

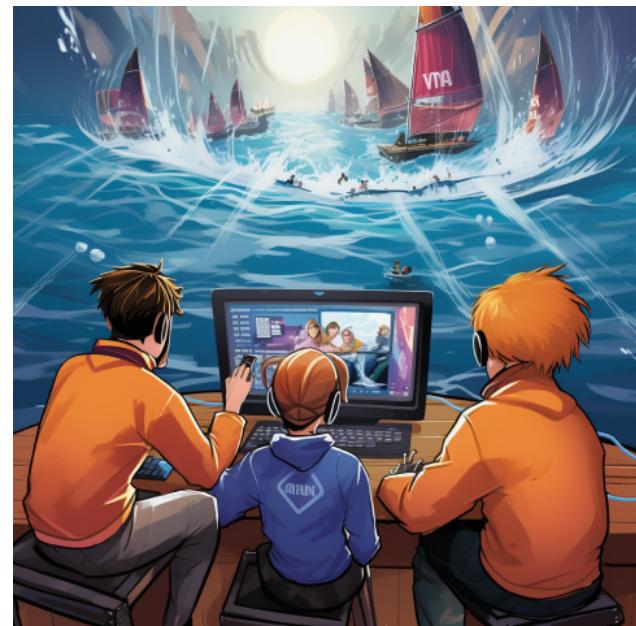
# RAPPORT DE PROJET

## AQUA.BOT

Groupe : TheBoys

Membres :

- Noël JUMIN
- Nathanaël BLAVO- BALLARIN
- Jules SIARD



TOULOUSE  
**INP N7**

**CESI**  
ÉCOLE D'INGÉNIEURS

**INSA** | INSTITUT NATIONAL  
DES SCIENCES  
APPLIQUÉES  
TOULOUSE



## Remerciements

Avant tout développement sur ce défi technique à forte plus value pédagogique, il me paraît opportun de commencer ce rapport par des remerciements, aux acteurs principaux de ce défi, à ceux qui nous ont énormément appris, à ceux qui ont fait de ce défi un moment très profitable et à ceux sans qui, ce défi n'aurait pas eu lieu.

M. Clément RAVARD, Chef de projet Open Innovation chez Schoolab, de nous avoir contacté sur LinkedIn pour participer à ce projet. Sans lui, nous n'aurions sans doutes jamais pris connaissance de ce défi d'envergure.

L'ensemble des acteurs du défi : les employés de SIREHNA, de nous proposer un défi d'une aussi grande qualité. En effet, grâce à leurs conseils, leurs points hebdomadaire et leur réactivité, nous avons pu monter en compétence dans un langage de programmation (Python). Également, nous avons pu monter en compétence en ROS2, une API très utile dans le monde de la robotique.

## Sommaire

<b>Remerciements</b>	<b>3</b>
<b>Sommaire</b>	<b>3</b>
<b>1. Phase d'observation</b>	<b>4</b>
1.1. Présentation des membres de l'équipe.	4
1.2. Déroulement du travail sur une semaine	4
1.3. Les outils utilisés	5
1.4. Le suivi de projet	5
<b>2. Phase de prototypage</b>	<b>6</b>
2.1. Step 1 : Reach zone	6
2.2. Step 2 : Patrol	7
2.3. Step 3 : Follow	8
<b>3. Difficultés rencontrées</b>	<b>10</b>
3.1. Difficultés résolues	10
3.2. Difficultés non-résolues	10
<b>CONCLUSION</b>	<b>10</b>

## 1. Phase d'observation

Cette partie aura pour but de préciser pourquoi et comment le groupe TheBoys a fonctionné lors de ce défi pour travailler en équipe dans le but d'être le plus efficace et productif.

### 1.1. Présentation des membres de l'équipe.



**Nathanaël BLAVO** : Développeur & Expert Python

École : CESI Toulouse    Filière : Systèmes-embarqués

Rôle :

- Appui technique Python3 ;
- Acquisition et traitement des données des capteurs ;
- Responsable reconnaissance des objets maritimes (machine learning).

[\*\*FIGURE 1 : CRÉDIT PHOTO H.RESSAYRES / CD31\*\*](#)

**Noël JUMIN** : Développeur & Expert ROS2

École : INSA Toulouse    Filière : Systèmes-embarqués

Rôle :

- Appui technique pour le ROS2 ;
- Acquisition et traitement des données des capteurs ;
- Garant de la motorisation et du module d'évitement / tracking.

[\*\*FIGURE 2 : PHOTO DE NOËL JUMIN\*\*](#)



**Jules SIARD** : Chef de projet & Appui technique

École : INP ENSEEIHT    Filière : Mécanique des fluides, énergétiques & environnement.

Rôle :

- Appui technique méthodes mathématiques ;
- Définition du trajet prévisionnel ;
- Rédactions des livrables.

[\*\*FIGURE 3 : CRÉDIT PHOTO F.ESTIVALS\*\*](#)

Les membres du groupes sont amis en dehors du contexte professionnel\*

### 1.2. Déroulement du travail sur une semaine

De manière générale, le groupe se réunit au minimum une demi-journée par semaine pour travailler ensemble et mettre en commun le travail réalisé chacun de son côté en autonomie. De plus, comme signifié, le groupe travaille et regarde les replay des réunions en autonomie la semaine. Les réunions de travail se déroulent soit chez Noël JUMIN ou chez Nathanaël BLAVO car ils bénéficient d'une connexion WIFI. Le groupe s'assoit autour d'une table et travaille en flex office.

## 1.3. Les outils utilisés

Cette sous-partie a pour but de présenter les outils que le groupe a utilisé pendant le défi. Deux types d'outils numériques sont à distinguer : les techniques et de gestion de projet. Par ailleurs, chacun des membres du groupe possède un Laptop avec Linux comme OS.

### 1) Les outils numériques techniques :

- **Gazebo** comme environnement de simulation numérique ;
- **RQT** pour afficher les topics ROS2 ;
- **RVIZ2** pour afficher les données acquis (pour voir le fonctionnement du LIDAR) ;
- **Visual Studio Code** comme environnement de développement ;
- **FoxGlove** pour afficher les données acquis (habitué à travailler avec cet outil) ;
- **YoloV5** comme outil de machine learning ;
- **Google Collaborative** pour entraîner plus rapidement notre IA.

### 2) Les outils numériques de gestion de projet :

- **Discord** pour partager des informations et des fichiers.
- **Snapchat** pour échanger sur les avancées et programmer les réunions de travail.
- **GitHub** pour faire notre versionnage et partager nos programmes.
- **GANTT** pour avoir un œil sur l'avancé des tâches.
- **La suite microsoft** pour plusieurs tâches.

## 1.4. Le suivi de projet

Pour être tenu au courant de l'avancée du projet au cours des deux mois, un diagramme de GANTT a été mis en place. Ce dernier a été réalisé sur « GANTT ONLINE ».

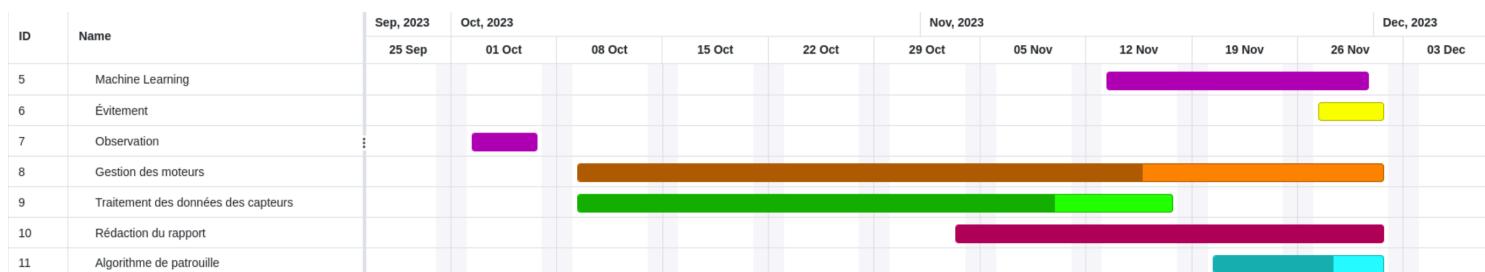


FIGURE 4 : DIAGRAMME DE GANT

## 2. Phase de prototypage

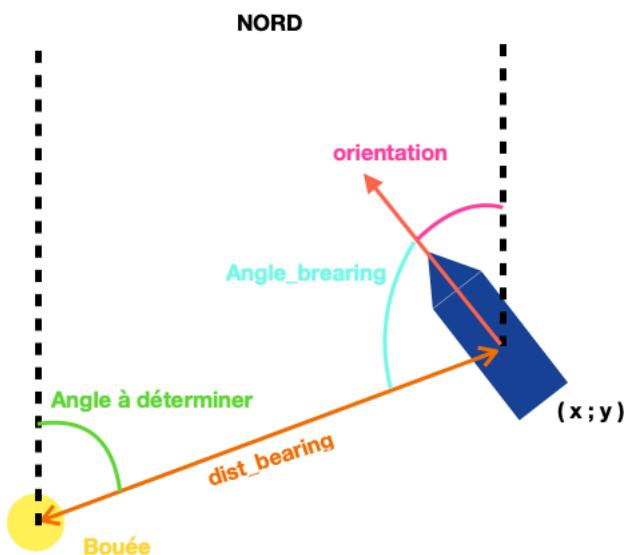
Cette partie a pour but de préciser le contenu de nos algorithmes et d'expliciter notre cheminement intellectuel et technologique.

### 2.1. Step 1 : Reach zone

Au commencement, le bateau doit rejoindre la bouée dans un environnement ne contenant ni ennemi, ni allié, ni obstacle. Pour cela, les données d'angle et de distance de la bouée par rapport au bateau sont acquises depuis le topic :

`/wamv/sensors/acoustics/receiver/range_bearing`

Ainsi, à partir de ces données, le bateau est en mesure de s'orienter et d'avancer vers la bouée.



**FIGURE 5 : SCHEMA DESCRIPTIF DE L'OBTENTION DE L'ANGLE À DÉTERMINER**

## 2.2. Step 2 : Patrol

Une fois que le bateau a réussi à atteindre la zone de la bouée, il faut maintenant réfléchir à la manière la plus efficace pour quadriller la carte.

Pour ce faire, la solution adoptée est la suivante : le bateau parcourra une spirale d'Archimède autour de la bouée. Cette figure a pour avantage de parcourir uniformément une zone en réalisant des « tours » autour d'un point (la bouée dans notre cas). Par ailleurs, la distance D entre sillons est toujours constante (on la fixera à 66 mètres dans notre cas).

$$\begin{cases} x(u) = -\frac{Du}{2\pi} \sin(u) \\ y(u) = \frac{Du}{2\pi} \cos(u) \end{cases} \quad \text{avec } u = [\frac{3\pi}{2}; 19\pi]$$

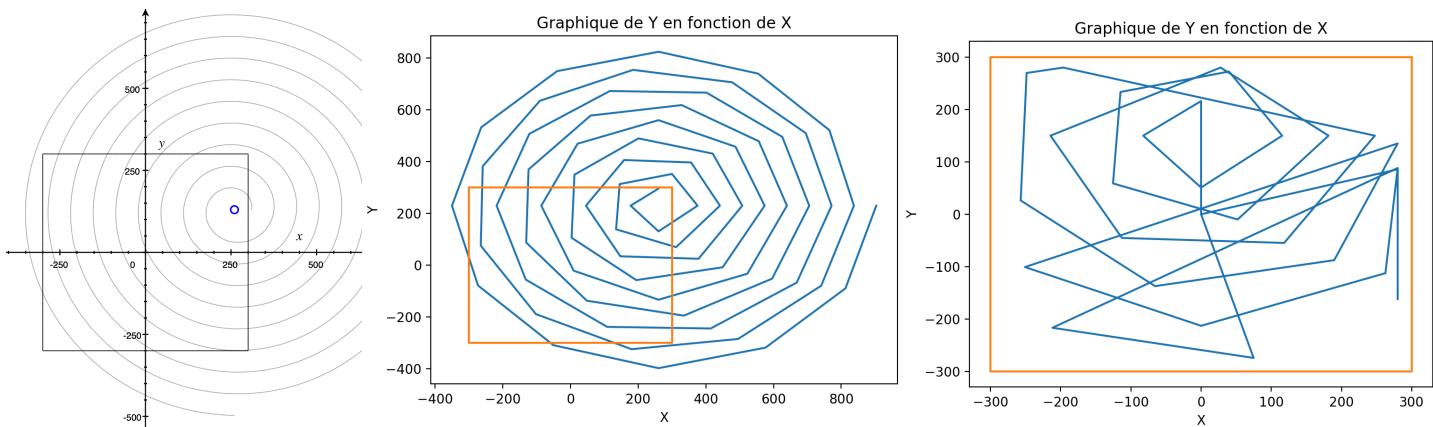
**FIGURE 6 : ÉQUATION PARAMÉTRIQUE DE LA SPIRALE D'ARCHIMÈDE**

Maintenant que le modèle théorique est fixé, il faut le simplifier car le bateau n'est pas en capacité de réaliser des trajectoires de marche en arc de cercle. La solution adoptée est de tronquer les périodes de la spirale en plusieurs segments. La première période sera scindée en 4 segments, la deuxième en 5 segments et la N-ième en N+3 segments.

Enfin, un nœud « patrol node » vient réajuster / supprimer les points en dehors de la carte en faisant des régressions linéaires des points voisins en dehors de la carte.

De ce fractionnement en segments, seuls les points sont connus. Ils sont déterminés grâce à l'algorithme Python « spirale.py » disponible en annexe. À présent, un dernier problème se pose, il faut supprimer et réajuster les segments en dehors de la carte.

Pour terminer, les deux listes « X\_c et Y\_c » contenant les coordonnées de notre spirale tronquée sont utilisées pour diriger le bateau après avoir atteint la zone de la bouée.



**FIGURE 7 : CHRONOLOGIE DE LA DÉFINITION DU TRAJET PRÉVISIONNEL**

Cependant, par manque de temps, ce code n'a pas été implémenté dans le programme final.

## 2.3. Step 3 : Follow

Pour suivre le bateau ennemi, il faut d'abord pouvoir l'identifier parmi tous les objets présents dans la simulation (bateau vert, rocher, phare et bouée). Afin de reconnaître le type d'objet que le bateau a dans son champ de vision (face à la caméra), la méthode de traitement d'image par YoloV5 a été utilisée.

La reconnaissance des différents objets s'est déroulée en 4 étapes :

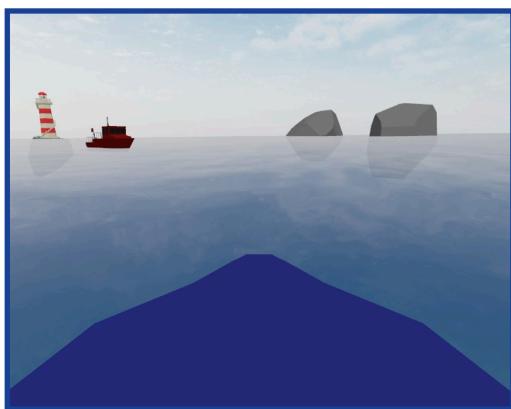
La première étape consiste à collecter une centaine d'images provenant de la caméra du bateau. Ces photos devaient faire apparaître les différents objets qui apparaissent dans la simulation (bateau ennemi et vert, rocher, phare et bouée). Pour cela, un code simple a été réalisé (`image_saver.py`) afin d'enregistrer une photo toutes les 2 secondes provenant du topic :

`/wamv/sensors/cameras/main_camera_sensor/image_raw`

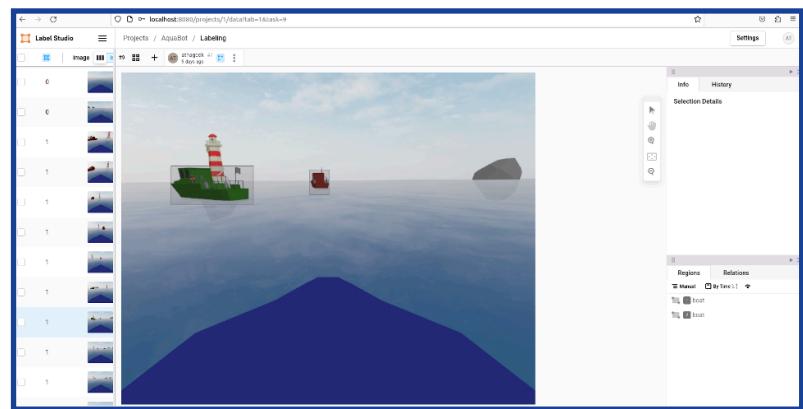
La seconde étape a pour rôle de labelliser nos images avec un outil prévu à cet effet (`label-studio`). Ce processus consiste à entourer un élément sur l'image et de lui donner un label. Ici, il se comptabilisera 5 types de labels car il y aura 5 modèles de reconnaissance d'objets :

- le label « Buoy » pour la renaissance de la bouée ;
- le label « Red boat » pour la reconnaissance du bateau ennemi ;
- le label « Green boat » pour la reconnaissance du bateau allié ;
- le label « Rock » pour la reconnaissance du roché ;
- le label « Light house » pour la reconnaissance du phare.

Ainsi, lorsque toutes les images seront labellisées, elles seront exportées au format « YOLO ».



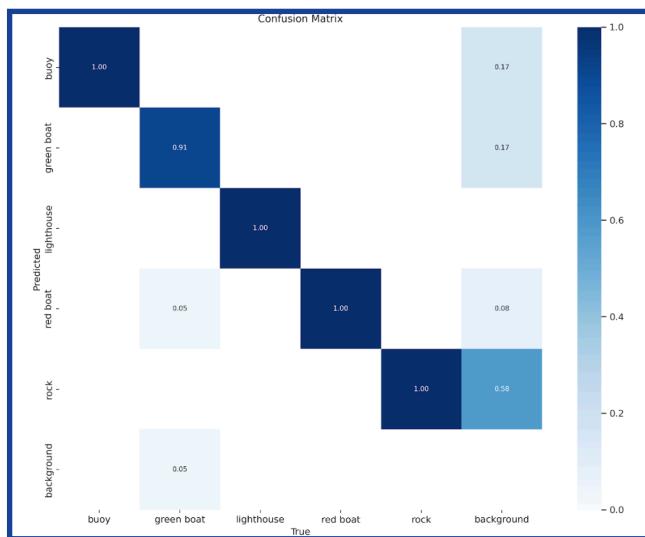
**FIGURE 8 : IMAGE COLLECTÉE**



**FIGURE 9 : LABELISATION**

La troisième étape est dédiée à l'entraînement du modèle de reconnaissance des objets. Ces modèles ont été entraînés grâce aux images labellisées via YOLOV5 : une famille d'architecture pour la reconnaissance d'objets. Par ailleurs, la puissance de calcul de Google Colab a été utilisée pour accroître la vitesse d'entraînement. Ce service gratuit permet d'exécuter du code Python en ligne en utilisant des GPU et CPU partagés.

Pour commencer à reconnaître des objets, un modèle simple pré-entraîné a été utilisé : « YOLO5 nano ». Un modèle léger permettant d'apprendre rapidement. Ce premier entraînement s'est fait sur 500 epochs (l'epoch correspond à un passage complet de l'ensemble des données d'apprentissage par l'algorithme d'apprentissage). De plus, l'algorithme s'arrêtera seul s'il s'aperçoit que son pourcentage d'apprentissage n'évolue plus. Le premier modèle se fera sur les « Red Boat ». Pour terminer, le modèle final comprenant les objets : « Buoy », « Green boat », « Rock » et « Lighthouse » s'entraînera à partir du premier.



**FIGURE 10 : MATRICE DE CONFIANCE DE L'ENTRAÎNEMENT DU MODÈLE FINAL**

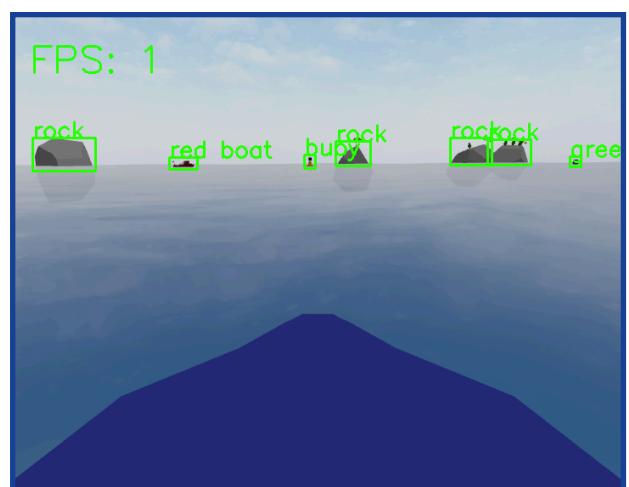
Enfin, la dernière étape concerne l'utilisation du modèle. Pour cela, un noeud est développé pour récupérer les images du topic mentionné précédemment. Ce code utilise la librairie

- OpenCV : spécialisé dans le traitement d'images en temps réel ;
- Pytorch : une librairie de machine learning compatible avec YOLOV5.

Ainsi, à partir de cette reconnaissance, l'angle de l'objet détecté par rapport à notre bateau peut être identifié. Mais surtout, l'ensemble des objets identifiable par la caméra peuvent être classés.



**FIGURE 11 : EXTRAIT D'UN RAPPORT D'ENTRAÎNEMENT DU MODÈLE PRINCIPAL**



**FIGURE 12: IDENTIFICATION EN TEMPS RÉEL DES DIFFÉRENTS OBJETS**

## 3. Difficultés rencontrées

Cette partie a pour but d'énumérer les difficultés rencontrées résolues et non résolues lors de ce projet et se fera sous forme de points numérotés.

### 3.1. Difficultés résolues

1. Estimer l'angle et la distance de la bouée par rapport au bateau.
2. Créer un modèle de machine learning pour détecter les objets.
3. Développement d'un algorithme de suivi du bateau ennemi (semi-réussi).

### 3.2. Difficultés non-résolues

1. Avoir une spirale parfaitement ajusté sur les bord de la carte avec un trajet cohérent point à point (éviter les repliements inter-sillons tronqués).
2. Implémentation du trajet de patrouille en forme de pseudo-spirale.
3. Développement d'un algorithme d'évitement des objets.

## CONCLUSION

Cette expérience professionnelle et technique a été très profitable pour nous bien que nous n'ayons pas pu mener le projet jusqu'au bout. Nous avons acquis des compétences dans plusieurs domaines tels que la programmation Python, ROS2 et le traitement d'images avec YoloV5.