SOMMAIRE

SOMMAIRE 2
AVANT-PROPOS 3
INTRODUCTION 4
LE TRAVAIL 6
Analyse du travail déjà effectué 6
Le simulateur 6
Le programme d'interface graphique 7
Le driver de la centrale inertielle 8
Démarche 9
Description du besoin 9
Planification 9
Outils de mise en œuvre 10
Travail réalisé 11
Système général 11
Simulateur 12
Dashboard 13
Débriefeur 18
Développement des interfaces graphiques ..... **Erreur ! Signet non défini.**
Livrables analyse du travail effectué 23
CONCLUSION 25
Debrief 25
Projection future 26
ANNEXES 27
Mode d'emploi de la suite 27
Utilisation de Qt 27

AVANT-PROPOS

Ce projet a été très enrichissant pour moi. Je tiens donc, dans un premier temps, 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 PFE. Il a été mon intermédiaire avec le Centre d'Entrainement Méditerranéen (CEM) durant toute la durée du projet et m'a beaucoup conseillé dans mes choix.

Je remercie de plus toutes les personnes du CEM qui m'ont accordé de leur temps afin de me guider dans ma recherche de solution à leur problème.

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 contexte professionnel. Les sujets proposés sont 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 projet que j'ai choisi pour effectuer mon PFE est une problématique émise par le Laboratoire de Génie Mécanique et de Génie Civil (LGMC) pour le compte du Centre d'Entrainement Méditerranéen (CEM). Le CEM est une structure de préparation et d'entrainement pour les professionnels de la course de voile. Mon projet était alors en lien avec les équipages de Nacra 17 professionnels de la structure.

Le Nacra17 est une classe de catamaran de sport en binôme. Cette discipline a été lancée en 2011 et est présente au Jeux Olympique 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 alors une grande précision des marins dans leurs manipulations.

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 donc utiliser leur expérience acquise en entrainement afin d'améliorer leur rendement.

Ces pour ces raisons que le CEM a émis la problématique d'un répétiteur de vol.

L'objectif était de développer un système permettant de récupérer et d'afficher des données de navigation pour les équipages lors de leurs entrainements. Ils serviraient alors à informer en temps réel les marins sur l'état du bateau.

Ce dispositif devait alors être adapté à un contexte d'entrainement en mer sur un bateau n'incluant pas d'électronique à son bord. Il serait composé d'un écran affichant une interface graphique très simpliste et très lisible des différentes données de navigation récupérées par les capteurs. Tout cela branché à une carte Raspberry Pi 3b incluant les programmes de l'interface et les drivers des capteurs. Ces derniers étants eux-mêmes inclus dans le boitier du système afin de le rendre le plus amovible possible.

Dans le but de répondre aux attentes des marins du CEM, j'ai donc été amené à réaliser une suite d'applications Qt comprenant trois interfaces graphiques : une interface d'entrainement, une interface de debriefing post entrainement et un simulateur de données.

LE TRAVAIL

I. ANALYSE DU TRAVAIL DEJA EFFECTUE

Ce projet se rapproche énormément du projet transversal auquel j'ai participé l'année précédente. L'objectif du projet était de développer une chaine de traitement de données pour voilier, allant de la capture des valeurs à leur visualisation sur une interface graphique. Ces similitudes m'ont amené à me baser sur ce travail déjà effectué pour développer ma solution en le faisant évoluer et en y ajoutant des fonctionnalités.

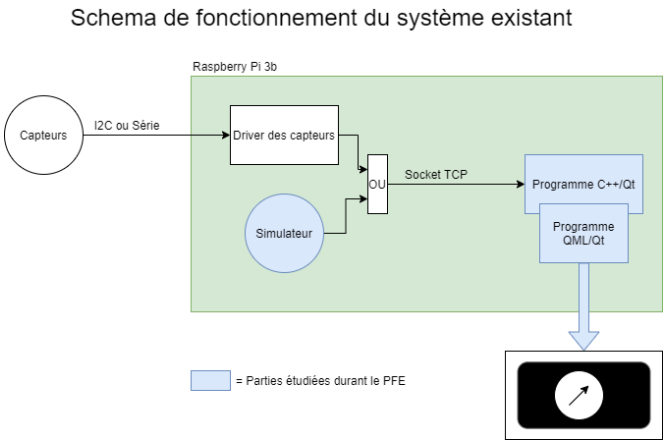


Figure 1 Schéma de fonctionnement du système existant

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. LE SIMULATEUR

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

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

2. LE PROGRAMME D'INTERFACE GRAPHIQUE

Ce programme est composé de deux parties : Une partie décrivant le fonctionnement du programme en C++/Qt et une deuxième partie en QML décrivant graphiquement l'interface. Ces deux programmes fonctionnent ensemble.

La partie C++ s'occuper de décrire le fonctionnement du système. C'est ce programme qui appelle la deuxième partie en QML en créant un composant basé sur le code QML désigné. Il inclut de plus, les différentes classes et méthodes nécessaires au fonctionnement de l'interface.

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 de la partie C++ pour fonctionner.

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 :

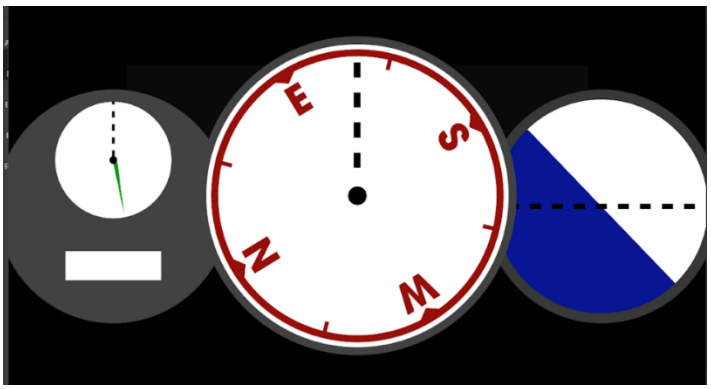


Figure 2 Interface graphique du système existant

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 indique la vitesse du vent ainsi que sa direction. Cette dernière fonctionnalité n'a pas été aboutie.

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 est fortement basé sur les drivers fournis par le fabriquant du capteur.

II. DEMARCHE

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 :

- Création d'une interface graphique lisible et ergonomique dans un contexte d'entrainement en bateau.
- 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
- Mémorisation des données à un moment donné
- Création d'une interface permettant la lecture des données mémorisées pour le débriefing des entrainements à terre

Afin de répondre à ces besoins, j'ai alors développé sur la base du projet existant, 2 programmes. Le premier étant une évolution de Dashboard existant avec une refonte totale de l'interface et un ajout de nombreuses fonctionnalités et le deuxième 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 3ème interface graphique.

2. PLANIFICATION

Le projet s'est déroulé en 4 parties, 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 parties ont été effectuées de manière chronologique selon le digramme de Gant en Annexe. Ce diagramme a été réalisé pour la présentation de la faisabilité de mon projet. Il a donc rencontré quelques évolutions.

3. OUTILS DE MISE EN ŒUVRE

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

- Langages de programmation:
  - C++/Qt
  - QML/Qt
- Librairie :
  - 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 :
  - 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 :
  - GitHub : Logiciel de versionnage de projet.
  - 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 :
  - 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 :
  - Adobe Illustrator : Ce logiciel d'infographie vectorielle m'a permis de designer la plupart des composants graphiques de mon projet.

III. TRAVAIL REALISE

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

1. SYSTEME GENERAL

Le système général est composé de trois programmes correspondants aux trois interfaces que j'ai été amené à développer. Le premier programme le « Nacra17Dashboard » permet la visualisation en temps réel des données importantes de navigation. Le deuxième programme est le « 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. Le dernier programme est le « Nacra17Debriefer », il permet de visualiser l'état du bateau à un temps donné en fonction des valeurs enregistrées par le Dashboard.

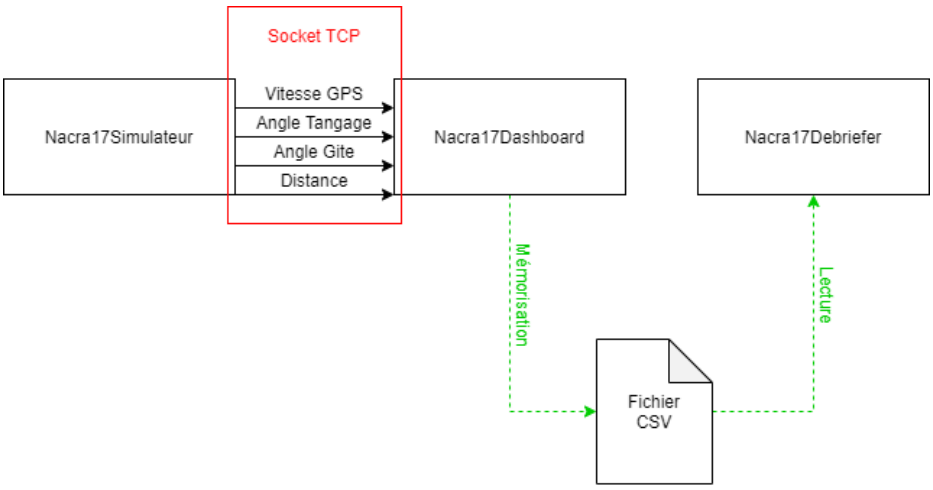


Figure 3 Schéma du fonctionnement général du système

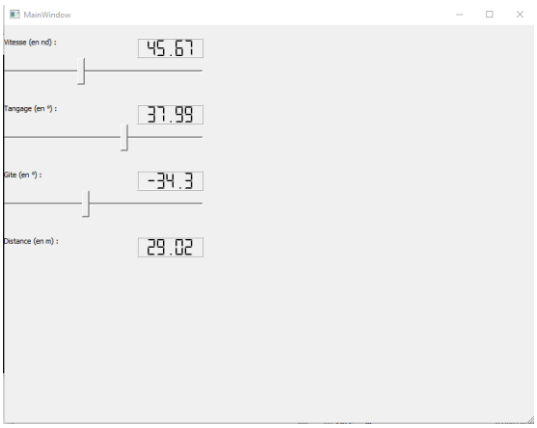
2. SIMULATEUR MANUEL

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 programme précédent. L'utilisateur est maintenant capable de données les valeurs qu'il souhaite à certaines données importantes du bateau : la gite (°), 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 :



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. Comme vous l'avez compris, il est 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.

3. 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.

Son interface se doit donc d'être très claires et ergonomique. Les marins doivent être capable 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 gite (-180° à 180°)
- Les seuils minimum et maximum du tangage (-180° à 180°)

Voici donc l'interface de paramétrage :

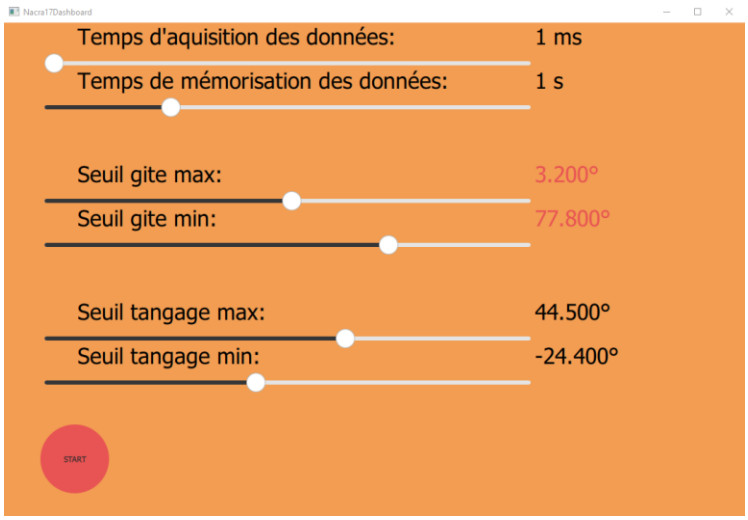


Figure 4 Interface « paramétrage » de Nacra17Dashboard

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.

Voici l'interface du tableau de bord telle que les marins pourraient le voir en entrainement.

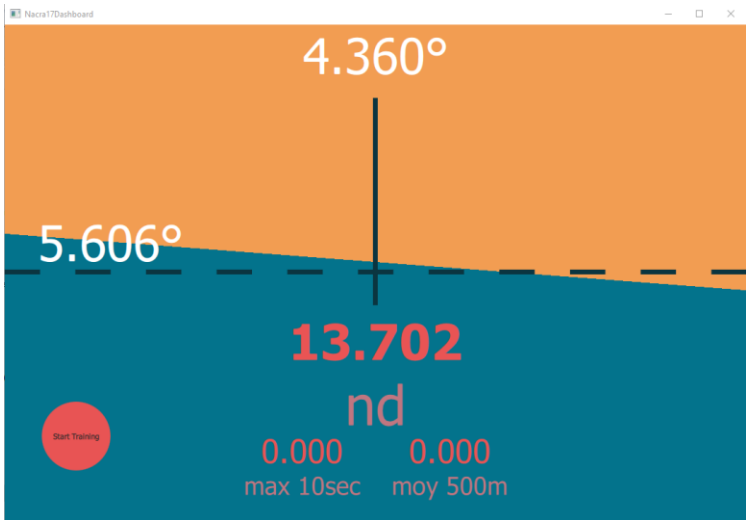


Figure 5 Interface « tableau de bord » de Nacra17Dashboard

A. 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).

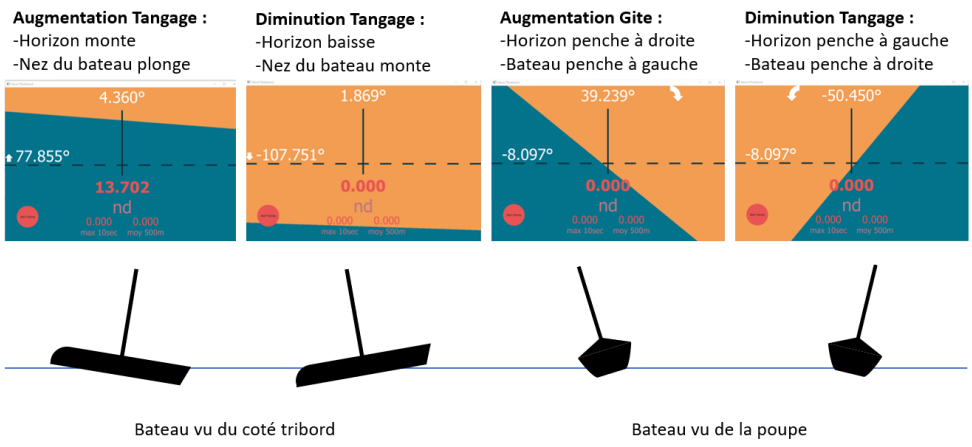


Figure 6 Tableau de description du fonctionnement de l'horizon artificiel

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.

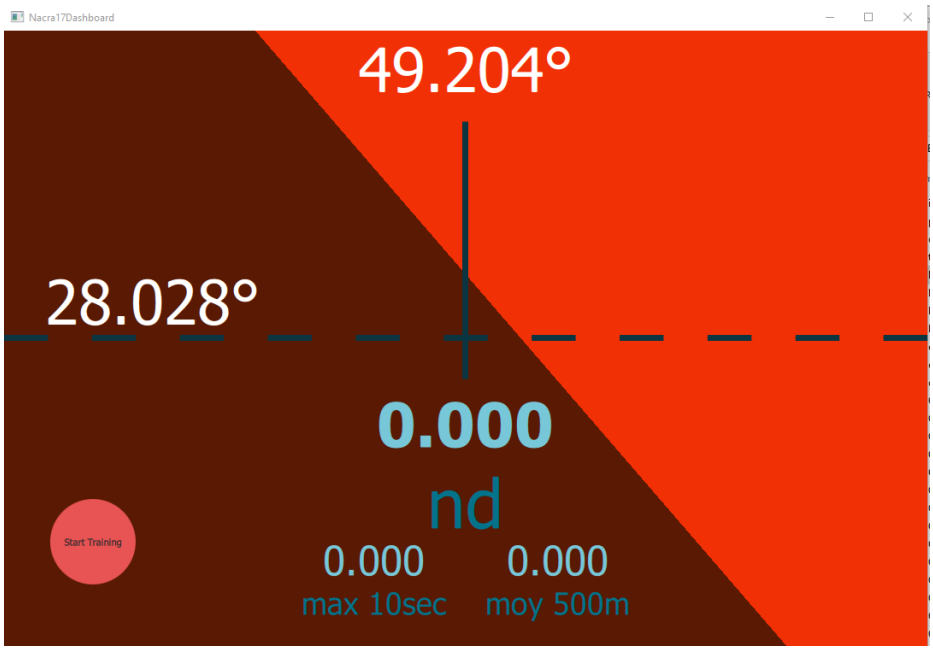


Figure 7 Interface lors d'une alerte de dépassement de seuil

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.

B. VALEURS NUMERIQUES 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

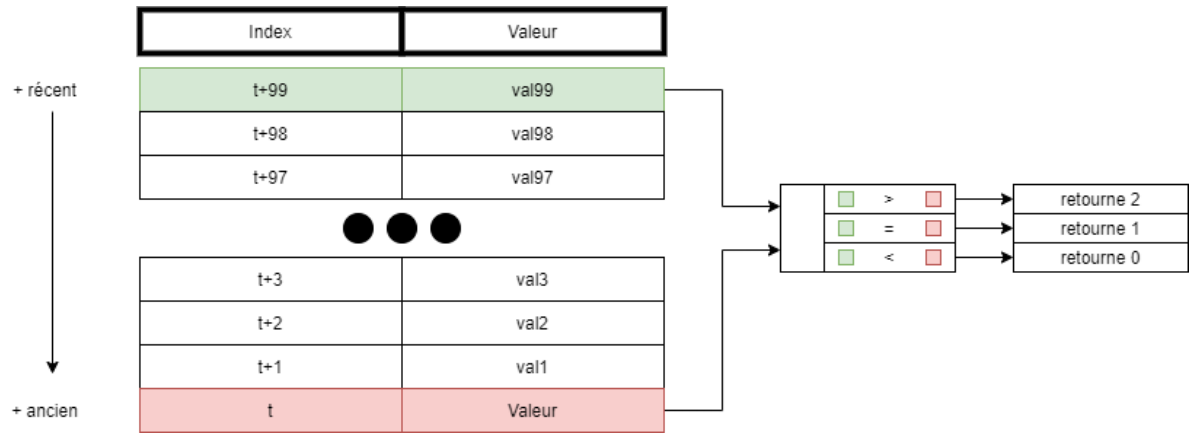


Figure 8 Fonctionnement du calcul de la tendance

A. MEMORISATION

Le tableau de bord a de plus une fonction d'enregistrement des données de l'entrainement. 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 entrainement.

Le contenu du fichier est le suivant :

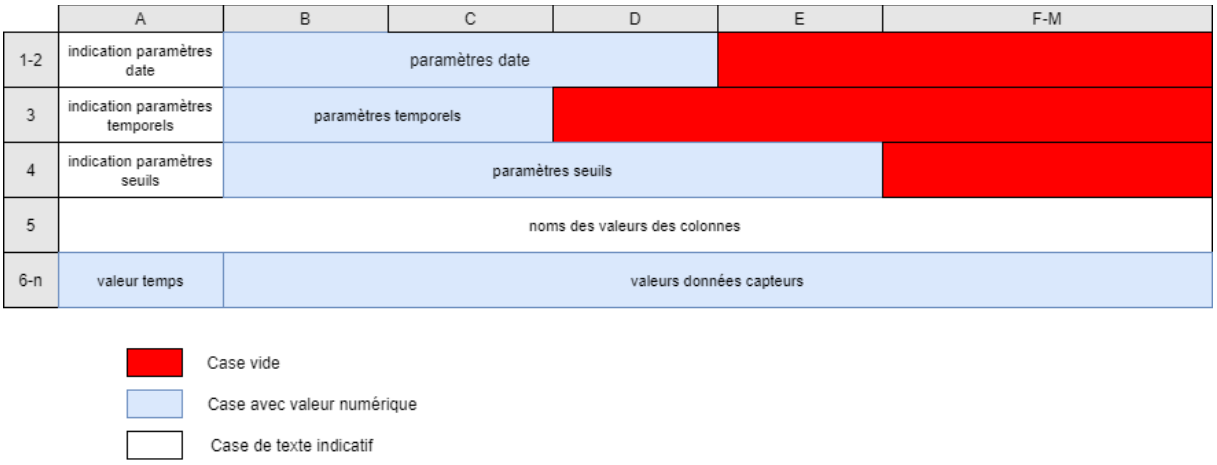


Figure 9 Tableau de description du fichier.csv

Date:	26	1	2021									
Heure:	21	8	34									
Param Temp	100	1										
Param Seuils	-9	6.4	-9.5	6.9								
T (s)	Distance par	Valeur Tang	Tendance Ta	Valeur Gite	Tendance Gi	Valeur Vites	Tendance Vi	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 chaine 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 chaine 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 chaine 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 3ème programme : le « Nacar17Debriefer ».

4. 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'entrainement 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'entrainement.

A. INTERFACE

Le Nacra17Debriefer est une sorte de tableau de bord affichant les données du bateau à un temps donné. L'utilisateur à la capacité de se déplacer dans le temps de l'entrainement à l'aide d'une timeline pour pouvoir analyser les données de navigations.

Voici l'interface du débriefer :

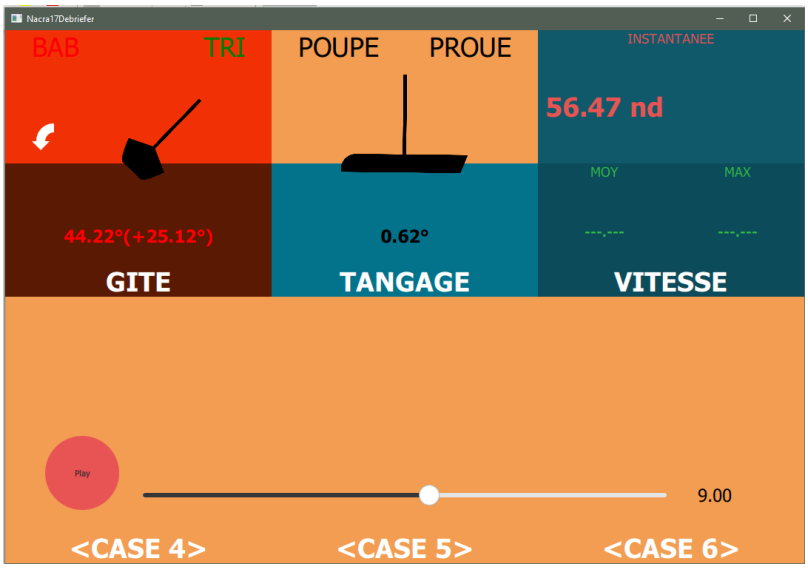


Figure 11 Tableau de bord du Débriefer

Le débriefer se divise en plusieurs cases qui correspondent aux diverses données analysées. Le point de vue n'est maintenant plus à la première personne mais à la 3ème, 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.

B. TIMELINE

La timeline est un composant QML du Nacra17Debriefer. Elle permet aux utilisateurs de manipuler le temps de l'entrainement 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 en réalité un composant qml nommé slider qui est en réalité une barre de contrôle fournies 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 programmé 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 donc augmenter d'un cran la timeline tous les deltas t. Ainsi on affiche de manière fidèle dans le temps les évènements de l'entrainement. 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 elle-même en fonction de la taille 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.

C. RECUPERATION DES DONNEES

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 QDialog. Cette dernière permet d'afficher une boîte de dialogue pour que l'utilisateur choisisse le fichier à lire.

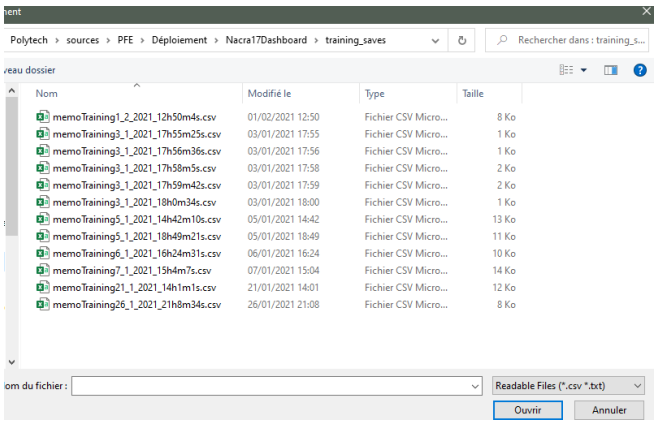


Figure 12 Boite de dialogue de choix d'entrainement

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

La chaine 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 :

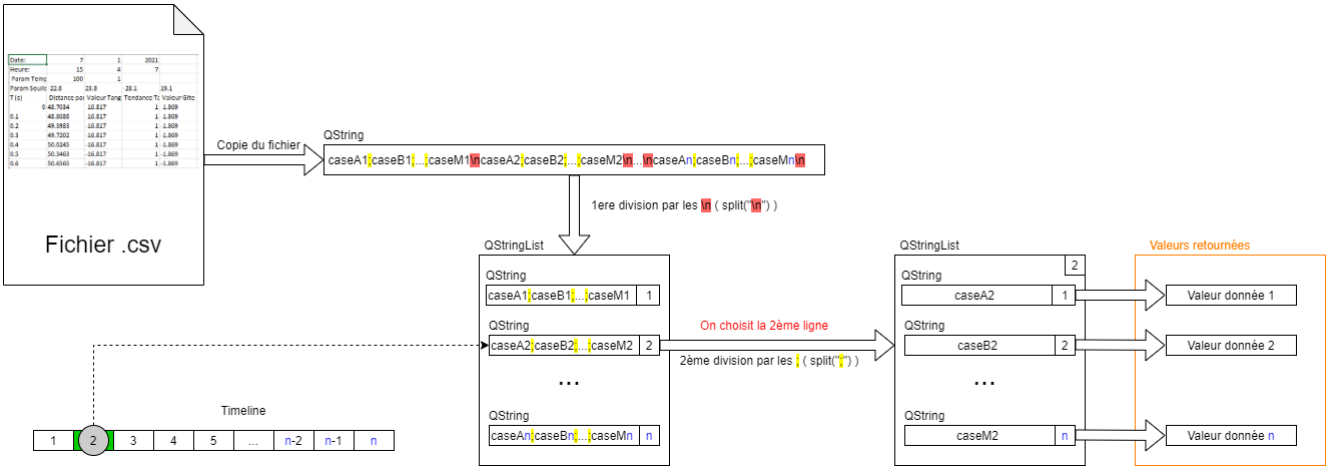


Figure 13 Schéma de fonctionnement de la lecture du fichier.csv

1) Tout d'abord, la chaine de caractère 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 chaine de caractère 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'entrainement 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.

IV. LIVRABLES 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 controle avec les différent paramètres modifiables	1
	Récupération des valeurs et traitement	0,5
	Envoie des données sur des sockets	0,5
Mémorisation 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 cetrale 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'entrainement (bouton start/stop, memorisation à ce moment la, affichage du temps)	1
	Adaptation de la couleur en fonction des valerus 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 14 Tableau de livrables avant réunion de faisabilité

Tasks	Sub-Tasks	Points /20
Manual simulator		2
	Creation of a control window with the various modifiable parameters	1
	Retrieval of values and processing	0,5
	Sends data over sockets	0,5
Memorization of data		2
	Automatic creation of a storage file	0,5
	Retrieving read data and writing to the file	0,5
	Creation of a write (and read) protocol	0,5
	Protocol allowing data to be read directly from the text file without a graphical interface	0,5
Memorized data reading		4
	Retrieving storage files	1
	Reading data	1
	Creation of an interface for reading recorded data	2
New graphic interface		8
	Settings interface	2
	Artificial horizon	1
	Numerical values of heel and pitch	0,5
	numerical values of the GPS speed	0,5
	display of trends (list, speed, pitch)	0,5
	numerical values of the average speed (creation of a system for averaging values)	1
	numerical values of the maximum speed	1
	creation of a training system (start / stop button, memorization at this moment, time display)	1
	Adaptation of the color according to the values recovered	0,5
Implementation on raspberry		1
	Adaptation of the program for Raspbian	
Manual		1
	Writing of a system operating manual	
Project documentation		2
	Project domcumentation and programm comments for potential continuation of the project	
		Total
		20

Figure 15 Tableau de livrable après réunion de faisabilité

Les livrables sont dans l'ensemble assez similaires. Seules quelques parties changent. Après la décision de ne me concentrer que sur la partie software du projet, j'ai décidé de supprimer la partie d'interfaçage des capteurs et de distribuer ces points à une partie non présente dans le premier tableau. Ces 2 points sont donc attribués à la partie de développement d'une interface de paramétrage du dashboard. La partie rapport de projet se transforme, elle, en partie documentation du projet.

Le travail effectué est fonctionnel dans l'ensemble, seules certaines tâches n'ont pas été traitées par manque de temps ou de moyen. Mais la base existante permet d'avoir une idée du fonctionnement du système final et est prête à accueillir de nouvelles évolutions.

Le développement de mon système s'est effectué en parallèle de mon apprentissage de Qt et de ses fonctionnalités. Un programme comme Nacra17Dashboard n'a donc pas la même physionomie qu'un programme plus récent comme Nacra17Debriefer. Je pense donc que mon code peut encore être optimisé afin de rendre le système plus performant.

CONCLUSION

I. 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'entrainement 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. Je n'ai pas pu porter le programme sous linux par manque de temps.

Malgré cela, 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. Mon stage de fin d'étude sera d'ailleurs accès sur l'utilisation de la librairie Qt.

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

Ce projet n'est donc pas terminé et beaucoup d'évolution du système sont possible.

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 entrainements en mer existent. Auquel cas il faudra adapter le programme et le système.

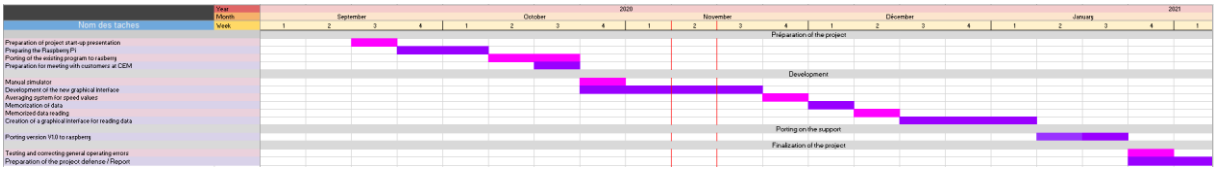
La mémorisation est actuellement effectuée dans des fichiers qui sont créés au niveau du fichiers .exe, il serait important de programmer l'enregistrement pour créer les fichiers dans un espace de stockage amovible afin de les transférer du tableau de bord à un ordinateur avec le Nacra17Debriefer d'installé.

La création du boitier 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'équipages du bateau.

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

ANNEXES



MODE D'EMPLOI DE LA SUITE

UTILISATION DE QT