le modèle COM (Component Object Model), permet de piloter Word comme si l'utilisateur le faisait lui-même, mais de façon logicielle. Cela nécessite que Word soit installé sur la machine exécutant le script.

Pour insérer le contenu du fichier `.comment` à l'endroit prévu à cet effet, un signet est défini dans le modèle au préalable. La fonction `bookmark.Range.InsertAfter()` est ensuite utilisée pour insérer le contenu du fichier `.comment` au sein de la section 6. Puis, la fonction `doc.TablesOfContents(1).Update()` est utilisée pour mettre à jour la table des matières du fichier Word.

### 2.3.3 Réalisation de l'interface de sélection de fichiers

Comme dit plus haut, pour fonctionner, ce script a besoin de deux fichiers : un fichier `.xml` duquel il va extraire les informations relatives aux tests et un fichier `.comment` duquel sont extraits les commentaires de l'opérateur des tests. Pour sélectionner ces fichiers, une interface a été réalisée. Sur cette interface, il est également possible de sélectionner le dossier de sortie, dossier dans lequel sera généré le fichier Word. L'interface réalisée est la suivante :

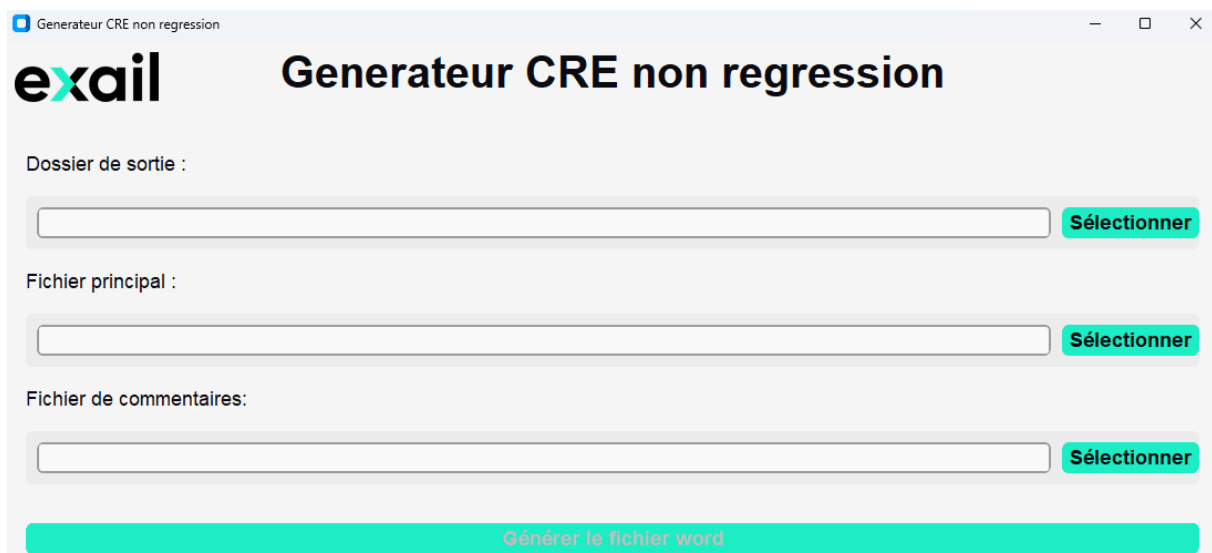
The image shows a web-based interface titled "exail Generateur CRE non regression". It has a light blue header with the "exail" logo and the title. Below the header, there are three sections for file selection. The first section is labeled "Dossier de sortie :" and contains a text input field and a green button labeled "Sélectionner". The second section is labeled "Fichier principal :" and also contains a text input field and a green button labeled "Sélectionner". The third section is labeled "Fichier de commentaires:" and contains a text input field and a green button labeled "Sélectionner". At the bottom of the interface, there is a large green button labeled "Générer le fichier word".

FIGURE 2.26 – Interface de sélection des fichiers et du dossier de sortie

Afin d'éviter que le programme soit lancé sans que l'utilisateur ait choisi un dossier de sortie et un fichier `.xml`, le bouton permettant de générer le fichier Word est désactivé tant que ces deux champs ne sont pas remplis. La sélection d'un fichier `.comment` n'est pas obligatoire puisqu'il est possible que l'opérateur n'ait fait aucun commentaire lors des tests. Cette interface a été réalisée aux couleurs de l'entreprise pour respecter la charte IHM (Interface Homme Machine).

#### Génération d'un fichier `.exe`

Ce programme ayant pour objectif d'être utilisé sur une machine sécurisée et n'ayant pas accès à Internet, il est nécessaire de générer un `.exe` car il sera impossible d'y installer les librairies Python nécessaires. Pour le générer, la librairie `PyInstaller` est utilisée. Lors

de la génération du fichier, il est essentiel d'ajouter certaines librairies au binaire généré pour assurer le fonctionnement de l'application et qu'elle se porte toute seule. Pour ce faire, la commande suivante est utilisée :

```
python -m PyInstaller --onefile --hidden-import=customtkinter
--hidden-import=lxml --hidden-import=win32com.client
--add-data "ressources;ressources" --windowed main.py
```

Cette commande utilise plusieurs options de PyInstaller :

- **onefile** : Cette option permet de regrouper tout le code et les dépendances en un seul fichier exécutable, ce qui facilite le déploiement et l'utilisation sur des machines sans accès à Internet ;
- **hidden-import** : Elle est utilisée pour inclure des modules qui ne sont pas automatiquement détectés par PyInstaller. Dans ce cas, les modules `customtkinter`, `lxml`, et `win32com.client` sont ajoutés manuellement ;
- **add-data** : Cette option permet d'inclure des fichiers de données supplémentaires nécessaires à l'application. Ici, le dossier `ressources` est inclus pour s'assurer que toutes les ressources nécessaires sont disponibles ;
- **windowed** : Cette option indique que l'application doit être exécutée dans une fenêtre, sans afficher une console, ce qui est souvent souhaitable pour les applications graphiques.

Une fois cette commande exécutée, un fichier `.exe` autonome est généré dans le dossier `dist` de votre projet. Ce fichier peut ensuite être copié sur la machine cible et exécuté sans nécessiter d'installation supplémentaire.

## 3. Bilan de mon activité de stagiaire

### 3.1 Présentation des résultats

#### 3.1.1 Sujet principal du stage

Pour rappel, avant le début de mon stage, l'entreprise ne disposait d'aucun outil interne permettant de visualiser des trames CAN à partir d'un fichier de logs, de les décoder, ni d'y associer des fonctionnalités avancées telles que le tracé des signaux en fonction du temps.

Aujourd'hui cet outil est complètement fonctionnel. En effet, les tests effectués à partir de fichiers de logs récupérés sur les prototypes des cartes associées au POD NRJ ont été concluants. Cet outil unifie des fonctionnalités jusqu'alors réparties sur divers logiciels, permettant à l'utilisateur de tout analyser à partir d'une seule application. Ce travail a été présenté à plusieurs ingénieurs et techniciens qui pourraient avoir besoin de l'utiliser et qui ont pu faire des retours afin de faire évoluer l'outil. Toutes les personnes à qui il a été présenté ont été satisfaites du travail réalisé, ce qui montre l'utilité de ce travail.

Afin que cet outil reste utilisé même après mon départ, les documents relatifs à son utilisation et à la compréhension du code sont en cours de réalisation, cela afin de permettre à l'entreprise de le faire évoluer et d'en faire l'outil à utiliser dans le cadre de l'analyse de trames CAN.

Bien que le fonctionnement de l'outil ait été validé, il l'a été seulement pour des logs provenant d'un seul POD et d'une taille faible en comparaison avec la réalité. En l'absence de fichiers de logs récupérés directement sur véhicule, il est difficile de valider la robustesse du code dans des conditions réalistes, notamment avec de gros volumes de données et des opérations de tri complexes via l'interface.

#### 3.1.2 Sujet secondaire du stage

Pour le second sujet que j'ai traité au cours de ce stage, quand il m'a été attribué, tout le CRE devait être réalisé à la main, ce qui était très fastidieux.

Au moment où ce document est écrit, l'outil de génération de CRE a été complètement testé et permet de générer un fichier Word qui respecte les normes imposées par le modèle de l'entreprise, tout en remplissant un maximum d'informations pour éviter à la personne en charge des essais de le modifier autant que possible. Afin d'assurer que l'application fonctionne sur toutes les machines, elle a été testée sur différents appareils fonctionnant

sur Windows 10 et Windows 11 et même sur des ordinateurs cryptés et sécurisés. Ces tests ont montré que, en ayant l'application ainsi que les fichiers ressources nécessaires (le fichier de configuration et les deux parties du modèle Word), la génération du document Word est possible sur n'importe quel appareil sans installation complémentaire. Ce sujet est donc totalement finalisé et déjà prêt à être utilisé.

## **3.2 Prolongements possibles de la mission réalisée**

Concernant le premier sujet de ce stage, certaines évolutions sont envisageables.

### **3.2.1 Ajout de l'analyse en temps réel**

Actuellement, l'outil ne permet que le décodage en différé des fichiers de logs. Les loggers utilisés (IXXAT et CL2000) possédant tous deux la fonctionnalité d'envoyer les trames sur le port série en temps réel, il pourrait être intéressant d'ajouter une option temps réel qui permettrait de simplement brancher un des loggers en USB et de récupérer les trames de cette manière. Pour ce faire, l'implantation d'un thread est nécessaire. En effet, l'application étant un programme qui tourne en boucle et en attente d'actions de la part de l'utilisateur, il est nécessaire d'implanter un thread pour gérer toute la partie réception et l'ajout des trames dans les listes utilisées dans l'application.

### **3.2.2 Amélioration de la réactivité de la page HTML**

Un problème qui a été observé concerne la réactivité de la page HTML. Lors des tests avec un fichier de logs de 14 000 lignes, au lancement, la page HTML était peu réactive. Cela est dû au fait que toutes les données et le code de la page sont stockés dans le même fichier HTML. Pour résoudre cela, il a été envisagé de créer un fichier de type base de données qui serait utilisé par la page HTML pour alléger son code interne. Cette approche permettrait non seulement d'améliorer la réactivité de la page, mais aussi de structurer et de gérer les données de manière plus efficace.

En utilisant une base de données dédiée pour stocker les logs, il serait possible de séparer les données du code HTML, réduisant ainsi la charge initiale sur le navigateur. De plus, l'utilisation d'une API pour accéder aux données depuis la base de données vers la page HTML pourrait grandement améliorer les performances, en ne chargeant que les données nécessaires au moment opportun.

Enfin, il serait crucial d'optimiser le code JavaScript utilisé pour manipuler les données, en évitant les boucles inutiles et en utilisant des méthodes efficaces pour manipuler le DOM, réduisant ainsi le temps de traitement et améliorant l'expérience utilisateur.

### 3.3 Apports en termes d'expérience personnelle et disciplinaire

Ce stage de 14 semaines au sein d'Exail Robotics a été une expérience extrêmement enrichissante, tant sur le plan professionnel que personnel. Il m'a permis de développer et renforcer des compétences techniques, organisationnelles et personnelles dans un cadre d'ingénierie structuré et exigeant.

Sur le plan disciplinaire, ce stage m'a permis d'approfondir mes connaissances sur le protocole CAN (Controller Area Network), un sujet abordé en cours mais que j'ai pu explorer en profondeur grâce à la conception d'un outil de décodage de trames CAN. J'ai dû comprendre sa structure, ses limites et ses usages dans un contexte contraint, notamment avec la norme NMEA 2000 que je ne connaissais pas auparavant. J'ai approfondi ma compréhension de la modélisation du trafic CAN, ce qui m'a amené à analyser des documentations pour reproduire ce trafic dans un outil logiciel que j'ai développé intégralement. Ce développement m'a obligé à monter en compétence sur tout un écosystème Python que je n'avais jamais utilisé en pratique : bibliothèques de parsing (comme csv), d'interface graphique (CustomTkinter), d'export HTML ou Excel (pandas), génération d'exécutables (PyInstaller), structuration de bases de données de messages, etc. J'ai également acquis une compréhension approfondie de la structure des fichiers Word, ce qui a été essentiel pour le développement d'un outil de génération automatique de documents Word.

La gestion de projet a été un aspect clé de ce stage. J'ai participé à des points hebdomadaires avec mon tuteur de stage et l'ingénieur avec qui je collaborais, ce qui m'a permis de suivre l'avancement de mon travail et de prendre des décisions éclairées. Mon maître de stage a insisté sur la nécessité de communiquer une date de livraison au client, ce qui a ajouté une dimension supplémentaire à la gestion de mon temps et de mes priorités. Cela m'a permis de mieux comprendre l'importance de la planification et de la communication avec les clients dans un environnement professionnel. Cette collaboration avec des ingénieurs expérimentés m'a appris à travailler différemment, en intégrant des normes et des pratiques professionnelles, telles que la norme NMEA 2000 et en adaptant mon rythme à des besoins techniques et humains. Cette méthodologie m'a également permis de prendre conscience de la rigueur exigée par des projets industriels, en particulier en matière de sécurité logicielle dans des environnements sensibles.

Sur le plan personnel, ce stage m'a permis d'évoluer de façon significative. J'étais dans un rôle d'ingénieur en devenir, avec une autonomie réelle sur un projet à forte responsabilité. Contrairement à mon stage précédent, où j'étais principalement impliqué dans des tests de cartes en autonomie, ici j'étais responsable de la résolution d'un problème critique de

communication affectant un véhicule sous-marin livré à un client. Cette évolution de rôle m'a obligé à structurer mes démarches, à anticiper les erreurs et à me projeter sur les conséquences des choix techniques. Elle m'a aussi permis de voir que je pouvais être force de proposition dans un environnement exigeant et de livrer un outil conforme aux attentes, dans les temps, avec des résultats concrets et exploitables. Cette expérience a renforcé ma motivation et mon engagement, car j'ai découvert un domaine qui me passionne et dans lequel je souhaite évoluer.

En conclusion, ce stage a été une expérience extrêmement enrichissante qui m'a permis de développer des compétences techniques, organisationnelles et personnelles. Il a confirmé mon intérêt pour les systèmes embarqués et m'a donné la confiance nécessaire pour poursuivre une carrière dans ce domaine.

# Conclusion

Pendant ce stage de quatorze semaines chez Exail Robotics, j'ai développé un outil de décodage de trames CAN, une mission qui m'a permis d'approfondir mes connaissances sur le protocole CAN et de me familiariser avec la norme NMEA 2000. J'ai également développé un outil de génération automatique de documents Word, ce qui a nécessité une compréhension approfondie de la structure des fichiers Word. Ces missions m'ont permis de monter en compétence sur l'écosystème Python et de découvrir de nouvelles bibliothèques et outils.

L'objectif principal de ce stage était de développer des outils logiciels répondant à des besoins spécifiques dans le domaine des systèmes embarqués. Grâce aux enseignements dispensés dans ma formation, j'ai pu aborder ce stage avec une base solide en programmation et en gestion de projet. Cependant, la confrontation avec les réalités du terrain m'a permis de comprendre l'importance de l'adaptabilité et de la rigueur dans le respect des normes industrielles. Ce que j'ai le plus apprécié dans cette expérience, c'est l'autonomie et la responsabilité qui m'ont été accordées, ainsi que la collaboration avec des ingénieurs expérimentés.

Ce stage a été une expérience extrêmement enrichissante qui m'a permis de confronter mon projet professionnel à la réalité du terrain. Il a confirmé mon intérêt pour les systèmes embarqués et m'a donné la confiance nécessaire pour poursuivre une carrière dans ce domaine. Travailler dans une entreprise comme Exail Robotics, qui allie innovation et rigueur technique, est une perspective qui m'attire particulièrement. Je souhaite donc continuer à développer mes compétences dans ce domaine et j'envisage une carrière dans ce type d'entreprise à l'avenir. Cette expérience m'a également permis de prendre du recul sur mes aspirations professionnelles et de mieux comprendre les défis et les opportunités qui m'attendent dans le monde de l'ingénierie.

# Lexique

**API** : Application Programming Interface, une interface permettant à différents logiciels de communiquer entre eux.

**AUV** : Autonomous Underwater Vehicle, un véhicule sous-marin autonome utilisé pour diverses missions sous-marines.

**BMS** : Un BMS (Battery Management System) est un système électronique assurant la surveillance, la protection et la gestion des paramètres d'un pack batterie (tension, courant, température, état de charge).

**BUT** : Bachelor Universitaire de Technologie.

**CAN** : Controller Area Network, un protocole de communication utilisé principalement dans les systèmes embarqués.

**COM** : Component Object Model, un modèle de composant logiciel introduit par Microsoft.

**Console de debug** : interface affichant en temps réel des informations techniques destinées au diagnostic et au suivi de l'exécution du programme.

**CRE** : Compte Rendu d'Essais, un document détaillant les résultats des tests effectués.

**DB9** : Un type de connecteur.

**DVL** : Un DVL (Doppler Velocity Log) est un type de sonar utilisé pour mesurer la vitesse d'un véhicule par rapport à l'eau ou au fond marin, en utilisant l'effet Doppler.

**ESE** : Électronique et Systèmes Embarqués.

**FIFO** : Littéralement "First In First Out", une FIFO est une file d'attente où les données sont traitées dans l'ordre d'arrivée : le premier élément entré est le premier à sortir.

**Firmware** : Logiciel embarqué directement dans un composant matériel, assurant le contrôle de ses fonctions de bas niveau.

**GEII** : Génie Électrique et Informatique Industrielle.

**HTML** : HyperText Markup Language, le langage de balisage standard pour les pages web.

**IFREMER** : Institut Français de Recherche pour l'Exploitation de la Mer.

**IHM** : Interface Homme-Machine.

**Implémenté** : Développé et intégré de manière fonctionnelle dans le logiciel ou système.

**JSON** : JavaScript Object Notation, un format de données textuelles dérivé de la notation des objets du langage JavaScript.

**MCM** : Mine Countermeasures, des mesures et équipements utilisés pour détecter et neutraliser les mines marines.

**MDA** : Maritime Domain Awareness, la connaissance et la compréhension des activités et des conditions dans l'environnement maritime.

**NMEA** : National Marine Electronics Association, une norme de communication pour les systèmes électroniques marins.

**Parser** : Opération consistant à analyser un fichier ou un flux de données, selon une structure définie, afin d'en extraire les informations pertinentes ou d'en faciliter le traitement automatique.

**PGN** : Parameter Group Number, un identifiant utilisé dans les messages NMEA 2000 pour définir le type de message.

**Photonique** : Science et technologie de la génération, de la manipulation et de la détection de la lumière, principalement sous forme de photons.

**Python** : Un langage de programmation interprété, multiparadigme et multiplateforme.

**R&D** : Recherche et Développement.

**ROV** : Remotely Operated Vehicle, un véhicule sous-marin télécommandé utilisé pour diverses missions sous-marines.

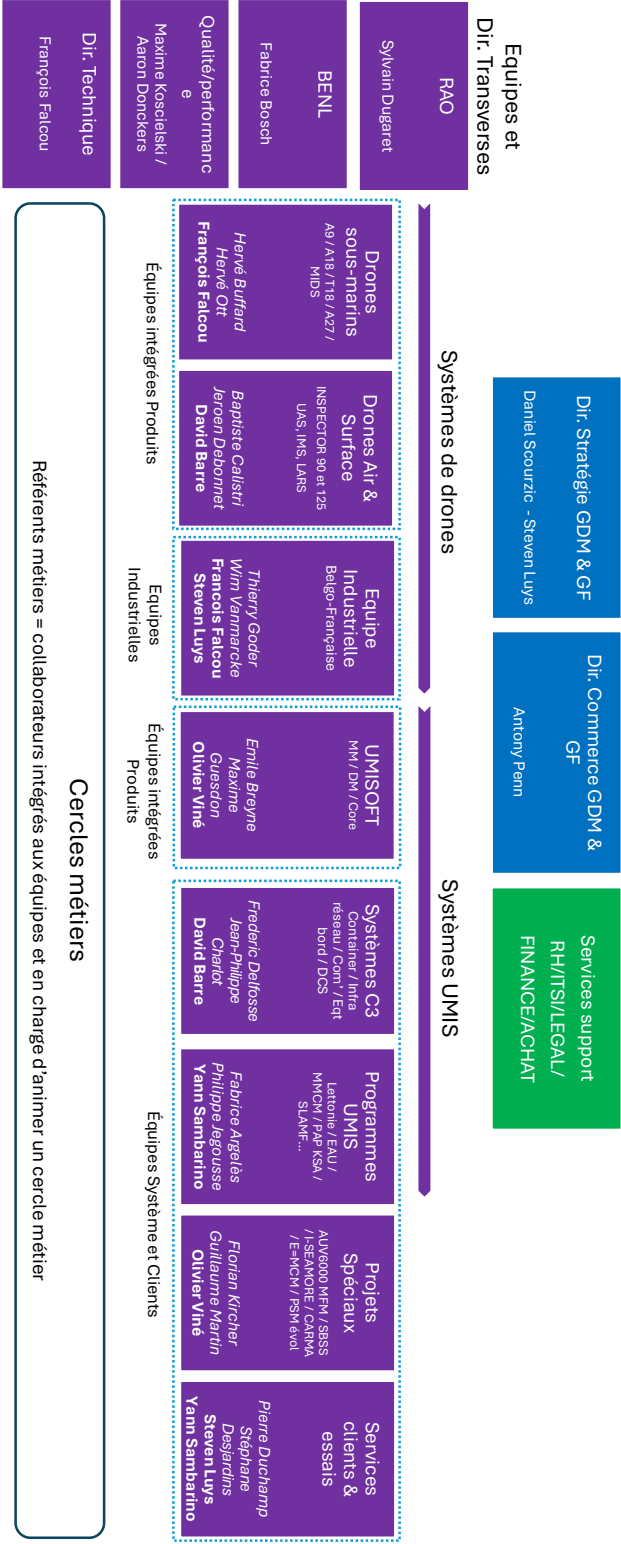
**Sérialisées** : Converties en un format structuré, tel que JSON, pour faciliter le stockage et la transmission des données.

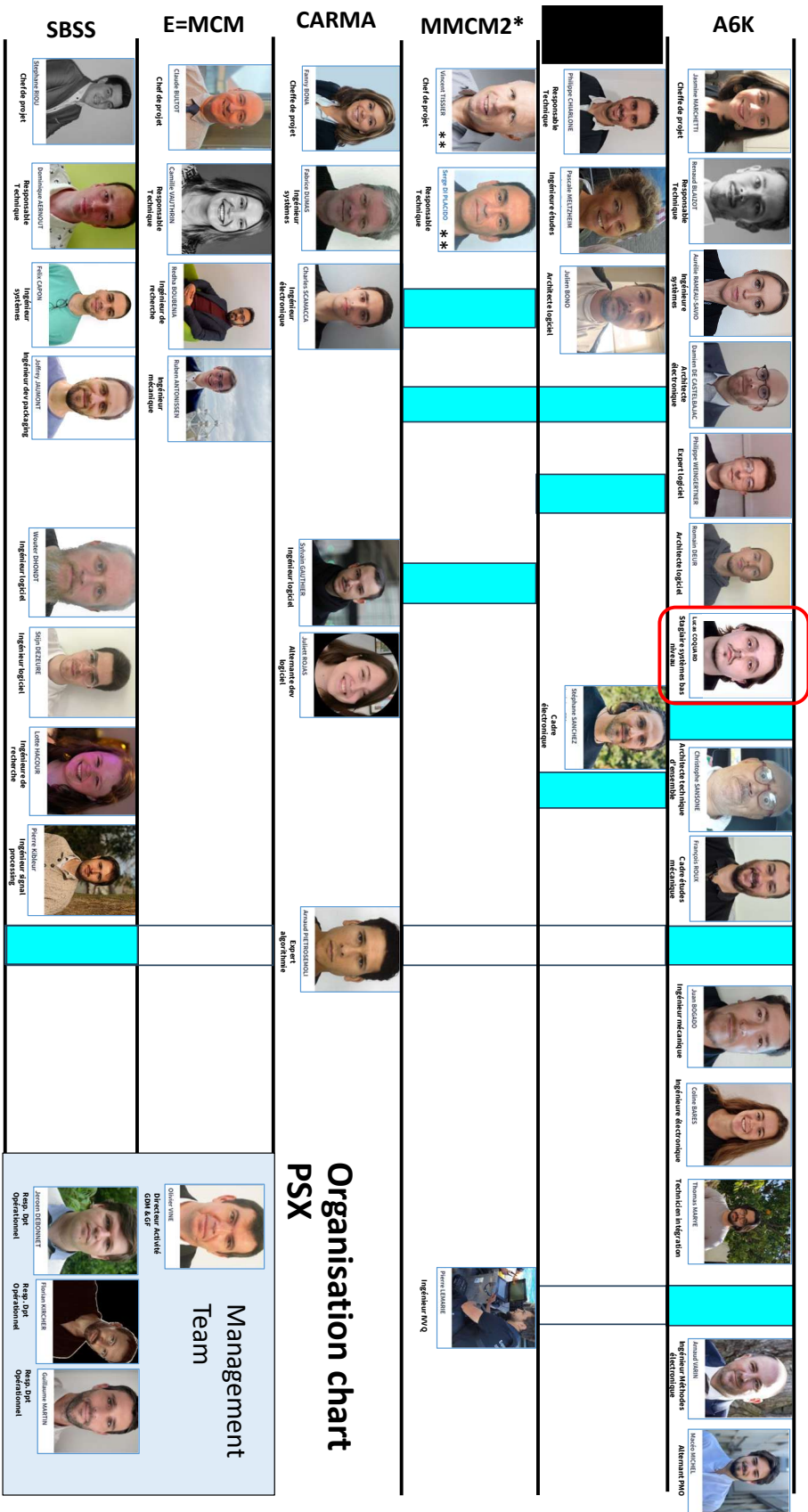
**SID** : Signal Identifier, identifiant d'un signal dans une trame de données.

**STM32** : Une famille de microcontrôleurs 32 bits basée sur l'architecture ARM Cortex-M.

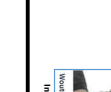
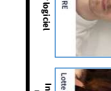
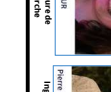
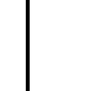
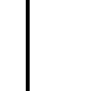
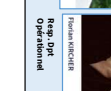
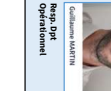
**XML** : eXtensible Markup Language, un langage de balisage conçu pour stocker et transporter des données.

Organisation opérationnelle Guerre des Mines/Grands Fonds

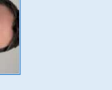




### SBSS

|                                                                                                                       |                                                                                                                              |                                                                                                                           |                                                                                                                               |                                                                                                                           |                                                                                                                           |                                                                                                                            |                                                                                                                             |                                                                                                                             |                                                                                                                             |                                                                                                                             |                                                                                                                             |                                                                                                                             |                                                                                                                             |
|-----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <br>Sébastien BLOU<br>Chef de projet | <br>Sébastien BLOU<br>Responsable Technique | <br>Sébastien BLOU<br>Ingénieur systèmes | <br>Sébastien BLOU<br>Ingénieur de packaging | <br>Sébastien BLOU<br>Ingénieur logiciel | <br>Sébastien BLOU<br>Ingénieur logiciel | <br>Sébastien BLOU<br>Ingénieur logiciel | <br>Sébastien BLOU<br>Ingénieur logiciel | <br>Sébastien BLOU<br>Ingénieur logiciel | <br>Sébastien BLOU<br>Ingénieur logiciel | <br>Sébastien BLOU<br>Ingénieur logiciel | <br>Sébastien BLOU<br>Ingénieur logiciel | <br>Sébastien BLOU<br>Ingénieur logiciel | <br>Sébastien BLOU<br>Ingénieur logiciel |
|-----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|

### Management Team

|                                                                                                                                |                                                                                                                                |                                                                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <br>Olivier FINE<br>Resp. Dpt. Opérationnel | <br>Olivier FINE<br>Resp. Dpt. Opérationnel | <br>Olivier FINE<br>Resp. Dpt. Opérationnel |
|--------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|

## Organisation chart

### PSX

  
Ingénieur R+D

\* En cours de transfert DSC / \*\* en mobilité

Annexe 2 : Organigramme du département projets spéciaux

# Résumé et Abstract

## Résumé

Ce stage de 14 semaines chez Exail Robotics a consisté en la conception d'un outil de décodage de trames CAN pour résoudre des problèmes de communication sur un véhicule sous-marin autonome destiné à l'IFREMER. L'objectif principal était de développer une application en Python permettant de visualiser, analyser et décoder les trames CAN à partir de fichiers de logs, tout en intégrant une logique de filtrage similaire à celle des microcontrôleurs STM32 des PODs du véhicule. L'outil permet également d'exporter les résultats sous forme de rapports Excel et HTML interactifs. En parallèle, un projet annexe a été réalisé : la génération automatique de documents Word à partir de fichiers XML de test, en respectant une charte graphique stricte imposée par l'entreprise. Ces travaux ont été menés dans un environnement technique contraint, notamment par les exigences de sécurité spécifiques aux systèmes embarqués à vocation militaire. Ce stage a permis de développer des compétences en traitement de données, en interfaces graphiques Python, en structuration logicielle modulaire, ainsi qu'en outillage électronique et tests matériels.

Mots-clés : Bus CAN, décodage, Python, STM32, système embarqué, interface graphique, export HTML, IFREMER, Exail Robotics, traitement de logs.

## Abstract

This 14-week internship at Exail Robotics focused on developing a CAN frame decoding tool to address communication issues on an autonomous underwater vehicle delivered to IFREMER. The primary objective was to create a Python application capable of visualizing, analyzing, and decoding CAN frames from log files, while incorporating a filtering logic similar to that of the STM32 microcontrollers in the vehicle's PODs. The tool also allows for the export of results in interactive Excel and HTML report formats. Additionally, a secondary project involved the automatic generation of Word documents from XML test files, adhering to strict corporate formatting guidelines. This work was conducted in a technically constrained environment, particularly due to the security requirements specific to military embedded systems. The internship provided an opportunity to develop skills in data processing, Python GUI development, modular software structuring, and electronic tooling and testing.

Keywords : CAN bus, decoding, Python, STM32, embedded system, GUI, HTML export, IFREMER, Exail Robotics, log processing.