

ECOLE POLYTECHNIQUE UNIVERSITAIRE DE MONTPELLIER

**Projet industriel de fin d'études
Département Electronique, Robotique,
Informatique Industrielle**

ANNEE 2020-2021

Oscar PRÊCHEUR

Répétiteur de vol

PROJET



ECOLE POLYTECHNIQUE UNIVERSITAIRE DE MONTPELLIER
UNIVERSITE MONTPELLIER II SCIENCES ET TECHNIQUES DU LANGUEDOC
Place Eugène Bataillon 34095 MONTPELLIER CEDEX 5
Tél. : 04 67 14 31 60 – Fax : 04 67 14 45 14
E-mail : scola@polytech.univ-montp2.fr



SOMMAIRE

SOMMAIRE	2
AVANT-PROPOS	3
INTRODUCTION.....	4
LE TRAVAIL.....	6
1. Analyse du travail déjà effectué	6
1.1. Le simulateur.....	7
1.2. Le programme d’interface graphique	7
1.3. Le driver de la centrale inertielle	8
2. Démarche	9
2.1. Description du besoin.....	9
2.2. Planification	10
2.3. Outils de mise en œuvre.....	10
3. Travail réalisé.....	12
3.1. Système général	12
3.2. Le Dashboard	13
3.3. Le Débriefeur	19
3.4. Le Simulateur manuel.....	23
4. Livrables et analyse du travail effectué	24
CONCLUSION	26
1. Debrief.....	26
2. Projection future	27
Bibliographie.....	28
ANNEXES.....	29
I. Modes d’emploi du système	29
II. Tableaux de hierarchie des programmes.....	33
III. Code source	35

AVANT-PROPOS

Ce projet a été très enrichissant pour moi, je tiens à remercier toutes les personnes m'ayant aidé dans sa réalisation.

Je remercie tout d'abord Laurent LATORRE, mon tuteur durant ce projet pour son aide et ses conseils dans l'exécution de ce projet.

Mes remerciements vont aussi à Loïc DARIDON, porteur du projet et mon principal interlocuteur durant ce Projet de Fin d'Etude (PFE). Il a été mon intermédiaire avec le Centre d'Entraînement Méditerranéen (CEM) durant toute la durée du projet et a su m'orienter dans sa réalisation

Je remercie de plus toutes les personnes du CEM qui m'ont accordé de leur temps et m'ont guidé dans ma recherche d'un système répondant à leur besoin.

INTRODUCTION

Dans le cadre de la formation d'ingénieur proposée par Polytech Montpellier, les étudiants sont amenés à effectuer un Projet de Fin d'Etudes (PFE). Ce projet vise à mettre en œuvre les connaissances acquises tout au long du cursus dans un contexte se rapprochant au plus du cadre professionnel. Les sujets proposés sont issus de vraies problématiques émises par des industriels, laboratoires ou par l'école elle-même. Les étudiants doivent alors réfléchir à une solution, déterminer sa faisabilité et la mettre en œuvre.

Le sujet que j'ai choisi pour effectuer mon PFE a été proposé par le Laboratoire de Génie Mécanique et de Génie Civil (LGMC) pour le compte du Centre d'Entraînement Méditerranée (CEM). Le CEM est une structure de préparation et d'entraînement pour les professionnels de la course de voile. Le projet auquel j'ai participé s'adresse aux équipages des catamarans « Nacra 17 ».

Le Nacra 17 est une classe de catamaran de sport se pilotant en binôme. Cette discipline a été lancée en 2011, elle est présente aux Jeux Olympiques depuis 2016. Les bateaux sont montés sur des foils leur permettant de « léviter » sur l'eau à certaines vitesses. Cette technologie leur permet de réduire le frottement de la coque avec l'eau et donc d'augmenter leur allure. Cela nécessite une grande précision de pilotage.

Lors des compétitions, les équipages n'ont pas le droit de recourir à des instruments électroniques pouvant leur donner des indications sur leur navigation. Ils doivent alors utiliser leur expérience acquise en entraînement afin d'optimiser le rendement du bateau.

Dans ce cadre, le CEM a émis le besoin d'un outil embarqué permettant d'afficher et de visualiser, enregistrer et rejouer des données de navigation: un répéteur de vol.

Cet outil proposerait :

- Une fonction d'affichage temps réel des données de navigation utilisable par les équipages durant leur entraînement en mer.

- Une fonction de rejou et simulation utilisable à terre lors des séances de débrief.

De plus, ce dispositif doit être conçu pour répondre à des contraintes d'utilisation particulières :

- Embarqué sur des bateaux dépourvus d'électronique
- Soumis aux conditions particulières d'un entraînement en mer.

La solution se compose d'un simple boîtier autonome incluant un écran, les capteurs et le calculateur. Il a été conçu sur la base d'un micro-ordinateur de type Raspberry Pi 3b incluant les programmes de l'interface et les drivers des capteurs.

Dans ce cadre j'ai choisi de baser mon travail sur la partie software du système. Mon but était donc de développer les interfaces homme-machine des fonctions d'affichage temps réel et de rejou.

LE TRAVAIL

1. ANALYSE DU TRAVAIL DEJA EFFECTUE

Ce projet fait suite au projet transversal auquel j'ai participé l'année dernière. L'objectif du projet était de développer une chaîne de traitement de données pour voilier, allant de la capture des valeurs à leur visualisation sur une interface graphique. Je suis donc repartie de cette base qui répondait parfaitement aux contraintes techniques et fonctionnelles pour développer ma solution en la faisant évoluer et en y ajoutant des fonctionnalités attendues.

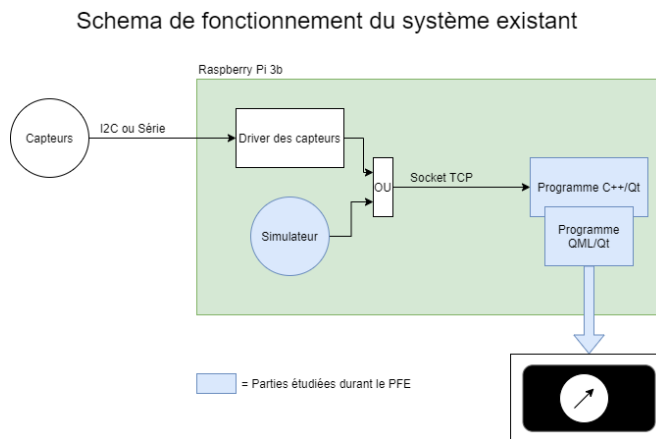


Figure 1 Schéma de fonctionnement du système existant

Le système en l'état comprenait 3 fonctions :

- Un simulateur automatique de données capteurs
- Un programme d'interface graphique pour la boussole et la gite
- Des drivers pour la centrale inertielle

1.1. LE SIMULATEUR

Le simulateur est un programme développé en C++ à l'aide de la librairie Qt. Son but est de remplacer les capteurs lors de la phase de développement du système. Il génère des données de valeurs arbitraires selon une sinusoïde et les envoie dans un socket TCP sous la forme de mots de 4 bytes lisibles par le programme d'interface graphique.

Il envoie des données dans différents ports correspondants à chaque capteur du système.

1.2. LE PROGRAMME D'INTERFACE GRAPHIQUE

Le programme est composé d'une partie graphique réalisée en QML et d'une partie calcul et fonctionnement en C++ .

La partie QML décrit la partie visible du système. Tous les composants graphiques de l'interface y sont détaillés dans leur forme comme dans leur comportement. Cette partie fait appel aux classes et aux méthodes du code C++.

Le langage de programmation QML est un langage déclaratif associé à Qt. Il permet de concevoir des interfaces utilisateurs.

Ce programme permet donc de récupérer les valeurs des données émises soit par les capteurs soit par le simulateur et de les afficher sur un écran selon le design et les animations décrites dans le programme QML.

Voici une capture d'écran de l'interface affichée au lancement du programme :

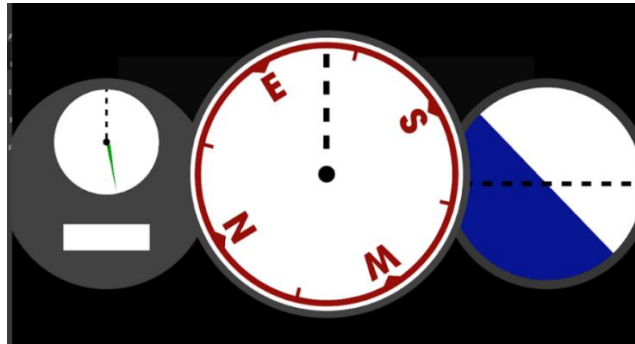


Figure 2 Interface graphique du système existant

On peut apercevoir le cadran de la boussole au milieu de l'écran. La partie rouge s'oriente en fonction des valeurs de la boussole. A sa droite le cadran de la gite dont la partie bleue est mobile. A gauche on peut apercevoir un cadran servant à indiquer la vitesse du vent ainsi que sa direction. Cette dernière fonctionnalité n'a pas été aboutie.

1.3. LE DRIVER DE LA CENTRALE INERTIELLE

Ce programme permet de faire l'interface entre les valeurs brutes émises par le capteur sur un port I2C ou série de la Raspberry et le programme d'interface graphique. Celui-ci est développé en Python et basé sur les drivers fournis par le fabricant du capteur.

2. DEMARCHE

2.1. DESCRIPTION DU BESOIN

Plusieurs réunions ont été programmées afin d'analyser le besoin du CEM et adapter la solution à développer. Les attentes du CEM étaient les suivantes :

- Une interface graphique lisible et ergonomique dans un contexte d'entraînement en bateau.
- Un affichage en temps réel de données nécessaires à la navigation des bateaux :
 - Valeur du tangage et sa tendance
 - Valeur de la gite et sa tendance
 - Valeur de la vitesse GPS et sa tendance
 - Indication de dépassement des seuils du tangage et de la gite
 - Vitesse moyenne sur un temps ou une distance donnée
- Une mémorisation des données sur un intervalle donné
- Une interface permettant la lecture des données mémorisées pour le débriefing des entraînements à terre

Afin de répondre à ces besoins, j'ai alors développé sur la base du projet existant, 2 programmes :

- Un programme tableau de bord (dashboard) réalisé sur la base du composant existant avec une refonte totale de l'interface graphique et l'ajout de nouvelles fonctionnalités.
- Un programme de lecture des données mémorisées en fonction du temps pour le debriefing.

Ces deux programmes sont accompagnés d'une nouvelle version du simulateur de capteurs permettant à l'utilisateur d'intervenir sur les valeurs à l'aide d'une nouvelle interface graphique.

2.2. PLANIFICATION

Le projet s'est déroulé en 4 phases, chacune composée de différentes tâches à réaliser :

- Préparation du projet (Analyse du problème, étude de la solution...)
- Développement de la solution
- Déploiement
- Finalisation du projet (Documentation, tests...)

Ces phases ont été effectuées de manière chronologique selon le diagramme de Gantt fourni en annexe. Une planification a été proposée en début de projet pour en valider la faisabilité.

2.3. OUTILS DE MISE EN ŒUVRE

Plusieurs outils ont été nécessaires dans la mise en œuvre de la solution :

- Langages de programmation:
 - o C++/Qt
 - o QML/Qt
- Librairie :
 - o Qt 5.15.1 version Open Source : Librairie orientée objet incluant un nombre important de fonctionnalités permettant le développement d'interfaces graphiques.
- Environnement de développement :
 - o QtCreator 4.15.0 : Cet IDE est fourni lors de l'installation de Qt. Il est adapté à l'utilisation de la librairie en incluant des outils spécialement développés pour elle.
- Versionnage :
 - o GitHub : logiciel de versionnage de projet.
 - o Visual Studio Code : j'ai utilisé cet IDE avec Git pour le versionnage de mon projet car ses outils sont bien plus ergonomiques que ceux proposés par QtCreator.

- Planification et gestion des activités :
 - o Trello : cette application est très simple d'utilisation et permet d'avoir un aperçu des tâches à effectuer afin d'organiser au mieux son travail.
- Graphismes :
 - o Adobe Illustrator : Ce logiciel d'infographie vectorielle m'a permis de designer la plupart des composants graphiques de mon projet.

3. TRAVAIL REALISE

Dans cette partie je vais traiter du travail effectué ainsi que du fonctionnement général du système au terme de ce PFE.

3.1. SYSTEME GENERAL

Le système général est composé de trois programmes correspondants aux trois interfaces que j'ai été amené à développer :

- **Le dashboard** « Nacra17Dashboard » qui permet de visualiser en temps réel des données importantes de navigation.
- **Le débriefer** « Nacra17Debriefeur », il permet de visualiser l'état du bateau à un temps donné en fonction des valeurs enregistrées par le Dashboard.
- **Le simulateur manuel** « Nacra17Simulateur », il permet à l'utilisateur de simuler des valeurs afin de les envoyer au Dashboard. Ce programme est à destination des développeurs afin d'aider au débogage du programme et n'est donc pas utile en fonctionnement normal.

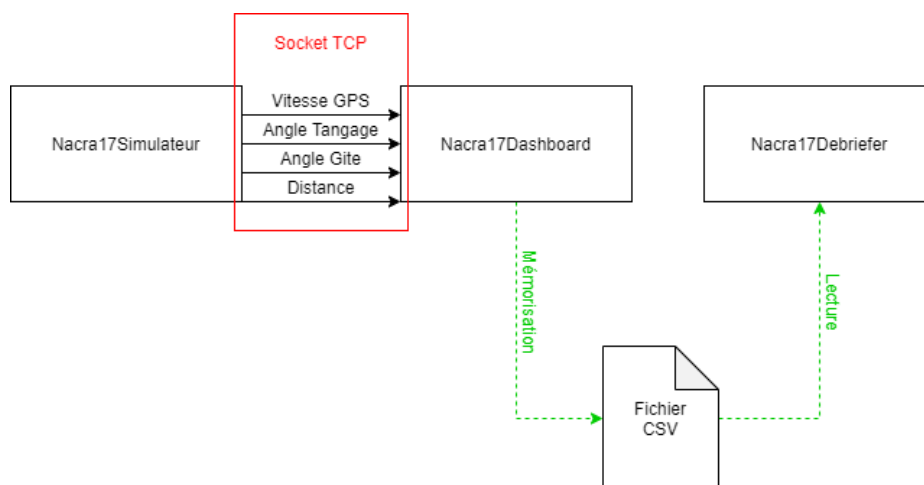


Figure 3 Schéma du fonctionnement général du système

3.2. LE DASHBOARD

Le dashboard est l'outil principal des marins, c'est le programme qui doit être intégré au système embarqué dans le bateau. Il permet d'indiquer en temps réel les différentes données du bateau pour que les marins puissent être informés de son comportement et des effets de leur manipulations. Dans ce projet, le programme dashboard porte le nom de Nacra17Dashboard.

3.2.1. Interface

Son interface se doit donc d'être très claires et ergonomique. Les marins doivent être capables de saisir les informations le plus rapidement possible pour être efficaces. Il faut donc que le design attire l'œil et soit impactant. Pour cela, le programme utilise en plus de l'affichage des valeurs, des animations et des jeux de couleurs rendant la prise d'information plus intuitive.

Dans ce programme, le tableau de bord se divise en 2 interfaces: une interface de paramétrage et une interface tableau de bord.

Les utilisateurs ont la possibilité de paramétrer :

- L'intervalle de temps entre chaque lecture des données capteur (1 à 10000ms)
- L'intervalle de temps entre chaque enregistrement des données lues (100ms à 10s)
- Les seuils minimum et maximum de la gîte (-180° à 180°)
- Les seuils minimum et maximum du tangage (-180° à 180°)

Ci-dessous, l'interface de paramétrage .

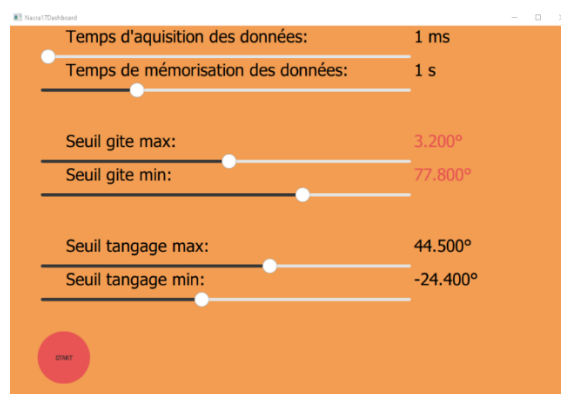


Figure 4 Interface « paramétrage » de Nacra17Dashboard

Lorsque l'utilisateur clique sur « Start » s'il a indiqué des paramètres sans erreurs (seuil min < seuil max) alors le tableau de bord s'affiche.

Ci-dessous, l'interface du tableau de bord telle que les marins pourraient le voir en entraînement.

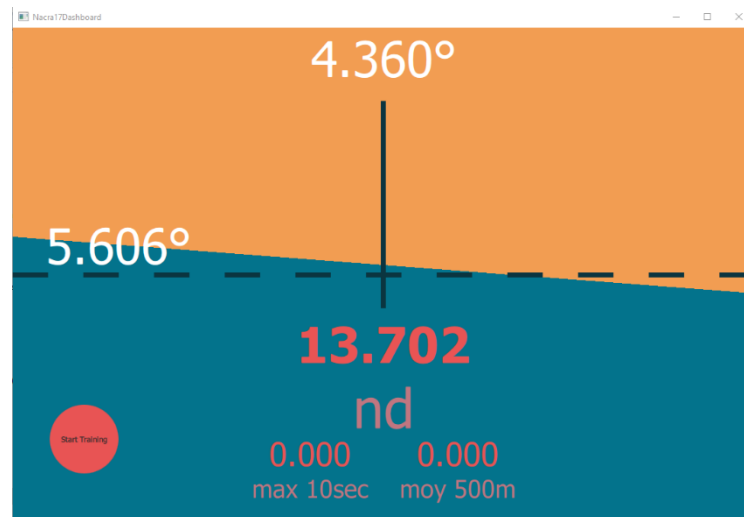


Figure 5 Interface « tableau de bord » de Nacra17Dashboard

3.2.2. Horizon Artificiel

Le tableau de bord est composé d'un horizon artificiel évoluant de manière dynamique en fonction des valeurs de la gite et du tangage.

Les translations vers le haut ou le bas symbolisent le tangage. Sa valeur est affichée en blanc à gauche (5.606° sur la Figure 5). Alors que les rotations à droite ou à gauche symbolisent la gite dont la valeur est affichée en haut (4.360° sur la Figure 5).

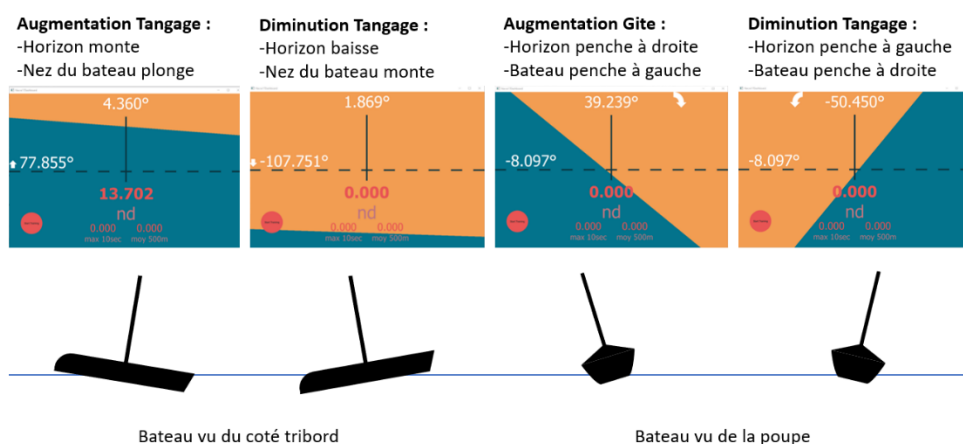


Figure 6 Tableau de description du fonctionnement de l'horizon artificiel

Lorsque les seuils paramétrés par les utilisateurs sont dépassés, l'interface alerte les marins de ce dépassement avec un changement de couleur de l'interface. Afin de déterminer cet état d'alerte, l'horizon artificiel devient rouge. Les caractères présents dans une couleur similaire virent au bleu afin de rester lisible.

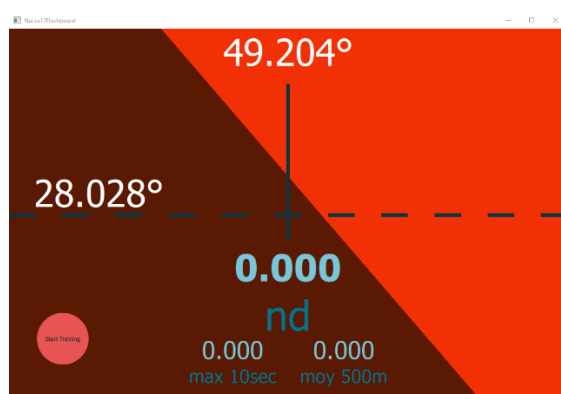


Figure 7 Interface lors d'une alerte de dépassement de seuil

L'animation de l'horizon artificiel se décrit dans le code du composant QML HorizonArtificiel.qml. C'est un rectangle de couleur bleue de base symbolisant l'eau. Son l'orientation et les coordonnées sont modifiées en fonction de la valeur de retour des méthodes getValGite() et getValTangage() de la classe valcapt appelée dans le code qml. La couleur du rectangle est changée lors du dépassement de seuil. L'air est symbolisé par l'arrière-plan de la fenêtre orange de base et qui peut lui aussi changer de couleur.

Lorsqu'on regarde l'interface, on va donc apercevoir ce que verrait l'équipage s'il regardait dans le sens de navigation du bateau.

3.2.3. Valeurs numériques et tendances

Le Dashboard affiche plusieurs valeurs numériques :

- Valeur de gite
- Valeur de tangage
- Valeur de vitesse
- Moyennage de vitesse selon le temps ou la distance (cette partie n'est pas fonctionnelle au moment où ce rapport est écrit)

Afin d'afficher les valeurs correspondantes sur le dashboard, les composants QML récupèrent les valeurs retournées par `getValGite()`, `getValTangage()` et `getValVitesse()`. Ces méthodes renvoient les valeurs des capteurs correspondant à leur nom.

Afin d'aider les marins à déterminer la tendance actuelle des valeurs, leur affichage sont accompagnés de flèches symbolisant cette dernière. Ces flèches ne s'affichent qu'à certains moments déterminés par la classe « tendance ».

Cette classe fonctionne à l'aide d'une FIFO de 100 cases. Le nombre. On enregistre en permanence les 100 dernières valeurs. Pour déterminer la tendance, on compare simplement la première et la dernière des valeurs afin de déterminer la tendance augment ou diminue.

Cette fonctionnalité permet de comparer 2 valeurs séparées dans le temps afin de déterminer si l'évolution est notable ou non. Pour rendre la rendre plus précise, il serait intéressant de comparer la valeur moyenne des 100 cases avec la dernière valeur. Cela permettrait de prendre en compte toutes les valeurs intermédiaires. Cette solution n'a pas été étudiée durant le PFE et peut être un sujet à développer à l'avenir

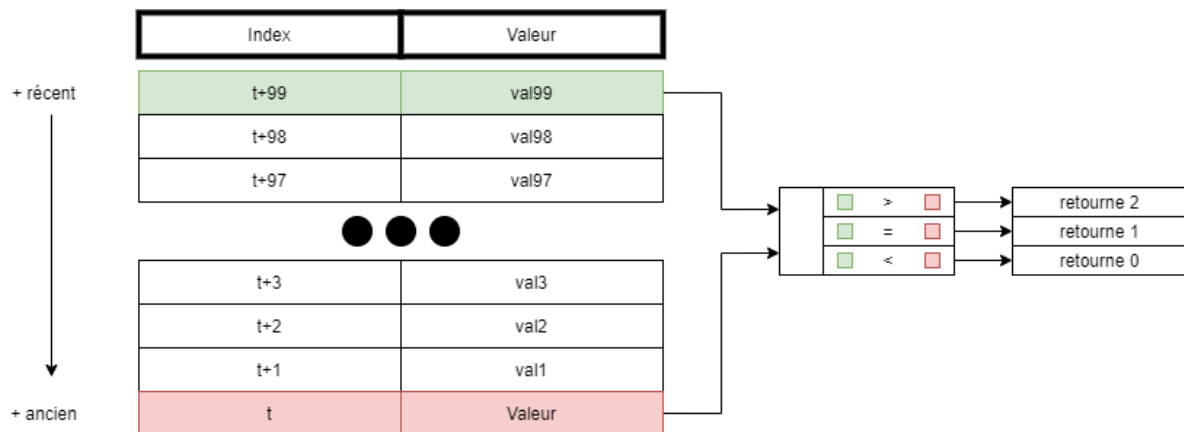


Figure 8 Fonctionnement du calcul de la tendance

3.2.4. Mémorisation

Le tableau de bord a de plus une fonction d'enregistrement des données de l'entraînement. Cette fonction se met en route en appuyant sur le bouton présent en bas à gauche de l'interface.

Cette action a pour conséquence la création d'un fichier .csv dans le répertoire « training_saves » à l'aide des classes QFile et QDir. Celui-ci est créé dans le même répertoire que le fichier Nacra17Dashbord.exe s'il n'existe pas déjà. Les classes QFile et QDir offrent des fonctionnalités d'interaction avec des fichiers et des répertoires externes au programme. Le fichier .csv porte un nom de la forme suivant :

memoTraining<jour>_<mois>_<année>_<heure>h<minute>m<seconde>s.csv

Cette forme permet aux utilisateurs de faire plusieurs enregistrements distincts au cours d'un même entraînement.

Le contenu du fichier est le suivant :

	A	B	C	D	E	F-M
1-2	indication paramètres date	paramètres date				
3	indication paramètres temporels	paramètres temporels				
4	indication paramètres seuils	paramètres seuils				
5	noms des valeurs des colonnes					
6-n	valeur temps	valeurs données capteurs				

<div style="width: 15px; height: 15px; background-color: red; border: 1px solid black;"></div>	Case vide
<div style="width: 15px; height: 15px; background-color: lightblue; border: 1px solid black;"></div>	Case avec valeur numérique
<div style="width: 15px; height: 15px; background-color: white; border: 1px solid black;"></div>	Case de texte indicatif

Figure 9 Tableau de description du fichier.csv

Date:	26	1	2021											
Heure:	21	8	34											
Param Temp	100	1												
Param Seuil	-9	6.4	-9.5	6.9										
T (s)	Distance par	Valeur Tang	Tendance T _g	Valeur Gite	Tendance G _i	Valeur Vites	Tendance V _i	Vitesse Moy	Vitesse Moy	Direction (d	Latitude (de	Longitude (degres)		
0	58.9398	-5.606	1 0.623	1 27.405	1 vit moy 500	vit moy 10s	direction	lat	long					
0.1	59.3756	-5.606	1 0.623	1 27.405	1 vit moy 500	vit moy 10s	direction	lat	long					
0.2	59.8239	-5.606	1 0.623	1 27.405	1 vit moy 500	vit moy 10s	direction	lat	long					
0.3	60.2639	-5.606	1 0.623	1 27.405	1 vit moy 500	vit moy 10s	direction	lat	long					
0.4	60.7121	-5.606	1 0.623	1 27.405	1 vit moy 500	vit moy 10s	direction	lat	long					
0.5	61.1604	-5.606	1 0.623	1 27.405	1 vit moy 500	vit moy 10s	direction	lat	long					

Figure 10 Capture d'écran d'un fichier d'enregistrement

Pour arriver à ce résultat, le programme de mémorisation écrit une chaîne de caractère QString dans le fichier à chaque coup d'horloge d'un timer de type QTimer. Le timer est paramétré via l'interface de paramétrage. La chaîne de caractères envoyée est la suivante :

```
<temps> ;<distance> ;<tangage> ;<tendance tangage> ;<gite> ;<tendance gite> ;<vitesse> ;<tendance
vitesse> ; « vit moy 500 » ; « vit moy 10s » ; « direction » ; « lat » ; « long » ; « \n »
```

Cette façon d'écrire permet de découper les données par case. Dans un fichier .csv, les colonnes sont séparées par un « ; » et les lignes par « \n ».

Les 5 dernières cases de chaque ligne sont des fonctionnalités absentes pour le moment du programme. En attendant, on envoie simplement une chaîne de caractères décrivant la donnée qui sera présente plus tard.

Ce fichier a été conçu pour être lisible sous excel, mais aussi avec l'utilisation du 3^{ème} programme : le « Nacar17Debrief ».

3.3. LE DEBRIEFER

Ce programme est à destination des marins après leurs sorties en mer, lorsque vient le moment du débriefing. Les données enregistrées lors de l'entraînement étant stockées dans un fichier.csv, celles-ci ne correspondent alors qu'à une suite de chiffres. Lors d'un debrief, il est important d'avoir une vision d'ensemble des effets des actions réalisées durant l'entraînement.

3.3.1. Interface

Le Nacra17Debriefeur est une sorte de tableau de bord affichant les données du bateau à un temps donné. L'utilisateur a la capacité de se déplacer dans le temps de l'entraînement à l'aide d'une timeline pour pouvoir analyser les données de navigations.

Voici l'interface du débriefer :

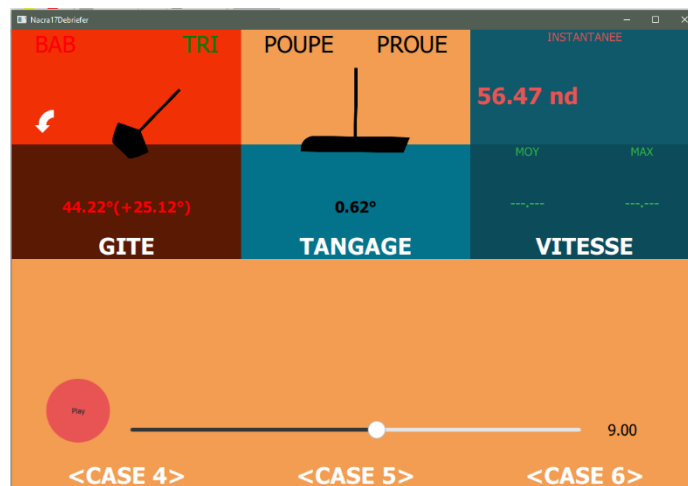


Figure 11 Tableau de bord du Débriefeur

Le débriefer se divise en plusieurs zones qui correspondent aux diverses données analysées. Le point de vue n'est maintenant plus à la première personne mais à la 3^{ème}, cela permet de visualiser le comportement du bateau et d'analyser de manière indépendante la gite et le tangage.

On retrouve le même code couleur que pour le Nacra17Dashboard. Lors du dépassement du seuil, la case en question vire au rouge (exemple de la Gite sur la figure 11). Là où dans le Dashboard ça n'était qu'une alerte, ici on peut savoir si ce dépassement vient de la gite ou du tangage. De plus l'excédent est indiqué entre parenthèse lors du dépassement.

3.3.2. Timeline

La timeline est un composant QML du Nacra17Debrief. Elle permet aux utilisateurs de manipuler le temps de l'entraînement de manière intuitive à l'image de la timeline d'un lecteur multimédia.

Sa valeur va permettre au programme de lecture de déterminer quel moment est sélectionné et ainsi afficher les données enregistrées à ce moment-là.

C'est un composant qml de type slider correspondant à une barre de contrôle fournie par Qt.

L'utilisateur n'a qu'à déplacer un curseur pour faire changer sa valeur. Ici la valeur retournée correspond à l'indice de la ligne à lire dans le fichier .csv (cf : C. Récupération des données)

La timeline peut fonctionner de 2 manière différente :

- Fonctionnement automatique : la timeline évolue en fonction du temps
- Fonctionnement manuel : l'utilisateur change lui-même la valeur à l'aide du curseur.

Pour la partie automatique, un bouton play/pause est implémenté afin d'incrémenter la position du curseur automatiquement à l'aide d'un timer qml fourni par Qt. La lecture s'adapte à l'intervalle de temps choisi pour l'enregistrement. On va augmenter d'un cran la timeline tous les deltas t. On affiche ainsi, de manière fidèle dans le temps, les événements de l'entraînement. Il est possible d'arrêter la lecture en appuyant à nouveau sur le bouton play/pause.

En ce qui concerne la taille de la timeline, celle-ci va s'adapter à la durée de la séquence (correspondant au nombre de lignes du fichier). Pour un fichier d'enregistrement de n lignes, la timeline sera créée avec n-5 crans. Les 5 crans manquants correspondent aux 5 premières lignes de paramétrages et d'indications.

3.3.3. Récupération des données

Afin de récupérer les données à afficher, le debriefer va lire le fichier .csv indiqué au préalable par l'utilisateur. Pour cela on utilise les classe QFile et QDir ainsi que la classe QFileDialog. Cette dernière permet d'afficher une boîte de dialogue pour que l'utilisateur choisisse le fichier à lire.

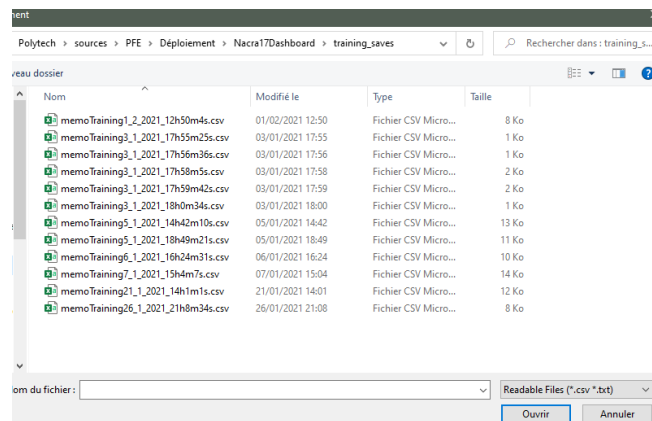


Figure 12 Boîte de dialogue de choix d'entraînement

Lorsque le fichier est choisi, son intégralité est copiée dans une QString. Ce type de variable est en réalité une simple chaîne de caractère héritant de Qt.

La chaîne de caractère est logiquement similaire à celle envoyée lors de l'écriture des données par le Dashboard. Elle se compose comme ceci :

```
caseA1;caseB1;...;caseM1\ncaseA2;caseB2;...;caseM2\n...\ncaseAn;caseBn;...;caseMn\n
```

Dans un premier temps, le programme va décomposer les 5 premières lignes afin d'en récupérer les valeurs de paramétrage utiles pour la lecture des données enregistrées. Une fois cette étape terminée, la lecture du fichier s'exécute de la façon suivante :

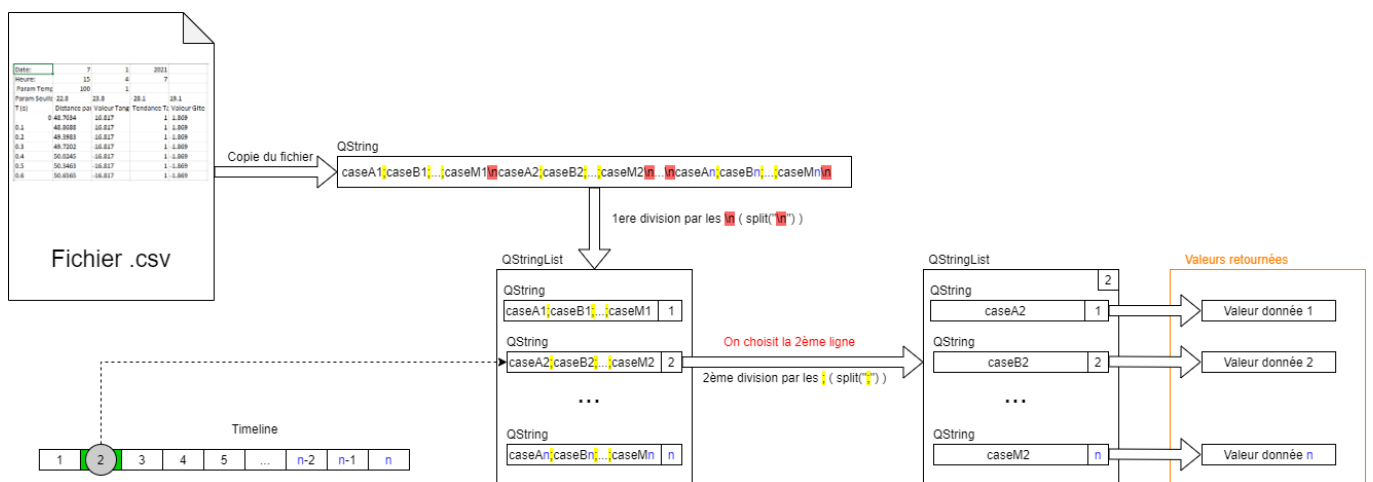


Figure 13 Schéma de fonctionnement de la lecture du fichier.csv

- 1) Tout d'abord, la chaîne de caractères comprenant l'intégralité du fichier va être divisée et devenir une QStringList. C'est une sorte de tableau de QString. On utilise la méthode `split(« \n »)` de la classe QString pour diviser la chaîne de caractères à chaque apparition du caractère choisi (ici « \n » : correspondant à un saut de ligne). On se retrouve avec cette QStringList dont chaque QString découpée possède un index.
- 2) On va à ce moment récupérer la valeur de la timeline et prendre alors la QString ayant l'index qui correspond (valeur de la timeline +5). On prend en compte la présence des 5 lignes de paramétrage dans le choix de la ligne. Une valeur à 1 de la timeline correspondra à la ligne 6 du fichier.
- 3) On a maintenant la ligne correspondant au moment de l'entraînement désigné par la timeline. Il suffit maintenant de la décomposer et d'assigner les valeurs aux variables correspondantes. Pour cela on utilise une seconde fois la méthode `split(« ; »)` mais avec « ; » en paramètre qui correspond à un changement de case. On récupère chaque valeur avec l'index de sa case.

Il ne reste maintenant plus qu'à afficher les données récupérées sur l'interface grâce au programme QML.

3.4. LE SIMULATEUR MANUEL

Le simulateur est un outil de débogage pour les développeurs. Mon travail étant basé sur la partie software du projet, j'ai décidé d'utiliser un simulateur pour remplacer temporairement les capteurs.

Ce programme est une évolution du simulateur développé lors du précédent projet . L'utilisateur est maintenant capable de d'imposer les valeurs qu'il souhaite à certaines données importantes du bateau : la gîte (°), le tangage (°) et la vitesse (nœud). Il est de plus capable de calculer la distance parcourue grâce à la vitesse indiquée.

Le programme envoie alors dans un port spécifique un mot de 4 bytes correspondant à la valeur de chaque donnée. Les numéros de port doivent donc correspondre exactement aux numéros de port codés dans le programme du dashboard.

L'interface du simulateur est comme ci-dessous :

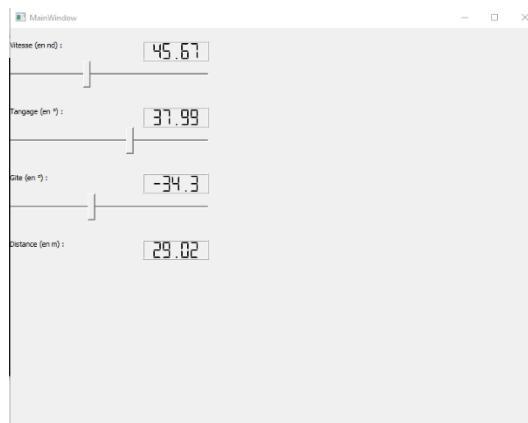


Figure 14 Copie d'écran de l'interface du simulateur manuel

Ce programme n'étant qu'un outil de développement, il fallait qu'il soit opérationnel le plus vite possible. Son interface n'est donc pas réalisée en QML mais directement dans le code .cpp grâce aux classe Qt disponibles. Il est effectivement possible de programmer une interface graphique avec Qt en C++ ou en QML.

L'avantage du QML est la personnalisation et la plus grande simplicité dans la création des composant graphiques complexes. Pour une interface simple comme le simulateur, il était inutile de créer des composants QML.

4. LIVRABLES ET ANALYSE DU TRAVAIL EFFECTUE

Au début du projet, un tableau de livrables a été édité. Ce tableau résume toutes les tâches et sous-tâches à effectuer afin de mener à bien le projet. Ce ne sont que des estimations et ce tableau est susceptible de changer au cours du temps. Voici à la suite le premier tableau de livrable et le deuxième édité à l'issue de la réunion de faisabilité du projet.

Taches	Sous-taches	Points /20
Simulateur manuel		2
	Création d'une fenêtre de contrôle avec les différent paramètres modifiables	1
	Récupération des valeurs et traitement	0.5
	Envoie des données sur des sockets	0.5
Mémoire des données		2
	Création automatique d'un fichier de mémorisation	0.5
	Récupération des données lues et écriture sur le fichier	0.5
	Création d'un protocole d'écriture (et de lecture)	0.5
	Protocole permettant la lecture des donnée directement sur le fichier texte sans interface graphique	0.5
Lecture des données mémorisées		4
	Récupération du fichiers de mémorisation	1
	Lecture des données	1
	Création d'une interface de lecture des données enregistrées	2
Interfacage des capteurs		2
	Interfacage de la centrale inertielle	1
	Interfacage du GPS	1
Nouvelle interface graphique		6
	Horizon artificiel	1
	Valeurs numérique de la gîte et du tangage	0.5
	valeurs numérique de la vitesse GPS	0.5
	affichage des tendances (gîte, vitesse, tangage)	0.5
	valeurs numérique de la vitesse moyenne (création d'un système de moyennage des valeurs)	1
	valeurs numérique de la vitesse max	1
	création d'un système d'entraînement (bouton start/stop, mémorisation à ce moment la, affichage du temps)	1
	Adaptation de la couleur en fonction des valeurs récupérées	0.5
Implémentation sur raspberry		1
	Adaptation du programme pour Raspbian	
Mode d'emploi		1
	Rédaction d'un manuel de fonctionnement du système	
Rapport de projet		2
	Rapport de projet et commentaires des programmes pour une potentielle reprise du projet	
		Total
		20

Figure 15 Tableau de livrables avant réunion de faisabilité

CONCLUSION

1. DEBRIEF

A la fin de ce PFE, le système fonctionne de façon partielle.

Le Nacra17Dashboard est capable de retranscrire de façon graphique en temps réelle les données de navigations dans une nouvelle interface plus ergonomique. Il permet maintenant d'enregistrer les données pour une utilisation futures. Une interface de paramétrage permet d'indiquer certains réglages concernant l'entraînement à venir. Le traitement du moyennage de la vitesse n'a cependant pas été aboutie.

Le Nacra17Debrifer quant à lui répond aux attentes du début de projet. Il n'est pas capable d'afficher le moyennage des vitesses vues que cette fonctionnalité n'existe pas encore. Il a été prévu pour rajouter des données dans le futur.

Le simulateur bien que n'étant qu'un outil, pourra servir aux futurs développeurs. Celui-ci est maintenant capable d'envoyé des données imposées par l'utilisateur.

Le projet dans l'ensemble est abouti. Mais certaines fonctions sont manquantes comme le moyennage de la vitesse. De plus le Nacra17Dashboard est prévu pour fonctionner sur linux or il a été développer sous Windows 10. Il m'a manqué du temps pour porter mon projet sur Linux. Qt est une librairie multi-plateforme, il est donc aisé de réaliser cette tâche si les réglages des versions linux et Windows sont similaires.

Ce projet a été très bénéfique pour moi. Il m'a permis de découvrir, en autonomie, les fonctionnalités de la librairies Qt, ainsi que le développement d'interface graphique. Cela m'a conforté dans mon choix de me concentrer vers le développement logiciel pour mon début de carrière professionnelle.

Mes connaissances en termes de gestion de projet ont été élargies et approfondies ce qui est pour moi un point très positif.

2. PROJECTION FUTURE

De nombreuses perspectives s'ouvrent pour la suite de ce projet.

Du côté de la partie software, il sera important de terminer les tâches que je n'ai pas réussi à finaliser. Le portage sous linux en est une essentielle à réaliser.

Afin de finaliser le système, il sera important de s'occuper de la partie hardware avec l'interfaçage des différents capteurs. Une recherche approfondie au niveau de l'écran du tableau de bord devra être réalisée. Le programme est développé pour être utilisé avec un écran tactile. Cette solution m'a parue intéressante en termes d'ergonomie, mais il se peut que d'autres solutions plus adaptées aux entraînements en mer existent. Auquel cas il faudra adapter le programme et le système.

La mémorisation est actuellement effectuée dans des fichiers qui sont créés au niveau du fichiers .exe, il serait important de programmer l'enregistrement pour créer les fichiers dans un espace de stockage amovible afin de les transférer du tableau de bord à un ordinateur avec le Nacra17Debrief d'installé.

La création du boîtier peut aussi être un sujet de projet pour finaliser le système. Celui-ci devra être adapté à l'environnement dans lequel il évoluera (étanchéité, résistance au sel...).

Un travail sur la connectivité pourrait aussi être mené afin, par exemple, d'envoyer en temps réel les informations à d'autres terminaux rendant les informations de navigation accessibles à d'autres personnes que l'équipage du bateau.

Ce projet n'est donc qu'un début et pourra, je pense, accueillir de nombreuses évolutions pour le rendre encore plus performant.

BIBLIOGRAPHIE

- Lien de commandes Git : <https://gist.github.com/aquelito/8596717>
- Tutoriel Raspberry Pi : https://wiki.qt.io/Raspberry_Pi_Beginners_Guide
- Ensemble des classes Qt : <https://doc.qt.io/qt-5/classes.html>
- Ensemble des plugins QML : <https://doc.qt.io/qt-5/qmltypes.html>
- Ensemble des modules Qt : <https://doc.qt.io/qt-5/qtmodules.html>
- Tutoriel de déploiement d'un programme Qt :
 - <https://www.siloged.fr/cours/QTCreator/DeployeruneapplicationQT.html>
 - <https://mechatronicsblog.com/cross-compile-and-deploy-qt-5-12-for-raspberry-pi/>
- Lien de téléchargement de l'installateur internet Qt : <https://www.qt.io/download-thank-you?hsLang=en>
- Exemple de code en QML :
https://www.youtube.com/watch?v=tfVxhL99c3s&ab_channel=KhoaTranProgrammer

ANNEXES

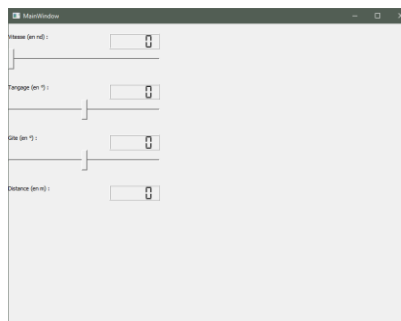
I. MODES D'EMPLOI DU SYSTEME

Modes d'emploi de la suite d'applications développées durant le PFE.

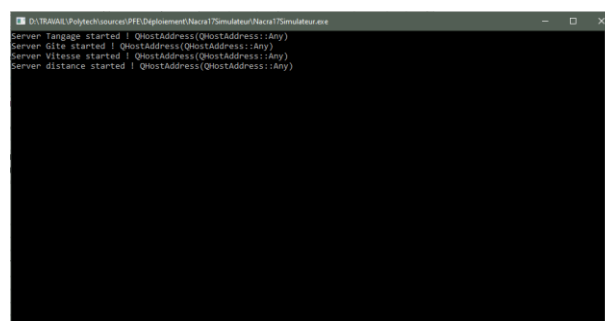
1. UTILISATION DU SIMULATEUR

ATTENTION : Pour garantir un bon fonctionnement, est important de toujours lancer Nacra17Simulateur **AVANT** le programme Nacra17Dashboard !

- 1) Se rendre dans le répertoire ***/sources/PFE/Déploiement/Nacra17Simulateur** présent dans le code source (voir annexe code source).
- 2) Lancer l'application **Nacra17Simulateur.exe**
- 3) La fenêtre suivante s'affiche :



Ainsi que la console suivante :



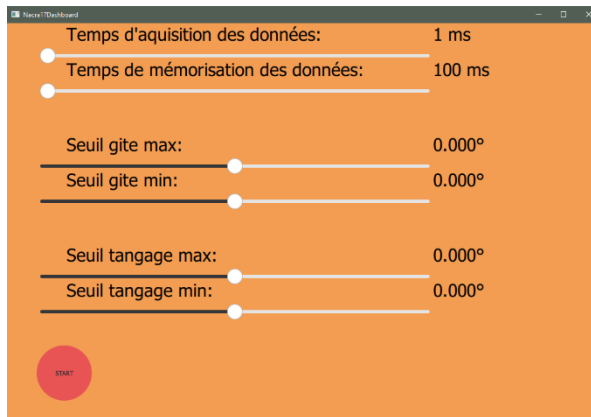
- 4) Le simulateur est en attente du Nacra17Dashboard , lancer l'application Nacra17Dashboard comme détaillé dans la partie 2. .
- 5) La console affiche :

```
Server Tangage started ! QHostAddress(QHostAddress::Any)
Server Gite started ! QHostAddress(QHostAddress::Any)
Server Vitesse started ! QHostAddress(QHostAddress::Any)
Server distance started ! QHostAddress(QHostAddress::Any)
Someone connected !!
Someone connected !!
Someone connected !!
Someone connected ijoij!!
```

Le simulateur est prêt à fonctionner. Pour changer les valeurs il suffit de faire coulisser les curseurs.

2. UTILISATION DU DASHBOARD

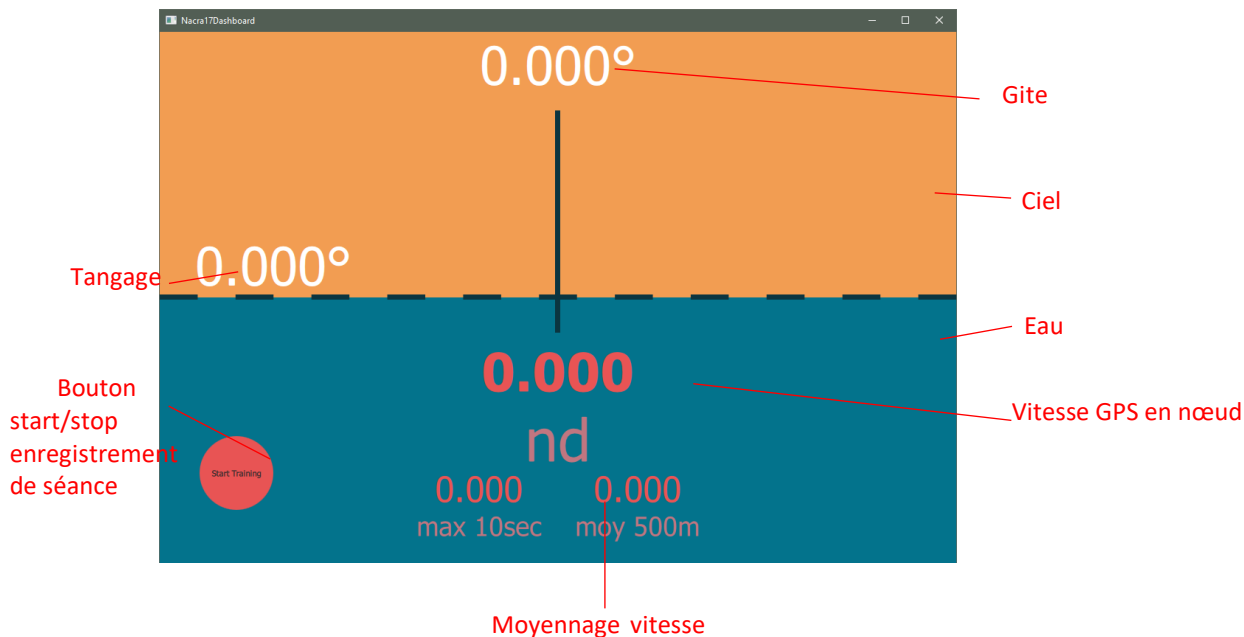
- 1) Se rendre dans le répertoire ***/sources/PFE/Déploiement/Nacra17Dashboard** présent dans le code source (voir annexe code source).
- 2) Lancer l'application **Nacra17Dashboard.exe**
- 3) La fenêtre de paramétrage suivante s'affiche :



- **Temps d'acquisition des données** : intervalle de temps entre 2 lectures des valeurs capteurs (conseillé : minimum)
- **Temps de mémorisation des données** : intervalle de temps entre chaque enregistrement des données en mode entraînement
- **Seuils** : valeurs de déclenchement d'alerte si dépassée

Paramétrer le tableau de bord et appuyer sur start

- 4) La fenêtre suivante s'affiche :

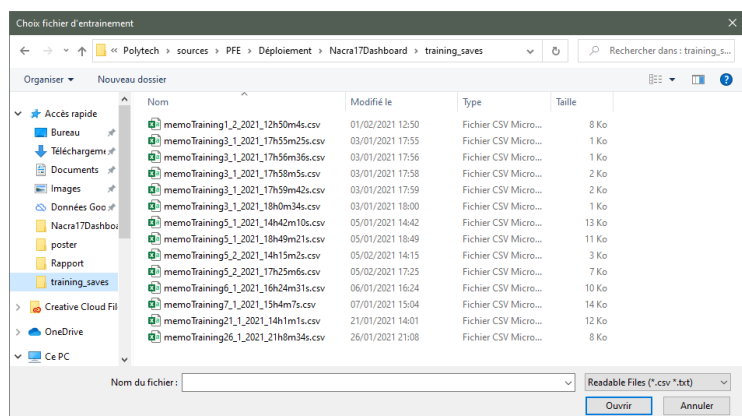


- 5) Pour enregistrer un entraînement : appuyer sur le bouton « Start Training », pour arrêter l'enregistrement : appuyer à nouveau sur le bouton. Un nouveau démarrage

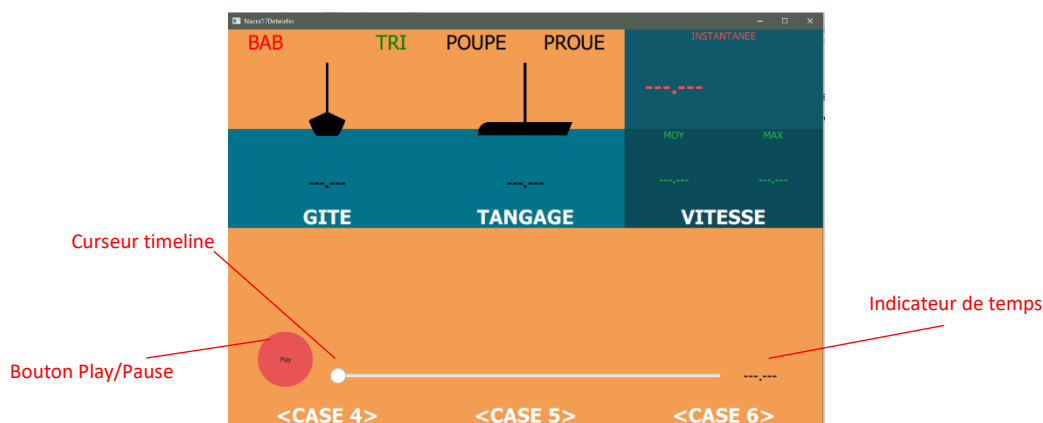
d'entrainement entraine la création d'un nouveau fichier de sauvegarde. Ces fichiers sont situés ***/sources/PFE/Déploiement/Nacra17Dashboard/Training_saves**

3. UTILISATION DU DEBRIEFER

- 1) Se rendre dans le répertoire ***/sources/PFE/Déploiement/Nacra17Debrief** présent dans le code source (voir annexe code source).
- 2) Lancer l'application **Nacra17Debrief.exe**
- 3) Une fenêtre de dialogue s'affiche. Se diriger vers le répertoire ***/sources/PFE/Déploiement/Nacra17Dashboard/Training_saves** et choisir le fichier .csv correspondant à l'entraînement à étudier (ils sont indiqués par leur date de création)



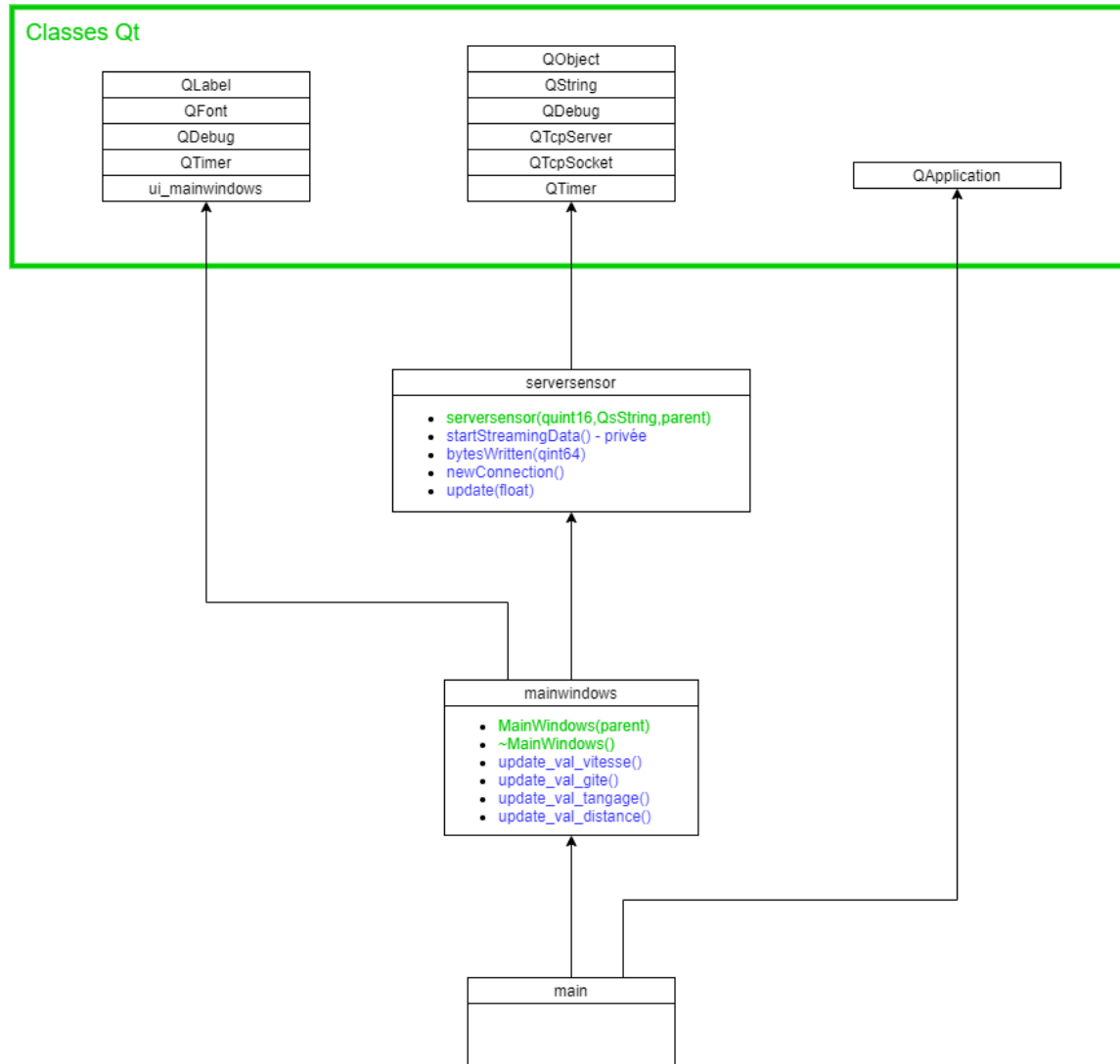
- 4) Après ouverture du fichier la fenêtre suivante s'affiche :



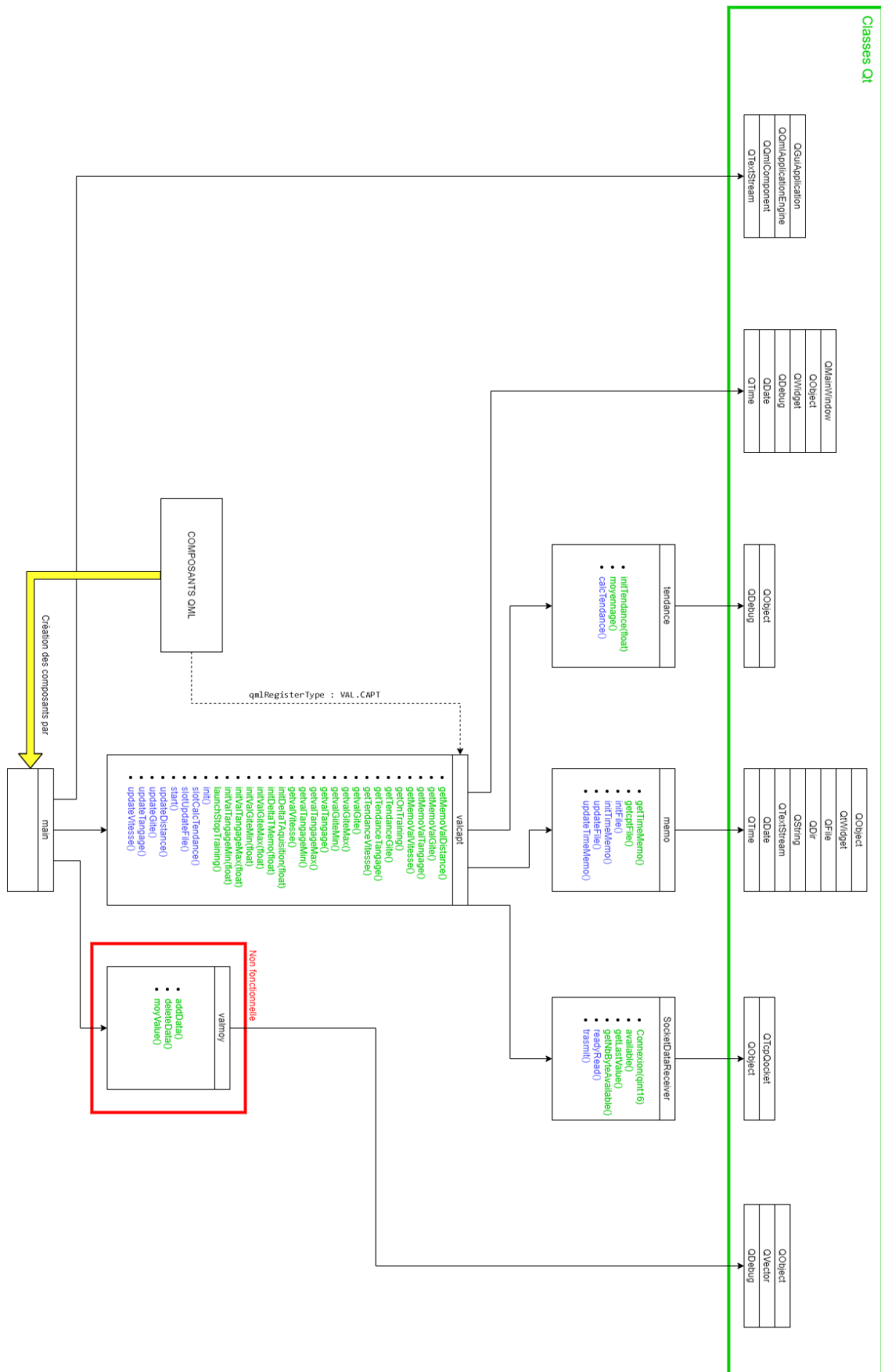
- 5) Lancer la lecture avec le bouton ou parcourir l'entraînement à l'aide du curseur de la timeline

II. TABLEAUX DE HIERARCHIE DES PROGRAMMES

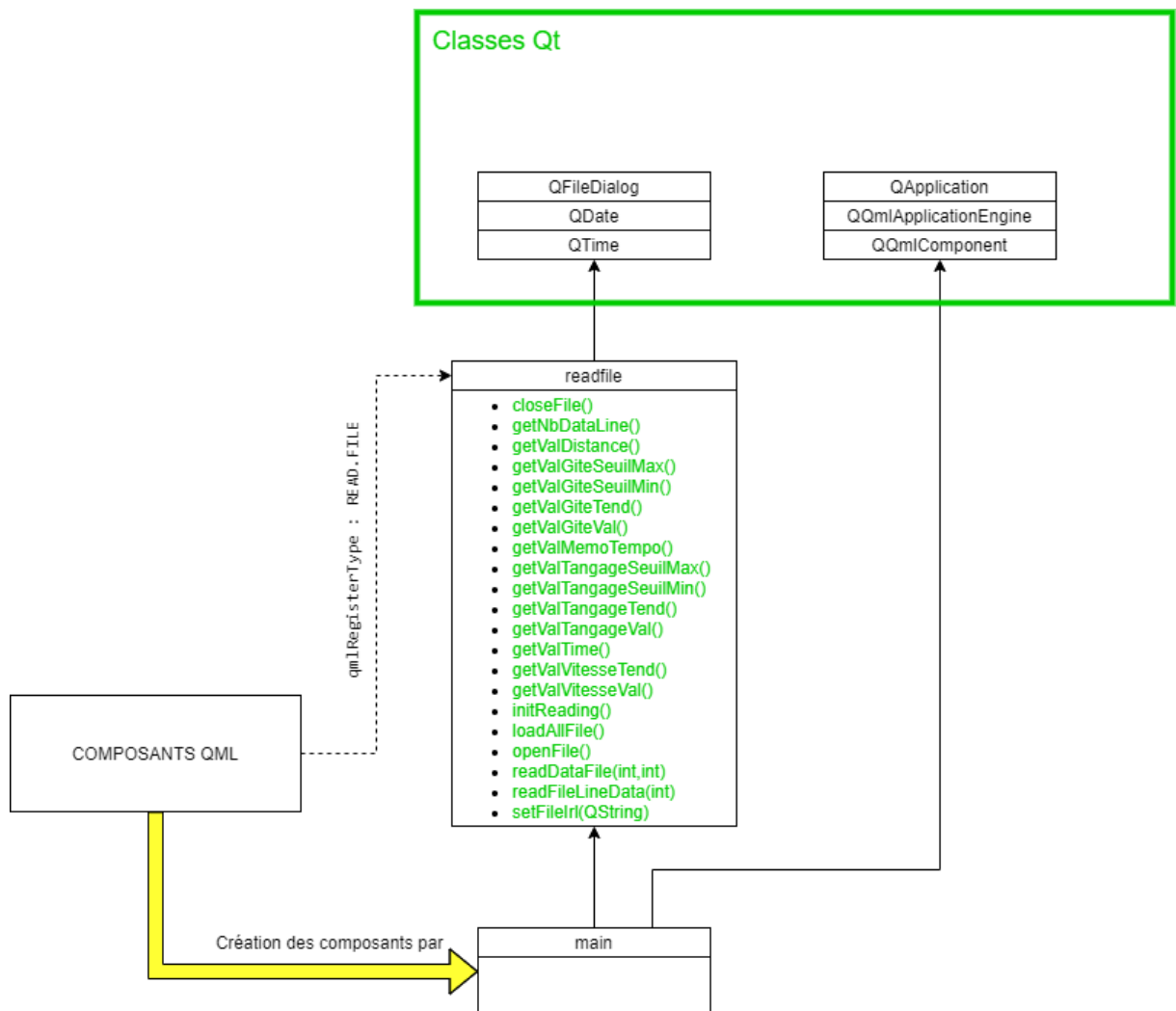
1. NACRA17SIMULATEUR



2. NACRA17DASHBOARD



3. NACRA17DEBRIEFER



III. CODE SOURCE

Le code source du projet se trouve à l'adresse git suivante :

<https://github.com/oscarprecheur/PFE.git>

ABSTRACT

Nacra 17 boats are sports catamarans. They are devoid of electronics on their board which makes it difficult to improve the precise analysis during training. The overall project proposed by the Mediterranean Training Center is to design an autonomous system including sensors, computer, and screen. It had to retrieve the navigation data, display it on the screen and save it.

This project is the continuation of a similar project begun last year. I had to improve it to make it able to answer at the demand of the Mediterranean Training Center.

During my End Studies Project, I focused on the software part of the system. I had to develop the graphical interfaces of several functions of the system.

The first function is a dashboard: the interface displays real time values on a screen, and it could save this data.

The second function is a debriefing function: this function read the saved data and displays them on another interface.

And the last one is a sensors simulator, to help during the debugging of the program.

To develop these interfaces, I choose to use the Qt library. It is a multi-language library allowing to develop graphical interfaces. It is available in Python or in C++. For this project I used the last one.

This project is not finished. There are many ways to improve it and make it efficient.