16/11/2020

Oscar Prêcheur

MEA5 Polytech Montpellier

FICHE TECHNIQUE

PFE : REPETEUR DE VOL POUR VOILIER

Ce document décrit le code de la partie dashboard du système

**PARTIE 2 : Nacra17Dashboard**

Table des matières

[CLASSES 2](#_Toc64035739)

[Memo 2](#_Toc64035740)

[Socketdatareceiver 3](#_Toc64035741)

[Tendance 4](#_Toc64035742)

[Valcapt 5](#_Toc64035743)

[COMPOSANTS QML 6](#_Toc64035744)

Fichier du projet disponible sur github https://github.com/oscarprecheur/PFE.git

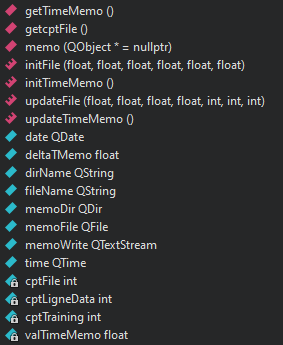
# CLASSES

Pour plus de details, se référer aux commentaires du code de chaque classes et au site <https://doc.qt.io/qt-5/classes.html> regroupant toutes les classes QT 5 et leur focntionnement

## Memo

Cette classe regroupe toutes les méthodes utilse au programme pour mémoriser les données dans un fichiers .csv lorsque l’utilisateur lance un entrainement

Elle est constituée des méthodes suivantes :



Variables privées

Variables publiques

Méthodes

Slot

Méthodes et Slot

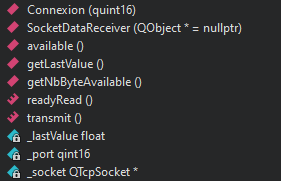
* getTimeMemo() : getter de la valeur de temps actuel de mémorisation
* getcptFile() : getter du compteur de ligne du fichier
* memo() : builder de la classe memo
* initFile(float, float, float, float, float, float) : méthode d’initialisation du fichier d’ecriture
* initTimeMemo() : initialisation du temps actuel de mémorisation
* updateFile(float,flaot,float,int,int,int() : mise à jour des donnés dans le fichier d’ecriture avec les données en paramètres
* updateTimeMemo() : mise à jour du temps de l’entrainement

Variables publiques et privées

*Détail en commentaires du fichier memo.h*

## Socketdatareceiver

Cette classe regroupe toutes les méthodes permettant la lecture des données provenant de la socket TCP



Méthodes

Variables privées

Slot

Méthodes et Slot

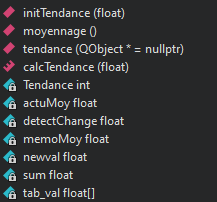
* Connexion(quint16) : connexion du programme aux ports corresponant au port numéro en paramètre
* SocketDataReceiver() : builder de la classe
* available() : données disponible dans la socket
* getLastValue(float, float, float, float, float, float) : getter de la derniere valeur lue
* getNbByteAvailable() : getter du nombre de byte dispo dans la socket
* readyRead(float,flaot,float,int,int,int() : programme pret à lire les données de la socket ?
* transmit() : méthode permettant la transmission des données à l’aide d’un connect

Variables publiques et privées

*Détail en commentaires du fichier memo.h*

## Tendance

Cette classe permet le calcul des tendance des données lues, elle permet de pouvoir afficher les flèches indicatives sur l’interface avec le programme QML



Méthodes

Variables privées

Slot

Méthodes et Slot

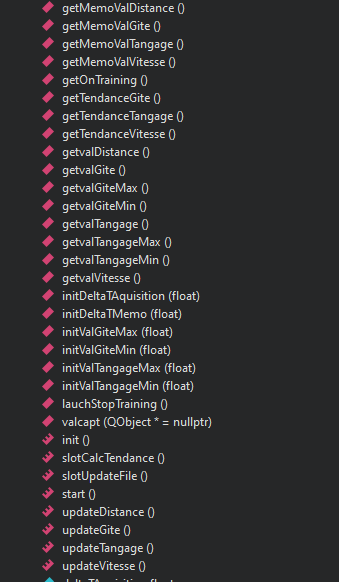
* initTendance(float) initialisatin de la valeur de la tendance
* moyennage(): calcue de la moyenne de la vitesse ( non fonctionnnel)
* tendance() : builder
* calcTendance(float): slot appellant la méthode pour calculer la valeur de la tendance

Variables publiques et privées

*Détail en commentaires du fichier tendance.h*

## Valcapt

Cette classe est la classe principale du programme, elle traite des données lues par le programme et les affiches grâce aux liens crées entre elle et les composants QML . Elle hérite de toutes les autres classes du programme. C’est le chef d’orchestre de l’acquisition et du traitement des données



Les variables sont détailées dans le code à l’aide des commentaires

Méthodes et Slot

* Les getters : les getters de la classe sont utilisables par le programme QML grâce à la création d’un registre de propriétés partagées par la classe et les composant QML. Ces composant peuvent donc appeler directement les getters et récupérer leur valeur de retour. On les utilise pour afficher les valeurs sur l’interface et faire réagir les composants selon ces valeurs.
* Les init : les méthode commençant par init initilaise les valeur de paramètre indiquées dans l’interface de paramétrage. On utilise des Q\_INVOKABLE qui permettent aux composants qml d’appeler ces méthode et de renvoyer les valeurs à la classe. On récupère donc des valeurs provenant des composants QML. C’est l’inverse des Q\_PROPERTY utilisés par les getters
* Les slot : les méthodes commençant par slot, permettent d’utiliser un connect avec une fonction prenant des paramètres différents du signal utilisé dans le connect. Le connect lance le slot qui va lui-même lancer la méthode en question
* Start() : méthode de lancement du fonctionnement de l’aquisition des données et de toute les fonctionnalitées du dashboard
* Les méthode update : permettent de récupérer de nouvelle valeurs capteur et les mettre à jour

Variables publiques et privées

*Détail en commentaires du fichier tendance.h*

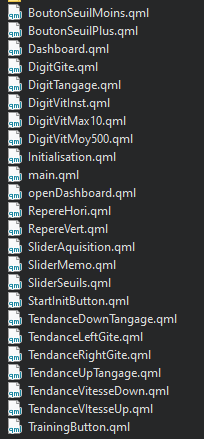
## COMPOSANTS QML

Le programme QML du dashboard est une composition de composants QML.

Chaque composant va être décrit dans un fichier propre. Ce composant sera alors appelé dans un fichier plus important. C’est une sorte d’encapsulation ou le composant appelant a une hiérarchie plus importante que le composant appelé.

Une interface graphique est donc un composant QML, constitué de plusieurs autre somposant QML moins importants eux même composés de composant QML….

Voici la liste des composants utilisés , les détails de chacun d’eux sont présent en commentaires du code



## Evolution à apporter

* Moyennage de la vitesse :
  + Calcul de la vitesse moyenne en fct d’une distance
  + Calcul de la vitesse moyenne en fct d’une durée
  + Affichage
* Enregistrement des données :
  + Test de mémoire nécessaire pour un entrainement type
  + Caractérisation de la taille de la carte SD
  + Programmation de l’enregistrement sur un espace de stockage
* Plateforme
  + Déploiement du programme sur raspberry PI

Pour apprendre les bases de Qt en Q++ : https://openclassrooms.com/fr/courses/1894236-programmez-avec-le-langage-c/1898935-initiez-vous-a-qt

Pour apprendre les bases du QML : https://qt.developpez.com/doc/4.7/qml-tutorial/