

# Mise en œuvre d'un récepteur AIS dans un drone marin de surface

---

RAPPORT DE STAGE 2A

**BENGUET Aimé Prince**

PARCOURS : Systèmes mécatroniques et robotiques (SYSMER)



**Enseignant référent :** M. SORIANO Thierry,  
responsable du parcours SYSMER

**Tuteur organisme d'accueil :** M. CADEL Hilaric,  
Chief Observer, KIETTA

Année universitaire 2022/2023

## ENGAGEMENT DE NON PLAGIAT

Je soussigné, .....BENGUET Aimé Prince....

N° carte d'étudiant : .....22107600.....

Déclare avoir pris connaissance de la charte des examens et notamment du paragraphe spécifique au plagiat.

Je suis pleinement conscient(e) que la copie intégrale sans citation ni référence de documents ou d'une partie de document publiés sous quelques formes que ce soit (ouvrages, publications, rapports d'étudiants, internet, etc...) est un plagiat et constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.

En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour produire et écrire ce document.

Fait le ...01/07/2023 à LA GARDE

Signature(s)



Ce document doit être inséré en première page de tous les rapports, dossiers et/ou mémoires.

## REMERCIEMENTS

L'élaboration de ce rapport est en partie le fait de l'aide et de l'assistance de plusieurs personnes que je tiens à remercier.

Je souhaite remercier monsieur Hilaric CADEL (tuteur de stage), Chef Observateur chez KIETTA ainsi que monsieur Hervé SOUCHE, ingénieur génie électrique responsable du maintien en condition opérationnel (MCO), pour l'encadrement technique et académique tout au long de ce stage.

Je tiens à remercier également monsieur Thierry LEON (responsable logistique), monsieur Benjamin FIORUCCI (QHSE manager), monsieur Éric BATHELLIER (Chef géophysicien), pour les expériences partagées ainsi que l'accueil chaleureux.

## RESUME ET MOTS-CLES

### Français :

Mon stage de deuxième année d'école d'ingénieurs s'est déroulé dans la société KIETTA, s'inscrivant dans le domaine de l'industrie pétrolière et gazière, plus précisément dans l'acquisition sismique marine.

Au cours de ce stage j'ai effectué des travaux sur un récepteur AIS (Autonomous Identification System), dispositif d'aide à la navigation. Le but était l'intégration de ce récepteur dans un drone de surface (USV, plus précisément RAV), plus particulièrement dans le système de contrôle commande. Pour ce faire, je me suis servi des outils et moyens utilisés au sein de l'entreprise et autres (notion de CPA) pour recevoir, décoder et traiter les données AIS. Ceci passe, notamment par l'utilisation de la carte Raspberry Pi, des outils de développement web tels que Node.js, JavaScript, Vue.js (conception d'interface graphique Web) et la mise en place d'algorithmes d'évitement de collision basé uniquement sur les données AIS.

### Anglais :

My second-year engineering school internship took place in the company KIETTA, which is specialized in the field of the oil and gas industry, more specifically in marine seismic acquisition.

During this internship I carried out work on an AIS receiver (Autonomous Identification System), an aid to navigation device. The goal was the integration of this receiver in an Unmanned Surface Vehicle (USV, precisely RAV) more precisely in the command control system. To do so, I used the tools and means used within the company and other means (like CPA notion) to receive, decode and process AIS data. This involves, in particular, the use of Raspberry Pi computer, web development tools such as Node.js, JavaScript, Vue.js (Web graphical interface design) and the implementation some collision avoidance algorithms based on the AIS data.

### Mots-Clés :

AIS, USV/RAV, Node.js, mqtt, CPA.

## SOMMAIRE

ENGAGEMENT DE NON PLAGIAT .....	2
REMERCIEMENTS .....	3
RESUME ET MOTS-CLES .....	4
INTRODUCTION .....	6
I. L'ENTREPRISE.....	7
I.1 Présentation .....	7
I.2 Structure .....	7
I.3 Activité .....	8
I.4 Les RAV (Recording Autonomous Vessel) chez KIETTA .....	10
II. TRAVAUX D'INITIATION.....	14
II.1 Initiation aux outils de développement web utilisés chez KIETTA.....	14
II.2 TOPO sur l'installation des logiciels et modules nécessaires au codage en Node.js, sous Raspberry Pi 4B, pour ce stage .....	17
III. L' AIS .....	17
III.1 Présentation du dispositif.....	17
III.2 Branchement.....	20
III.3 Décodage des trames AIS .....	24
IV. TRAITEMENT DES DONNEES AIS.....	29
IV.1 Modules Node.JS utilisés.....	29
IV.2 Notion de Closest Point of Approach (CPA) .....	32
IV.3 Recherche de paramètres supplémentaires d'aide à la prise de décision .....	38
IV.4 Algorithmes d'évitement de collision .....	45
IV.5 Simulateur développé pour tests d'évitement de collision.....	51
Les éléments de base .....	51
IV.5.1 Algorithme d'évitement de collision, cas « hors mission » .....	54
IV.5.2 Algorithme d'évitement de collision, cas d'un suivi de tronçons « en mission » .....	59
V. PERSPECTIVES.....	61
V.1 Restant à faire.....	61
V.2 A améliorer .....	62
CONCLUSION .....	64
BIBLIOGRAPHIE .....	65
GLOSSAIRE.....	66
ANNEXES .....	68

## INTRODUCTION

Dans l'optique de compléter ma formation de deuxième année d'école d'ingénieurs Systèmes Mécatroniques et Robotiques à SEATECH, j'ai effectué un stage assistant-ingénieur d'une durée de trois mois au sein de la société KIETTA, sur le site de La Seyne-sur-Mer. Cette entreprise qui s'inscrit dans le cadre de la recherche pétrolière offshore, se sert en partie de drones marins de surface autour desquels est développée une chaîne mécatronique complexe.

Je me suis intéressé à cette entreprise car les activités qui y sont développées correspondent à mon domaine d'étude et à mes centres d'intérêt, notamment, le pilotage de systèmes mécatroniques complexes, cas des drones marins de surface de l'entreprise KIETTA.

Ce stage consistait en l'intégration d'un récepteur AIS dans un drone de surface. Ceci du fait que l'entreprise compte rendre ses drones de plus en plus autonomes. Il était alors question d'intégrer l'AIS dans le flux de données d'un RAV et de pouvoir s'en servir en tant qu'aide à la navigation, pour un évitement de collision principalement.

Pour ce faire, je me suis penché en premier sur le dispositif, notamment son fonctionnement, puis j'ai travaillé sur des paramètres pouvant découler des données AIS et permettant une prise de décision effective. Enfin, j'ai effectué des tests sur mes paramètres et algorithmes d'évitement de collision sur un simulateur conçu sous Vue.js.

Les tâches réalisées, les méthodes employées ainsi que les résultats sont présents dans ce rapport.

## I. L'ENTREPRISE

### I.1 Présentation

L'entreprise Kietta a été créée en 2009. C'est une société à actions simplifiées (SAS) dont le siège est situé à Marseille au 165 avenue du Prado. Elle est divisée en deux sites distincts :

Un site à Marseille, où se trouve la direction ainsi que toutes les activités liées à la QHSE et à la géophysique.

Un site à la Seyne-sur-Mer situé dans la zone portuaire de Brégaillon afin d'avoir un accès permanent à la mer en raison de son activité. C'est là où est stockée une très grande majorité du matériel nécessaire aux missions (les RAV et autres).

Les employés travaillant à la Seyne ont donc un accès direct au matériel et peuvent ainsi effectuer des tests ou des réglages. Ceux de Marseille se rendent à la Seyne-sur-Mer lorsque cela est nécessaire.

Kietta rassemble une équipe d'experts dans le domaine des systèmes d'enregistrement sismique, des technologies de positionnement et de navigation, des navires marins autonomes, des télécommunications, de la robotique, des sous-systèmes de contrôle de profondeur automatisés et de la gestion de l'alimentation.

Ensemble, ils ont développé le premier système **FreeCable**, qui a été déployé avec succès à grande échelle en 2015. Kietta détient de nombreux brevets qui sont à la base de la technologie du système FreeCable.

Le système FreeCable est conçu pour fonctionner 24 heures sur 24, toute l'année. Kietta a conçu et mis en place des systèmes sophistiqués qui soutiennent l'équipe opérationnelle dirigée par des professionnels expérimentés couvrant la logistique, le HSE, la gestion des équipes, et la maintenance.

### I.2 Structure

Aujourd'hui KIETTA compte moins de 10 salariés, mais l'entreprise compte augmenter le nombre d'employés sous peu en vue du bon déroulement des projets à venir qui nécessitent une main d'œuvre importante.

Kietta s'appuie aussi en grande partie sur la sous-traitance car plusieurs domaines sont impliqués dans leurs activités. Les RAV nécessitent par exemple d'être entretenus sous différents aspects : de la coque, aux différents compartiments internes. C'est ainsi que j'ai pu assister à l'intervention de plusieurs entreprises lors de la maintenance des drones. Il s'agit par exemple de :

CEGELEC : travail sur les coques, vannes, filtres, etc. des RAV.

FOSELEV : Carénage, grutage des RAV (sortie et mise à l'eau), etc.

PLS : nettoyage des coques (RAV), antifouling, etc.

Durant mon stage, j'ai travaillé sur le site de la Seyne-sur-Mer au sein du département Opération.

Ce département sous la responsabilité de Mr Thierry Leon (yard Manager) est basé à la Seyne-sur-Mer. Il prend en compte la gestion logistique, des missions en mer, des tests et maintenances des RAV. Aidés par moment, comme j'ai pu le constater durant ce stage, par d'autres employés, ce département est composé principalement d'un petit groupe de 3 personnes :

- Mr Thierry Leon responsable logistique et du site de la Seyne.
- Mr Hilaric Cadel chef Observateur et spécialiste en sismique.
- Mr Hervé Souche : ingénieur génie électrique responsable du maintien en condition opérationnel.

Mr Laurent Velay est le Directeur Général de l'entreprise.

## I.3 Activité

Kietta est une entreprise spécialisée dans la recherche pétrolière offshore (en mer). Elle utilise plus précisément la recherche par prospection sismique. Cette méthode de recherche consiste à envoyer des



ondes en direction du sol grâce à des canons à air comprimé et on recueille les échos sur des hydrophones flottants (flûtes). La réflexion de ces ondes sera ensuite analysée afin de connaître la nature des couches géologiques de la zone étudiée. Par la suite, des récepteurs enregistrent les ondes qui ont traversé des couches terrestres. C'est la méthode utilisée par la majorité des entreprises de ce secteur. Voir schéma ci-dessous.

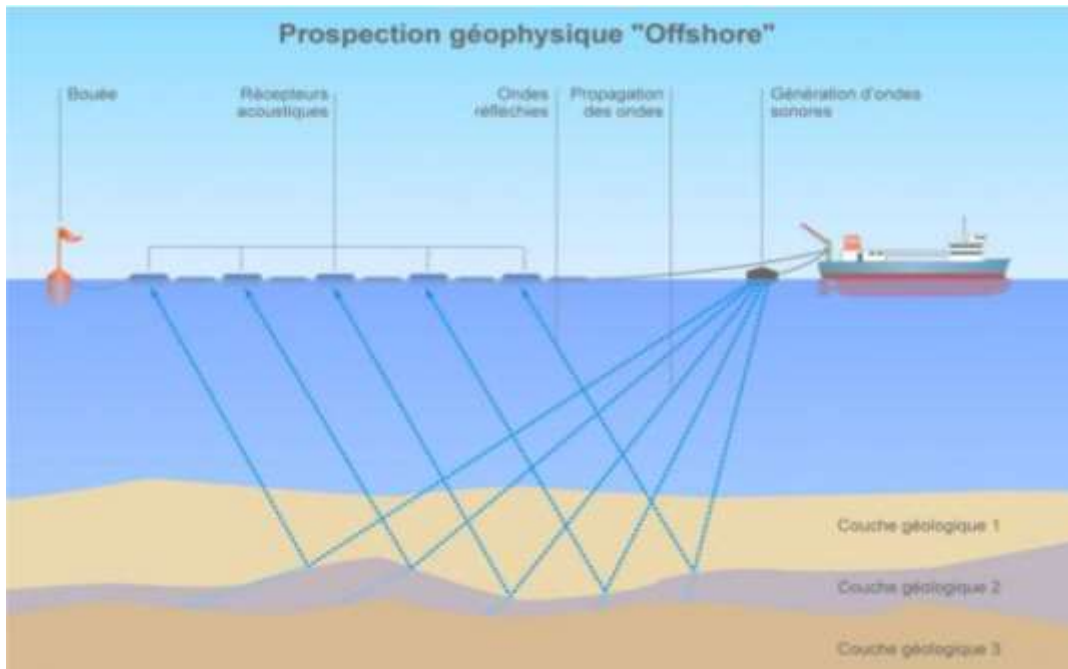


Figure 1 : Prospection sismique

Toutefois, les différents systèmes d'acquisition sismique marine souvent proposés aux clients ont chacun des limites : les flûtes tractées offrent une grande productivité, mais souffrent des limites d'un réseau de prise de vue et d'enregistrement essentiellement linéaire, et du bruit généré autour de la flûte à une vitesse moyenne de 5 nœuds. Les systèmes de fond marin utilisant des câbles ou des nœuds fournissent des données d'azimut complet supérieures, mais leur faible productivité entraîne des coûts très élevés et de longues durées de levé.

D'où le concept de **FreeCable**, propre à KIETTA.

Le système **FreeCable** est le résultat de 5 ans de concepts, de conceptions, de prototypes et d'essais à grande échelle. Tout a commencé par le défi d'offrir aux clients des données de meilleure qualité sans les aspects chronophages (activités ou des processus qui prennent beaucoup de temps) et onéreux des systèmes d'acquisition au fond de la mer.

Le système FreeCable s'inspire à bien des égards de l'acquisition sismique terrestre, dans laquelle les récepteurs sont indépendants de la source, et des données d'azimut complet et de long décalage peuvent être obtenues.

L'idée est de localiser une série de câbles parallèles à une profondeur contrôlée sous le bruit des vagues et de faire naviguer un navire source sur des lignes de tir perpendiculaires aux câbles.

La robotique rend cela possible, offrant un moyen rentable et flexible de déplacer les câbles en position, et sert de générateur d'énergie et d'unité d'enregistrement/transmission pour les récepteurs des câbles et les sous-systèmes de contrôle de profondeur.

L'entreprise utilise une méthode qui permet d'enregistrer sur 4 composantes, là où la plupart des concurrents n'en utilisent qu'un ou deux. Les données obtenues sont vendues par la suite aux clients de Kietta, c'est-à-dire à des compagnies pétrolières, qui les analysent afin de déterminer s'il y a du pétrole exploitable ou non dans la zone inspectée.

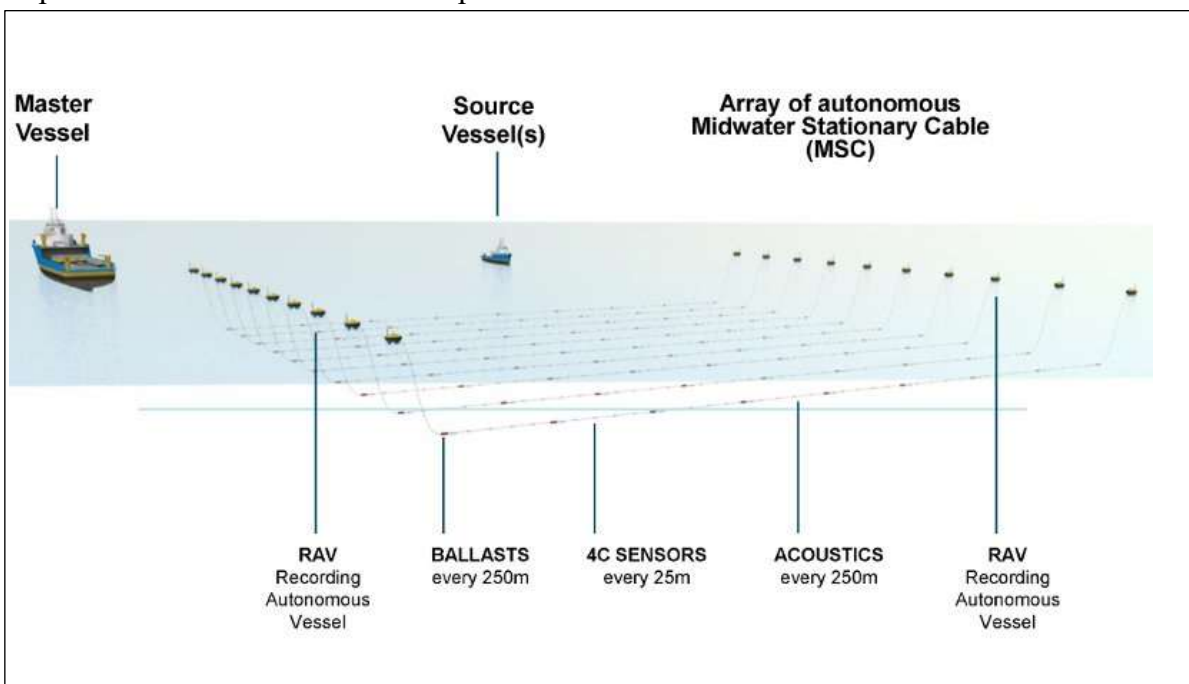


Figure 2 : Système FreeCable

## I.4 Les RAV (Recording Autonomous Vessel) chez KIETTA

### ▪ Définition :

Un **RAV** est un navire de petite taille, piloté à distance, dont les fonctions principales sont de positionner géographiquement le câble sismique à l'emplacement souhaité, d'exercer une tension sur le câble pour le resserrer, de contrôler la profondeur d'immersion nominale de l'entrée des extrémités de câble, pour générer de l'énergie pour les composants électroniques intégrés dans le câble et pour servir de relais de surface à des fins de télécommunication. Deux RAV par câble sont utilisés : le RAV en amont est appelé RAV de tête, le RAV en aval est appelé RAV de queue.

### ▪ Description

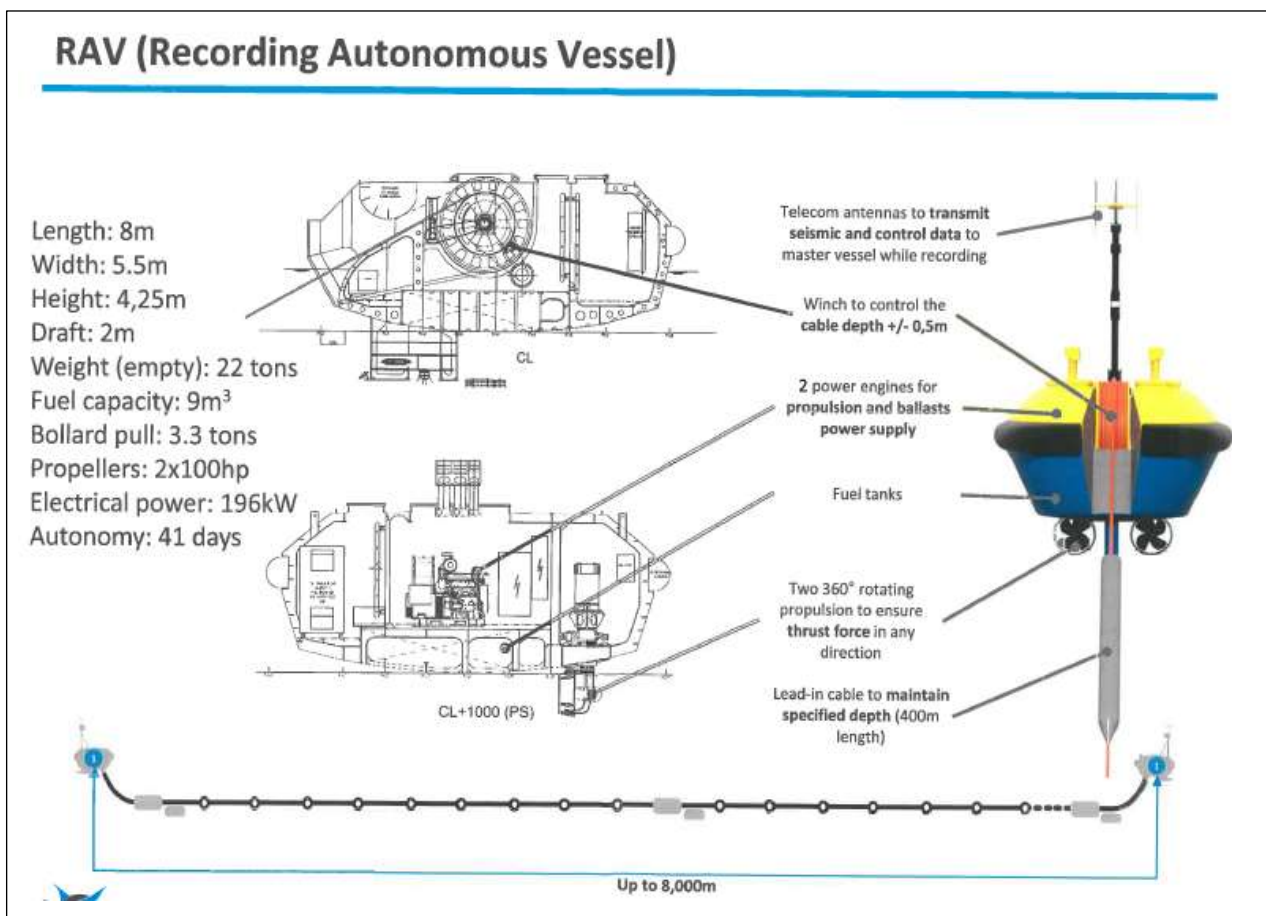


Figure 3 : Description du RAV

Le RAV est équipé de deux propulseurs azimutaux dans une coque simple en forme de bateau, ce qui lui confère d'être très maniable et navigable.

Dans le pont principal du RAV, il y'a un "treuil MSC (Midwater Stationary Cable) avec dispositif de "dévidage" pour la manipulation de la section d'entrée du MSC (câble lead-in) de 400 m de long rangé sur le tambour du treuil.

L'intérieur du navire est divisé en différents compartiments étanches comprenant :

plusieurs machines telles que groupe électrogène, système d'échappement, commutateur électrique et un moteur électrique de propulsion, etc.

On note en particulier le compartiment électronique, d'une grande importance, situé à l'arrière du navire. Ceci, pour les commandes informatiques du navire, le système électronique de commande MSC, le matériel de radiocommunication et d'autres produits électroniques.

Comme équipements on peut citer le VCC qui est l'ordinateur de contrôle du navire (comprenant la CPU, les E / S numériques et analogiques et le matériel réseau), le récepteur GPS, le contrôleur de ballast, l'équipement de commutation de réseau, le modem satellite iridium, l'équipement de modem radio (WIFI, VHF, UHF), l'AHRS, l'INS, l'AIS ...etc.

#### ▪ Fonctionnement

Le fonctionnement du RAV est automatisé et ce, à l'aide des systèmes d'automatisation des navires conventionnels. Le contrôle à distance du RAV permet le contrôle de tous les équipements à bord (propulsion, treuil, mât, OFE, etc.) du début à la fin de l'arrêt des moteurs principaux.

Un RAV est interchangeable en service en tant que RAV de tête ou de queue, en fonction du courant et de la direction du trajet.

Le RAV est équipé d'un système diesel-électrique pour alimenter la propulsion électrique, des commandes basées sur des automates programmables et pour alimenter le câble sismique lui-même. Ses automates programmables sont au nombre de 7 et communiquent entre eux pour le pilotage du drone. Le système diesel est composé de deux groupes électrogènes qui produisent l'alimentation principale du navire.

Son système de propulsion électrique comprend deux propulseurs azimutaux équipés d'un moteur électrique chacun, tous les entraînements motorisés (les entraînements à fréquence variable (EFV)), ainsi que tous les éléments d'interface, les composants et les raccords nécessaires au bon fonctionnement de l'équipement. Leur puissance d'entrée nominale doit être au maximum de 100 kW à 1 800 tr / min pour chaque propulseur.

#### ▪ Système de contrôle et communication

Le navire est conçu pour fonctionner à distance sur une liaison de télémétrie radio numérique pendant les opérations sismiques normales. Tous les systèmes de contrôle du RAV sont accessibles via le protocole IP principal du réseau RAV. Les fonctions du navire telle que la navigation, la propulsion, la direction, la climatisation, les communications ainsi que la production et la gestion de l'énergie, sont contrôlées par un système à base d'automates programmables.

Les fonctions du navire sont contrôlées localement par :

- un réseau distribué de panneaux PLC (Programmable Logic Controller) à écrans tactiles sur chaque armoire électrique,
- Un panneau PC à écran tactile HMI centralisé monté dans le compartiment électronique

Les fonctions du navire sont contrôlées à distance par une interface informatique avec le VCC monté dans l'armoire OFE.

Le système IHM du RAV donne la possibilité de contrôler à distance la climatisation, le pilotage des propulseurs, la vitesse des hélices du propulseur etc..., mais aussi de lancer un test d'endurance.

Le panneau PC HMI à écran tactile est accessible depuis le réseau Ethernet :

Les automates ont une interface directe avec le VCC et le panneau PC tactile (qui a une interface avec le réseau Ethernet du RAV) fournit une liaison Ethernet MODBUS au VCC ; ainsi toutes les commandes des automates (par exemple le contrôle du treuil, mât, la commande de pilotage des propulseurs... etc.) doivent pouvoir passer par le VCC.

## II. TRAVAUX D'INITIATION

### II.1 Initiation aux outils de développement web utilisés chez KIETTA

Avant de rentrer dans le sujet de ce stage, j'ai été initié aux outils de développement logiciels nécessaires. Il s'agit, entre autres, de JavaScript, Node.js, HTML, CSS, Vue.js, Prime Vue, le protocole MQTT, le protocole WebSocket, écriture et lecture de fichiers avec extension **JSON** (JavaScript Object Notation).

- Un objet JSON (JavaScript Object Notation) est un format de données léger, facile à lire et à écrire, basé sur la notation des objets JavaScript. Il est composé de paires clé-valeur, où chaque clé est une chaîne de caractères et chaque valeur peut être de type différent.

Les fichiers au format JSON sont très utilisés dans le flux de données partagés entre drones et **CO** (contrôle/commande), d'où l'importance de s'y intéresser dès le départ.

- JavaScript : ce langage est principalement utilisé pour le développement côté client, ce qui signifie qu'il est exécuté dans le navigateur web de l'utilisateur.
- Node.js : Node.js est un environnement d'exécution côté serveur construit sur le moteur JavaScript V8 de Google Chrome. Il permet d'exécuter du code JavaScript sur le serveur plutôt que dans un navigateur web.
- HTML : C'est un système standardisé de balisage des fichiers texte pour obtenir des effets de police, de couleur, de graphisme et d'hyperlien sur les pages du World Wide Web.
- CSS : Il s'agit d'un langage de feuille de style utilisé pour décrire la présentation d'un document écrit en HTML ou XML.

- Vue.js : Framework JavaScript open-source utilisé pour construire des interfaces utilisateurs interactives et réactives côté client.
- PrimeVue : Une bibliothèque de composants d'interfaces utilisateurs pour Vue.js.
- MQTT: Message Queuing Telemetry Transport. Il s'agit d'un protocole de messagerie léger, ouvert et basé sur la publication/abonnement, conçu pour les communications M2M (Machine-to-Machine) et IoT (Internet of Things).
- Protocole WebSocket : Le protocole WebSocket est un protocole de communication bidirectionnelle, en temps réel et basé sur le TCP (Transmission Control Protocol), qui permet une communication continue et permanente entre un navigateur web (client) et un serveur.

Après une prise en main de ces outils, la première tâche a été de concevoir une interface graphique simple qui permet de modifier les champs d'un fichier au format JSON. Ceci, afin de bien assimiler principalement l'utilisation des protocoles mqtt et WebSocket, des objets JSON, de Vue.js et Node.js. Cette interface ressemble à ceci :

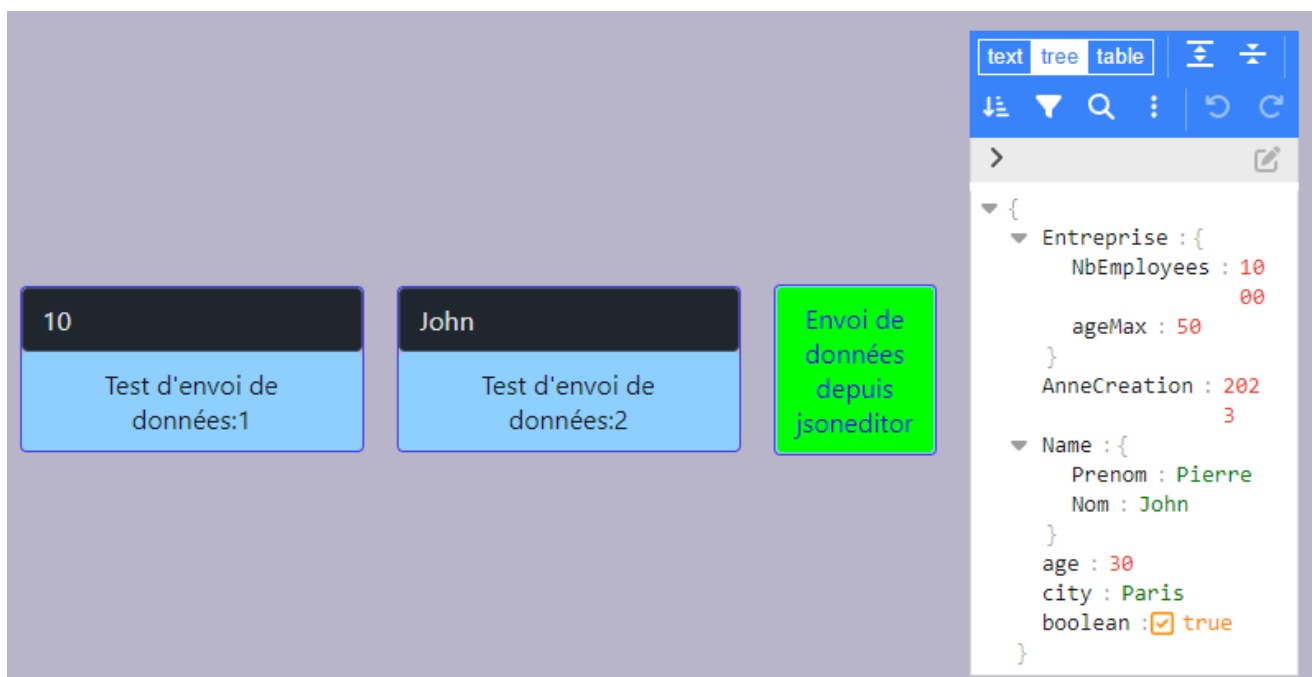


Figure 4 : Interface graphique d'initiation aux outils de développement Web

L'outil à l'extrémité droite est un éditeur JSON permettant de visualiser et de modifier des données au format JSON de manière conviviale.

Dans le champ **Test d'envoi de données :1**, je modifie le nombre d'employés par la nouvelle valeur entrée. Il s'agit de modifier le dernier champ du topic " **Entreprise/NbEmployes**" (voir figure ci-dessus). Les topics MQTT étant des chaînes de caractères utilisées pour identifier les canaux de

communication auxquels les clients MQTT peuvent s'abonner pour recevoir des messages ou publier des messages.

Le champ de droite permet de modifier le champ 'Nom', du topic "**Name/Nom**" (voir figure ci-dessus).

Une idée des tâches réalisées en internes :

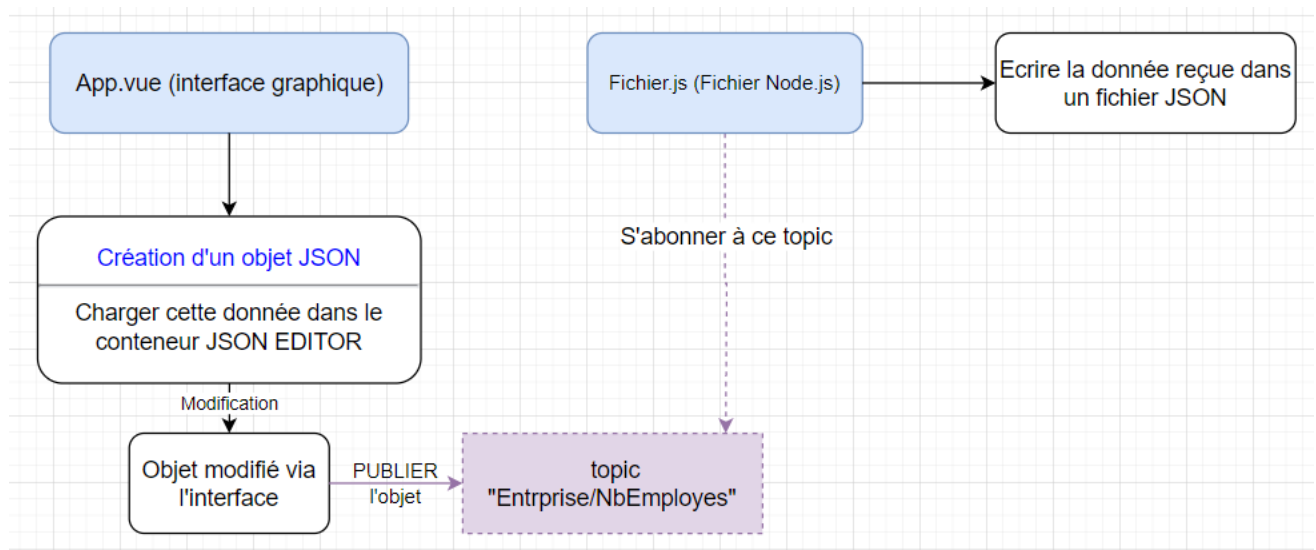


Figure 5: Modification d'un fichier JSON



```
65 // Pour charger une donnée json dans le conteneur jsoneditor
66 const container1 = document.getElementById('jsoneditor');
67 editor = new JSONEditor({target:container1,props:{}});
68
69 const initialData={
70   json:{
71     "Entreprise": {
72       "NbEmployes": 1000,
73       "ageMax": 50,
74     },
75   },
76   "AnneCreation": 2023,
77   "Name": {"Prenom":"Pierre","Nom":"John"},
78   "age": 30,
79   "city": "Paris",
80   boolean : true,
81 }
82 }
83
84 editor.set(initialData);
85 })
86 function publishNbEmployees() {
87   const modifiedData = editor.get();
88   const client = mqtt.connect('ws://192.168.1.76:8083/');
89   let topic1='Entreprise/NbEmployes';
90   client.publish(topic1, JSON.stringify(modifiedData));
91 }
```

Figure 6 : Modification d'un fichier JSON- Portion de code sous Vue.js

## II.2 TOPO sur l'installation des logiciels et modules nécessaires au codage en Node.js, sous Raspberry Pi 4B, pour ce stage

Un certain nombre de logiciels ont été installés sur la carte Raspberry Pi 4B utilisée.

Je fournis la démarche à suivre en [Annexe 1](#).

## III. L'AIS

### III.1 Présentation du dispositif

Le Système d'Identification Automatique de Navire (AIS) est un système d'informations de navires et de positionnement. Il permet aux navires qui en sont équipés d'effectuer automatiquement et dynamiquement des mises à jour de leur position, leur vitesse, leur cap et d'autres informations telles que l'identité du navire, son type, son numéro **MMSI (Maritime Mobile Service Identity ; numéro unique à neuf chiffres attribué à chaque navire équipé d'un système de communication radio)**, la nature du fret, les ports de départ et d'arrivée, les dimensions du navire, etc. Il partage ces données avec d'autres navires qui ont le même équipement. Ainsi, ces navires annoncent périodiquement leur données dynamiques (position, cap, vitesse, etc.) et statiques (nom du navire, type du navire, etc.), en utilisant la technologie TDMA (Time Division Multiple Access) similaire à la façon dont les téléphones portables le font pour éviter les interférences mutuelles. Dans un réseau TDMA, le temps est divisé en intervalles fixes appelés "slots" (créneaux). Chaque utilisateur du réseau se voit attribuer un ou plusieurs de ces créneaux, pendant lesquels il peut transmettre ses données de manière séquentielle.

L'AIS enregistre la position et les déplacements grâce à son système GPS/GNSS embarqué. A intervalles réguliers, des messages AIS sont transmis et d'autres sont reçus en provenance de navires équipés d'un émetteur AIS, sur deux canaux Très Haute Fréquence (VHF) alternativement. Les données AIS sont en effet émises sur deux fréquences radio dédiées : AIS 1 : **161,975 MHz**, canal 87B et AIS 2 : **162,025 MHz**, canal 88B.

Il existe plusieurs types d'appareils AIS. Ils sont listés ci-dessous :

- **Emetteurs-récepteurs de classe A.**

Il s'agit du même type que les émetteurs-récepteurs de classe B, à la différence qu'ils sont conçus pour être installés dans des navires de taille importante tels que les paquebots et les cargos. Les émetteurs-récepteurs de classe A émettent un signal VHF à une puissance plus élevée que les émetteurs-récepteurs de classe B, ce qui permet au signal d'être intercepté par des navires plus éloignés, ainsi que des transmissions plus fréquentes. Les émetteurs-récepteurs de classe A sont obligatoires sur tous les navires de plus de 300 tonneaux bruts effectuant des traversées internationales, ainsi que dans certains types de navires de passagers soumis au mandat **SOLAS** (International Convention for the Safety of Life at Sea).

Les transpondeurs AIS Classe A ont une **puissance d'émission jusqu'à 12.5 watts** et transmettent généralement les informations **toutes les 2 à 12 secondes** suivant la vitesse du bateau. Leur portée d'émission peut atteindre **20 miles ou plus** si une antenne appropriée est utilisée. [\[Réf. 2\]](#)

- **Émetteurs-récepteurs de classe B.**

Ils sont similaires aux émetteurs-récepteurs de classe A, mais sont normalement moins chers en raison d'exigences moins strictes en termes de performance. Les émetteurs-récepteurs de classe B transmettent à une puissance moindre et connaissent une fréquence de transmission moins élevée que les émetteurs-récepteurs de classe A.

Les AIS de classe B fonctionnent à une puissance inférieure à **2 watts** et émettent environ **toutes les 30 secondes** avec une portée d'émission qui dépendra de l'installation de l'antenne. Certains appareils affichent une classe B+, avec une puissance de 5 W et une fréquence de 5 secondes.

[\[Réf. 2\]](#)

La différence entre un récepteur AIS et un transpondeur AIS est que le récepteur AIS ne fait que recevoir les données des navires équipés d'un transpondeur AIS, tandis que le transpondeur AIS peut à la fois émettre et recevoir les données. Concrètement, le récepteur AIS permet donc de visualiser le trafic maritime autour de soi, mais pas de transmettre sa propre identité, position, vitesse et route aux autres navires. Le transpondeur AIS permet en revanche de se faire identifier par les autres navires et les stations côtières, ce qui augmente la sécurité et la prévention des collisions.

- **Paramétrage d'un AIS**

Pour paramétrer un AIS, il faut d'abord obtenir un numéro MMSI, Maritime Mobile Service Identity, auprès de l'autorité compétente. En France, c'est **l'Agence Nationale des Fréquences** ou ANFR qui distribue les numéros MMSI aux navires équipés d'un AIS ou d'une radio VHF. Le MMSI est un code numérique de 9 chiffres qui permet d'identifier le navire et son propriétaire.

Il faut enfin configurer le transpondeur à la première mise en route, en entrant le numéro MMSI, nom du navire, type de navire et sa taille. **Le paramétrage est définitif**, il sera impossible de modifier les données enregistrées par la suite sans l'intervention du fabricant. La dernière étape consiste à vérifier que l'AIS reçoit bien le signal GPS et qu'il émet et reçoit correctement les données des autres navires et des stations. [\[Réf. 3\]](#)

- **Quelques avantages de l'AIS**

L'AIS a pour rôle d'éviter les collisions en mer et d'assurer un suivi de route en temps réel. Lorsque la visibilité est faible ou les conditions de navigation dangereuses, l'AIS peut être d'une grande aide car il fonctionne comme un radar en temps réel.

Pour les activités de pêche, les transpondeurs AIS ne servent pas qu'à éviter les collisions mais permettent aussi aux pêcheurs d'alerter les bateaux aux alentours de la pose de filet et suggérer des modifications de route.

Enfin pour les activités de loisirs, l'AIS permet de réaliser un suivi de la position d'un professionnel ou d'un plaisancier en navigation, directement sur Internet. Des sites tels que <https://www.marinetraffic.com/en/ais/home/centerx:8.800/centery:-0.744/zoom:13> permettent à n'importe qui de visualiser les infos AIS d'un skipper par exemple.

### ▪ Les inconvénients

Le principal inconvénient est que les bateaux n'émettent pas tous un signal AIS. Les bateaux de plaisances n'ont aucune obligation de cet ordre. Seules les unités de commerce de plus de 300 tonnes émettent à coup sûr : cargos et autres navires de commerce.

## III.2 Branchement

Pour ce stage, l'AIS utilisé est l'AIS TR-200 de la société ADVANSEA. Il est utilisé en réception uniquement.



Figure 7 -a: AIS TR200-ADVANSEA

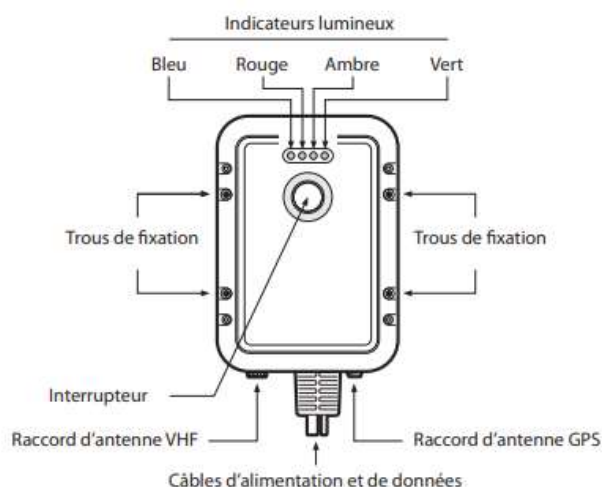


Figure 7-b: Aperçu de l'AIS TR200

Les câbles raccordés à l'AIS TR-200 assurent les connexions suivantes :

- Alimentation électrique
- Port de données NMEA0183 pour se connecter aux Chartplotter (dispositif électronique utilisé pour afficher les messages AIS reçus en tant que navires).
- RS232 pour se connecter à un PC (Raspberry Pi 4B pour mon cas).
- Entrée de l'interrupteur externe (Non utilisé).

Par ailleurs, il existe deux raccords supplémentaires pour l'antenne GPS et l'antenne VHF.

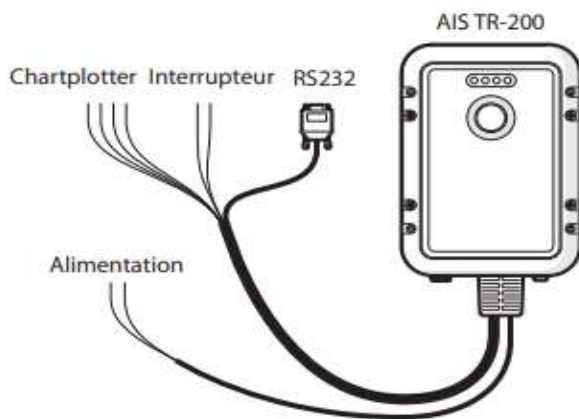


Figure 8 : Câbles raccordés à l'AIS TR 200

- J'ai branché l'AIS, conformément à cette image. Seuls l'interrupteur et le câble du Chartplotter qui sert à afficher les messages AIS reçus en tant que navires n'ont pas été connectés. En effet, le but est de réaliser l'affichage des navires soi-même et de l'intégrer au système de contrôle/commande.
- Le câble RS232 a été connecté à une carte Raspberry pi 4B via un LOT de câbles ruban LL31941 SCSI CSA AWMIA 105C 300V FT-1 auquel on connecte des connecteurs DB-9 femelle. En effet, le câble RS232 ne permet pas une connexion directe à la carte Raspberry Pi : il faut des câbles adaptés pour une connexion aux GPIO (General Purpose Input Output) de la carte.
- L'alimentation est une source de tension de 12 V.
- Les raccords d'antennes VHF et GPS ont été connectés (les antennes étaient déjà installées).

Je fournis en [Annexe 3](#) les images du câblage.



Figure 9-a: LOT de câbles ruban  
LL31941 SCSI CSA AWMIA  
105C 300V FT-1



Figure 9-b :  
Raccord\_VHF\_PL259



Figure 9-c :  
Raccord\_VHF\_PL259



Figure 9-d : Connecteur  
femelle DB-9 femelle.



Figure 9-e : Connecteur DB-9 mâle.

- J'ai travaillé dans le drone 1, en ce sens que le flux de données en provenance de l'AIS a été lié au circuit de données de ce drone. Nous avons alors disposé le matériel afin de créer la configuration mettant en lien le drone 1 (RAV 1) et le contrôle/commande (CO). Nous avons cependant, remplacé la communication Wi-Fi entre les deux (car en condition réel la communication RAV-CO se fait à distance), par une connexion filaire RJ45.



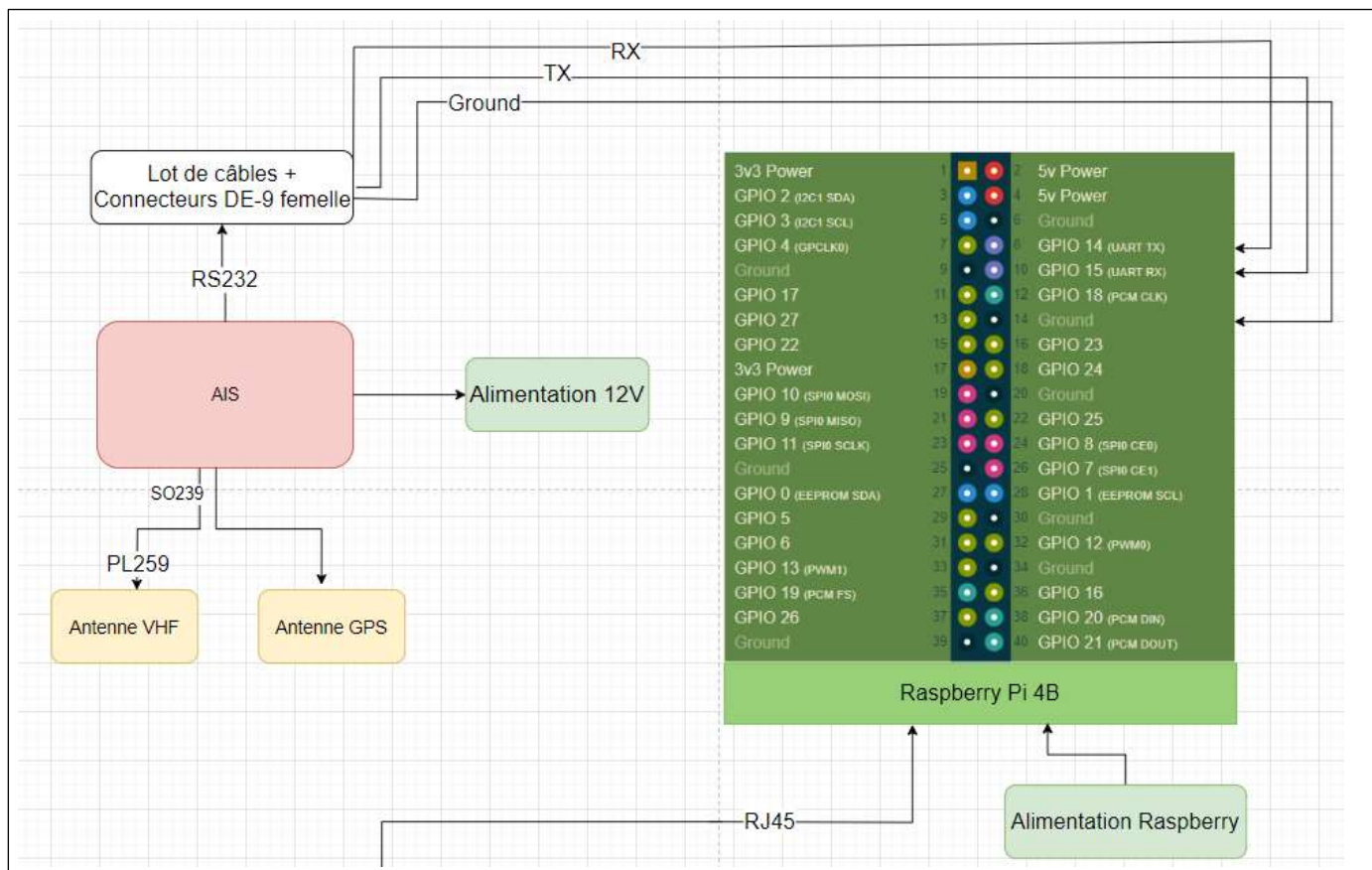


Figure 10-a : Aperçu des connexions AIS/Drone durant le stage.

CO : système de contrôle-commande.

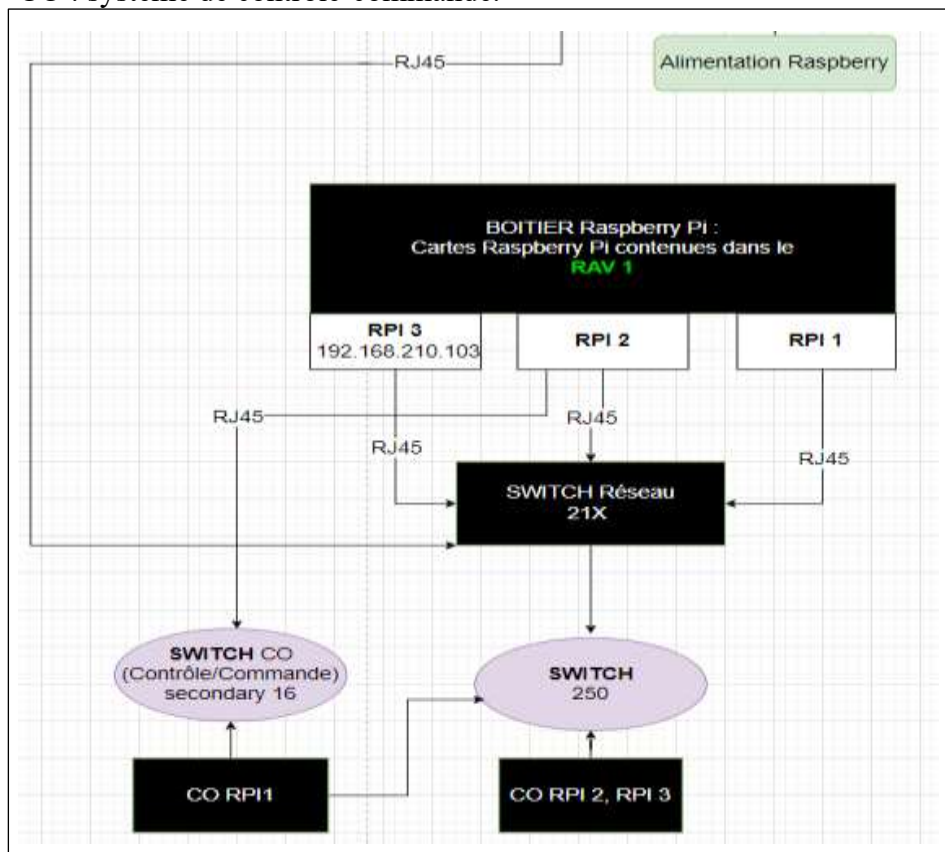


Figure 10-b : Aperçu des connexions AIS/Drone établie durant le stage.

Je fournis en [ANNEXE 4](#) un aperçu des connexions RAV - Contrôle/Commande en situation réelle.

### III.3 Décodage des trames AIS

Une fois l'AIS branché, j'ai configuré le port série de la carte Raspberry Pi pour recevoir les données et les afficher dans un premier temps : il s'agit des données spécifiques aux AIS (concernant les navires) et des données NMEA, format de données utilisé pour les instruments électroniques d'aide à la navigation.

- La norme **NMEA 0183** est une spécification pour la communication entre équipements marins, dont les équipements GPS. Elle est définie et contrôlée par la **National Marine Electronics Association** basée à Severna Park au Maryland (États-Unis d'Amérique).

Il existe plus d'une trentaine de trames différentes ayant chacune leur propre syntaxe. Les premiers caractères transmis (les 5 caractères suivant le symbole \$, début de trame) donnent des renseignements sur le type d'équipement utilisé pour la géolocalisation et sur le type de trame utilisée : Par exemple : **\$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,,0000\*0E** est une trame GPS de type GGA.



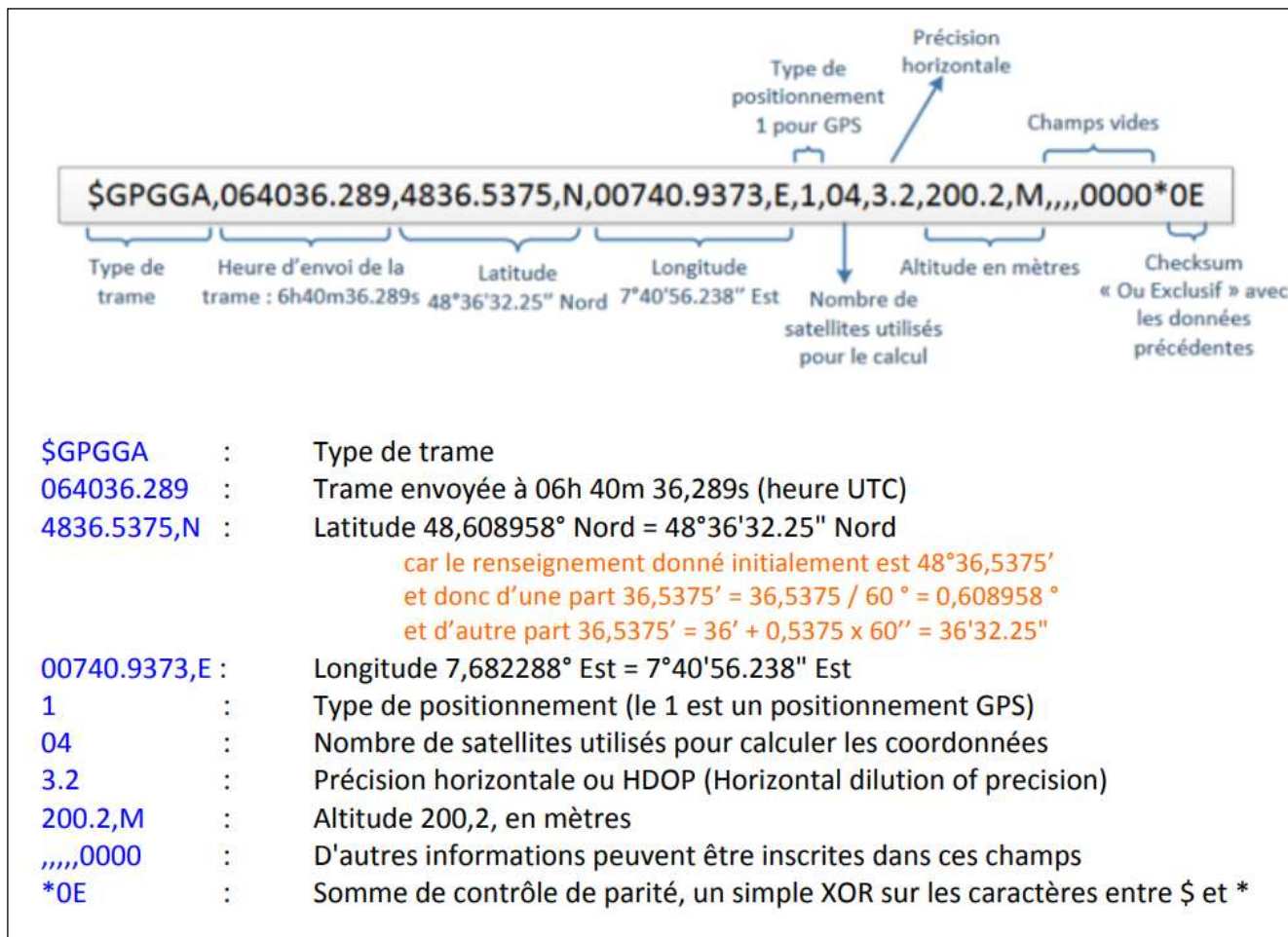


Figure 11 : Trame NMEA-Type GGA [Réf. 5].

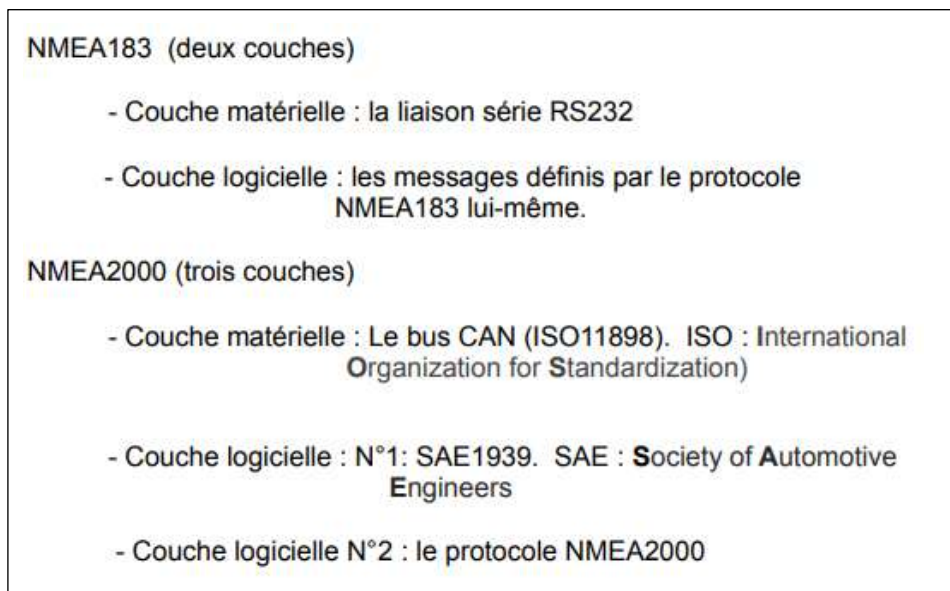


Figure 12 : Protocole NMEA.

Les données NMEA envoyées par l'AIS sont issues du protocole NMEA183.

- Outre les trames NMEA, l'AIS envoie des types de données qui lui sont spécifiques. Il s'agit des données :

**AIVDM** (AI pour 'AIS') : Rapport de position des cibles.

**AIVDO** : Compte rendu de position concernant son propre bateau.

Ci-dessous un paquet de données AIVDM typique :

**!AIVDM,1,1,,B,177KQJ5000G?to`K>RA1wUbN0TKH,0\*5C**

La signification des différents champs des trames !AIVDM est fournie en [ANNEXE 5](#)

Il existe plusieurs types de trames !AIVDM selon la nature des données partagées :

C'est ainsi que des données !AIVDM de type 1, 2 ou 3 fournissent des **informations de navigation** (cap, vitesse, etc.) et les données de type 24, des **données statiques** (nom du bateau, type du navire, dimensions, etc.). Le type est donné dans la trame envoyée par l'AIS.

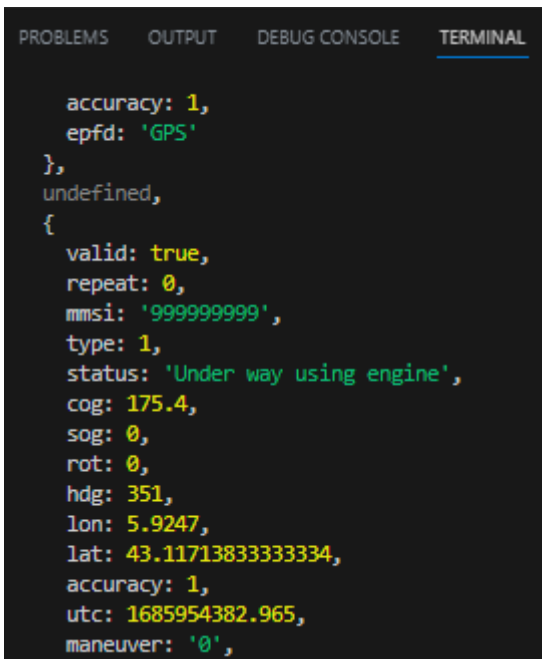
Parameters	Description
<b>timestamp_pretty</b>	time in format dd/mm/yyyy hh:mm:ss
<b>timestamp</b>	Unix time stamp (seconds since 01/01/1970 00:00:00)
<b>msgid</b>	The AIS message the signal was issued
<b>targetType</b>	AIS type A or B
<b>mmsi</b>	MMSI number of the ship
<b>lat</b>	Latitude in decimal format
<b>long</b>	Longitude in decimal format
<b>posacc</b>	Position accuracy
<b>SOG</b>	Speed Over Ground in 0.1 knot
<b>COG</b>	Course Over Ground in 0.1°
<b>shipType</b>	Ship type of the vessel
<b>dimBow</b>	The dimension between the AIS transmitter and the bow of the ship in meters
<b>draught</b>	Draught of the ship in 0.1 meter
<b>dimPort</b>	The dimension between the AIS transmitter and the port side (left) of the boat in meters
<b>dimStarboard</b>	The dimension between the AIS transmitter and the starboard side (right) of the boat in meters
<b>dimStern</b>	The dimension between the AIS transmitter and the stern of the ship in meters
<b>month</b>	Month the signal was issued (between 1 and 12)
<b>week</b>	Week number the signal was issued
<b>imo</b>	IMO number of the ship
<b>country</b>	Flag of the ship

Figure 13 : Quelques données fournies par l'AIS.

- J'ai procédé au décodage des trames AIS afin qu'elles soient interprétables. Le codage se fait en Node.Js.

Pour décoder les trames AIS j'ai eu besoin d'un décodeur : **ais-stream-decoder**. Il s'agit d'un module fourni par le gestionnaire de paquets officiel de Node.js. Le code de décodage est fourni en [ANNEXE 6](#).

Un exemple des trames décodées :



```
accuracy: 1,
epfd: 'GPS'
},
undefined,
{
  valid: true,
  repeat: 0,
  mmsi: '999999999',
  type: 1,
  status: 'Under way using engine',
  cog: 175.4,
  sog: 0,
  rot: 0,
  hdg: 351,
  lon: 5.9247,
  lat: 43.11713833333334,
  accuracy: 1,
  utc: 1685954382.965,
  maneuver: '0',
```

Figure 14 : Trames AIS décodées

Dans la suite je n'ai retenu que certains champs qui sont utiles et qui interviennent très souvent dans les trames reçues. Il s'agit de :

**COG** : représente le cap (direction) du navire par rapport au nord vrai (0 degré) mesuré en degrés. C'est la direction dans laquelle le navire se déplace par rapport au sol et non par rapport à l'eau.

**SOG** : représente la vitesse du navire par rapport au sol en nœuds (1 nœud  $\Leftrightarrow$  1 mille marins par heure).

**LAT** : latitude.

**LON** : longitude.

**HDG** (Healing) : contrairement au COG, le HDG est la direction vers laquelle le navire pointe, indépendamment de sa vitesse ou de la dérive causée par le courant ou le vent.

Les autres champs n'interviennent pas toujours dans les trames ou sont peu pertinents pour mes calculs.

J'ai effectué le décodage dans un module Node.js **"ais.js"**.

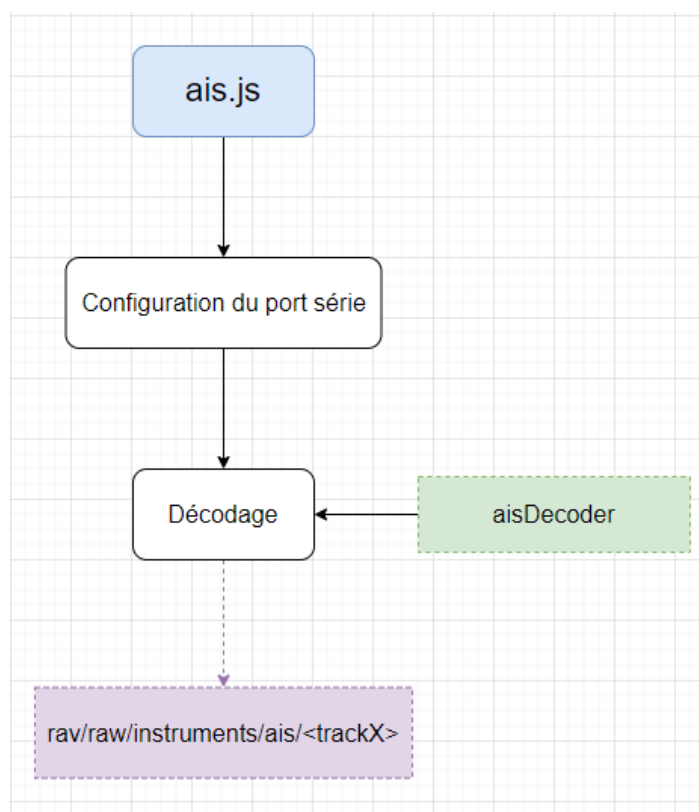
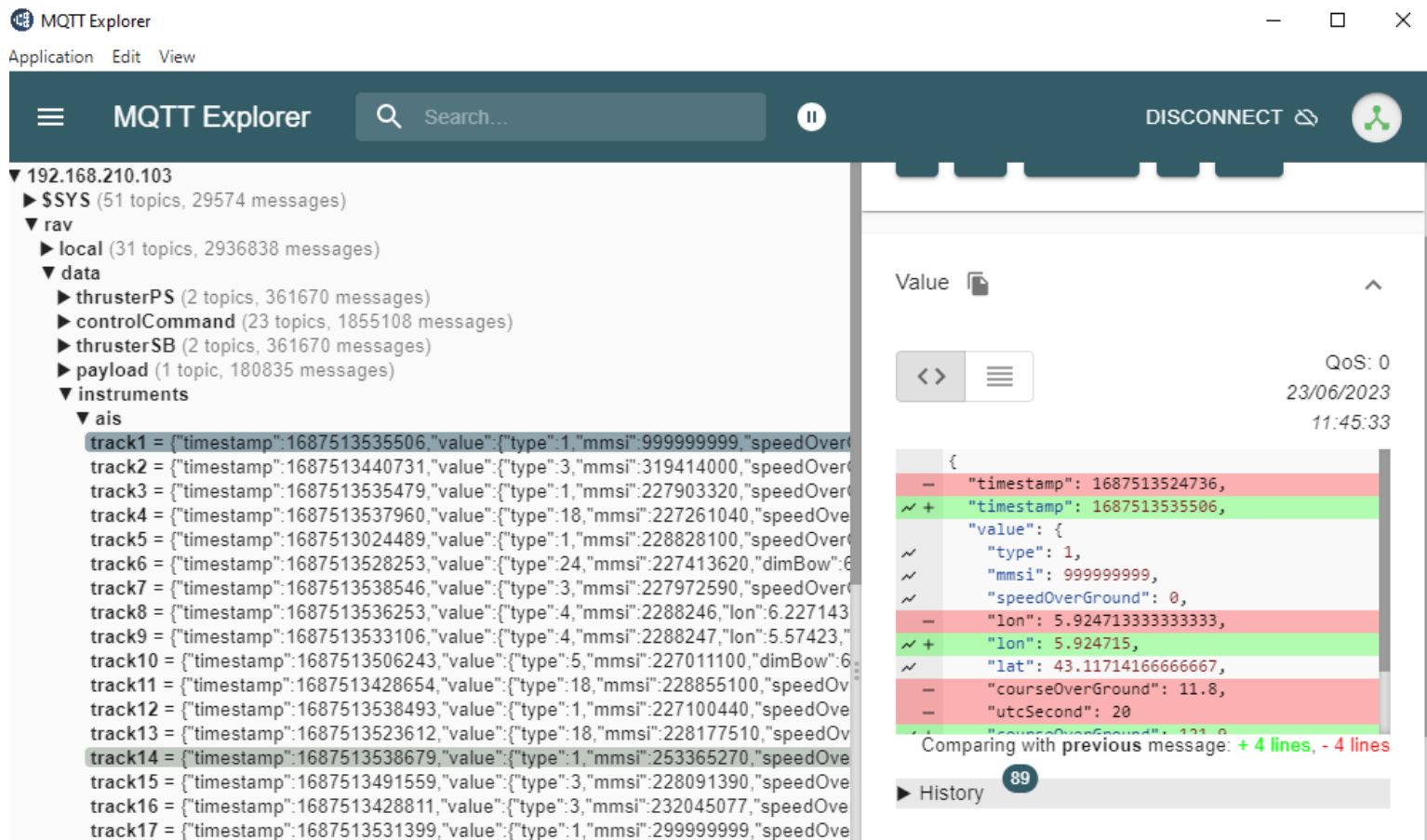


Figure 15 : Module de décodage des trames AIS –"ais.js"

Les trames décodées sont ensuite envoyées au **broker MQTT** (intermédiaire qui facilite la communication entre les clients MQTT), sous le topic **rav/raw/instruments/ais/<trackX>**.

Les données peuvent être visualisées sur le logiciel **MQTT Explorer**. Chaque trame possède un identifiant unique **"trackX"** :



The screenshot shows the MQTT Explorer application interface. On the left, a tree view displays the topic structure: **192.168.210.103** (51 topics, 29574 messages) → **SSYS** (31 topics, 2936838 messages) → **data** (2 topics, 361670 messages) → **thrusterPS** (23 topics, 1855108 messages) → **thrusterSB** (2 topics, 361670 messages) → **payload** (1 topic, 180835 messages) → **instruments** (1 topic, 180835 messages) → **ais**. The main pane displays a list of decoded AIS messages (track1 through track17) as JSON objects. The right pane shows a detailed view of a selected message (track14) with its JSON structure: `{ "timestamp": 1687513538679, "value": { "type": 1, "mmsi": 253365270, "speedOverGround": 0, "lon": 5.924713333333333, "lat": 43.11714166666667, "courseOverGround": 11.8, "utcSecond": 20 } }`. The interface also includes a search bar, a disconnect button, and a history panel at the bottom right.

Figure 16 : Trames décodées, visualisées sur MQTT Explorer

Je m'abonne ensuite à ce topic pour récupérer les trames décodées et les traiter.

## IV. TRAITEMENT DES DONNEES AIS

### IV.1 Modules Node.JS utilisés

Pour le traitement des données AIS, je me sers dans un premier temps, d'un module (fichier) Node.js nommé **"aisData.js"**. A travers ce module, je récupère les trames AIS décodées et je vérifie si les trames sont obsolètes (je les écarter) ou non (je les met à jour). Elles sont obsolètes si l'instant (fonction **Date.now()** en Node.js) de la dernière mise à jour est différente de l'instant actuel d'un seuil de temps donné. Ceci, afin de cesser de partager des trames obsolètes sur le broker MQTT (topic



`rav/data/instruments/ais/<trackX>`) : 1 minute comme seuil, par exemple, dans mes tests. En situation réelle, il faudrait augmenter ce seuil.

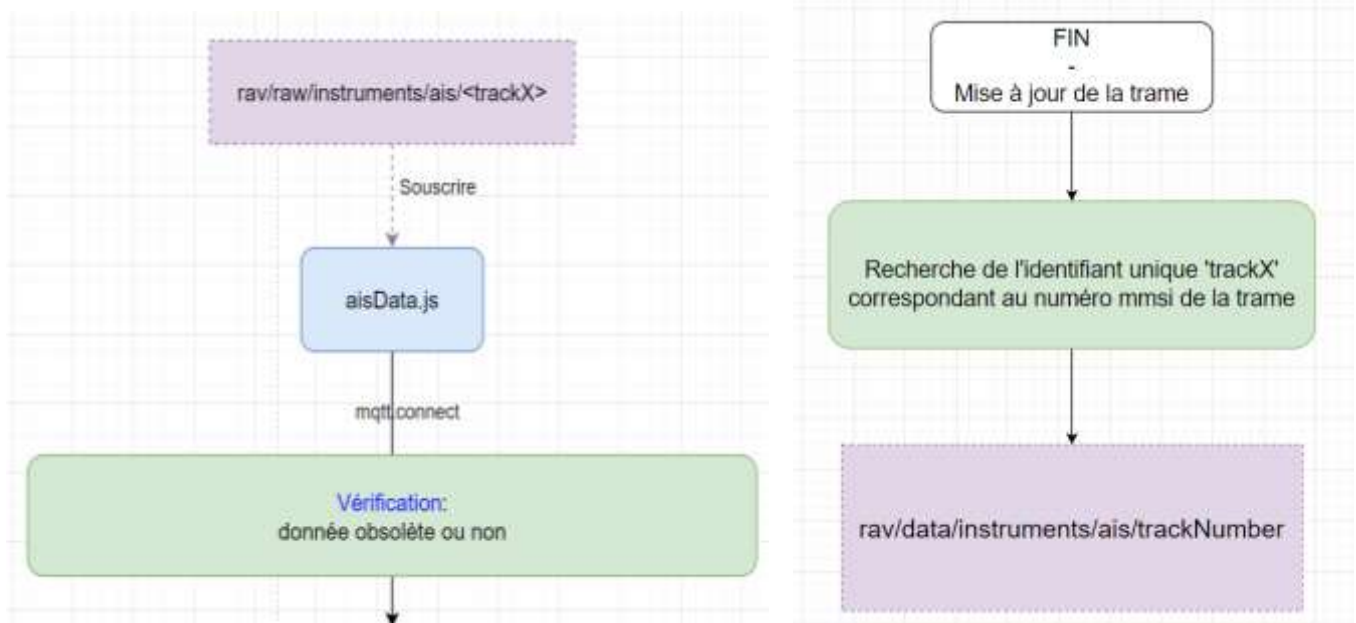


Figure 17 : Mise à jour des trames AIS

Plus en détail, l'algorithme résumant ce module de décodage est le suivant :

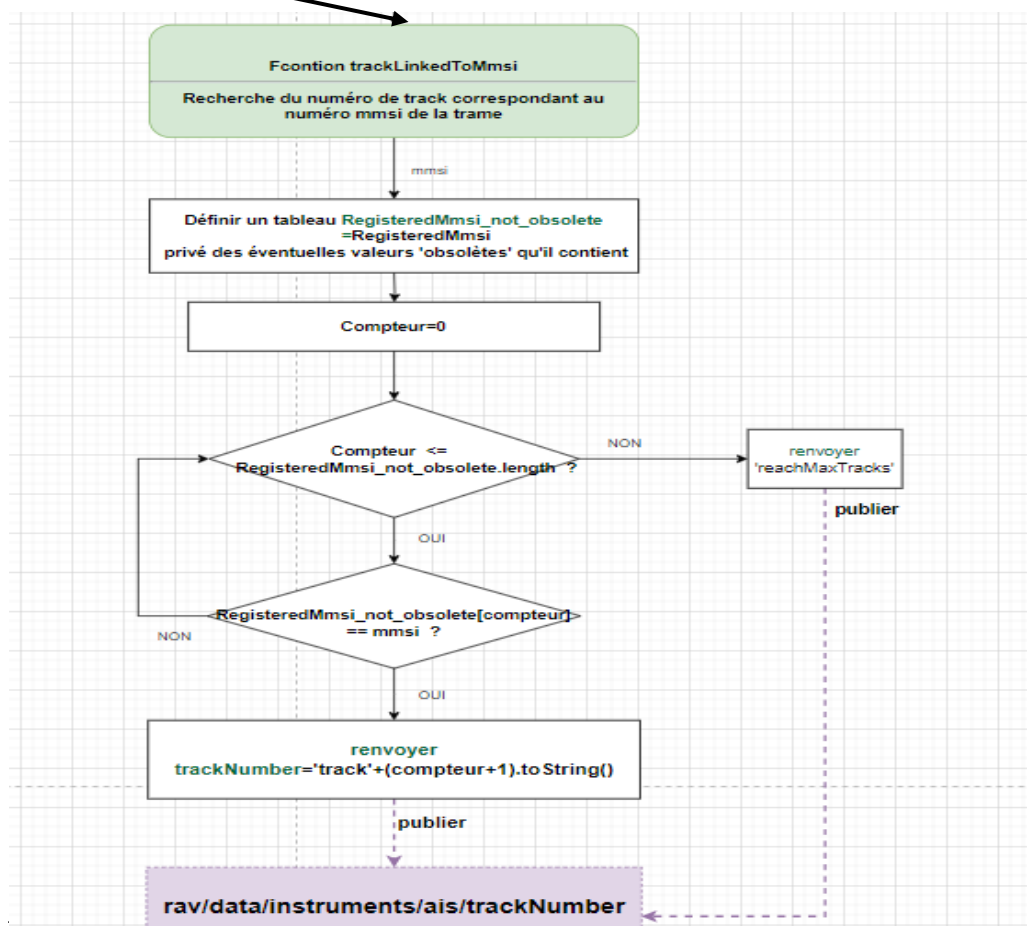
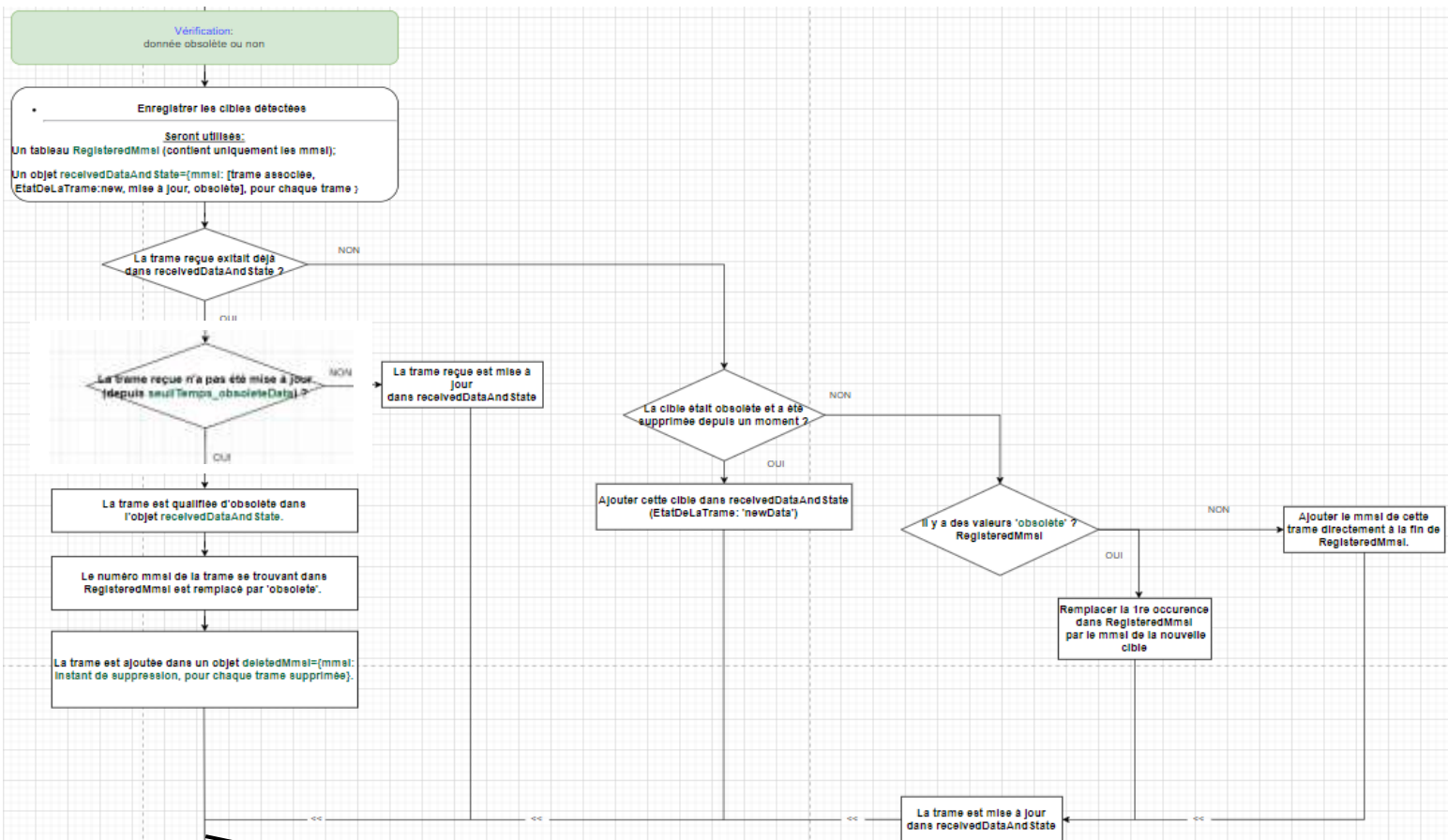


Figure 18 : Algorithme de mise à jour des trames AIS.

Les trames mises à jour sont envoyées au broker mqtt du drone 1 sur le topic `rav/data/instruments/ais/<trackX>`.

## IV.2 Notion de Closest Point of Approach (CPA)

Le **CPA**, Closest Point of Approach, est la notion centrale des dispositifs anticollision. Il s'agit d'un point estimé où la distance entre deux mobiles (dont au moins l'un est en mouvement) atteindra sa valeur minimale (cette distance est le **DCPA** ; le temps au bout duquel aura lieu ce rapprochement minimal est le **TCPA**). Cette estimation est utilisée pour évaluer le risque de collision de deux navires.

[\[Réf. 6\]](#) et [\[Réf. 7\]](#).

Cette notion permet principalement de rendre compte de grandeurs telles que le DCPA (Distance at Closest Point of Approach), le TCPA (Time to Closest Point of Approach) et le **BCPA** (Bearing of the Closest Point of Approach), qui sont alors utilisées pour des algorithmes d'évitement de collisions.

La théorie graphique est la suivante : Dans notre cas, **Le point B serait le RAV, et le point A une cible quelconque. Les points A et B sont sur une position à l'instant où les calculs du CPA sont effectués. Donc, la configuration ci-dessous évolue, le principe étant le même.**



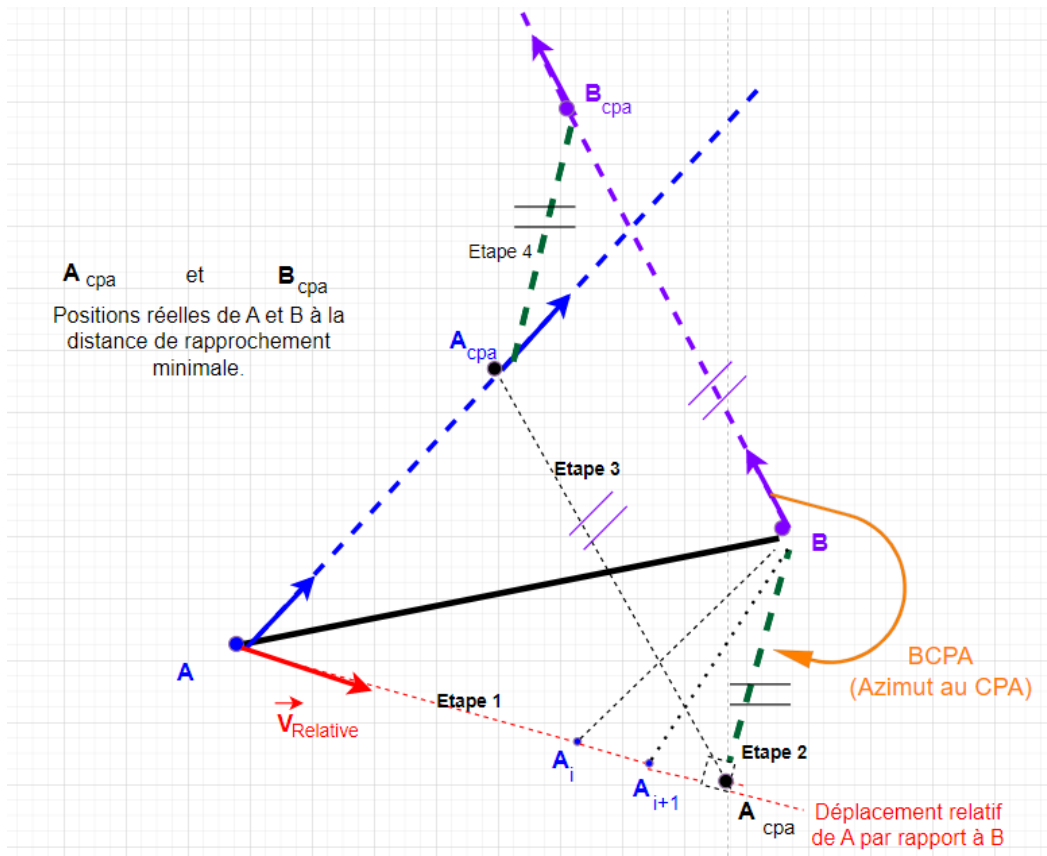


Figure 19 : Théorie graphique – CPA.

Le **DCPA** apparaît en pointillés vert, le **BCPA** ), en orange (en réalité le  $BCPA = \text{angle}(\text{Axe } B\_Nord, \text{axe } B\_Acpa, )$  ; sur la figure je le prends équivalent au relèvement vis-à-vis de la cible au CPA. Il suffit d'ajouter le Heading du point B pour obtenir l'azimut réel au CPA).  $TCPA = \text{distance}(AA_{cpa}) / V_{relative}$  ( $V_{relative}$  différent de 0). La vitesse relative n'est nulle que si une cible se déplace avec le même cap, à la même vitesse : on n'aurait pas de risque de collision avec cette cible.

Pour effectuer les différents calculs, j'ai choisi un repère orthonormé adapté, me permettant de me servir, entre autres, des fonctions fournies par la bibliothèque **wgs84-util** de Node.js. Ce repère, ainsi que les angles remarquables utilisés sont les suivants :

- Le point B est assimilé au RAV et le point A, à une cible quelconque. Le repère est centré sur le RAV pour tous les calculs.
- Les calculs sont valables à vitesse et à cap constant. Toutefois, les calculs sont effectués à la fréquence de réception/mise à jour des trames AIS : les calculs sont donc aussi valables en cas de

changement de cap ou de vitesse des cibles (ou du RAV). Si une variable change au niveau des couples (COG\_RAV, SOG\_RAV) et (COG\_cible, SOG\_cible), tous les calculs sont mis à jour.

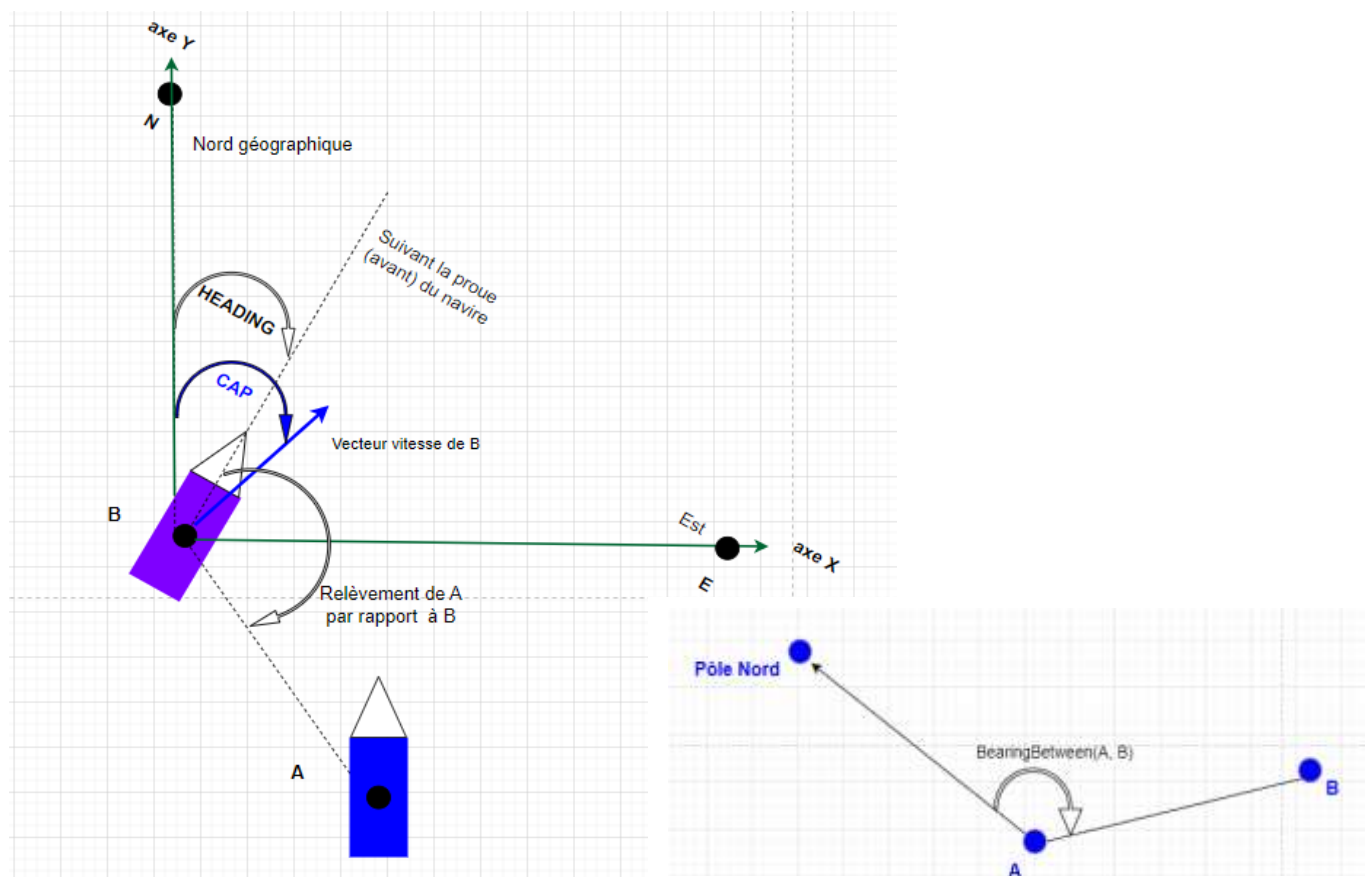


Figure 20 : Repère de calculs et angles de navigation utilisés.

⇒ Les premiers calculs à effectuer sont ceux du DCPA et du TCPA :

$$\begin{aligned}\vec{V}_{RELATIVE} &= (\vec{V}_A - \vec{V}_B) \\ \vec{AA}_t &= (\vec{V}_A - \vec{V}_B) \cdot t \\ \vec{BA}_t &= (\vec{BA} + \vec{AA}_t)\end{aligned}$$

$$DCPA = \min(\|\vec{BA}_t\|)$$

$$\Rightarrow \|\vec{BA}_t\|^2 = BA^2 + AA_t^2 + 2 \cdot \vec{BA} \cdot \vec{AA}_t$$

$$(E) \quad BA_t^2 = BA^2 + [(\vec{V}_A - \vec{V}_B) \cdot t]^2 + 2 \cdot \vec{BA} \cdot (\vec{V}_A - \vec{V}_B) \cdot t$$

- Recherche du minimum de la distance entre A et B (DCPA) :

$$\frac{d(BA_t^2)}{dt} = 2 \cdot t \cdot (\vec{V}_A - \vec{V}_B)^2 + 2 \cdot \vec{BA} \cdot (\vec{V}_A - \vec{V}_B) \cdot t$$

$$\frac{d(BA_t^2)}{dt} = 0 \Rightarrow t = T CPA = \frac{\vec{BA} \cdot (\vec{V}_A - \vec{V}_B)}{(\vec{V}_A - \vec{V}_B)^2} = \frac{\vec{BA} \cdot \vec{V}_{RELATIVE}}{(\vec{V}_{RELATIVE})^2}$$

En insérant TCPA dans l'équation donnant la distance entre A et B, on obtient :

$$BA_t^2(t = TCPA) = DCPA^2 = BA^2 - \frac{(\vec{BA} \cdot \vec{V}_{RELATIVE})^2}{(\vec{V}_{RELATIVE})^2}$$

- Notons :

$BB_{BA}$  = BearingBetween (B, A)

Il s'agit de l'azimut  $\vec{BN} - \vec{BA}$  (où N représente le Nord géographique)

$COG_p$  = Cap du point p.

Coordonnées dans (B,  $\vec{BE}$ ,  $\vec{BN}$ ) :

$$\vec{BA} = [\|\vec{BA}\| \cdot \cos(90 - BB_{BA}); \|\vec{BA}\| \cdot \sin(90 - BB_{BA})]$$

$$\vec{V}_{RELATIVE} = [\|\vec{V}_A\| \cdot \cos(90 - COG_A) - \|\vec{V}_B\| \cdot \cos(90 - COG_B); \|\vec{V}_A\| \cdot \sin(90 - COG_A) - \|\vec{V}_B\| \cdot \sin(90 - COG_B)]$$

90° intervient afin de me ramener à un repère orthonormé ordinaire (en partant des angles de navigation fournis par wgs84).

Ces grandeurs vectorielles sont ensuite utilisées pour compléter le calcul du TCPA et du DCPA.

⇒ Le calcul du BCPA a été calculé comme suit :

### Notons :

- $A_{cpa}$  : La position de la cible (par rapport à B) au CPA, obtenue par projection orthogonale de B sur le chemin relatif de A par rapport à B.
- $(\vec{B}, \vec{BE}, \vec{BN})$  le repère orthonormé centré sur le point B (Assimilé au RAV), dont l'axe  $X=\vec{BE}$  et l'axe  $Y=\vec{BN}$  : N représentant le Nord géographique et E, l'Est.
- $(Dr)$  la droite représentant le chemin relatif de A par rapport à B. Son équation est de la forme :  $(Dr) : ax+by+c=0$

- $\vec{Heading}_B$  le vecteur unitaire correspondant au Heading du point B (RAV)

Il est question de trouver l'angle  $(\vec{Heading}_B, \vec{BA}_{cpa})$ , qui représente le **BCPA**.

Pour cela je trouve cet angle en calculant :

$$\text{Arcos} [ (\vec{Heading}_B \cdot \vec{BA}_{cpa}) / (||\vec{BA}_{cpa}|| \cdot ||\vec{Heading}_B||) ]$$

D'où la nécessité de calculer les coordonnées du vecteur  $\vec{BA}_{cpa}$  dans le repère de calculs  $(\vec{B}, \vec{BE}, \vec{BN})$ , autrement dit, les coordonnées de  $A_{cpa}$  car celles de  $B$

Sont connues : B étant l'origine du repère  $(X_B = Y_B = 0)$ .

- $X_{rel}$  et  $Y_{rel}$  : coordonnées du vecteur vitesse relative dans  $(\vec{B}, \vec{BE}, \vec{BN})$ . Ce vecteur vitesse est un vecteur directeur de la droite  $(Dr)$ . En rappelant que pour une droite d'équation  $ax+by+c=0$ , le vecteur de coordonnées  $(-b ; a)$  définit un vecteur directeur. J'assimile ce vecteur directeur aux coordonnées inconnues a et b, au vecteur vitesse relative de coordonnées  $X_{rel}$  et  $Y_{rel}$ . Ce qui entraîne  $a=Y_{rel}$   
Et  $b=-X_{rel}$ .

- Pour trouver les coordonnées de  $A_{cpa}$ , je me sers des relations suivantes :

$$(1) : A \in (Dr)$$

Ses coordonnées vérifient donc l'équation de la droite  $(Dr)$ .

$$D'où: Y_{rel} \cdot X_A - X_{rel} \cdot Y_A + C = 0$$

$$\Leftrightarrow C = X_{rel} \cdot Y_A - Y_{rel} \cdot X_A$$

$$(2) : \vec{B} \vec{A}_{cpa} \perp \vec{V}_{RELATIVE}$$

$$\Rightarrow \vec{B} \vec{A}_{cpa} \cdot \vec{V}_{RELATIVE} = 0$$

Avec (1) et (2), on obtient le système suivant :

$$\begin{cases} (X_{A_{CPA}} - X_B) \cdot X_{rel} + (Y_{A_{CPA}} - Y_B) \cdot Y_{rel} = 0 \\ Y_{rel} \cdot X_{A_{CPA}} - X_{rel} \cdot Y_{A_{CPA}} + C = 0 \end{cases}$$

(En traduisant l'appartenance de  $A_{cpa}$  à la droite (Dr))

La constante C est connue et est fonction des coordonnées de A (Voir calculs précédents).

La résolution de ce système mène à :

$$\begin{cases} X_{A_{CPA}} = X_B - (Y_{A_{CPA}} - Y_B) \cdot \frac{Y_{rel}}{X_{rel}} \\ Y_{A_{CPA}} = \frac{1}{X_{rel}^2 + Y_{rel}^2} * [X_{rel} \cdot C + Y_{rel} \cdot (X_{rel} \cdot X_B + Y_{rel} \cdot Y_B)] \end{cases}$$

Pour trouver C, il nous faut les coordonnées du point A.

- Or les coordonnées de A dans le repère ( $B, \vec{B}\vec{E}, \vec{B}\vec{N}$ ) sont les coordonnées de

$$\vec{B} \vec{A} : \begin{cases} BA \cdot \cos(90 - BB_{BA}) \\ BA \cdot \sin(90 - BB_{BA}) \end{cases}$$

Les 90° interviennent afin de se ramener à un repère orthonormé courant

Avec  $BB_{BA}$  le BearingBetween (B, A) : Azimut entre ces deux points.

- Le vecteur unitaire :

$$\vec{Heading}_B = \begin{cases} \cos(90 - \text{Heading}_B) \\ \sin(90 - \text{Heading}_B) \end{cases}$$

$$\text{BCPA} = \text{Arcos} [ (\vec{Heading}_B \cdot \vec{B} \vec{A}_{cpa}) / (||\vec{B} \vec{A}_{cpa}|| \cdot ||\vec{Heading}_B||) ]$$

⇒ ce stade, les trois paramètres DCPA, TCPA et BCPA peuvent être implémentés sous Node.js.



### IV.3 Recherche de paramètres supplémentaires d'aide à la prise de décision

Les trois paramètres précédents ne sont pas suffisants pour une prise de décision effective, surtout dans le cas où on s'intéresse à plusieurs cibles.

Je choisis d'effectuer la prise de décision vis à vis des cibles, sur un principe de base qui est le suivant : Je cherche premièrement à déterminer un niveau de criticité de chaque cible. Pour cela, je définis trois cercles de sécurité aux rayons bien choisis.

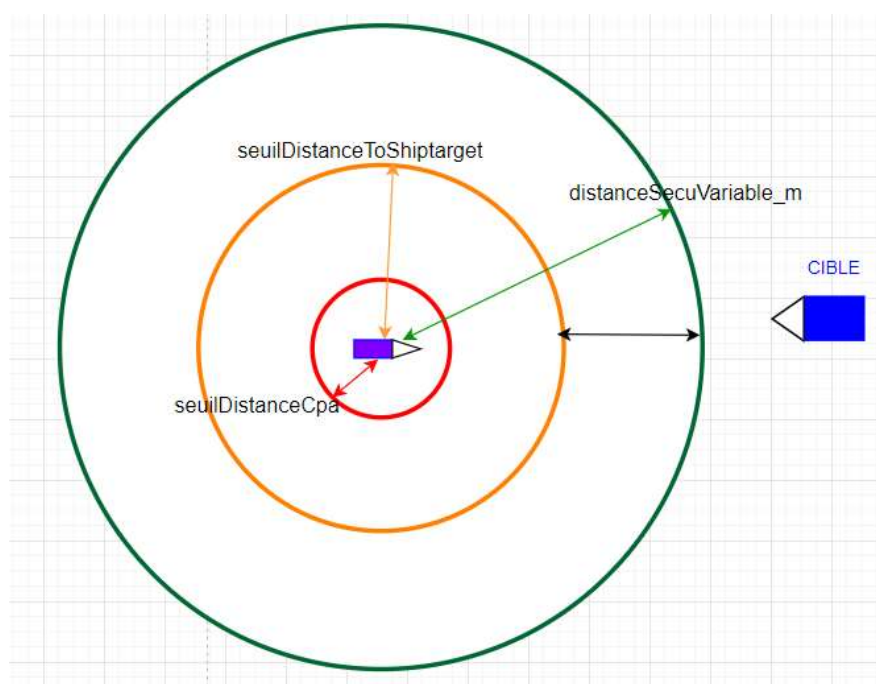


Figure 21 : Principe de base de la prise de décision

#### Le cercle rouge :

Il a un rayon égal à une valeur de **DCPA** critique (**seuilDistanceDCPA**) : on choisit un seuil de DCPA en deçà duquel il y a risque de collision. Le calcul de TCPA et DCPA se faisant à chaque instant, on vérifie pour chaque cible détectée par l'AIS, si le DCPA par rapport à cette cible est inférieur à

**seuilDistanceDCPA**. Si tel est le cas, la cible est potentiellement à risque et on lui assigne un niveau de criticité 0 (level\_0).

### Le cercle orange :

Définit la distance de sécurité **seuilDistanceToTargetShip** à garder vis-à-vis d'une cible (lors d'un évitement de collision par exemple). Logiquement, cette distance doit être supérieure à **seuilDistanceDCPA** (rayon du cercle rouge).

### Le cercle vert :

En se basant sur les données de navigation d'une cible, on cherche à connaître à quelle distance (**distanceSecuVariable\_m**) elle se trouvera du RAV, telle que **TempsVariable** (temps à fixer) plus tard elle sera sur le cercle de sécurité orange : il s'agit du rayon du cercle en vert, différent selon la cible.

Ainsi, **TempsVariable** doit être une approximation du temps qu'il faut au RAV pour suivre une consigne de cap. Dans ce cas, le RAV aura effectué une manœuvre d'évitement et gardé la cible dans le cercle de sécurité orange après une durée égale à **TempsVariable**.

**En résumé, les manœuvres d'évitement doivent être déclenchées lorsque la cible rentre dans ce cercle vert.**

Je cherche ensuite à construire une variable binaire **criticityLevel** qui définit le niveau de criticité d'une cible, en fonction de 4 bits bien choisis.

Les calculs sont implémentés dans un module Node.js "**aisCPA.js**". Je récupère les trames décodées via mqtt, sur le topic **rav/data/instruments/ais/<trackX>**. X un entier naturel, inférieur à seuil (100 dans mes tests ; il y a autant de cibles détectées que de valeurs de X).

J'effectue ensuite les calculs à partir de ces trames. Ci-dessous un bout de l'algorithme lié à ce module de calculs Node.js :

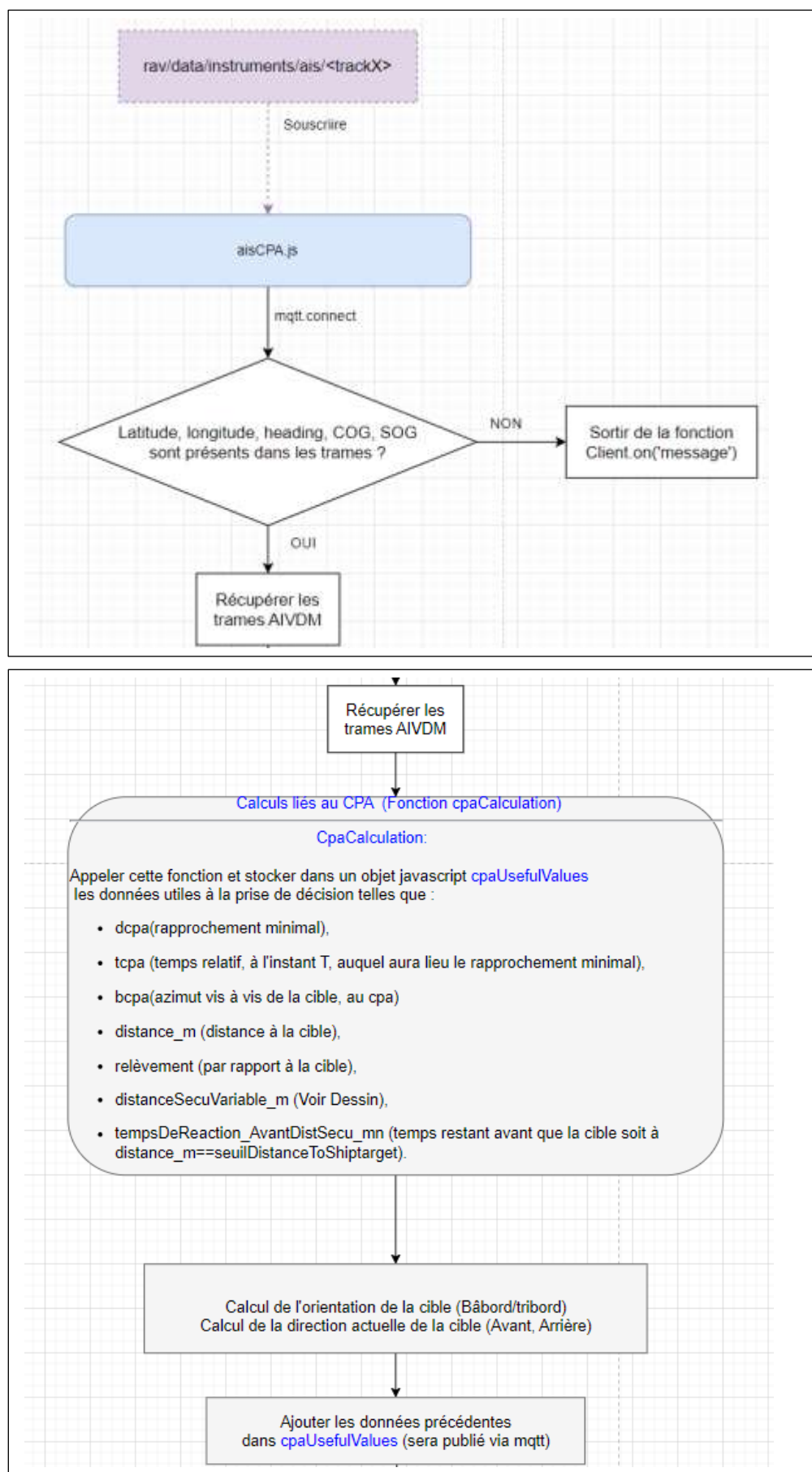
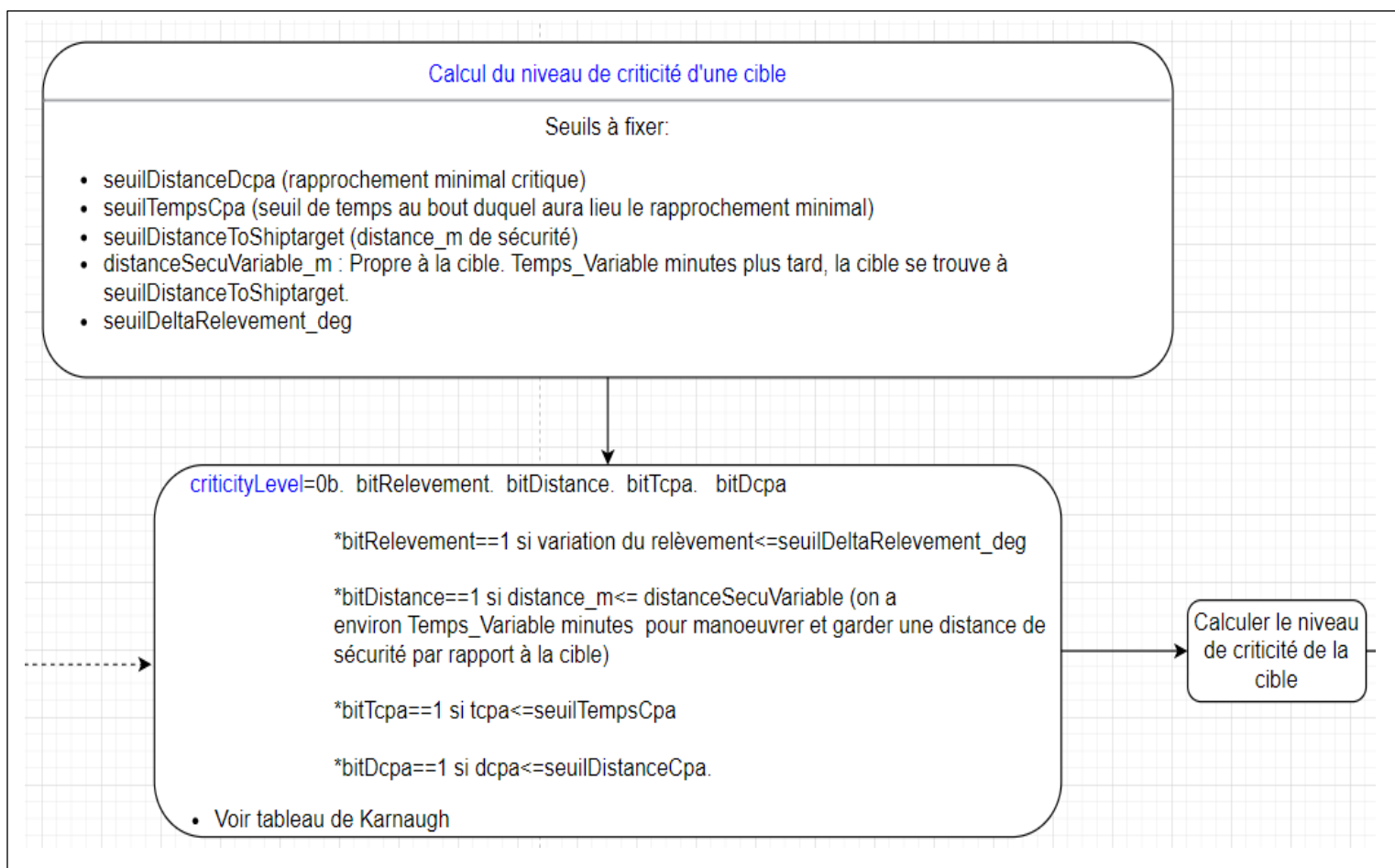


Figure 22 : Portion d'algorithme-module 'aisCPA.js'.



⇒ Le niveau de criticité est déterminé comme suit :



#### NOTES:

- On ne s'intéresse qu'aux situations dans lesquelles **bitDcpa==1**: c'est-à-dire que la cible passera près du RAV à une distance inférieure ou égale à seuilDcpa (cible critique).
- Dans le cas contraire, la cible est sans danger.

bitRelevement bitDistance bitTcpa bitDcpa	00	10	01	11
00	SafeTarget	SafeTarget	Level0	Level1
10	SafeTarget	SafeTarget	Level0	Level1
01	SafeTarget	SafeTarget	Level0	Level2
11	SafeTarget	SafeTarget	Level0	Level3

Tableau de KARNAUGH

Figure 23 : Niveau de criticité des cibles.

⇒ Il faut ensuite classer les cibles par ordre de priorité dans la prise de décision, selon des critères de comparaison. Ci-dessous le principe :

Critères de comparaison choisis :

\*Dans le **Level0** les cibles sont classées suivant **dcpa**(du min au max)

\*Dans le **Level1** les cibles sont classées suivant **tcpa**(du min au max)

\*Dans le **Level2** les cibles sont classées suivant **distanceSecuVariable\_m**: à cette distance (donnant le rayon du [cercle de sécurité vert](#)), la cible se trouvera **TempsVariable**(à fixer) minutes plus tard dans le [cercle de sécurité orange](#), si elle garde son cap et sa vitesse.

La cible la plus critique est celle ayant une 'distanceSecuVariable\_m' la plus grande ( i.e elle se rapproche plus vite que les autres et on sait à l'avance quand il faut manœuvrer grâce à cette variable; cette donnée est plus concluante que la vitesse relative cible/Rav).

\*Ensuite, les cibles sont classées globalement, tous les niveaux inclus.

⇒ **Les principaux paramètres supplémentaires calculés sont** : le temps (**t\_seuil**) au bout duquel la cible se trouvera dans le cercle de sécurité orange (**dseuil**), la distance de sécurité variable **distanceSecuVariable\_m** (à cette distance les manœuvres d'évitement doivent être déclenchées), le relèvement par rapport à une cible, l'orientation (Bâbord/Tribord de la cible), la direction (Avant/Arrière) de la cible, et d'autres paramètres qui seront traités dans les sections suivantes.

▪ Le calcul de **t\_seuil** est le suivant :

(E) donne la distance entre le RAV et une cible (**BA<sub>i</sub>**) à chaque instant. **BA** étant la distance à un instant considéré comme initial.

Sachant : (E)

$$BA_t^2 = BA^2 + [(\vec{V}_A - \vec{V}_B) \cdot t]^2 + 2 \cdot \vec{BA} \cdot (\vec{V}_A - \vec{V}_B) \cdot t$$

Le temps  $t_{seuil}$  cherché est tel que  $BA_t^2(t_{seuil}) = d_{seuil}^2$ .

L'équation du second degré qui en résulte conduit à :

$$t_{seuil} = \frac{-2 \cdot \vec{BA} \cdot (\vec{V}_A - \vec{V}_B) \pm \sqrt{\Delta}}{2 \cdot (\vec{V}_A - \vec{V}_B)^2}$$

$$\Delta = 4 \cdot [\vec{BA} \cdot (\vec{V}_A - \vec{V}_B)]^2 - 4 \cdot (\vec{V}_A - \vec{V}_B)^2 \cdot (BA^2 - d_{seuil}^2).$$

Ainsi, lorsque  $t_{seuil}$  vaut **TempsVariable** (au bout de ce temps, la cible passe du cercle vert au cercle orange), on sait que la cible vient de franchir (se trouve à **distanceSecuVariable\_m** du RAV) le cercle en vert et qu'il faut déclencher les manœuvres d'évitement.

Seulement, nous serions obligés d'attendre cet instant pour connaître à quelle distance de la cible il faut manœuvrer.

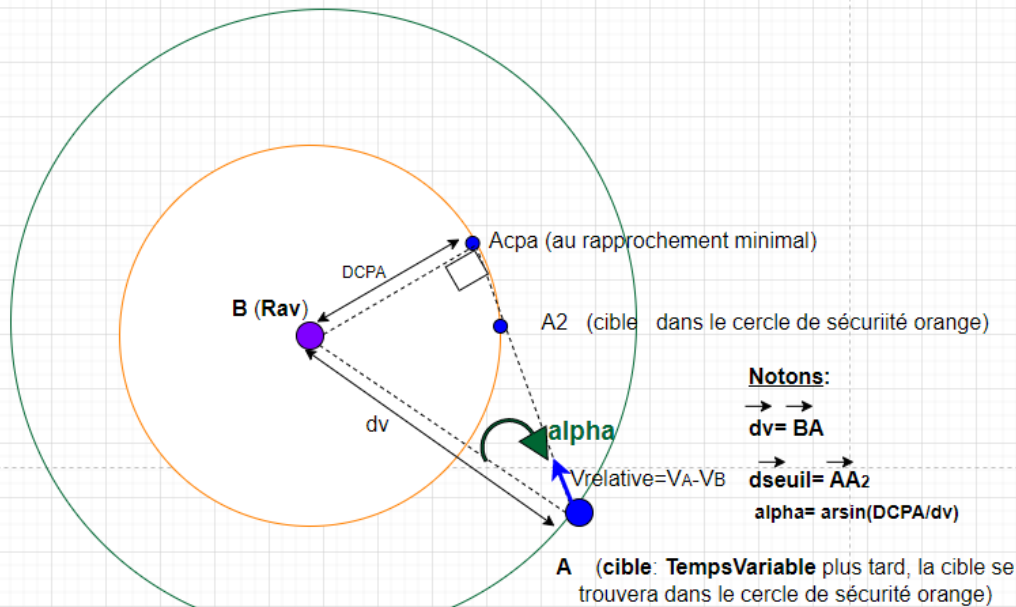
- Je calcule alors la distance **distanceSecuVariable\_m** de manière à l'avoir en avance. Je peux en plus l'utiliser comme critère de classification des cibles au niveau 2 (level\_2), celui à partir duquel il faut déclencher les manœuvres d'évitement : la cible ayant une **distanceSecuVariable\_m** la plus grandes est la plus critique (voir explications et images précédentes).

**Note :** Si les cibles se dirigent toujours droit sur le RAV (DCPA=0):

**distanceSecuVariable\_m(rayon du cercle vert)=dseuil (rayon du cercleOrange) +**

**Vitesse\_Relative\_cible \* TempsVariable.** Car par définition du cercle vert, la cible part du cercle vert au cercle orange en une durée **TempsVariable** (on peut vérifier ce cas particulier à partir du cas général ci-dessous).

Seulement, les cibles ne se dirigent pas toujours droit sur le RAV. Le calcul général est le suivant :



**TempsVariable** est à fixer :

il doit approcher le temps nécessaire pour effectuer un changement de cap (par le RAV).  
 Ceci permettra de donner une consigne de cap au Rav qu'il suivra en gardant une distance de sécurité (2me cercle, orange) vis-à-vis de la cible, **TempsVariable** plus tard.

$$\alpha = \arcsin(DCPA/dv)$$

- L'équation cherchée, partant de (E) :

$$BA_t^2 = BA^2 + [(V_A - V_B) \cdot t]^2 + 2 \cdot BA \cdot (V_A - V_B) \cdot t$$

est la suivante :

$$(1) d_{seuil}^2 = dv^2 + [V_{relative} \cdot tvar]^2 + 2 \cdot (dv \cdot V_{relative}) \cdot tvar$$

Le produit scalaire  $(\vec{dv} \cdot \vec{V_{relative}})$  est gênant car on souhaite obtenir  $dv$  :

On a :

$$(\vec{dv} \cdot \vec{V_{relative}}) = |\vec{dv}| \cdot |\vec{V_{relative}}| \cdot \cos(\vec{dv}, \vec{V_{relative}})$$

Et l'angle  $(\vec{dv}, \vec{V_{relative}}) = 180 - \alpha$  ( $\alpha = \arcsin(DCPA/dv)$ )

=>

$$(\vec{dv} \cdot \vec{V_{relative}}) = |\vec{dv}| \cdot |\vec{V_{relative}}| \cdot \cos(180 - \arcsin(DCPA/dv))$$

- En simplifiant (relation trigonométrique) :

$$(2) (\vec{dv} \cdot \vec{V_{relative}}) = -|\vec{dv}| \cdot |\vec{V_{relative}}| \cdot \sqrt{1 - \frac{DCPA^2}{dv^2}}$$

J'insère cette valeur du produit scalaire dans l'équation (1) :

J'obtiens une équation du 4me degré en  $dv$  :

$$dv^4 + dv^2 \cdot (-2 \cdot d_{seuil}^2 - 2 \cdot tvar^2 \cdot V_{relative}^2) + ([d_{seuil}^2 - tvar^2 \cdot V_{relative}^2]^2 + 4 \cdot tvar^2 \cdot V_{relative}^2 \cdot DCPA^2) = 0$$

On trouve ainsi  $dv^2$  dans un premier temps, puis  $dv$

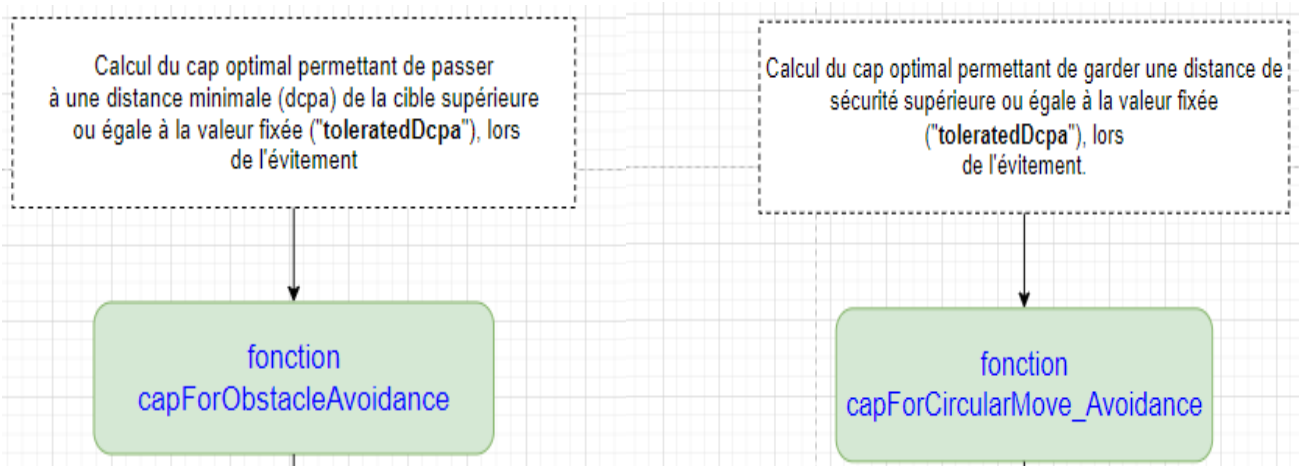
Un aperçu des données calculées et publiées via MQTT :

```
{
  "dcpa_nmi": "safeDcpaValue",
  "relevement_deg": 0,
  "distance_m": 399.9964,
  "distance_m": 399.9988,
  "azimutCpa_deg": 117.54314021017764,
  "cibleZoneCroisementCpa": "Derrière",
  "cibleCurrentDirection": "Arrière",
  "cibleOrientation": "Tribord",
  "mmsi": "227261040BIS",
  "deltaRelevement_deg": 0,
  "criticalityLevel": "safeTargetShip"
}
```

Figure 24 : Données décisionnelles-MQTT Explorer

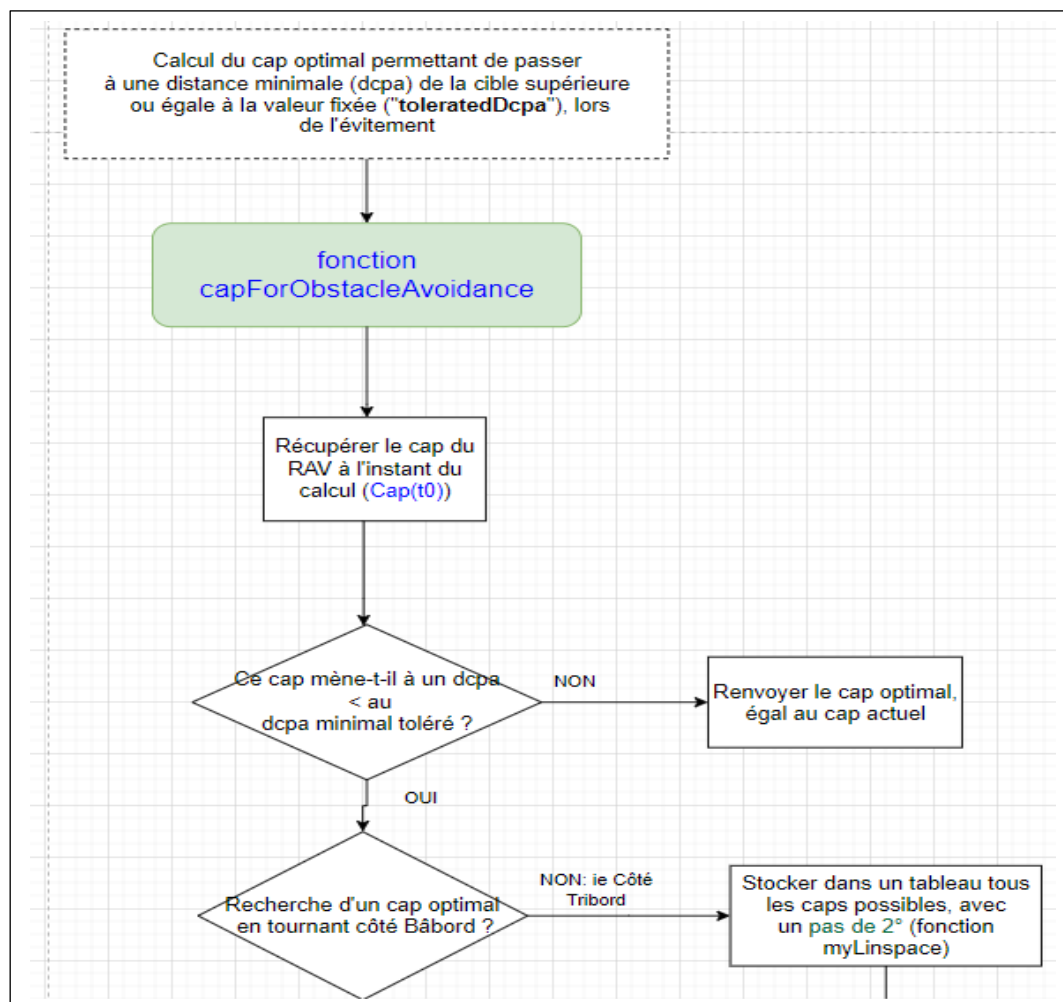
## IV.4 Algorithmes d'évitement de collision

Après les premiers paramètres de prise de décision, j'ai mis en place des algorithmes d'évitement de collision basés sur une recherche de caps optimaux.



▪ Fonction **capForObstacleAvoidance** :

La variable “toleratedDcpa” doit être prise égale au rayon du cercle de sécurité orange (seuilDistanceToTargetShip), vu dans la section précédente. A travers cette fonction je cherche le 1er cap menant à un **DCPA**  $\geq$  **toleratedDcpa** = **seuilDistanceToTargetShip**, qui nécessite une variation minimale par rapport au cap à l’instant de la prise de décision. Ci-dessous l’algorithme lié à cette fonction :



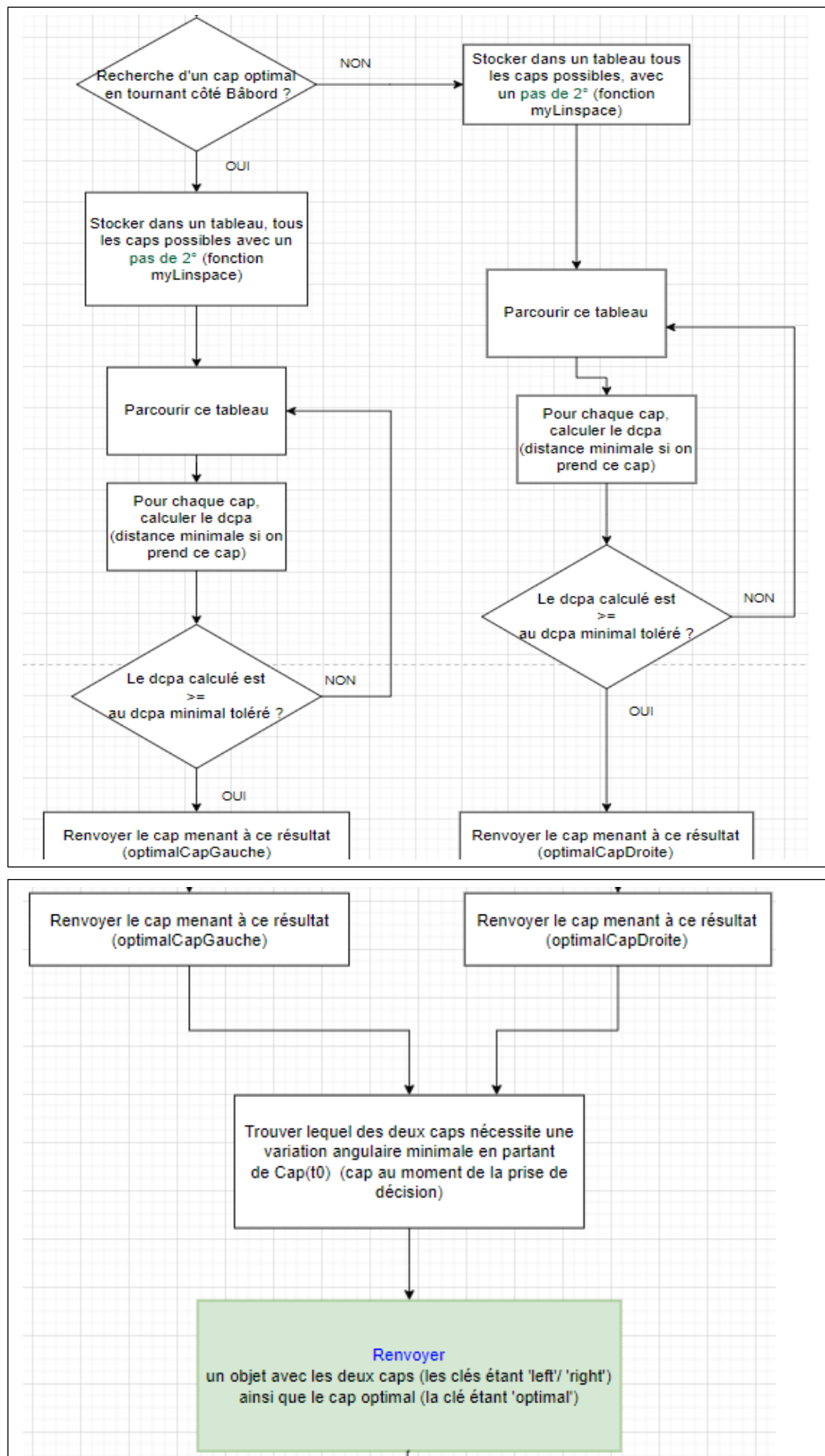
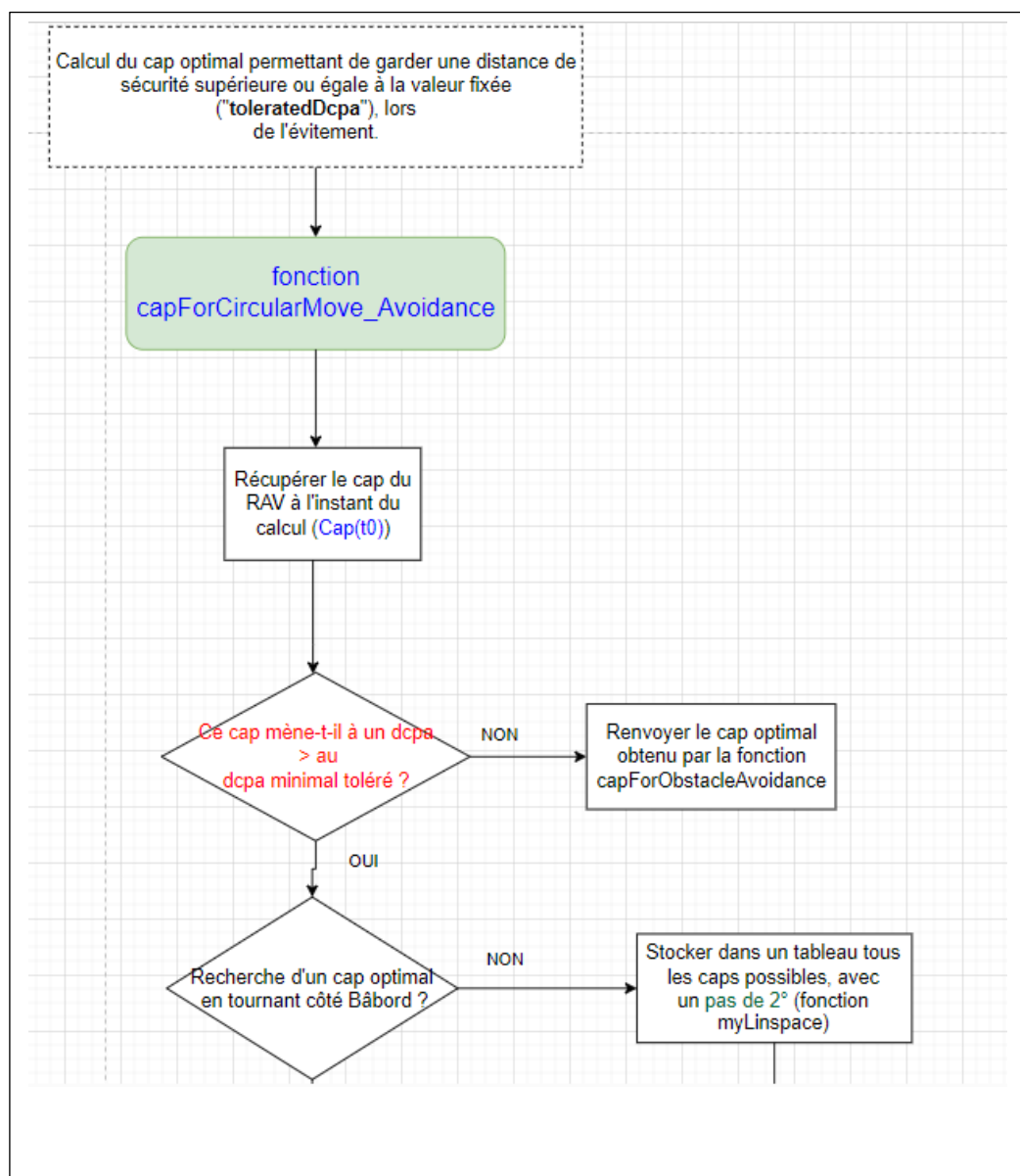




Figure 25-a : Portion d'algorithme- cap optimaux.

▪ La fonction **capForCircularMove\_Avoidance** :

Cette fois-ci, je calcule le cap permettant de garder précisément une distance de sécurité (égale à "toleratedDcpa" = seuilDistanceToTargetShip). L'algorithme associé à cette fonction est le suivant:



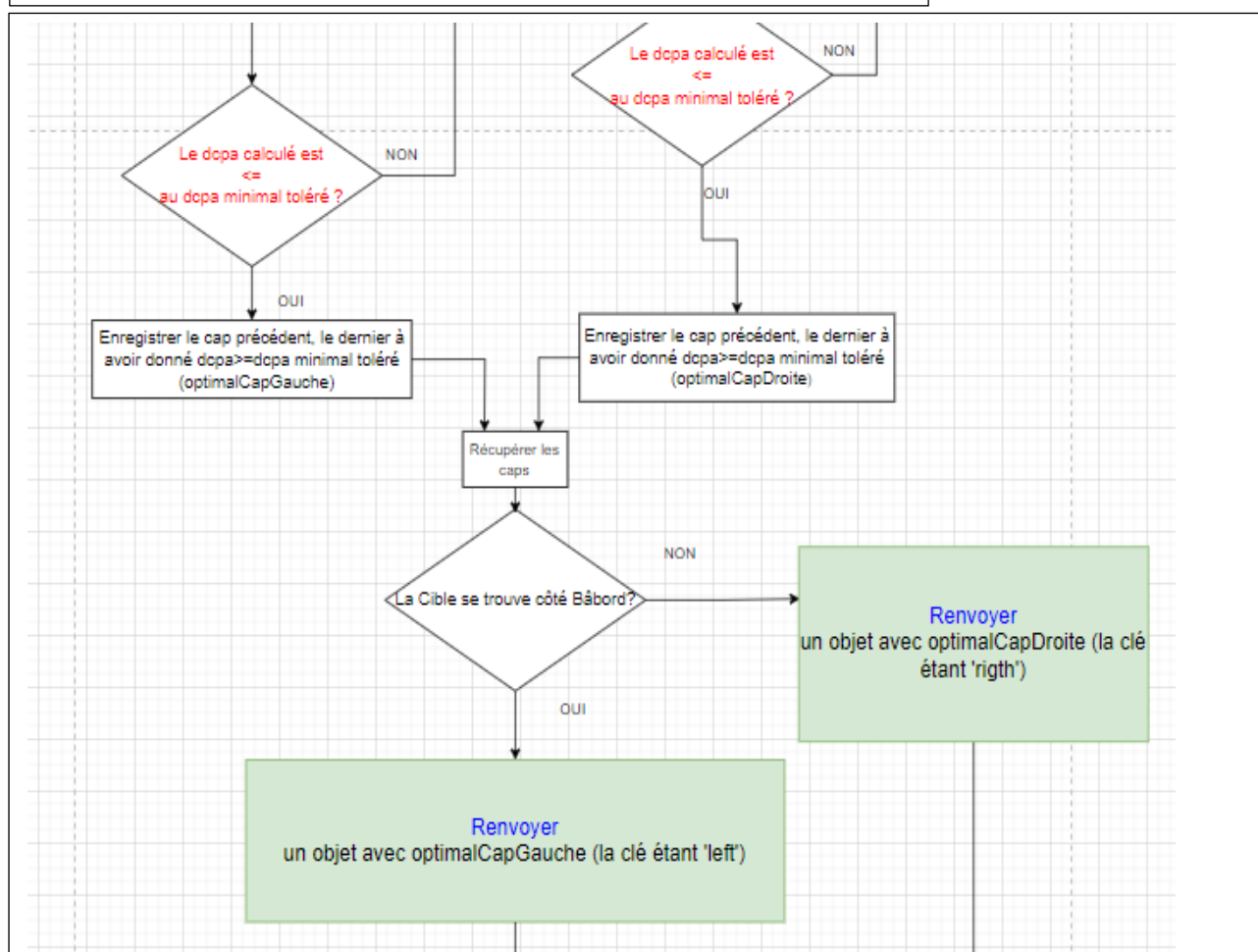
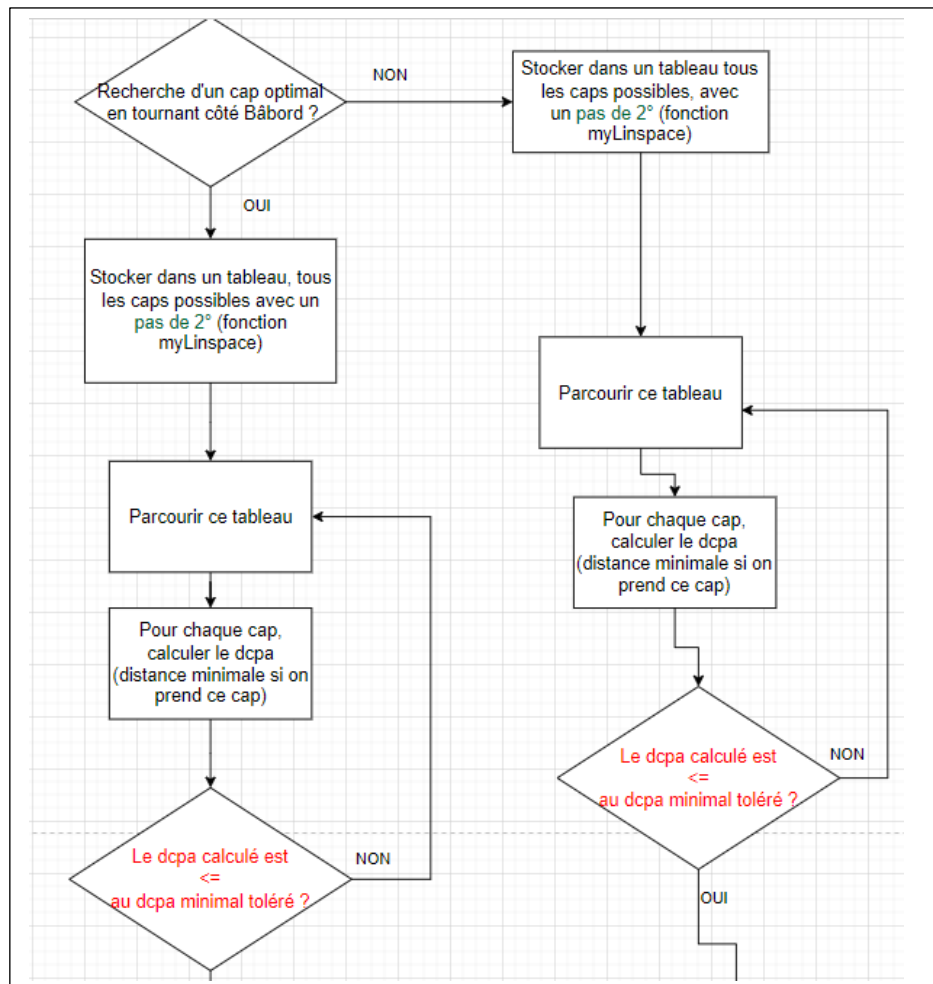


Figure 25-b : Portion d'algorithme- cap optimaux.

A ce stade, toutes les données obtenues peuvent être envoyées (en utilisant un objet JavaScript) au broker MQTT, sur les différents topic `rav/local/instruments/ais/<trackX>`. Je m'abonnerai à ces topics pour utiliser les données obtenues sur le simulateur.

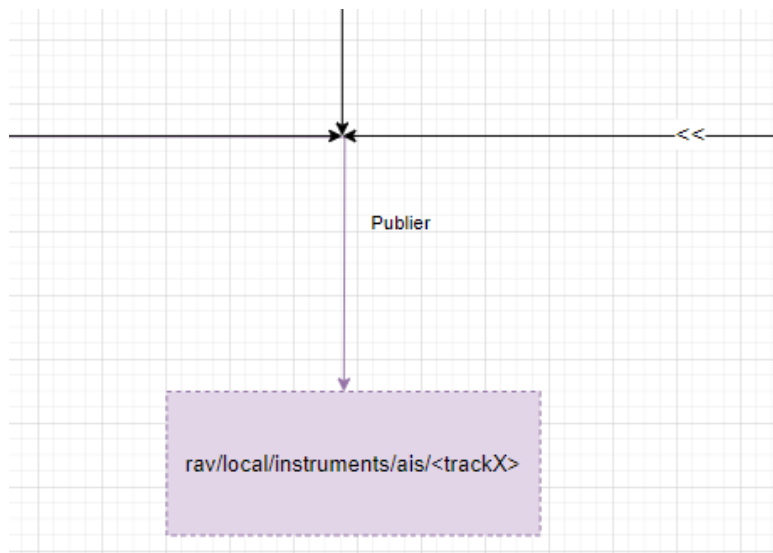


Figure 26 : Un aperçu des valeurs de caps optimaux calculés :

```
~ "capForObstacleAvoidance": {
~   "left": 222.10000000000002,
~   "right": 222.10000000000002,
~   "optimal": [
~     "currentCap",
~     222.10000000000002
~   ]
~ },
~ "capForCircularMove_avoidance": {
~   "right": 222.10000000000002
~ },
~ "priorityOrder": "safeTargetShip",
```

Figure 27 : Aperçu cap optimal- MQTT Explorer

## IV.5 Simulateur développé pour tests d'évitement de collision

### Les éléments de base

Dans le but de tester les différents paramètres précédents, j'ai développé un simulateur de tests de mes algorithmes d'évitement de collisions.

L'algorithme de base utilisé dans ce simulateur est fondé sur une machine d'états à 4 niveaux :

**Step1, step2, step3, safeNavigation.**

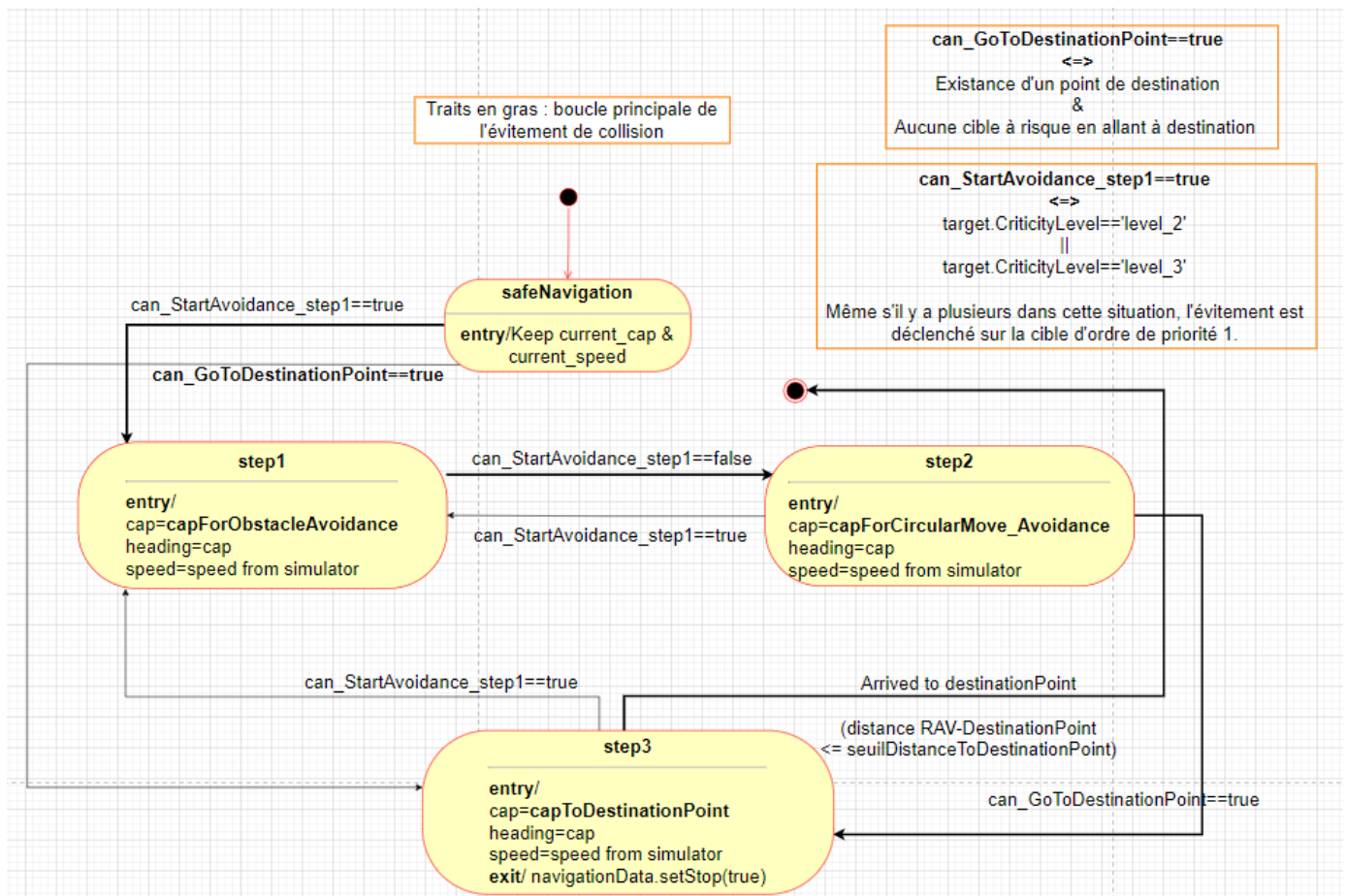


Figure 28 -a : Machine à états-algorithme d'évitement de collision.

- Si une cible est au niveau de criticité `level_2` ou `level_3`, l'étape `step1` est déclenchée (`can_StartAvoidance_step1==true`) : c'est le début

de l'évitement. Ensuite, les changements d'étapes sont définis grâce à une machine à états finis ;

- Sinon, on est dans l'étape **safeNavigation**

### **step1 :**

Lancement de l'algorithme d'évitement de collision par le **calcul du cap optimal** permettant de passer près de la cible, à un `dcpa` donné (`dcpa >= toleratedDCPA`).

En suivant ce cap, dès que la cible n'est plus à risque, on passe au `step2`.

### **Step2 :**

Dans cette étape, on vérifie si on a un point de destination (car le RAV peut être piloté suivant ce mode selon lequel il doit se rendre à un point de destination donné) et si on peut se rendre directement à ce point de destination.

En effet, j'ai une fonction **capToDestinationPoint** qui permet d'obtenir le cap menant directement à un point donné (point au format GeoJson). Un point P au format GeoJson étant de la forme `P={ 'type' : 'point', 'coordinates' : [longitude, latitude] }`.

- Si on ne peut se rendre directement au point de destination (présence d'un navire à risque), **on contourne la cible** pour s'y rendre.
- Sinon, on passe au `step3`.

S'il n'y a pas de point de destination le RAV décrit un mouvement circulaire autour de la cible.

### **Step3 :**

**On se rend directement au point de destination** : consigne de `cap=capToDestinationPoint(...)`.

Lorsque le RAV arrive au point de destination, il s'arrête. On passe dans un état `safeNavigation`.

La boucle peut reprendre à tout moment, si le niveau de criticité de la cible passe au niveau de criticité `level_2` ou `level_3`.

Concernant l'interface graphique, je me sers des données et paramètres calculés plus haut (partagés via MQTT), de module de fonctions Node.js, d'application Vue.js qui ont accès à des bases de données locales (fichier store.js et storeNav.js) permettant de “communiquer ” entre elles.

Un aperçu des modules utilisés pour l'application :

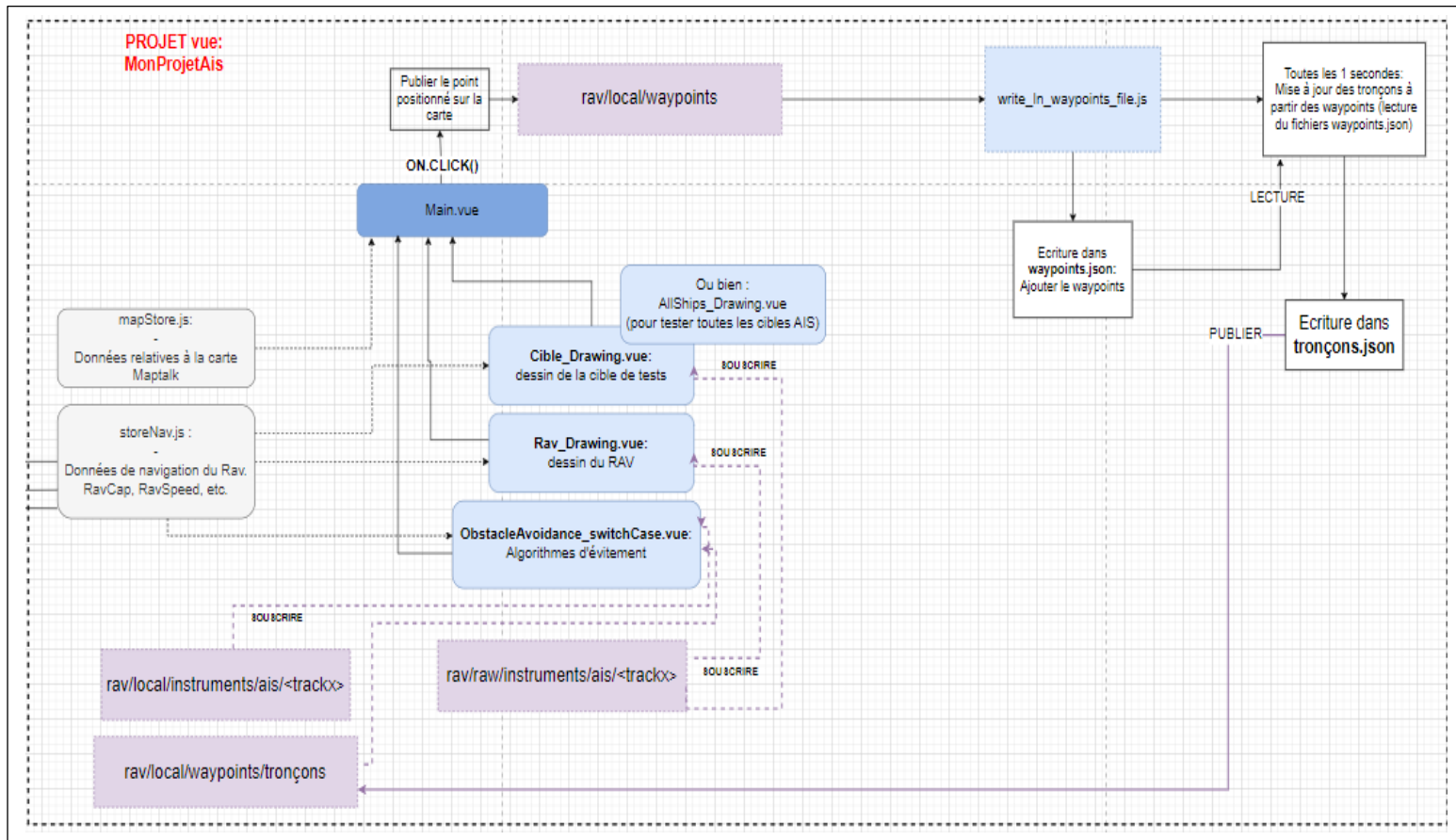
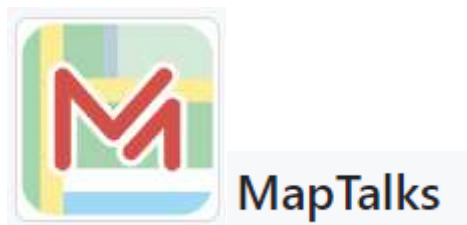


Figure 28 -b : Modules utilisés pour le simulateur.

Je dessine les cibles et le RAV sur une carte, grâce à la bibliothèque wgs84-util de Node.js :



La bibliothèque **MapTalks** est une bibliothèque JavaScript open-source qui permet de créer des cartes interactives et des visualisations géospatiales.

### IV.5.1 Algorithme d'évitement de collision, cas « hors mission ».

❖ Dans un premier temps, j'effectue des tests sur une cible (dessinée en bleu), dont je simule le mouvement. Le RAV est dessiné en violet. Je teste l'algorithme d'évitement « hors mission », c'est-à-dire que le RAV ne suit pas des chemins (tronçons) bien définis, mais éventuellement un simple point de destination.

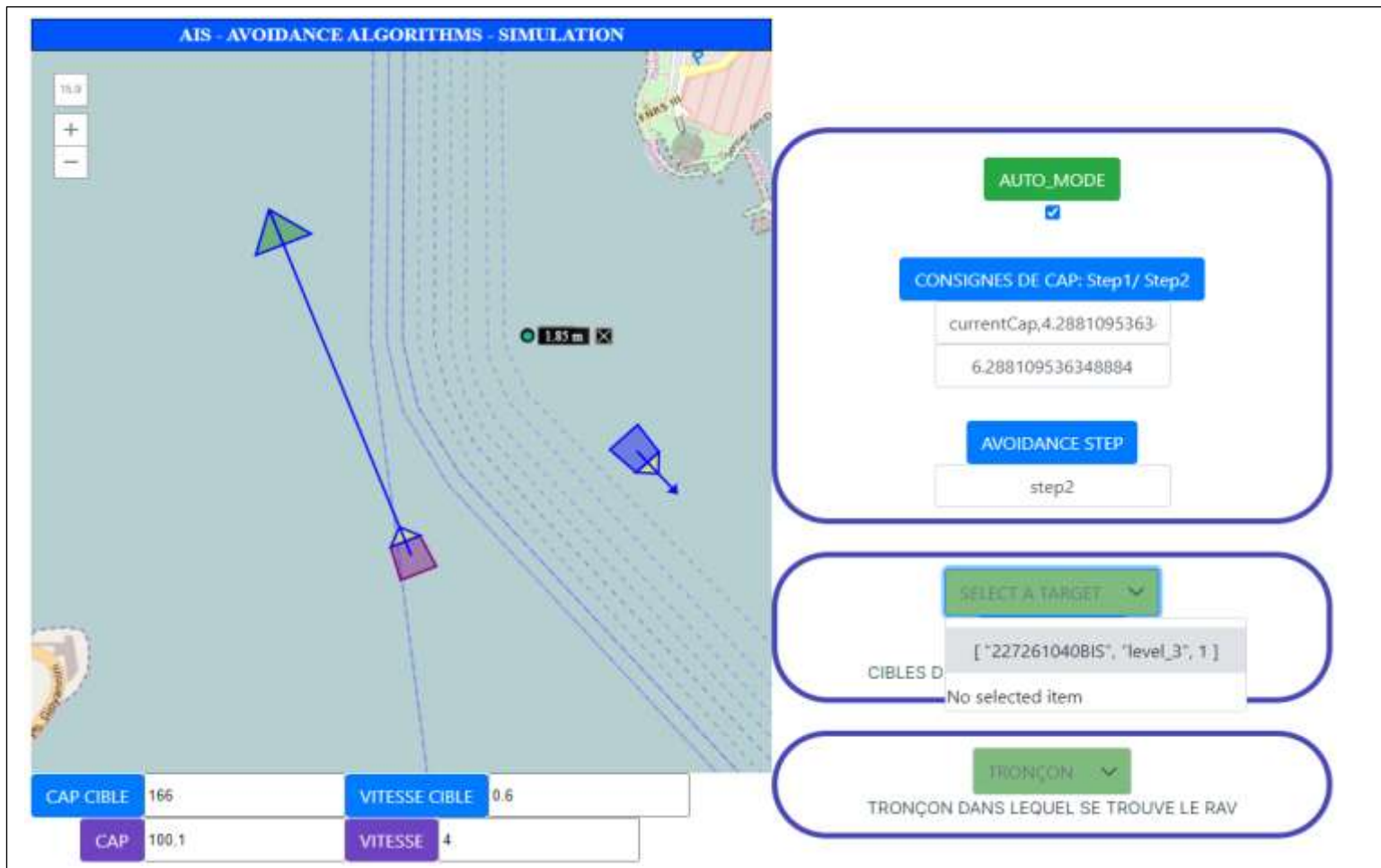


Figure 29 : Simulateur-une cible de tests.

#### Commentaires :

- Le bloc SELECT A TARGET : contient les cibles à risque à travers un champ [**'mmsi', niveau de criticité, ordre de priorité**]. Ce champ est mis à jour lorsque les données sont reçues.

Ici, j'isole la cible à éviter en cliquant sur la ligne qui lui correspond.

- La cible est dessinée en triant les trames reçues sur le topic `rav/local/instruments/ais/<trackX>`.



En effet, Je n'entre dans le code que si le mmsi associé à une trame reçue est égal à celui de la cible que je teste (alors pour les tests sur une cible, je fixe ce numéro mmsi d'avance via le bloc SELECT A TARGET).

Ainsi, lorsque je vais m'intéresser à toutes les cibles détectées par l'AIS, il faudra également isoler la cible à éviter de par son numéro mmsi.

- La flèche sur le RAV représente son cap (dans les tests cap=heading) : En gardant ce cap, le Rav se trouvera à l'extrémité de la flèche 5 min plus tard (défini ainsi dans l'application servant à dessiner le RAV).

- Le bouton **“Auto\_Mode”** permet lancer l'évitement de collision de façon automatique, lorsque les conditions sont réunies pour cela.

- Les consignes de cap :

Au step1 il s'agit du cap optimal pour éviter la cible, au step2 il s'agit du cap permettant un mouvement circulaire (rayon=rayon du cercle orange vu plus haut).

L'information **“CurrentCap, 4.288...”** indique que le cap actuel (4.288...) permet déjà de passer à une distance de sécurité de la cible. Donc le cap optimal (calculé au step1) vaut le cap actuel.

S'il fallait éviter la cible côté bâbord (gauche), on aurait une consigne de la forme **“Left, valeur du cap”**.

S'il fallait éviter la cible côté tribord (droit), on aurait une consigne de la forme **“Right, valeur du cap”**.

- Le bloc Tronçon sera abordé plus tard.

- Le bloc AVOIDANCE STEP : indique l'étape de l'évitement en cours. Ici le RAV est dans le **step2**, c'est-à-dire qu'il contourne la cible après l'avoir détectée.

- ❖ Je dessine ensuite toutes les cibles détectées par l'AIS (cibles en jaune). La dimension des navires sur la carte n'est pas à l'échelle réelle.



Figure 30 : Simulateur- cibles détectées par l'AIS

### Commentaires :

- Dans le cas où je m'intéresse à toutes les cibles détectées par l'AIS, j'utilise un bloc "SELECT A TARGET" présentant toutes les cibles potentiellement à risque.

Ensuite, je coche **premièrement le bouton “Auto\_Mode”**, puis je sélectionne la ligne correspondant à la cible d'ordre de priorité 1. Ainsi, dès que les conditions sont réunies, l'algorithme d'évitement en 4 étapes se déclenche sur cette cible uniquement.

- Un moyen plus autonome de lancer l'évitement est de ne pas cocher (sur la DropDown) manuellement la ligne correspondant à la cible la plus critique, mais **d'isoler cette cible d'une autre manière (code commenté dans le module “MyAppMaptalk\_obstacleAvoidance\_switchCase.vue”) :**

En effet, les données de toutes les cibles sont reçues au niveau du simulateur, dans une fonction `client.on('message', ...)`. Cette fonction est utilisée pour définir un gestionnaire d'événement qui sera appelé chaque fois que le client reçoit un message via le protocole MQTT. Lorsque le client détecte l'arrivée d'un nouveau message, il déclenche l'événement "message", et le gestionnaire d'événement spécifié (la fonction passée en argument) est exécuté.

Dans cette autre option, la cible à éviter est isolée en n'entrant dans le gestionnaire d'événement que si le mmsi associé à la trame reçue est celui d'une trame d'ordre de priorité 1. Par la suite, l'algorithme d'évitement en 4 étapes se déclenche sur cette cible uniquement lorsque les conditions seront réunies (cible au niveau de criticité level\_2 ou level\_3).

- Sur le bloc SELECT A TARGET, on voit quelle cible est la plus critique :

Il s'agit de la cible dont le numéro mmsi est **247552000**. On peut vérifier cela en lisant les informations données par **le panel en bleu** qui apparaît lorsque le curseur de la souris survole une cible (la cible critique dans l'image précédente).

### Remarque :

Avec plusieurs cibles un problème se pose si les données reçues au niveau du simulateur sont reçues à une faible fréquence. Dans mon cas, je recevais les données décisionnelles (au niveau du simulateur) en provenance du topic `rav/local/instruments/ais/<trackX>` après décodage (via ais.js) et mise à jour des trames AIS (via aisData.js) : ainsi, pour les trames qui sont reçues toutes les 3 minutes par exemple,

leurs paramètres de prises de décisions associés (Exemple : ordre de priorité) ne sont pas correctement mis à jour. De ce fait, une trame peut apparaître (sur la dropDown 'select a target') à l'ordre de priorité 10 alors qu'elle est passée récemment à l'ordre de priorité 1, et l'évitement qui est censé se faire sur cette cible ne se fait pas car l'ordre de priorité n'a pas été mis à jour.

Il faut trouver un moyen de toujours envoyer les données décisionnelles au niveau du simulateur, à une certaine fréquence. Par exemple : pour chaque cible, au lieu d'envoyer

Cela est possible par exemple en envoyant toujours à une certaine fréquence et pour chaque cible, les données liées à la dernière trame mise à jour (en partant du module aisData.js qui réalise la mise à jour). Ainsi, on va au pire des cas considérer une cible comme immobile et l'avoir en vue tout de même (au lieu de son absence au niveau du simulateur). Lorsque cette trame sera mise à jour, les paramètres associés à la cible le seront aussi et on répète l'envoi périodique des données.

C'est ce soucis (fréquence des données décisionnelles en provenance de [rav/local/instruments/ais/<trackX>](#)) qui fait la différence entre la simulation avec une cible (où les données de la cible sont envoyées constamment) et celle avec toutes les autres.

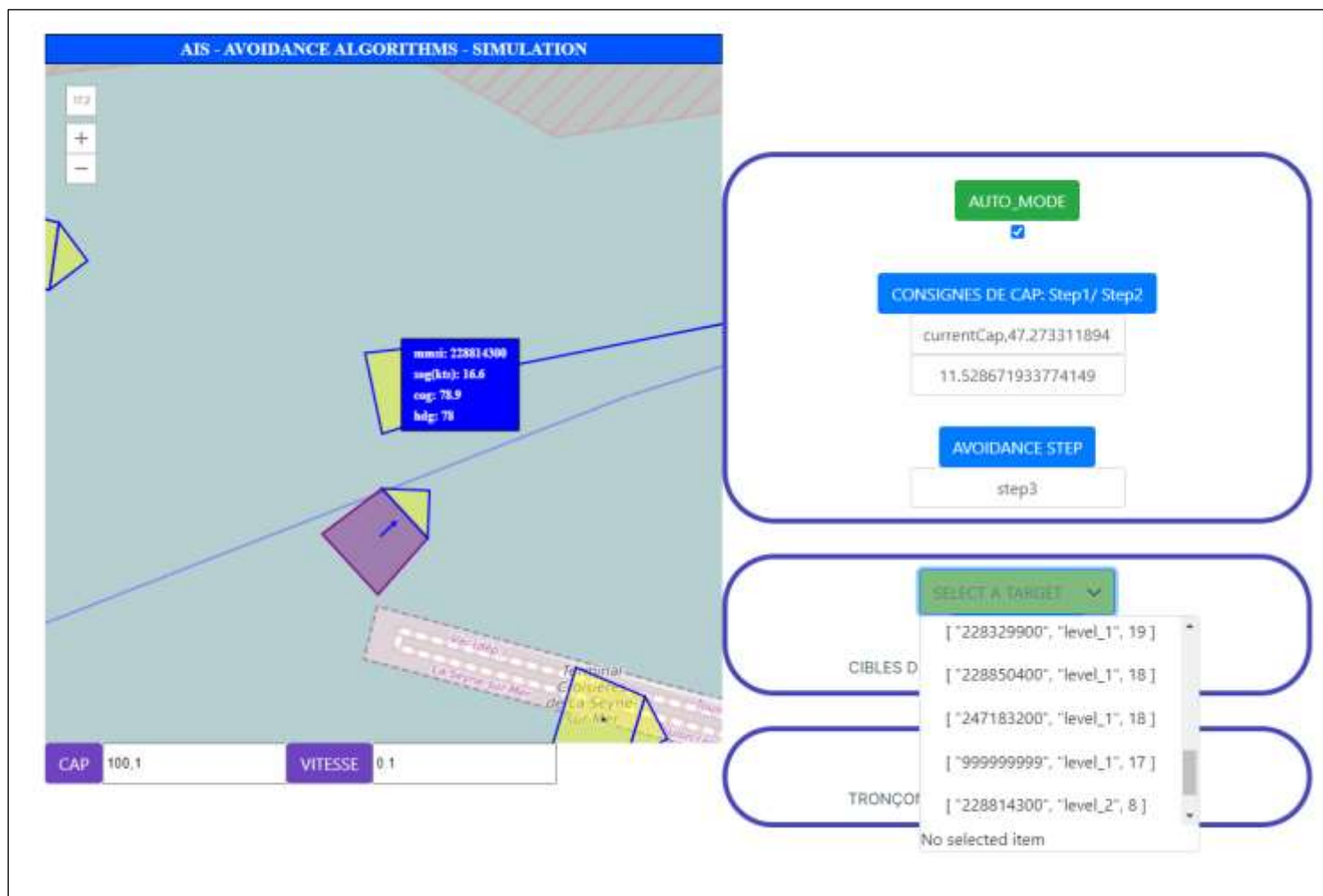


Figure 31 : Exemple- problème de rafraîchissement des données décisionnelles.

La cible de numéro mmsi=228814300 (sur le panel en bleu et sur la dernière ligne de la DropDown “SELECT A TARGET”) est d’ordre de priorité 1 en réalité, mais elle est toujours présente à l’ordre de priorité 8 pendant un certain temps (lors du test effectué) : ceci empêche de déclencher l’évitement sur cette cible. De même, 2 cibles ont l’ordre de priorité 18 : une l’a été depuis un certain temps car elle n’a pas été mise à jour.

#### IV.5.2 Algorithme d’évitement de collision, cas d’un suivi de tronçons « en mission ».

Dans le logiciel de contrôle commande des drones, il y a un **mode pilotage en waypoints**. Dans ce mode de pilotage, les drones doivent suivre des points bien définis sur une carte. C’est-à-dire que le RAV doit suivre des **tronçons** spécifiques (un tronçon est défini par deux points sur une carte).

Je m’intéresse ici aux algorithmes « en mission ».

Pour ce faire, il est question de pouvoir connaître dans quel tronçon se trouve le RAV (déterminé par une fonction **belong\_To\_tronçon**) et de connaître, entre autres, quel tronçon est le plus proche du RAV (une fonction **distance\_To\_tronçon** détermine la distance du RAV à un tronçon donné). Ci-dessous une illustration du principe autour des tronçons :

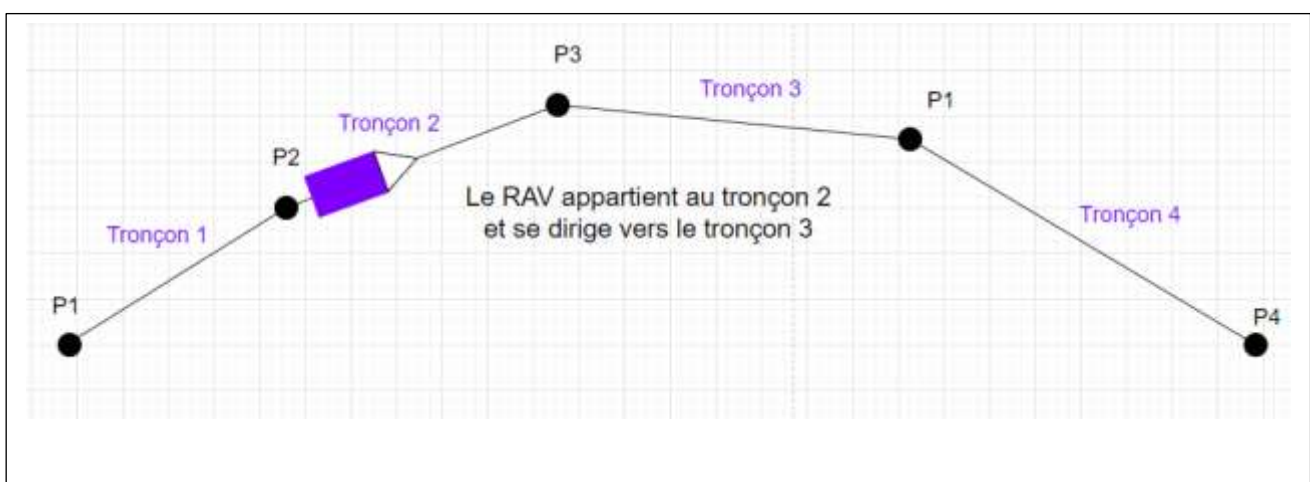


Figure 32-a : Déplacement suivant des tronçons



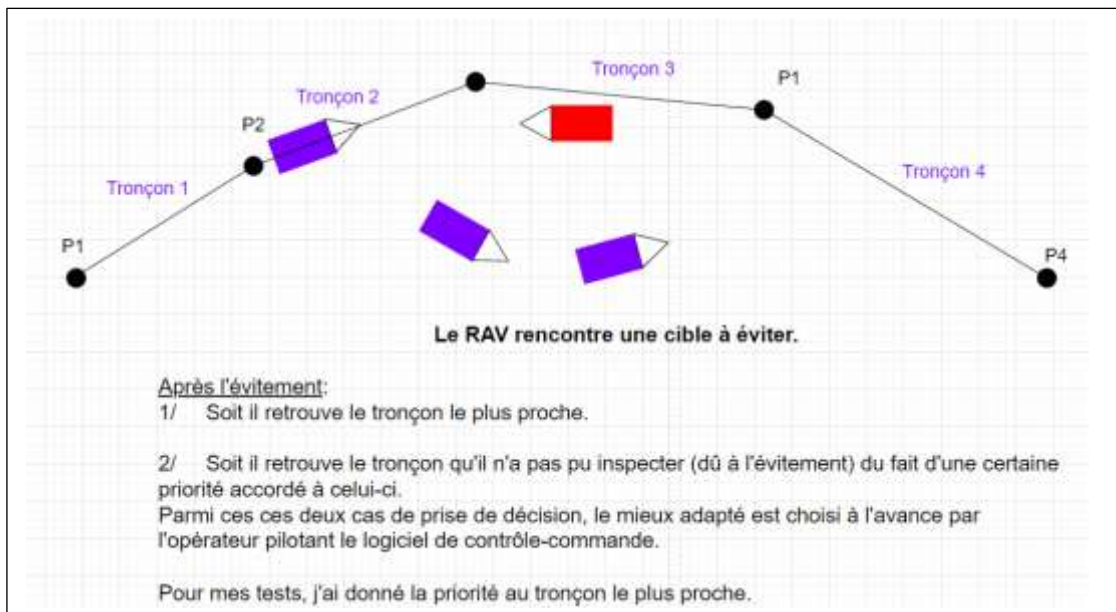


Figure 32-b : Déplacement suivant des tronçons

⇒ Dans un premier temps, j'ai testé le fonctionnement de l'algorithme donnant le tronçon auquel le RAV appartient, ainsi que la distance du RAV à un tronçon donné.

Un aperçu de ce que j'obtiens sur le simulateur.

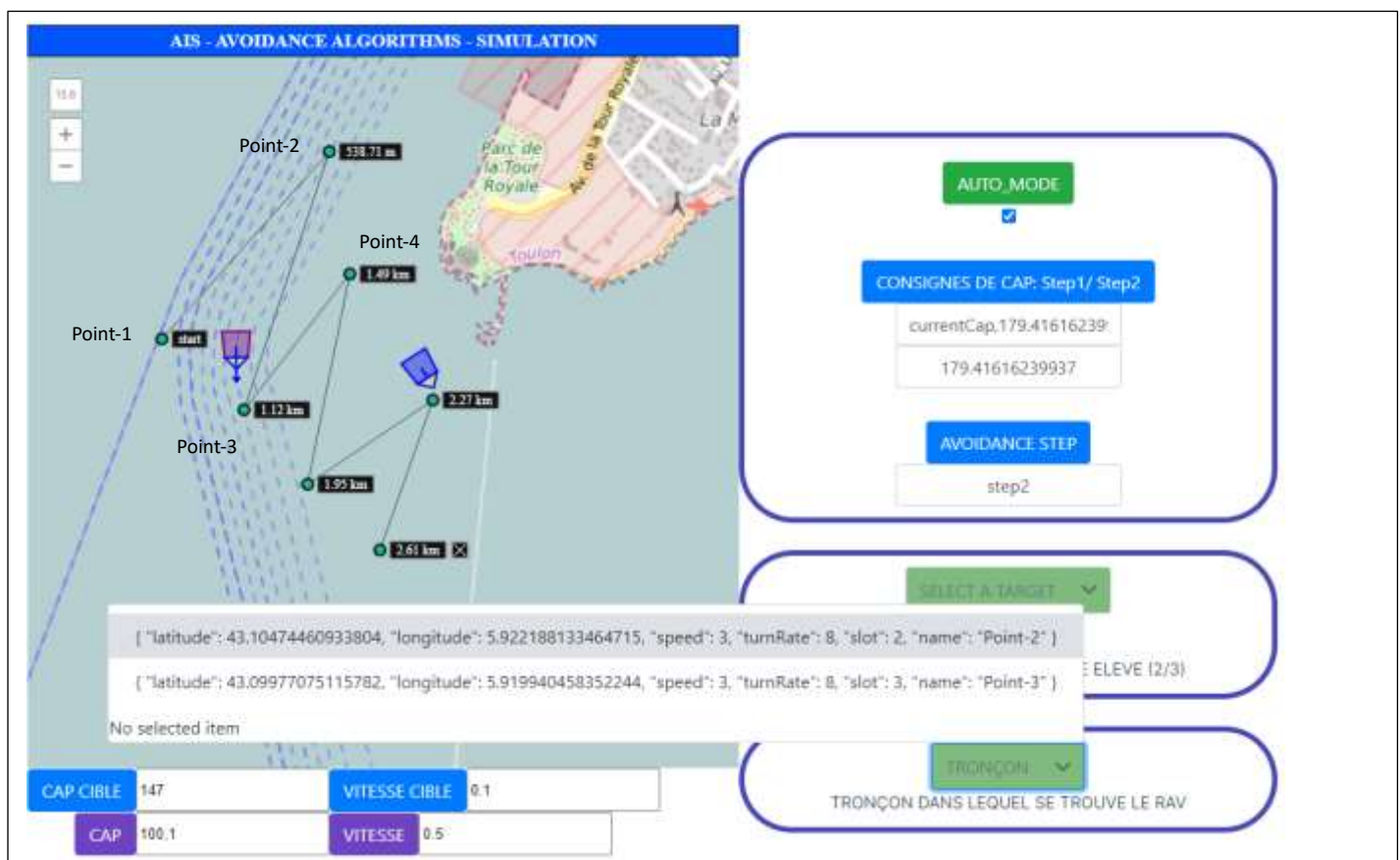


Figure 33 : Tronçons-tests

**Commentaires :**

- Ici, le RAV se trouve dans le tronçon formé par les Points 2 et 3 sur la carte (numéros attribués selon l'ordre dans lequel je les ai positionnés sur la carte). Ces points sont enregistrés avec certaines informations (latitude, longitude, speed, turnRate, slot, name) dont le 'slot'. Cette variable **slot** définit la priorité accordée à un point : Ainsi, via le logiciel de contrôle-commande le RAV se dirige vers chaque point dans l'ordre croissant des slots.

C'est une variable utilisée dans le logiciel de contrôle-commande et définie pour chaque point par l'utilisateur pilotant le logiciel : dans mon cas j'attribue les slot (donc une priorité aux points, selon l'ordre dans lequel ils ont été positionnés sur la carte).

- ⇒ Par la suite il est question de tester **l'algorithme d'évitement « en mission »** : Imposer au RAV de suivre des tronçons définis et s'il y a une cible à risque, éviter la cible par l'algorithme de base (step1/step2/step3/safeNavigation). Ensuite, retrouver le tronçon le plus proche.
- ⇒ Je n'ai pas terminé les tests de cet algorithme. Cependant, les fonctions (implémentées dans le module **aisCpa\_functions2.js**) donnant la distance à un tronçon ainsi que le tronçon auquel le RAV appartient, ont été testées et peuvent être utilisées. Sans oublier, les fonctions de recherche de cap optimaux (implémentées dans **aisCpa\_functions.js** (fonctions exportables dans des modules Node.js) et **aisCpa\_functions2.js** (fonctions exportables dans des modules Vue.js)).

## V. PERSPECTIVES

### V.1 Restant à faire...

Les données essentielles de suivi des cibles détectées par l'AIS sont accessibles via le broker mqtt du drone 1. Par la suite, il sera question d'utiliser les modules, fonctions et paramètres de prise de décision déterminés ainsi que d'adapter les algorithmes d'évitement d'obstacles au système de contrôle-commande.

L'intégration dans les autres RAV et les tests en situation réelle seraient encore plus concluants.



Il faudrait aussi vérifier que les trames AIS arrivent à une fréquence correcte. J'ai eu l'impression de recevoir certaines trames à des intervalles de plus de 5 minutes. Dans ce cas, l'algorithme d'évitement s'applique difficilement sur une telle cible : c'est ce que j'ai constaté sur le simulateur.

## V.2 A améliorer

- L'algorithme d'évitement de collision « en mission » tient en partie sur l'ordre de priorité (variable slot) accordé aux points de destination. Lorsque les 'slots' ont été bien déterminés à l'avance, le RAV aura pour destinations les différents points dans l'ordre des 'slots' croissant.

La prise de décision après un évitement doit être clairement énoncée à l'avance : est-ce que RAV repassera par des tronçons qu'il a évité ? si oui est-ce qu'il le fait immédiatement après l'évitement de collision ou doit-il d'abord terminer les tronçons les plus proches et revenir sur ces tronçons non inspectés ?

- Dans le simulateur, j'effectue l'évitement en isolant la cible (par son mmsi) à éviter via une DropDown (SELECT A TARGET). Cela m'a permis de tester en même temps la criticité et l'ordre de priorité, de visu. Pour améliorer l'évitement automatique, il faut un test en début de code que j'ai proposé et commenté dans le module **MyAppMaptalk\_ObstacleAvoidance\_switchCase.vue** (gestionnaire d'évènement du simulateur client.on("message"), 2 premiers blocs de commentaires avec "IF") :

Est-ce qu'on a une trame d'ordre de priorité 1 ? Si oui, enregistrer son mmsi dans une base de données locale (Exemple : "storeNav.js" dans mon cas).

Les prochaines fois qu'on entre dans le code des algorithmes d'évitement, on doit rentrer dans le programme si et seulement si le mmsi associé à la trame reçue est celui enregistré précédemment (celui de la trame à éviter).

Ainsi, dès qu'une cible sera de niveau level\_2 ou level\_3, et d'ordre de priorité 1 l'algorithme pourra être déclenché directement sur celle-ci : c'est ce que j'ai proposé.

- L'algorithme d'évitement basé sur la machine à états finis (step1/step2/step3/safeNavigation) est conçu sur la base d'un mouvement du RAV à vitesse non nulle. Le RAV ne s'arrête que sur un point de destination. Sinon, s'il n'y a pas de point de destination et qu'il y a une cible à éviter, il la contourne : on pourrait bien décider de s'arrêter plutôt en situation réelle. Car à vitesse constante, le RAV ne peut que contourner la cible, ce qui n'est pas toujours optimal.
- Voir Remarque - fin du paragraphe IV.5.1 (simulation avec plusieurs cibles).

## CONCLUSION

A l'issu de ce stage assistant-ingénieur au sein de la société KIETTA, je tire un bilan positif et une expérience enrichissante tant sur le plan académique que professionnel. J'ai eu l'occasion d'appréhender le métier d'ingénieur, de comprendre les attentes et le quotidien d'un ingénieur, plus particulièrement, mécatronique. J'ai travaillé tout au long de ce stage sur le volet programmation et informatique embarquée de mon domaine d'étude et je ressors très enrichi. Notamment, par la connaissance d'un nouveau moyen de pilotage de systèmes mécatroniques complexes (les drones marins de surface dans ce cas précis) par des outils de développement web, des connexions réseaux, et de nombreux outils de programmations et informatique embarquée.

Le but était d'intégrer un récepteur AIS dans un drone de surface. Les drones étant en maintenance tout au long du stage, l'intégration matérielle et les tests en situation réelle n'ont pas été effectués. Cependant, les données AIS utiles, acquises et traitées ont été intégrées dans le système de contrôle-commande des drones : ainsi, les cibles détectées peuvent déjà être repérées/dessinées, suivies et mises à jour sur un drone, et les paramètres de prise de décision qui leur sont associés sont également disponibles via le broker mqtt de ce drone (drone 1). Concernant le simulateur qui m'a permis de tester certains algorithmes, il ne peut être utilisé directement par KIETTA car le système de contrôle-commande possède déjà un mode de simulation. Les modules et algorithmes utiles aux évitements d'obstacles pourront être récupérés pour une intégration logicielle directement dans le contrôle-commande.

Je suis reconnaissant envers KIETTA de m'avoir permis d'effectuer ce stage grâce auquel mon expérience professionnelle se développe un peu plus. Après ce stage très orienté software, je compte acquérir par la suite encore plus d'expérience au niveau hardware, en mécatronique ou plus particulièrement en robotique.

## BIBLIOGRAPHIE

- [Réf. 1] : <https://www.framboise314.fr/utiliser-luart-port-serie-du-raspberry-pi-4/>
- [Réf. 2] : <https://www.icom-france.com/fr/catalogue/comprendre-l-ais.php>
- [Réf. 3] : <https://www.bateaux.com/article/23699/ais-une-revolution-electronique-securite-expliquee>
- [Réf. 4] : Académie de Nantes. « Protocole NMEA183 ».
- [Réf. 5] : <https://gpsd.gitlab.io/gpsd/AIVDM.html>
- [Réf. 6]: The Closest Point of Approach (CPA). De la théorie à la pratique ; Annaig Labornez1
- [Réf. 7]: CPA calculation method based on AIS position prediction; National Engineering Research Center for Water Transport Safety (WTSC)- Intelligent Transport Systems Research Center (ITSC), Wuhan University of Technology.
- [Réf. 8] : [https://receiverhelp.trimble.com/alloy-gnss/en-us/NMEA-0183messages\\_GBS.html](https://receiverhelp.trimble.com/alloy-gnss/en-us/NMEA-0183messages_GBS.html)
- [Réf. 9] : <https://www.aggssoft.com/ais-decoder.htm>
- [Réf. 10] : <https://nodejs.org/en/docs/guides/timers-in-node>
- [Réf. 11] : <https://www.npmjs.com/package/wgs84-util>
- [Réf. 12] : [https://devdocs.io/javascript/global\\_objects/string/replace](https://devdocs.io/javascript/global_objects/string/replace)
- [Réf. 13] : <https://www.w3schools.com/nodejs/default.asp>

## GLOSSAIRE

**AIS:** Automatic Identification System

**USV:** Unmanned Surface Vehicle

**RAV :** Recording Autonomous Vehicle

**Node.Js :** Node.js est un environnement d'exécution côté serveur construit sur le moteur JavaScript V8 de Google Chrome. Il permet d'exécuter du code JavaScript sur le serveur plutôt que dans un navigateur web.

**JavaScript :** langage de programmation informatique orienté objet couramment utilisé pour créer des effets interactifs dans les navigateurs Web.

**HTML :** HyperText Markup Language. C'est un système standardisé de balisage des fichiers texte pour obtenir des effets de police, de couleur, de graphisme et d'hyperlien sur les pages du World Wide Web.

**CSS:** Cascading Style Sheets. Il s'agit d'un langage de feuille de style utilisé pour décrire la présentation d'un document écrit en HTML ou XML.

**Vue.js :** Framework JavaScript open-source utilisé pour construire des interfaces utilisateur interactives et réactives côté client.

**MQTT:** Message Queuing Telemetry Transport. Il s'agit d'un protocole de messagerie léger, ouvert et basé sur la publication/abonnement, conçu pour les communications M2M (Machine-to-Machine) et IoT (Internet of Things).

**CPA:** Closest Point of Approach. Un point estimé dans lequel la distance entre deux objets, dont au moins un est en mouvement, atteindra sa valeur minimale.

**JSON :** Un fichier JSON (JavaScript Object Notation) est un format de données léger, facile à lire et à écrire, basé sur la notation des objets JavaScript. Un fichier JSON est composé de paires clé-valeur, où chaque clé est une chaîne de caractères et chaque valeur peut être de différents types de données.

**Bâbord :** C'est le côté gauche d'un navire ou d'un bateau lorsque l'observateur regarde vers l'avant. C'est également le côté qui est marqué en rouge sur les feux de navigation du bateau.

**Tribord :** C'est le côté droit d'un navire ou d'un bateau lorsque l'observateur regarde vers l'avant. C'est également le côté qui est marqué en vert sur les feux de navigation du bateau.

**Relèvement :** En navigation, le relèvement est l'angle formé entre la ligne de visée d'un objet ou d'un point fixe et une direction de référence, généralement le nord vrai. Le relèvement est utilisé pour déterminer la position du navire ou de l'observateur.

**Heading :** Terme anglais utilisé en navigation pour désigner le cap ou l'orientation d'un navire ou d'un aéronef par rapport au nord géographique.

**COG (Cap Over Ground)** : C'est le cap ou la direction vraie du navire par rapport au sol, exprimé en degrés. Le COG prend en compte le mouvement du navire dû aux courants et au vent.

**SOG (Speed Over Ground)** : C'est la vitesse du navire par rapport au sol, exprimée en nœuds. Le SOG prend en compte le mouvement du navire dû aux courants et au vent.

**DCPA (Distance of Closest Point of Approach)** : C'est la distance minimale la plus proche entre deux navires.

**TCPA (Time to Closest Point of Approach)** : C'est le temps estimé avant que les deux navires atteignent la distance minimale la plus proche (DCPA). Il est généralement exprimé en minutes ou en secondes.

**BCPA (Bearing of Closest Point of Approach)** : C'est l'azimut ou la direction par rapport au nord du point le plus proche entre deux navires au moment de la distance minimale la plus proche (DCPA).

## ANNEXES

### ❖ ANNEXE 1 : TOPO SUR UNE PREMIERE INSTALLATION DE LOGICIELS

#### ▪ REMOTE SSH

Depuis son PC, sous **Visual Studio Code** :

Télécharger **Remote SSH**, ajouter l'extension **VOLAR** (Vue Language Features) qui fournit une coloration syntaxique améliorée pour les fichiers Vue.js, mettant en évidence correctement les balises, les propriétés et les expressions Vue.js dans le code pour une meilleure lisibilité, se connecter à la carte Raspberry Pi.

Configuration SSH (Secure Shell) : Pour utiliser l'extension Remote SSH, il faut d'abord configurer une connexion SSH vers la machine distante. Cela implique de générer une paire de clés SSH (clé privée et clé publique) sur sa machine locale et de copier la clé publique sur la machine distante :

- Ajouter dans le **fichier ssh** du pi (carte Raspberry), depuis Remote ssh,

la clé se trouvant dans "**id\_rsa-pub**" (C:\Users\DOP1\ssh).

Si le dossier id\_rsa.pub n'existe pas encore il faut le créer via un terminal sur notre pc, par la commande : **ssh-keygen -t rsa**.

Si le fichier "authorized keys" n'existe pas sur RPI il faut le créer.

- si le pi refuse l'hôte lors d'une connexion, aller dans ce fichier et supprimer le KNOWN HOST...
- Pour générer la clé sur le pi, dans le fichier ./ssh/id\_rsa.pub :

Faire : **ssh-keygen**

- Ensuite, vous pouvez copier le contenu de votre fichier "id\_rsa.pub" provenant de votre PC et le coller dans le fichier "authorized\_keys" sur le Raspberry Pi.

#### ▪ Node.js , NPM (gestionnaire de paquets de Node.js), Vue.js :

**Installations sur la carte Raspberry Pi 4B**



- **sudo apt-get install npm** (<https://vuejs.org/guide/quick-start.html#creating-a-vue-application>).
- **node -v** et **npm -v** : pour vérifier si npm et node.js ont été téléchargés et leur version également.

- Il y a un souci en installant **Node.js** comme suit :

#### **sudo apt-get install nodejs:**

Le logiciel a installé la version 12 de nodejs au lieu de 18. Il faut alors se diriger sur

leur site et prendre la commande utile :

**curl -sL https://deb.nodesource.com/setup\_lts.x | sudo -E bash - | sudo apt-get install -y nodejs.**

On installe ainsi la version la plus récente.

- Installation de Vue.js :

**npm init vue@latest** (<https://vuejs.org/guide/quick-start.html#creating-a-vue-application>)

- Installer Volar sur le pi également (en allant dans les fichiers du projet vue créé : Exemple *Helloworld.vue*, celui-ci est en blanc et VS CODE propose d'ajouter l'extension qu'il faut).
- Dans le fichier du répertoire de projet créé, au niveau du fichier "package.json" compléter le champ 'dev' se trouvant dans "scripts" par : "dev" : "vite --host".
- **npm run dev** : Pour lancer le site web.

Pour vous déconnecter en toute sécurité de votre Raspberry Pi depuis Visual Studio Code, ouvrez un terminal dans Visual Studio Code. Si vous êtes connecté en SSH à votre Raspberry Pi, tapez 'exit' pour vous déconnecter de la session SSH. Si vous avez des processus en cours d'exécution dans le terminal, arrêtez-les en utilisant la commande Ctrl + C.

- Tapez "**sudo shutdown -h now**" pour arrêter proprement votre Raspberry Pi.

Attendez que le Raspberry Pi s'arrête complètement.

## ❖ ANNEXE 2: CONFIGURATION AMONT DU PORT SERIE

## ▪ Démarches nécessaires avant l'utilisation du port série

Configurez premièrement le Raspberry Pi pour utiliser l'UART. Par défaut, l'UART est utilisé pour la console série, vous devez donc le désactiver pour pouvoir l'utiliser à des fins de communication. Pour ce faire, suivez les étapes énoncées sur le lien : [\[Réf. 1\]](#) :

- Faire : **sudo nano /boot/cmdline.txt** (Supprimer "console=serial0,115200" ou "console=ttyAMA0,115200" : Afin de configurer le Raspberry Pi pour utiliser l'UART, car par défaut, l'UART est utilisé pour la console série, il faut donc le désactiver pour pouvoir l'utiliser à des fins de communication.)

- Dans **sudo nano /boot/cmdline.txt** :

Compléter par : enable uart, disable bluetooth ;

dtoverlay=disable-bt enable\_uart=1

ensuite en ligne de commande:

sudo systemctl disable hciuart (le système qui lance le Bluetooth);

sudo systemctl status bluetooth; sudo systemctl disable bluetooth

- Ensuite redémarrer le pi :

sudo reboot, et **laisser le fichier de config sur une seule ligne.**

- ls /dev/serial\*: voir les ports série

- J'ai installé raspi-serial et wiring pi depuis le tuto freenove :

En ligne de commande, faire :

sudo apt-get update

git clone https://github.com/WiringPi/WiringPi

cd WiringPi

./build

- gpio readall: Pour lire les GPIO

❖ ANNEXE 3 : BRANCHEMENT DE L'AIS

Figure : Branchement AIS 1



Figure : Branchement AIS 2



## ❖ ANNEXE 5 : CHAMPS DES TRAMES !AIVDM

Un paquet de données AIVDM typique :

`!AIVDM,1,1,,B,177KQJ5000G?tO`K>RA1wUbN0TKH,0*5C`

**Le champ 1**, !AIVDM, l'identifie comme un paquet AIVDM.

**Le champ 2** (1 dans cet exemple) est le nombre de fragments dans le message en cours d'accumulation. La taille de la charge utile de chaque phrase est limitée par le maximum de 82 caractères de NMEA 0183, il est donc parfois nécessaire de diviser une charge utile sur plusieurs phrases de fragment.

**Le champ 3** (1 dans cet exemple) est le numéro de fragment de cette phrase. Ce sera à base unique. Une phrase avec un nombre de fragments de 1 et un nombre de fragments de 1 est complète en soi.

**Le champ 4** (vide dans cet exemple) est un ID de message séquentiel pour les messages à plusieurs phrases.

**Le champ 5** (B dans cet exemple) est un code de canal radio. L'AIS utilise le côté haut du duplex à partir de deux canaux radio VHF : le canal AIS A est de 161,975 MHz (87B) ; Le canal AIS B est de 162,025 MHz (88B). Dans la nature, les codes de canal « 1 » et « 2 » peuvent également être rencontrés ; les normes ne prescrivent pas une interprétation de celles-ci mais c'est assez évident.

**Le champ 6** (177KQJ5000G?tO`K>RA1wUbN0TKH dans cet exemple) est la charge utile des données. Nous décrirons comment décoder cela dans les sections suivantes.

**Le champ 7** (0) est le nombre de bits de remplissage requis pour remplir la charge utile de données à une limite de 6 bits, allant de 0 à 5. De manière équivalente, soustraire 5 de cela indique combien de bits les moins significatifs du dernier quartet de 6 bits dans la charge utile des données doit être ignorée.

Le suffixe séparé par \* (\\*5C) est la somme de contrôle d'intégrité des données NMEA 0183 pour la phrase, précédée de "\*". Il est calculé sur la phrase entière, y compris la balise AIVDM mais en excluant le "!".



À titre de comparaison, voici un exemple de phrase à plusieurs fragments avec un champ d'ID de message non vide :

!AIVDM,2,1,3,B,55P5TL01VIaAL@7WKO@mBplU@<PDhh000000001S;AJ::4A80?4i@E53,0\*E

!AIVDM,2,2,3,B,1@000000000000,2\*55

## ❖ ANNEXE 6 : DECODAGE DE TRAMES AIS- NODE.JS

```
1  const raspi = require('raspi');
2  const Serial = require('raspi-serial').Serial;
3  //const Decoder = require('ais-decoder-ts').Decoder;
4  const AisDecoder = require('ais-stream-decoder').default
5  const aisDecoder = new AisDecoder();
6
7  const mqtt = require('mqtt');
8  const client = mqtt.connect('mqtt://192.168.210.103:1883/');
9
10
11 function publishData(data,topic) {
12     client.publish(topic, data, (err) => {
13         if (err) {
14             console.error('Erreur lors de la publication des données'+
15                 'Uart reçues depuis ais.js:', err);
16         }
17     });
18 }
19
20
21
22 // PORT SERIE //
23 raspi.init(()=>{console.log('Raspberry Pi initialized.')});
24 const serial = new Serial({
25     portId: '/dev/serial0',
26     baudRate: 38400 ,
27     dataBits: 8,
28     stopBits: 1,
29     parity: 'none'
30 });
31 serial.open(() => {console.log('Serial port opened.')});
32
33
34
35
36 // RECEPTION - DECODAGE SUR LE PORT SERIE
```

Figure : Décodage de trames AIS – Node.js (a)



```
36 // RECEPTION - DECODAGE SUR LE PORT SERIE
37 let receivedFlowData = ''
38 serial.on('data', (data) => {
39
40     data = data.toString()
41     receivedFlowData = receivedFlowData + data
42     if (!receivedFlowData.includes('\r\n')) {
43         return
44     }
45     if (!receivedFlowData.startsWith('!AIVDM')) {
46         receivedFlowData = ''
47         return
48     }
49
50
51     //
52     let startOfIncompleteData
53     if(receivedFlowData.split('\r\n')[1] !== ''){
54         startOfIncompleteData=receivedFlowData.split('\r\n')[1]
55         receivedFlowData=receivedFlowData.split('\r\n')[0]
56
57         try{aisDecoder.write(receivedFlowData)}
58         catch(error){ console.log('caught error');return}
59
60         receivedFlowData=startOfIncompleteData;
61         //à cette chaîne de caractères seront ajoutées les suivantes en entrant dans
62         // serial.on('message',...) la prochaine fois
63     }
64     else{
65         receivedFlowData=receivedFlowData.replace(/\r\n/g, '');
66
67         try{aisDecoder.write(receivedFlowData)}
68         catch(error){ console.log('caught error') ;return}
69
70         receivedFlowData = ''
71     }
72
73     return
74 }
```

Figure : Décodage de trames AIS – Node.js (b)

```
78 // DECODAGE
79 aisDecoder.on('data', (decodedTrame) => { //donnée décodée renvoyée en '{}'
80     let mmsi_=JSON.parse(decodedTrame)["mmsi"]
81     let AisTopic='rav/raw/instruments/ais/'+mmsi_
82     let AisTopic2='rav/raw/instruments/ais/'+ 'TESTESTESTEST'
83
84
85     //Conserver les données utiles
86     let decodedTrameToSend=JSON.parse(decodedTrame)
87     const Usefulkeys = [
88         'mmsi',
89         'type',
90         'lat',
91         'lon',
92         'speedOverGround',
93         'courseOverGround',
94         'heading',
95         "dimBow",
96         "dimStern",
97         "dimPort",
98         "dimStarboard",
99     ];
100     for (let key in decodedTrameToSend) {
101         if (! (Usefulkeys.includes(key)) ) {
102             delete decodedTrameToSend[key];
103         }
104     }
105     //Stream decoder renvoie 'null' pour des valeurs nulles (exemple: COG ET SOG).
106     //Pour pouvoir effectuer des calculs sans soucis:
107     if(decodedTrameToSend.speedOverGround==null){decodedTrameToSend.speedOverGround=0}
108     if(decodedTrameToSend.courseOverGround==null){decodedTrameToSend.courseOverGround=0}
109     if(decodedTrameToSend.heading==null){decodedTrameToSend.heading=0}
110
111     publishData(JSON.stringify(decodedTrameToSend),AisTopic)
112     publishData( JSON.stringify({'mmsi':'testestestes', 'lat':10, 'lon':11}), AisTopic2)
113 }
```

Figure : Décodage de trames AIS – Node.js (c)