ROV'NET

version 0.2

BTS SNIR LaSalle Avignon 2020

Table des matières

1 Le projet

1.1 Table des matières

— README
— Changelog
— A propos
— Licence GPL

1.2 Informations

Auteur

Servan Tenaille <servan.tenaille@gmail.com>
Anthony Bonnet <bonnet.anthony0@gmail.com>

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/rovnet/

2 Changelog

r192 | abonnet | 2020-04-02 17 :04 :45 +0200 (jeu. 02 avril 2020) | 1 ligne

Réajustement documentation

r191 | abonnet | 2020-04-02 16 :56 :03 +0200 (jeu. 02 avril 2020) | 1 ligne

Réajustement documentation

r190 | abonnet | 2020-04-02 16 :44 :39 +0200 (jeu. 02 avril 2020) | 1 ligne

Suppression éléments non interessant pour le client dans le tag 0.1

r189 | abonnet | 2020-04-02 16 :41 :05 +0200 (jeu. 02 avril 2020) | 1 ligne

Création du tag 0.1

r188 | stenaille | 2020-04-02 16 :30 :44 +0200 (jeu. 02 avril 2020) | 2 lignes

mise a jour documentation

r187 | stenaille | 2020-04-02 16 :28 :51 +0200 (jeu. 02 avril 2020) | 2 lignes

mise a jour documentation

r186 | stenaille | 2020-04-02 16 :24 :35 +0200 (jeu. 02 avril 2020) | 2 lignes

Correction documentation

r185 | stenaille | 2020-04-02 16 :16 :57 +0200 (jeu. 02 avril 2020) | 2 lignes

correction documentation

r184 | stenaille | 2020-04-02 16 :15 :42 +0200 (jeu. 02 avril 2020) | 2 lignes

Mise a jour documentation

r183 | abonnet | 2020-04-02 16 :11 :18 +0200 (jeu. 02 avril 2020) | 1 ligne

Mis a jour documentation

r182 | stenaille | 2020-04-02 16 :10 :15 +0200 (jeu. 02 avril 2020) | 2 lignes

mise a jour des return

r181 | abonnet | 2020-04-02 16 :08 :04 +0200 (jeu. 02 avril 2020) | 1 ligne

Reéinitialisation de la base de données

r180 | stenaille | 2020-04-02 16 :03 :01 +0200 (jeu. 02 avril 2020) | 2 lignes

Suppression fn du main.cpp

r179 | abonnet | 2020-04-02 16 :01 :11 +0200 (jeu. 02 avril 2020) | 1 ligne

Suppression BDD obsolete

r178 | stenaille | 2020-04-02 15 :59 :19 +0200 (jeu. 02 avril 2020) | 2 lignes

Suppression fn de manette.h

r177 | abonnet | 2020-04-02 15 :59 :02 +0200 (jeu. 02 avril 2020) | 1 ligne

Suppréssion fichier sql obsolete

r176 | stenaille | 2020-04-02 15 :55 :25 +0200 (jeu. 02 avril 2020) | 2 lignes

Rectification documentation

r175 | abonnet | 2020-04-02 15 :52 :46 +0200 (jeu. 02 avril 2020) | 1 ligne

Mis a jour documentation

r174 | stenaille | 2020-04-02 15 :47 :03 +0200 (jeu. 02 avril 2020) | 2 lignes

retrait des fn dans rov.h

r173 | abonnet | 2020-04-02 15 :45 :09 +0200 (jeu. 02 avril 2020) | 1 ligne

Mis a jour documentation

r172 | stenaille | 2020-04-01 15 :57 :03 +0200 (mer. 01 avril 2020) | 2 lignes

Création slot enregisterMesureBDD() et signal enregistrerMesures()

r171 | abonnet | 2020-03-31 18 :59 :11 +0200 (mar. 31 mars 2020) | 1 ligne

Modification bug chargement des photos -> conteneur informationsPhotos de la méthode chargerCampagne() doit être vidé pour chaque campagne

r170 | tvaira | 2020-03-31 18 :51 :35 +0200 (mar. 31 mars 2020) | 2 lignes

Déplacement du define SANS_DETECTION dans camera.h

r169 | abonnet | 2020-03-31 18 :22 :23 +0200 (mar. 31 mars 2020) | 1 ligne

Modification structure Photo -> cette dernière ne comprend plus les données issues des capteurs

r168 | abonnet | 2020-03-31 18 :19 :11 +0200 (mar. 31 mars 2020) | 1 ligne

Ajout define SANS_DETECTION

r167 | abonnet | 2020-03-31 16 :32 :35 +0200 (mar. 31 mars 2020) | 1 ligne

Modification de l'archivage des photos -> ces dernières sont enregistré dans la BDD directement à la prise

r166 | tvaira | 2020-03-30 18 :24 :09 +0200 (lun. 30 mars 2020) | 2 lignes

Révision de code (voir mail)

r165 | stenaille | 2020-03-29 20 :15 :03 +0200 (dim. 29 mars 2020) | 2 lignes

Ajout suppression image en local et suppression dossier en local

r164 | abonnet | 2020-03-29 18 :55 :07 +0200 (dim. 29 mars 2020) | 1 ligne

Implementation méthode permettant de modifie l'etat des bouton de l'ihmReglageVideo

r163 | tvaira | 2020-03-29 18 :53 :18 +0200 (dim. 29 mars 2020) | 2 lignes

Démarrer/Arrêter Vidéo

r162 | abonnet | 2020-03-29 18 :18 :38 +0200 (dim. 29 mars 2020) | 1 ligne

Modification IHMReglageVideo -> choix de la camera disponible

r161 | abonnet | 2020-03-29 15 :16 :10 +0200 (dim. 29 mars 2020) | 1 ligne

Création d'un dossier du nom de la campagne à la création de cette dernière + enregistrement des photos prise dans ce dossier

r160 | abonnet | 2020-03-29 14 :30 :09 +0200 (dim. 29 mars 2020) | 1 ligne

mis en ordre du code

r159 | stenaille | 2020-03-28 18 :42 :48 +0100 (sam. 28 mars 2020) | 2 lignes

Ajout suppression des photos non voulu lors de l'archivage

r158 | abonnet | 2020-03-28 18 :24 :45 +0100 (sam. 28 mars 2020) | 1 ligne

Les photos s'enregistresous format PNG dans le dossier choisi

r157 | abonnet | 2020-03-28 18 :01 :12 +0100 (sam. 28 mars 2020) | 1 ligne

Modification méthode chargerCampagne() -> prend en compte le nombre de photos presente dans la base de données

r156 | abonnet | 2020-03-28 17 :01 :35 +0100 (sam. 28 mars 2020) | 1 ligne

Modification bug validerCampagne

r155 | abonnet | 2020-03-28 16 :27 :08 +0100 (sam. 28 mars 2020) | 1 ligne

Modification structure BDD ajout champs cheminSauvegarde à la table campagne

r154 | abonnet | 2020-03-28 16 :02 :22 +0100 (sam. 28 mars 2020) | 1 ligne

Implementation méthode updateCampagneBDD

r153 | stenaille | 2020-03-28 13 :58 :57 +0100 (sam. 28 mars 2020) | 2 lignes

Correction bug suppression photo

r152 | stenaille | 2020-03-27 18 :39 :27 +0100 (ven. 27 mars 2020) | 2 lignes

Modification methode supprimerCampagneBDD

r151 | stenaille | 2020-03-27 16 :52 :34 +0100 (ven. 27 mars 2020) | 2 lignes

Modification méthode archiverCampagne et supprimerCampagneBDD

r150 | stenaille | 2020-03-27 16 :39 :43 +0100 (ven. 27 mars 2020) | 2 lignes

Ajout méthode Archiver et suppressionBDD

r149 | abonnet | 2020-03-27 12 :57 :34 +0100 (ven. 27 mars 2020) | 1 ligne

Ajout liste des techniciens connus

r148 | abonnet | 2020-03-26 18 :01 :35 +0100 (jeu. 26 mars 2020) | 1 ligne

Modification méthode enregistrerCampagneBDD ->mise a jour base de données campagnes.sqlite

r147 | abonnet | 2020-03-26 17 :40 :23 +0100 (jeu. 26 mars 2020) | 1 ligne

Modification méthode enregistrerCampagneBDD ->mise a jour base de données campagnes.sqlite

r146 | abonnet | 2020-03-26 16 :33 :25 +0100 (jeu. 26 mars 2020) | 1 ligne

Modification methode chargerCampagnes() -> mise a jour base de données campagnes.sqlite

r145 | stenaille | 2020-03-26 16 :21 :16 +0100 (jeu. 26 mars 2020) | 1 ligne

Ajout diagrammeDeClassePersonnel

r144 | stenaille | 2020-03-26 15 :23 :46 +0100 (jeu. 26 mars 2020) | 1 ligne

Ajout diagramme de classe : diagrammeClasseDeplacerLeRobot, diagrammeClassePiloterLeBras, diagrammeClasseRecevoir↩ Mesures, diagrammeClasseVisualiserMesures

r143 | stenaille | 2020-03-26 14 :07 :03 +0100 (jeu. 26 mars 2020) | 2 lignes

Modification diagramme cas d'utilisation

r142 | tvaira | 2020-03-26 13 :34 :44 +0100 (jeu. 26 mars 2020) | 1 ligne

Modelisation BD

r141 | tvaira | 2020-03-25 17 :25 :20 +0100 (mer. 25 mars 2020) | 1 ligne

Agencement cas utilisation tenaille

r140 | stenaille | 2020-03-25 17 :16 :37 +0100 (mer. 25 mars 2020) | 1 ligne

Ajout diagramme recevoirMesure, visualiserMesures, casUtilisationPersonnel

r139 | abonnet | 2020-03-25 14 :43 :19 +0100 (mer. 25 mars 2020) | 1 ligne

ajout diagramme de cas d'utilisation

r138 | abonnet | 2020-03-25 13 :54 :22 +0100 (mer. 25 mars 2020) | 1 ligne

Modification class IHMCreationCampagne -> héritage QDialog

r137 | abonnet | 2020-03-25 12 :27 :45 +0100 (mer. 25 mars 2020) | 1 ligne

Rectification diagramme cas d'utilisation

r136 | tvaira | 2020-03-24 16 :14 :45 +0100 (mar. 24 mars 2020) | 1 ligne

Maj diagramme de séquence demarrerUneCampagne

r135 | tvaira | 2020-03-24 16 :13 :59 +0100 (mar. 24 mars 2020) | 2 lignes

Modification classe IHMAccueil en mode modale (type QDialog)

r134 | abonnet | 2020-03-24 15 :05 :30 +0100 (mar. 24 mars 2020) | 1 ligne

Ajout diagramme sequence et deploiement

r133 | abonnet | 2020-03-24 15 :03 :14 +0100 (mar. 24 mars 2020) | 1 ligne

modification diagramme deploiement

r132 | abonnet | 2020-03-24 13 :26 :51 +0100 (mar. 24 mars 2020) | 1 ligne

Création diagramme de deploiement

r131 | abonnet | 2020-03-24 12 :47 :45 +0100 (mar. 24 mars 2020) | 1 ligne

Création diagramme sequence piloterLaCamera et demarreruneCampagne

r130 | tvaira | 2020-03-21 16 :54 :23 +0100 (sam. 21 mars 2020) | 1 ligne

Validation BD SQLite v1.1

r129 | abonnet | 2020-03-21 14 :39 :48 +0100 (sam. 21 mars 2020) | 1 ligne

Modification BD ajout campagne-v1.1.sql

r128 | stenaille | 2020-03-20 17 :46 :19 +0100 (ven. 20 mars 2020) | 1 ligne

ajout .png diagramme piloterLeBras

r127 | stenaille | 2020-03-20 17 :45 :11 +0100 (ven. 20 mars 2020) | 1 ligne

Ajout diagramme de sequence piloter le bras

r126 | abonnet | 2020-03-20 15 :41 :11 +0100 (ven. 20 mars 2020) | 1 ligne

Modification BD

r125 | abonnet | 2020-03-20 15 :25 :48 +0100 (ven. 20 mars 2020) | 1 ligne

Modification BD + ajout lieu class Campagne

r124 | abonnet | 2020-03-20 15 :18 :55 +0100 (ven. 20 mars 2020) | 1 ligne

Modification BD + ajout lieu class Campagne

r123 | stenaille | 2020-03-20 11 :13 :08 +0100 (ven. 20 mars 2020) | 1 ligne

Ajout element diagramme de sequence

r122 | tvaira | 2020-03-20 07 :34 :40 +0100 (ven. 20 mars 2020) | 1 ligne

Validation du fichier SQL

r121 | stenaille | 2020-03-19 19 :36 :12 +0100 (jeu. 19 mars 2020) | 2 lignes

Ajout du de la fonction archiverCampagne

r120 | tvaira | 2020-03-19 18 :40 :29 +0100 (jeu. 19 mars 2020) | 1 ligne

Validation sd deplacerLeRobot

r119 | stenaille | 2020-03-19 18 :26 :04 +0100 (jeu. 19 mars 2020) | 2 lignes

Mise a jour structure Mesure

r118 | stenaille | 2020-03-19 18 :23 :25 +0100 (jeu. 19 mars 2020) | 2 lignes

Ajout structure Mesures

r117 | abonnet | 2020-03-19 18 :20 :23 +0100 (jeu. 19 mars 2020) | 1 ligne

Modification BDD

r116 | abonnet | 2020-03-19 18 :20 :05 +0100 (jeu. 19 mars 2020) | 1 ligne

Modification BDD

r115 | abonnet | 2020-03-19 18 :19 :37 +0100 (jeu. 19 mars 2020) | 1 ligne

Modification BDD

r114 | stenaille | 2020-03-19 18 :19 :09 +0100 (jeu. 19 mars 2020) | 1 ligne

Ajout campagnesArchives

r113 | abonnet | 2020-03-19 18 :09 :17 +0100 (jeu. 19 mars 2020) | 1 ligne

Ajout fichier campagnesEnCours-v1.sql

r112 | stenaille | 2020-03-19 18 :00 :25 +0100 (jeu. 19 mars 2020) | 1 ligne

Ajout de bouml-rovnet-tenaille

r111 | stenaille | 2020-03-19 17 :59 :02 +0100 (jeu. 19 mars 2020) | 1 ligne

Suppression boulm-rovnet-tenaille

r110 | stenaille | 2020-03-19 17 :51 :00 +0100 (jeu. 19 mars 2020) | 1 ligne

Ajout des parties manquantes

r109 | stenaille | 2020-03-19 17 :45 :19 +0100 (jeu. 19 mars 2020) | 1 ligne

r108 | abonnet | 2020-03-19 17 :35 :08 +0100 (jeu. 19 mars 2020) | 1 ligne

Modification BDD

r107 | abonnet | 2020-03-19 17 :33 :49 +0100 (jeu. 19 mars 2020) | 1 ligne

Modification BDD

r106 | abonnet | 2020-03-19 17 :20 :30 +0100 (jeu. 19 mars 2020) | 1 ligne

Modification BDD

r105 | stenaille | 2020-03-19 17 :18 :46 +0100 (jeu. 19 mars 2020) | 1 ligne

REcommit bouml

r104 | abonnet | 2020-03-19 17 :14 :01 +0100 (jeu. 19 mars 2020) | 1 ligne

modification BDD

r103 | abonnet | 2020-03-19 17 :07 :10 +0100 (jeu. 19 mars 2020) | 1 ligne

modification BDD

r102 | abonnet | 2020-03-19 17 :04 :58 +0100 (jeu. 19 mars 2020) | 1 ligne

Modification BDD

r101 | abonnet | 2020-03-19 17 :04 :21 +0100 (jeu. 19 mars 2020) | 1 ligne

Modification BDD

r100 | tvaira | 2020-03-19 17 :03 :48 +0100 (jeu. 19 mars 2020) | 1 ligne

Suppression de 2.lock

r99 | abonnet | 2020-03-19 16 :54 :55 +0100 (jeu. 19 mars 2020) | 1 ligne

Implémentation méthodes permettant d'ajouter une nouvelle campagne dans la BDD

r98 | abonnet | 2020-03-19 16 :51 :14 +0100 (jeu. 19 mars 2020) | 1 ligne

modification BDD

r97 | abonnet | 2020-03-19 16 :38 :14 +0100 (jeu. 19 mars 2020) | 1 ligne

Deplacement bdd

r96 | abonnet | 2020-03-19 16 :38 :02 +0100 (jeu. 19 mars 2020) | 1 ligne

Deplacement bdd

r95 | stenaille | 2020-03-19 16 :29 :48 +0100 (jeu. 19 mars 2020) | 1 ligne

modification diagramme séquence déplacer le robot format png

r94 | stenaille | 2020-03-19 16 :26 :18 +0100 (jeu. 19 mars 2020) | 1 ligne

diagrame séquence déplacer le robot format png

r93 | stenaille | 2020-03-19 16 :23 :28 +0100 (jeu. 19 mars 2020) | 1 ligne

diagrame séquence déplacer le robot

r92 | stenaille | 2020-03-19 16 :21 :17 +0100 (jeu. 19 mars 2020) | 2 lignes

reglage bug configuration camera

r91 | tvaira | 2020-03-19 16 :19 :05 +0100 (jeu. 19 mars 2020) | 2 lignes

Ajout du déploiement des fichiers SQLite de base

r90 | abonnet | 2020-03-19 15 :54 :19 +0100 (jeu. 19 mars 2020) | 1 ligne

Implémentation méthodes permettant de charger les campagnes présente dans la BDD

r89 | tvaira | 2020-03-19 08 :29 :10 +0100 (jeu. 19 mars 2020) | 4 lignes

Ajout des méthodes executer() et recuperer() dans la classe BaseDeDonnees Suppression des warnings

r88 | abonnet | 2020-03-18 18 :01 :11 +0100 (mer. 18 mars 2020) | 1 ligne

Implémentation classe BaseDeDonnees

r87 | stenaille | 2020-03-18 17 :59 :40 +0100 (mer. 18 mars 2020) | 1 ligne

Ajout boulm-rovnet-tenaille

r86 | abonnet | 2020-03-18 17 :49 :36 +0100 (mer. 18 mars 2020) | 1 ligne

Implémentation classe BaseDeDonnees

r85 | stenaille | 2020-03-18 17 :32 :44 +0100 (mer. 18 mars 2020) | 2 lignes

Ajout campagnesArchives

r84 | abonnet | 2020-03-18 17 :28 :05 +0100 (mer. 18 mars 2020) | 1 ligne

Changement nom classe BaseDeDonnee -> BaseDeDonnees

r83 | abonnet | 2020-03-18 17 :18 :20 +0100 (mer. 18 mars 2020) | 1 ligne

Modification campagneEnCours -> campagnesEnCours

r82 | abonnet | 2020-03-18 16 :46 :13 +0100 (mer. 18 mars 2020) | 1 ligne

Ajout fichier SQLite campagneEnCours

r81 | abonnet | 2020-03-18 15 :52 :18 +0100 (mer. 18 mars 2020) | 1 ligne

Realisation sd_prendreUnePhoto

r80 | stenaille | 2020-03-18 15 :47 :56 +0100 (mer. 18 mars 2020) | 2 lignes

Ajout de la classe BaseDeDonnee

r79 | stenaille | 2020-03-18 15 :04 :58 +0100 (mer. 18 mars 2020) | 2 lignes

Correction warning

r78 | abonnet | 2020-03-18 13 :39 :32 +0100 (mer. 18 mars 2020) | 1 ligne

Ajout diagrammeSequenceSysteme

r77 | abonnet | 2020-03-18 13 :13 :41 +0100 (mer. 18 mars 2020) | 1 ligne

Modification classe albumPhoto -> IHMAlbumPhoto

r76 | abonnet | 2020-03-17 17 :26 :48 +0100 (mar. 17 mars 2020) | 1 ligne

Implémentation méthode permettant de garder l'etat d'une photo lors de la reprise d'une campagne

r75 | abonnet | 2020-03-17 16 :14 :47 +0100 (mar. 17 mars 2020) | 1 ligne

Implémentation méthode permettant de garder l'etat d'une photo lors de la reprise d'une campagne

r74 | abonnet | 2020-03-17 14 :21 :25 +0100 (mar. 17 mars 2020) | 1 ligne

Implémentation méthode permettant de changer le status d'une photo dans l'album photo

r73 | tvaira | 2020-03-16 18 :45 :45 +0100 (lun. 16 mars 2020) | 2 lignes

Exemple QSignalMapper

r72 | abonnet | 2020-03-16 17 :09 :31 +0100 (lun. 16 mars 2020) | 1 ligne

Implémentation méthodes permettant d'avoir les informations d'une photo dans l'album photo

r71 | tvaira | 2020-03-16 16 :32 :12 +0100 (lun. 16 mars 2020) | 2 lignes

Correction erreur erase() campagneEnCours

r70 | abonnet | 2020-03-16 15 :53 :30 +0100 (lun. 16 mars 2020) | 1 ligne

Implémentation méthodes permettant de supprimer la campagne selectionne

r69 | abonnet | 2020-03-16 14 :56 :53 +0100 (lun. 16 mars 2020) | 1 ligne

Implementation méthodes permettant de reprendre une campagne arreté avec la bonne durée de campagne

r68 | abonnet | 2020-03-16 14 :00 :20 +0100 (lun. 16 mars 2020) | 1 ligne

Ajout d'une structure photo + modification de la classe albumPhoto en IHMAlbumPhoto

r67 | abonnet | 2020-03-15 17 :22 :51 +0100 (dim. 15 mars 2020) | 1 ligne

Implementation méthodes permettant de creer une nouvelle campagne

r66 | abonnet | 2020-03-14 13 :50 :57 +0100 (sam. 14 mars 2020) | 1 ligne

Modification sd :visualiser environnement

r65 | abonnet | 2020-03-14 13 :50 :44 +0100 (sam. 14 mars 2020) | 1 ligne

Modification sd :visualiser environnement

r64 | tvaira | 2020-03-14 07 :52 :53 +0100 (sam. 14 mars 2020) | 1 ligne

Correction sd visualierEnvironnement

r63 | tvaira | 2020-03-14 07 :40 :33 +0100 (sam. 14 mars 2020) | 1 ligne

Ne pas commiter le dossier lock

r62 | stenaille | 2020-03-13 20 :09 :44 +0100 (ven. 13 mars 2020) | 2 lignes

debug chemin image

r61 | abonnet | 2020-03-13 19 :35 :46 +0100 (ven. 13 mars 2020) | 1 ligne

Ajout dossier bouml au projet rovnet

r60 | abonnet | 2020-03-13 18 :10 :23 +0100 (ven. 13 mars 2020) | 1 ligne

Implementation classe IHMAccueil

r59 | abonnet | 2020-03-12 15 :30 :35 +0100 (jeu. 12 mars 2020) | 3 lignes

Implémentation méthodes permettant d'envoyer les trames correspondantes au pilotage de la caméra

r58 | stenaille | 2020-03-12 12 :56 :52 +0100 (jeu. 12 mars 2020) | 2 lignes

Ajout Humidité

r57 | stenaille | 2020-03-12 12 :25 :16 +0100 (jeu. 12 mars 2020) | 2 lignes

Documentation

r56 | stenaille | 2020-03-12 11 :53 :54 +0100 (jeu. 12 mars 2020) | 2 lignes

correction bug trame

r55 | abonnet | 2020-03-12 11 :25 :48 +0100 (jeu. 12 mars 2020) | 2 lignes

Modification classe manette -> heritage QGamepad

r54 | stenaille | 2020-03-12 10 :59 :35 +0100 (jeu. 12 mars 2020) | 2 lignes

Ajout structure bouton et modification contruction trame ordre et pince

r53 | abonnet | 2020-03-12 10 :53 :36 +0100 (jeu. 12 mars 2020) | 2 lignes

Mise a jour connect manette

r52 | abonnet | 2020-03-12 10 :26 :15 +0100 (jeu. 12 mars 2020) | 2 lignes

Documentation

r51 | stenaille | 2020-03-11 13 :57 :06 +0100 (mer. 11 mars 2020) | 2 lignes

Mise à jour code

r50 | abonnet | 2020-03-11 13 :37 :31 +0100 (mer. 11 mars 2020) | 1 ligne

Ajustation de la résolution de la caméra

r49 | stenaille | 2020-03-11 13 :36 :13 +0100 (mer. 11 mars 2020) | 2 lignes

Ajout des trames pince

r48 | stenaille | 2020-03-11 11 :00 :21 +0100 (mer. 11 mars 2020) | 2 lignes

optimisation du code sur la partie test des etat de la manette

r47 | stenaille | 2020-03-10 22 :20 :38 +0100 (mar. 10 mars 2020) | 2 lignes

Modification du code (maniere de tester l'etat de la manette)

r46 | stenaille | 2020-03-10 21 :40 :47 +0100 (mar. 10 mars 2020) | 2 lignes

Organisation du code

r45 | stenaille | 2020-03-10 17 :05 :27 +0100 (mar. 10 mars 2020) | 2 lignes

Organisation code

r44 | tvaira | 2020-03-09 12 :02 :19 +0100 (lun. 09 mars 2020) | 1 ligne

Revisions de code

r43 | abonnet | 2020-03-08 16 :04 :13 +0100 (dim. 08 mars 2020) | 1 ligne

Modification de la relation rov-ihm en relation bidirectionnelle

r42 | stenaille | 2020-03-07 18 :08 :42 +0100 (sam. 07 mars 2020) | 2 lignes

correction code non compilable

r41 | abonnet | 2020-03-06 17 :56 :17 +0100 (ven. 06 mars 2020) | 1 ligne

Mise a jour documentation

r40 | stenaille | 2020-03-06 17 :08 :45 +0100 (ven. 06 mars 2020) | 2 lignes

Ajout des différentes structures EtatManetteDeplacement

r39 | stenaille | 2020-03-06 16 :51 :30 +0100 (ven. 06 mars 2020) | 2 lignes

Modification du code pour correction de bug lors du pilotage du bras

r38 | abonnet | 2020-03-06 15 :12 :38 +0100 (ven. 06 mars 2020) | 1 ligne

Création classe ReglageVideo et AlbumPhoto

r37 | tvaira | 2020-03-06 06 :52 :47 +0100 (ven. 06 mars 2020) | 2 lignes

Test 18.04 + define NO_GAMEPAD pour une utilisation sans manette

r36 | stenaille | 2020-03-05 17 :18 :36 +0100 (jeu. 05 mars 2020) | 2 lignes

modification de la methode creationTramePilotage

r35 | abonnet | 2020-03-05 16 :16 :42 +0100 (jeu. 05 mars 2020) | 1 ligne

Implementation des methodes permettant de capturer plusieurs photos en un campagne et de pouvoir les visualiser clairement

r34 | abonnet | 2020-03-05 14 :14 :59 +0100 (jeu. 05 mars 2020) | 1 ligne

Implementation methodes permettant de capturer une image et de l'afficher dans l'album photo

r33 | stenaille | 2020-03-05 13 :49 :38 +0100 (jeu. 05 mars 2020) | 2 lignes

Ajout de la création des trames de pilotage

r32 | stenaille | 2020-03-04 17 :05 :47 +0100 (mer. 04 mars 2020) | 2 lignes

Création des connexions et des slots pour les bouton A, B, X, Y

r31 | abonnet | 2020-03-04 16 :46 :45 +0100 (mer. 04 mars 2020) | 1 ligne

Implementation methode permettant d'avoir les reglage de la vidéo dans une nouvelle fenetre

r30 | stenaille | 2020-03-04 16 :05 :29 +0100 (mer. 04 mars 2020) | 2 lignes

rectification du code

r29 | stenaille | 2020-03-04 15 :38 :44 +0100 (mer. 04 mars 2020) | 2 lignes

oublie

r28 | stenaille | 2020-03-04 15 :24 :49 +0100 (mer. 04 mars 2020) | 3 lignes

Ajout de l'affichage des valeurs températures et radiations de la trame capteur

r27 | tvaira | 2020-03-04 14 :01 :11 +0100 (mer. 04 mars 2020) | 2 lignes

Support pour OpenCV 2 (Ubuntu 16.04) ou 3 (Ubuntu 18.04)

r26 | abonnet | 2020-03-04 11 :14 :21 +0100 (mer. 04 mars 2020) | 2 lignes

Suppression label données telemetrie

r25 | abonnet | 2020-03-04 10 :01 :04 +0100 (mer. 04 mars 2020) | 1 ligne

Incrustation telemetrie flux video

r24 | abonnet | 2020-02-21 15 :01 :22 +0100 (ven. 21 févr. 2020) | 1 ligne

Implémentation méthodes permettant une incrustation de l'heure et des données télémétriques sur le flux vidéo

r23 | stenaille | 2020-02-20 20 :21 :15 +0100 (jeu. 20 févr. 2020) | 3 lignes

Mise à jour du code (retrait fonction inutile et ajout du code d'envoye des trames déplacement)

r22 | stenaille | 2020-02-19 18 :19 :29 +0100 (mer. 19 févr. 2020) | 2 lignes

Modification bug

r21 | stenaille | 2020-02-19 18 :00 :22 +0100 (mer. 19 févr. 2020) | 2 lignes

Modification de la connexion creerTrameDeplacement

r20 | abonnet | 2020-02-19 16 :48 :47 +0100 (mer. 19 févr. 2020) | 1 ligne

Ajout methode getCamera

r19 | stenaille | 2020-02-19 16 :03 :10 +0100 (mer. 19 févr. 2020) | 2 lignes

création de la connexion creerTrameDeplacement

r18 | abonnet | 2020-02-18 13 :11 :01 +0100 (mar. 18 févr. 2020) | 1 ligne

Implémentation méthodes permettant de modifier les parametres du flux video

r17 | stenaille | 2020-02-17 13 :53 :38 +0100 (lun. 17 févr. 2020) | 2 lignes

suppression du todo des structures

r16 | stenaille | 2020-02-17 13 :52 :40 +0100 (lun. 17 févr. 2020) | 3 lignes

Créations des structures de la manette et ajout des connexion entre la mannette et l'application Rov

r15 | tvaira | 2020-02-15 14 :24 :56 +0100 (sam. 15 févr. 2020) | 2 lignes

Insertion d'un TODO pour une structure Manette

r14 | stenaille | 2020-02-14 17 :12 :57 +0100 (ven. 14 févr. 2020) | 2 lignes

Todo connexion entre creationTrameDeplacement et creerTrameDeplacement

r13 | stenaille | 2020-02-14 17 :03 :42 +0100 (ven. 14 févr. 2020) | 5 lignes

création du signal creationTrameDeplacement(char deplacementAxeX, int puissance, char deplacementAxeY) dans la classe manette et la méthode slot creerTrameDeplacement(char deplacementAxeX, int puissance, char deplacementAxeY) dans la classe rov

r12 | stenaille | 2020-02-14 16 :48 :16 +0100 (ven. 14 févr. 2020) | 2 lignes

Ajout documentation classe manette

r11 | stenaille | 2020-02-14 16 :33 :36 +0100 (ven. 14 févr. 2020) | 2 lignes

Amélioration du code de la fonction creerTrameDeplacement()

r10 | abonnet | 2020-02-14 15 :58 :20 +0100 (ven. 14 févr. 2020) | 3 lignes

Implémentation des méthodes permettant de décoder la trame de télémtrie et afficher les données correspondantes

r9 | stenaille | 2020-02-14 15 :50 :55 +0100 (ven. 14 févr. 2020) | 2 lignes

Documentation

r8 | stenaille | 2020-02-14 14 :30 :28 +0100 (ven. 14 févr. 2020) | 2 lignes

ajout class manette

r7 | stenaille | 2020-02-14 14 :27 :25 +0100 (ven. 14 févr. 2020) | 2 lignes

Ajout de la methode detecterManette()

r6 | abonnet | 2020-02-14 13 :50 :04 +0100 (ven. 14 févr. 2020) | 2 lignes

Implémentation des méthodes permmettant d'afficher le flux vidéo sur l'IHM

r5 | abonnet | 2020-02-13 16 :18 :17 +0100 (jeu. 13 févr. 2020) | 1 ligne

r4 | abonnet | 2020-02-13 16 :17 :44 +0100 (jeu. 13 févr. 2020) | 2 lignes

Ajout des attribut de la caméra

r3 | abonnet | 2020-02-13 13 :33 :24 +0100 (jeu. 13 févr. 2020) | 2 lignes

Ajout des classes camera et rov

r2 | abonnet | 2020-02-12 14 :28 :56 +0100 (mer. 12 févr. 2020) | 2 lignes

Création du projet Qt

r1 | www-data | 2020-02-01 15 :03 :29 +0100 (sam. 01 févr. 2020) | 1 ligne

Creating initial repository structure

3 README

3.1 Projet

3.1.1 Présentation

Les objectifs du projet ROV'NET sont de se déplacer dans un milieu contaminé afin de faire des prises de vues :

- Le déplacement se fera à partir d'un châssis en liaison filaire à 4 roues motorisées indépendamment.
- Le ROV sera équipé :
  - d'une caméra d'aide au déplacements et/ou de capteurs d'obstacles
  - d'un capteur de température et de radioactivité
  - d'un dispositif de prise de vue motorisé
  - d'un bras de robotique avec pince de préhension

3.1.2 Base de données SQLite

```
PRAGMA foreign_keys = ON;

--
-- Structure de la table `campagne`
--

CREATE TABLE IF NOT EXISTS `campagne` (
    `IdCampagne`    INTEGER PRIMARY KEY AUTOINCREMENT,
    `IdTechnicien`  INTEGER NOT NULL,
    `nom`   TEXT NOT NULL UNIQUE,
    `lieu`  TEXT NOT NULL,
    `cheminSauvegarde`  TEXT NOT NULL,
    `date`  DATETIME NOT NULL,
    `duree` INTEGER NOT NULL,
    `enCours` NUMERIC NOT NULL,
    UNIQUE(`IdCampagne`, `IdTechnicien`),
    FOREIGN KEY(IdTechnicien) REFERENCES technicien(IdTechnicien)
);

--
-- Structure de la table `mesure`
--

CREATE TABLE IF NOT EXISTS `mesure` (
    `IdMesure`  INTEGER PRIMARY KEY AUTOINCREMENT,
    `IdCampagne`    INTEGER NOT NULL,
    `heure` DATETIME NOT NULL,
    `temperature`   REAL NOT NULL,
    `radiation` REAL NOT NULL,
    `humidite`  REAL NOT NULL,
    UNIQUE(`IdMesure`, `IdCampagne`),
    FOREIGN KEY(IdCampagne) REFERENCES campagne(IdCampagne)
);

--
-- Structure de la table `photo`
--

CREATE TABLE IF NOT EXISTS `photo` (
    `IdPhoto`   INTEGER PRIMARY KEY AUTOINCREMENT,
    `IdCampagne`    INTEGER NOT NULL,
    `cheminImage`   TEXT NOT NULL,
    `aGarder`   NUMERIC NOT NULL,
    UNIQUE(`IdPhoto`, `IdCampagne`),
    FOREIGN KEY(IdCampagne) REFERENCES campagne(IdCampagne)
);

--
-- Structure de la table `technicien`
--

CREATE TABLE IF NOT EXISTS `technicien` (
    `IdTechnicien`  INTEGER PRIMARY KEY AUTOINCREMENT,
    `nom`   TEXT NOT NULL,
    `prenom`    TEXT NOT NULL
);
```

3.1.3 Recette

- Servan Tenaille
  - Prendre en charge une manette par le logiciel
  - Recevoir et Visualiser les mesures des capteurs de température et d'irradiation
  - Déplacer le robot
  - Piloter le bras articulé
  - Envoyer les ordres de déplacement au robot et au bras
  - Archiver les mesures
- Anthony Bonnet
  - Démarrer une campagne
  - Visualiser l'environnement (le flux vidéo de la caméra et les données de télémétrie)
  - Recevoir les données de télémétrie
  - Prendre une photo
  - Configurer le contrôle de la caméra
  - Archiver les photos

3.1.4 Informations

Auteur

Servan Tenaille <servan.tenaille@gmail.com>
Anthony Bonnet <bonnet.anthony0@gmail.com>

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/rovnet/

4 A propos

Auteur

Servan Tenaille <servan.tenaille@gmail.com>
Anthony Bonnet <bonnet.anthony0@gmail.com>

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/rovnet/

5 Licence GPL

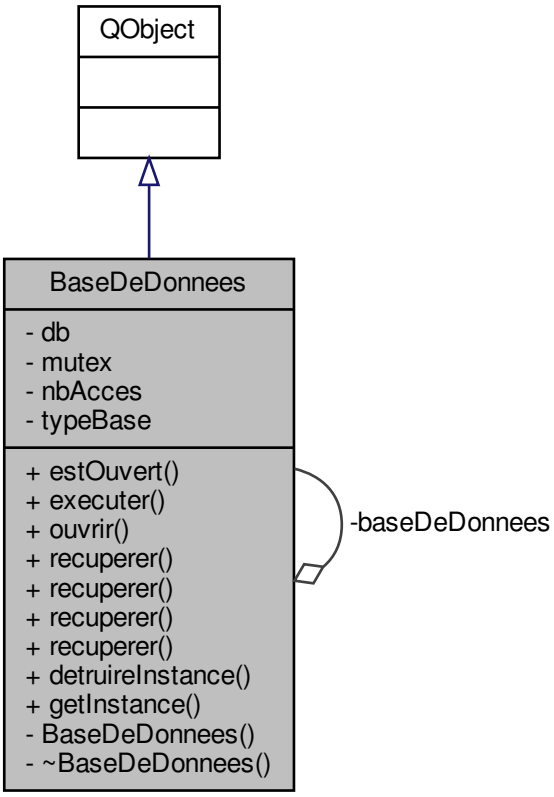
6 Documentation des classes

6.1 Référence de la classe BaseDeDonnees

Class permettant de s'interfacer avec la base de données.

```
#include "basededonnees.h"
```

Graphe de collaboration de BaseDeDonnees :



Fonctions membres publiques

- bool estOuvert ()
  Permet de savoir si la base de données est ouverte ou non.
- bool executer (QString requete)
  Permet d'executer la requete passé en paramètre au format SQL.
- bool ouvrir (QString fichierBase)
  Permet d'ouvrir le fichier de base de données passé en paramètre.
- bool recuperer (QString requete, QString &donnees)
  Permet d'executer la requete passé en paramètre au format SQL, et remplit le QString de sa réponse. Cette requête permet de récuperer un champs d'un enregistrement.
- bool recuperer (QString requete, QStringList &donnees)
  Permet d'executer la requete passé en paramètre au format SQL, et remplit le QStringListe de sa réponse. Cette requête permet de récuperer plusieurs champs d'un enregistrement.
- bool recuperer (QString requete, QVector< QString > &donnees)
  Permet d'executer la requete passé en paramètre au format SQL, et remplit le QVector<QString> de sa réponse. Cette requête permet de récuperer un champs de plusieurs enregistrements.
- bool recuperer (QString requete, QVector< QStringList > &donnees)
  Permet d'executer la requete passé en paramètre au format SQL, et remplit le QVector<QStringList> de sa réponse. Cette requête permet de récuperer plusieurs champs de plusieurs enregistrements.

Fonctions membres publiques statiques

- static void detruireInstance ()
  Permet de detruire l'instance en cours, Static elle est accessible depuis n'importe où
- static BaseDeDonnees * getInstance (QString type="QSQLITE")
  Permet de créer une instance de BDD ou de récuperer celle deja en cours, cette méthode controle l'instanciation des objet BaseDeDonnees. Static elle est accessible depuis n'importe où

Fonctions membres privées

- BaseDeDonnees (QString type)
  Constructeur de la classe BaseDeDonnees en privé afin de controller ses appels.
- ~BaseDeDonnees ()
  Destructeur de la classe BaseDeDonnees.

Attributs privés

- QSqlDatabase db
  Objet de type QSqlDatabase permettant la connexion avec la base de données.
- QMutex mutex
  Objet de type QMutex permettant de protéger l'objet db, en autorisant son accès par un seul thread à la fois.

Attributs privés statiques

- static BaseDeDonnees * baseDeDonnees = nullptr
  Objet de type BaseDeDonnees accessible uniquement depuis une méthode static.
- static int nbAcces = 0
  Attribut de type int contenant le nombre d'accès en cours à la base de données.
- static QString typeBase = "QSQLITE"
  Attribut de type QString contenant le type de la base de données (MySQL, SQLite, ...)

6.1.1 Description détaillée

Class permettant de s'interfacer avec la base de données.

Définition à la ligne 23 du fichier basededonnees.h.

6.1.2 Documentation des constructeurs et destructeur

6.1.2.1 BaseDeDonnees()

```
BaseDeDonnees::BaseDeDonnees (
            QString type )  [private]
```

Constructeur de la classe BaseDeDonnees en privé afin de controller ses appels.

Paramètres

type	

Définition à la ligne 15 du fichier basededonnees.cpp.

Références db, et typeBase.

Référencé par getInstance().

```
00016 {
00017     #ifdef DEBUG_BASEDEDONNEES
00018     qDebug() << Q_FUNC_INFO << type;
00019     #endif
00020     db = QSqlDatabase::addDatabase(type);
00021     typeBase = type;
00022 }
```

6.1.2.2 ~BaseDeDonnees()

```
BaseDeDonnees::~BaseDeDonnees ( )  [private]
```

Destructeur de la classe BaseDeDonnees.

Définition à la ligne 24 du fichier basededonnees.cpp.

```
00025 {
00026     #ifdef DEBUG_BASEDEDONNEES
00027     qDebug() << Q_FUNC_INFO;
00028     #endif
00029 }
```

6.1.3 Documentation des fonctions membres

6.1.3.1 detruireInstance()

```
void BaseDeDonnees::detruireInstance ( )  [static]
```

Permet de detruire l'instance en cours, Static elle est accessible depuis n'importe où

Définition à la ligne 44 du fichier basededonnees.cpp.

Références baseDeDonnees, et nbAcces.

Référencé par IHMAccueil : :~IHMAccueil().

```
00045 {
00046     if(baseDeDonnees != nullptr)
00047     {
00048         if(nbAcces > 0)
00049             nbAcces--;
00050
00051         #ifdef DEBUG_BASEDEDONNEES
00052         qDebug() << Q_FUNC_INFO << "nbAcces restants" << nbAcces;
00053         #endif
00054
00055         if(nbAcces == 0)
00056         {
00057             delete baseDeDonnees;
00058             baseDeDonnees = nullptr;
00059         }
00060     }
00061 }
```

6.1.3.2 estOuvert()

```
bool BaseDeDonnees::estOuvert ( )
```

Permet de savoir si la base de données est ouverte ou non.

Renvoie

un booleen correspondant à l'état d'ouverture de la base de données

Définition à la ligne 98 du fichier basededonnees.cpp.

Références db, et mutex.

```
00099 {
00100     QMutexLocker verrou(&mutex);
00101     return db.isOpen();
00102 }
```

6.1.3.3 executer()

```
bool BaseDeDonnees::executer (
            QString requete )
```

Permet d'executer la requete passé en paramètre au format SQL.

Paramètres

requete	

Renvoie

un booleen correspondant à l'état de retour de la requête

Définition à la ligne 104 du fichier basededonnees.cpp.

Références db, et mutex.

Référencé par IHMAccueil : :ajouterPhotoBDD(), IHMAccueil : :archiverCampagne(), IHMAccueil : :enregisterMesureBDD(), IHMAccueil : :enregistrerCampagneBDD(), IHMAccueil : :modifierCampagneBDD(), et IHMAccueil : :supprimerCampagne().

```
00105 {
00106     QMutexLocker verrou(&mutex);
00107     QSqlQuery r;
00108     bool retour;
00109
00110     if(db.isOpen())
00111     {
00112         if(requete.contains("UPDATE") || requete.contains("INSERT") || requete.contains("DELETE"))
00113         {
00114             retour = r.exec(requete);
00115             #ifdef DEBUG_BASEDEDONNEES
00116             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00117             #endif
00118             if(retour)
00119             {
00120                 return true;
00121             }
00122             else
00123             {
00124                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00125                 return false;
00126             }
00127         }
00128         else
00129         {
00130             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00131             return false;
00132         }
00133     }
00134     else
00135         return false;
00136
00137 }
```

6.1.3.4 getInstance()

```
BaseDeDonnees * BaseDeDonnees::getInstance (
             QString type = "QSQLITE" )  [static]
```

Permet de créer une instance de BDD ou de récuperer celle deja en cours, cette méthode controle l'instanciation des objet BaseDeDonnees. Static elle est accessible depuis n'importe où

Paramètres

type	

Renvoie

Instance de la BDD

Définition à la ligne 31 du fichier basededonnees.cpp.

Références baseDeDonnees, BaseDeDonnees(), et nbAcces.

Référencé par IHMAccueil : :IHMAccueil().

```
00032 {
00033     if(baseDeDonnees == nullptr)
00034         baseDeDonnees = new BaseDeDonnees(type);
00035
00036     nbAcces++;
00037     #ifdef DEBUG_BASEDEDONNEES
00038     qDebug() << Q_FUNC_INFO << "nbAcces" << nbAcces;
00039     #endif
00040
00041     return baseDeDonnees;
00042 }
```

6.1.3.5 ouvrir()

```
bool BaseDeDonnees::ouvrir (
            QString fichierBase )
```

Permet d'ouvrir le fichier de base de données passé en paramètre.

Paramètres

fichierBase	

Renvoie

booleen définissant si l'accès BDD s'est réalisé correctement

Définition à la ligne 63 du fichier basededonnees.cpp.

Références db, mutex, et typeBase.

Référencé par IHMAccueil : :archiverCampagne(), IHMAccueil : :chargerCampagnes(), IHMAccueil : :enregisterMesureBDD(), IH↩MAccueil : :enregistrerCampagneBDD(), IHMAccueil : :ouvrirArchive(), IHMAccueil : :ouvrirGraphiques(), IHMAccueil : :rechercher↩Campagne(), et IHMAccueil : :supprimerCampagne().

```
00064 {
00065     if(typeBase != "QSQLITE")
00066         return false;
00067     QMutexLocker verrou(&mutex);
00068     if(!db.isOpen())
00069     {
00070        db.setDatabaseName(fichierBase);
00071
00072        #ifdef DEBUG_BASEDEDONNEES
00073        qDebug() << Q_FUNC_INFO << db.databaseName();
00074         #endif
00075
00076        if(db.open())
00077        {
00078
00079            #ifdef DEBUG_BASEDEDONNEES
00080            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Ouverture réussie à %1").arg(
      db.databaseName());
00081            #endif
```

```
00082
00083             return true;
00084         }
00085         else
00086         {
00087             #ifdef DEBUG_BASEDEDONNEES
00088             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : impossible d'ouvrir la base de données !"
      );
00089             #endif
00090             QMessageBox::critical(nullptr, QString::fromUtf8("BaseDeDonnees"), QString::fromUtf8("Impossible
       d'ouvrir la base de données !"));
00091             return false;
00092         }
00093     }
00094     else
00095         return true;
00096 }
```

6.1.3.6 recuperer() [1/4]

```
bool BaseDeDonnees::recuperer (
              QString requete,
              QString & donnees )
```

Permet d'executer la requete passé en paramètre au format SQL, et remplit le QString de sa réponse. Cette requête permet de récuperer un champs d'un enregistrement.

Paramètres

requete	
donnees	

Renvoie

un booleen correspondant à l'état de retour de la requête

Définition à la ligne 139 du fichier basededonnees.cpp.

Références db, et mutex.

Référencé par IHMAccueil : :ajouterPhotoBDD(), IHMAccueil : :creerCampagne(), IHMAccueil : :enregistrerCampagneBDD(), IH↩ MAccueil : :modifierCampagneBDD(), IHMAccueil : :ouvrirArchive(), IHMAccueil : :ouvrirGraphiques(), IHMAccueil : :rechercher↩ Campagne(), IHMAccueil : :recupererCampagneEnCours(), IHMAccueil : :recupererIdCampagne(), IHMAccueil : :recupererNb↩ Photos(), IHMAccueil : :recupererPhotos(), et IHMAccueil : :supprimerPhotoLocal().

```
00140 {
00141     QMutexLocker verrou(&mutex);
00142     QSqlQuery r;
00143     bool retour;
00144
00145     if(db.isOpen())
00146     {
00147         if(requete.contains("SELECT"))
00148         {
00149             retour = r.exec(requete);
00150             #ifdef DEBUG_BASEDEDONNEES
00151             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00152             #endif
00153             if(retour)
00154             {
00155                 r.first();
00156
00157                 if(!r.isValid())
00158                 {
00159                     #ifdef DEBUG_BASEDEDONNEES
00160                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Résultat non valide !");
00161                     #endif
```

```
00162                     return false;
00163                 }
00164
00165                 if(r.isNull(0))
00166                 {
00167                     #ifdef DEBUG_BASEDEDONNEES
00168                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Aucun résultat !");
00169                     #endif
00170                     return false;
00171                 }
00172                 donnees = r.value(0).toString();
00173                 #ifdef DEBUG_BASEDEDONNEES
00174                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00175                 #endif
00176                 return true;
00177             }
00178             else
00179             {
00180                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00181                 return false;
00182             }
00183         }
00184         else
00185         {
00186             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00187             return false;
00188         }
00189     }
00190     else
00191         return false;
00192 }
```

6.1.3.7 recuperer() [2/4]

```
bool BaseDeDonnees::recuperer (
            QString requete,
            QStringList & donnees )
```

Permet d'executer la requete passé en paramètre au format SQL, et remplit le QStringListe de sa réponse. Cette requête permet de récuperer plusieurs champs d'un enregistrement.

Paramètres

requete	
donnees	

Renvoie

un booleen correspondant à l'état de retour de la requête

Définition à la ligne 194 du fichier basededonnees.cpp.

Références db, et mutex.

```
00195 {
00196     QMutexLocker verrou(&mutex);
00197     QSqlQuery r;
00198     bool retour;
00199
00200     if(db.isOpen())
00201     {
00202         if(requete.contains("SELECT"))
00203         {
00204             retour = r.exec(requete);
00205             #ifdef DEBUG_BASEDEDONNEES
00206             qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString, QStringList)> retour %1 pour
       la requete : %2").arg(QString::number(retour)).arg(requete);
00207             #endif
```

```
00208             if(retour)
00209             {
00210                 r.first();
00211 
00212                 if(!r.isValid())
00213                 {
00214                     #ifdef DEBUG_BASEDEDONNEES
00215                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Résultat non valide !");
00216                     #endif
00217                     return false;
00218                 }
00219 
00220                 for(int i=0;i<r.record().count();i++)
00221                     if(!r.isNull(i))
00222                         donnees << r.value(i).toString();
00223                 #ifdef DEBUG_BASEDEDONNEES
00224                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00225                 #endif
00226                 return true;
00227             }
00228             else
00229             {
00230                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00231                 return false;
00232             }
00233         }
00234         else
00235         {
00236             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00237             return false;
00238         }
00239     }
00240     else
00241         return false;
00242 }
```

6.1.3.8 recuperer() [3/4]

```
bool BaseDeDonnees::recuperer (
            QString requete,
            QVector< QString > & donnees )
```

Permet d'executer la requete passé en paramètre au format SQL, et remplit le QVector<QString> de sa réponse. Cette requête permet de récuperer un champs de plusieurs enregistrements.

Paramètres

requete	
donnees	

Renvoie

un booleen correspondant à l'état de retour de la requête

Définition à la ligne 244 du fichier basededonnees.cpp.

Références db, et mutex.

```
00245 {
00246     QMutexLocker verrou(&mutex);
00247     QSqlQuery r;
00248     bool retour;
00249     QString data;
00250 
00251     if(db.isOpen())
00252     {
00253         if(requete.contains("SELECT"))
00254         {
```

```
00255             retour = r.exec(requete);
00256             #ifdef DEBUG_BASEDEDONNEES
00257             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00258             #endif
00259             if(retour)
00260             {
00261                 while ( r.next() )
00262                 {
00263                     data = r.value(0).toString();
00264 
00265                     #ifdef DEBUG_BASEDEDONNEES
00266                     qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00267                     #endif
00268 
00269                     donnees.push_back(data);
00270                 }
00271                 #ifdef DEBUG_BASEDEDONNEES
00272                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00273                 #endif
00274                 return true;
00275             }
00276             else
00277             {
00278                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00279                 return false;
00280             }
00281         }
00282         else
00283         {
00284             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00285             return false;
00286         }
00287     }
00288     else
00289         return false;
00290 }
```

6.1.3.9 recuperer() [4/4]

```
bool BaseDeDonnees::recuperer (
            QString requete,
            QVector< QStringList > & donnees )
```

Permet d'executer la requete passé en paramètre au format SQL, et remplit le QVector<QStringList> de sa réponse. Cette requête permet de récuperer plusieurs champs de plusieurs enregistrements.

Paramètres

requete	
donnees	

Renvoie

un booleen correspondant à l'état de retour de la requête

Définition à la ligne 292 du fichier basededonnees.cpp.

Références db, et mutex.

```
00293 {
00294     QMutexLocker verrou(&mutex);
00295     QSqlQuery r;
00296     bool retour;
00297     QStringList data;
00298 
00299     if(db.isOpen())
00300     {
```

```
00301         if(requete.contains("SELECT"))
00302         {
00303             retour = r.exec(requete);
00304             #ifdef DEBUG_BASEDEDONNEES
00305             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00306             #endif
00307             if(retour)
00308             {
00309                 while ( r.next() )
00310                 {
00311                     for(int i=0;i<r.record().count();i++)
00312                         data << r.value(i).toString();
00313 
00314                     #ifdef DEBUG_BASEDEDONNEES
00315                     qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00316                     for(int i=0;i<r.record().count();i++)
00317                         qDebug() << r.value(i).toString();
00318                     #endif
00319 
00320                     donnees.push_back(data);
00321 
00322                     data.clear();
00323                 }
00324                 #ifdef DEBUG_BASEDEDONNEES
00325                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00326                 #endif
00327                 return true;
00328             }
00329             else
00330             {
00331                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00332                 return false;
00333             }
00334         }
00335         else
00336         {
00337             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00338             return false;
00339         }
00340     }
00341     else
00342         return false;
00343 }
```

6.1.4 Documentation des données membres

6.1.4.1 baseDeDonnees

```
BaseDeDonnees * BaseDeDonnees::baseDeDonnees = nullptr  [static], [private]
```

Objet de type BaseDeDonnees accessible uniquement depuis une méthode static.

Définition à la ligne 27 du fichier basededonnees.h.

Référencé par detruireInstance(), et getInstance().

6.1.4.2 db

```
QSqlDatabase BaseDeDonnees::db  [private]
```

Objet de type QSqlDatabase permettant la connexion avec la base de données.

Définition à la ligne 30 du fichier basededonnees.h.

Référencé par BaseDeDonnees(), estOuvert(), executer(), ouvrir(), et recuperer().

6.1.4.3 mutex

```
QMutex BaseDeDonnees::mutex  [private]
```

Objet de type QMutex permettant de protéger l'objet db, en autorisant son accès par un seul thread à la fois.

Définition à la ligne 31 du fichier basededonnees.h.

Référencé par estOuvert(), executer(), ouvrir(), et recuperer().

6.1.4.4 nbAcces

```
int BaseDeDonnees::nbAcces = 0  [static], [private]
```

Attribut de type int contenant le nombre d'accès en cours à la base de données.

Définition à la ligne 29 du fichier basededonnees.h.

Référencé par detruireInstance(), et getInstance().

6.1.4.5 typeBase

```
QString BaseDeDonnees::typeBase = "QSQLITE"  [static], [private]
```

Attribut de type QString contenant le type de la base de données (MySQL, SQLite, ...)

Définition à la ligne 28 du fichier basededonnees.h.

Référencé par BaseDeDonnees(), et ouvrir().

La documentation de cette classe a été générée à partir des fichiers suivants :

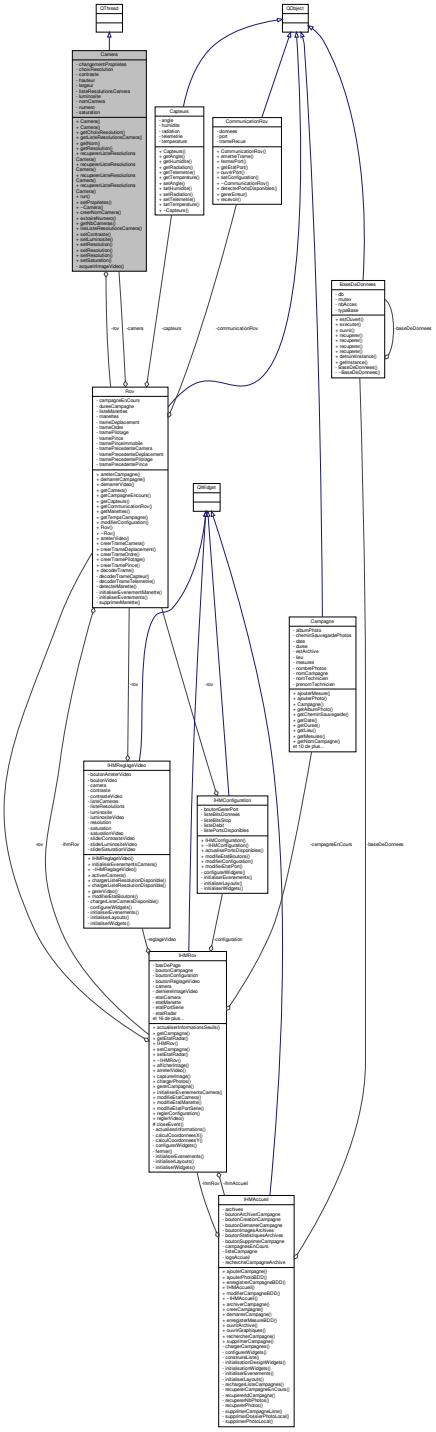
- basededonnees.h
- basededonnees.cpp

6.2 Référence de la classe Camera

Class permettant de mettre en place une communication avec la camera.

```
#include "camera.h"
```

Graphe de collaboration de Camera :



Connecteurs publics

- void setContraste (int contraste)
  Modifie le contraste de la caméra.
- void setLuminosite (int luminosite)
  Modifie la luminosite de la caméra.
- void setResolution (int largeur, int hauteur)
  Modifie la résolution (largeur x hauteur)
- void setResolution (QSize resolution)
  Modifie la résolution (largeur x hauteur)
- void setResolution (int choix)

Modifie la résolution (index dans la liste)
— void setSaturation (int saturation)
Modifie la saturation de la caméra.

Signaux

— void finVideo ()
Envoie un signal lorsque la vidéo est interrompu.
— void nouvelleImage (QPixmap image)
Envoie un signal lorsque une nouvelle image du flux vidéo est disponible.

Fonctions membres publiques

— Camera (Rov *rov, int numero, int choixResolution=-1)
Constructeur de la classe Camera.
— Camera (Rov *rov, QString nomCamera, int choixResolution=-1)
Constructeur de la classe Camera.
— int getChoixResolution ()
Récupère le choix de la resolution active.
— QList< QSize > getListeResolutionsCamera ()
Retourne la liste des résolutions supportées par la caméra.
— QString getNom () const
Retourne le nom de la caméra.
— QSize getResolution ()
Récupère la resolution active.
— void recupererListeResolutionsCamera ()
Récupère la liste des résolutions supportées par la caméra sélectionnée.
— void recupererListeResolutionsCamera (int numero)
Récupère la liste des résolutions supportées par la caméra à partir de son numéro.
— void recupererListeResolutionsCamera (QString nomCamera)
Récupère la liste des résolutions supportées par la caméra à partir de son nom.
— void recupererListeResolutionsCamera (QCameraInfo &cameraInfo)
Récupère la liste des résolutions supporté par la caméra.
— void run ()
Démarre une nouveau thread afin de capturer le flux video et l'envoyer à l'IHM.
— void setProprietes (cv : :VideoCapture &camera)
Après l'acquisition d'une nouvelle frame modifie les propriété de la caméra si ceux-ci ont été modifié par l'IHM.
— ~Camera ()
Destructeur de la classe Camera.

Fonctions membres publiques statiques

— static QString creerNomCamera (int numero)
Retourne le nom de caméra associé a son numéro.
— static int extraireNumero (QString nomCamera)
Retourne le numéro de caméra associé a son nom.
— static int getNbCameras ()
Retourne le nombre de caméras connectés.
— static QList< QSize > lireListeResolutionsCamera (QCameraInfo &cameraInfo)
Retourne la liste des résolutions supportés par la caméra passé en parametre.

Fonctions membres privées

— void acquerirImageVideo (cv : :VideoCapture &camera, cv : :Mat &frame)
Fait l'acquisition d'une nouvelle frame.

Attributs privés

- bool changementProprietes
  Attribut désignant si une propriete de la caméra doit être modifiée.
- int choixResolution
  Choix dans la liste contenant les résolutions supportés par la caméra.
- double contraste
  Attribut contenant le constraste de la vidéo.
- int hauteur
  Attribut contenant la hauteur (heigth) en pixels de la vidéo.
- int largeur
  Attribut contenant la largeur (width) en pixels de la vidéo.
- QList< QSize > listeResolutionsCamera
  Liste contenant les résolutions supportés par la caméra.
- double luminosite
  Attribut contenant la luminosite de la vidéo.
- QString nomCamera
  Attribut contenant le nom de la caméra sélectionnée.
- int numero
  Attribut contenant le numéro de la caméra sélectionnée.
- Rov * rov
  Objet rov permettant de récuperer les dernière mesures issues des capteurs.
- double saturation
  Attribut contenant la saturation de la vidéo.

6.2.1 Description détaillée

Class permettant de mettre en place une communication avec la camera.

Définition à la ligne 58 du fichier camera.h.

6.2.2 Documentation des constructeurs et destructeur

6.2.2.1 Camera() [1/2]

```
Camera::Camera (
            Rov * rov,
            int numero,
            int choixResolution = -1 )
```

Constructeur de la classe Camera.

Paramètres

rov	
numero	
choixResolution	

Définition à la ligne 12 du fichier camera.cpp.

Références contraste, creerNomCamera(), getNbCameras(), hauteur, largeur, luminosite, nomCamera, recupererListeResolutionsCamera(), saturation, et setResolution().

```
00012                                                   : rov(rov), numero(
      numero), largeur(LARGEUR_DEFAUT), hauteur(
      HAUTEUR_DEFAUT), luminosite(SEUIL_DEFAUT),
```

```
      contraste(SEUIL_DEFAUT), saturation(SEUIL_DEFAUT),
      changementProprietes(false), choixResolution(
      choixResolution)
00013 {
00014     #if CV_VERSION_MAJOR == 3
00015     qDebug() << Q_FUNC_INFO << "OpenCV" << CV_VERSION_MAJOR << CV_VERSION_MINOR;
00016     #else
00017     qDebug() << Q_FUNC_INFO << "OpenCV" << CV_MAJOR_VERSION << CV_MINOR_VERSION;
00018     #endif
00019
00020     Camera::getNbCameras();
00021
00022     nomCamera = Camera::creerNomCamera(numero);
00023     recupererListeResolutionsCamera();
00024     if(choixResolution == -1)
00025         setResolution(largeur, hauteur);
00026     else
00027         setResolution(choixResolution);
00028     qDebug() << Q_FUNC_INFO << this;
00029     qDebug() << Q_FUNC_INFO << "numero" << numero << "nomCamera" <<
      nomCamera;
00030     qDebug() << Q_FUNC_INFO << "largeur" << largeur << "hauteur" <<
      hauteur;
00031     qDebug() << Q_FUNC_INFO << "luminosite" << luminosite << "contraste" <<
      contraste << "saturation" << saturation;
00032 }
```

6.2.2.2 Camera() [2/2]

```
Camera::Camera (
            Rov * rov,
            QString nomCamera,
            int choixResolution = -1 )
```

Constructeur de la classe Camera.

Paramètres

rov	
nomCamera	
choixResolution	

Définition à la ligne 34 du fichier camera.cpp.

Références contraste, extraireNumero(), getNbCameras(), hauteur, largeur, luminosite, nomCamera, numero, recupererListe↩ ResolutionsCamera(), saturation, et setResolution().

```
00034                                                       : rov(rov),
      nomCamera(nomCamera), largeur(LARGEUR_DEFAUT),
      hauteur(HAUTEUR_DEFAUT), luminosite(SEUIL_DEFAUT),
      contraste(SEUIL_DEFAUT), saturation(SEUIL_DEFAUT),
      changementProprietes(false), choixResolution(
      choixResolution)
00035 {
00036     #if CV_VERSION_MAJOR == 3
00037     qDebug() << Q_FUNC_INFO << "OpenCV" << CV_VERSION_MAJOR << CV_VERSION_MINOR;
00038     #else
00039     qDebug() << Q_FUNC_INFO << "OpenCV" << CV_MAJOR_VERSION << CV_MINOR_VERSION;
00040     #endif
00041
00042     Camera::getNbCameras();
00043
00044     numero = Camera::extraireNumero(nomCamera);
00045
00046     recupererListeResolutionsCamera();
00047     if(choixResolution == -1)
00048         setResolution(largeur, hauteur);
00049     else
00050         setResolution(choixResolution);
00051
00052     qDebug() << Q_FUNC_INFO << this;
```

```
00053     qDebug() << Q_FUNC_INFO << "numero" << numero << "nomCamera" <<
      nomCamera;
00054     qDebug() << Q_FUNC_INFO << "largeur" << largeur << "hauteur" <<
      hauteur;
00055     qDebug() << Q_FUNC_INFO << "luminosite" << luminosite << "contraste" <<
      contraste << "saturation" << saturation;
00056 }
```

6.2.2.3 ∼Camera()

```
Camera::∼Camera ( )
```

Destructeur de la classe Camera.

Définition à la ligne 58 du fichier camera.cpp.

```
00059 {
00060     qDebug() << Q_FUNC_INFO << this;
00061 }
```

6.2.3 Documentation des fonctions membres

6.2.3.1 acquerirImageVideo()

```
void Camera::acquerirImageVideo (
             cv::VideoCapture & camera,
             cv::Mat & frame )  [private]
```

Fait l'acquisition d'une nouvelle frame.

Paramètres

camera	
frame	

Définition à la ligne 63 du fichier camera.cpp.

Référencé par run().

```
00064 {
00065     camera >> frame;
00066 }
```

6.2.3.2 creerNomCamera()

```
QString Camera::creerNomCamera (
             int numero )  [static]
```

Retourne le nom de caméra associé a son numéro.

Renvoie

le nom de caméra associé a son numéro

Paramètres

numero	

Définition à la ligne 294 du fichier camera.cpp.

Référencé par Camera(), Rov : :demarrerCampagne(), et recupererListeResolutionsCamera().

```
00295 {
00296     QString nom;
00297
00298     if(numero >= 0)
00299     {
00300         nom = QString("/dev/video") + QString::number(numero);
00301     }
00302     return nom;
00303 }
```

6.2.3.3 extraireNumero()

```
int Camera::extraireNumero (
            QString nomCamera )  [static]
```

Retourne le numéro de caméra associé a son nom.

Renvoie

le numéro de caméra associé a son nom

Paramètres

nomCamera	

Définition à la ligne 277 du fichier camera.cpp.

Références numero.

Référencé par Camera().

```
00278 {
00279     int numero = -1;
00280     QString video = "/dev/video";
00281
00282     if(nomCamera.contains(video))
00283     {
00284         QString n = nomCamera.mid(video.length(), nomCamera.length());
00285         bool ok;
00286         qDebug() << Q_FUNC_INFO << "nom" << nomCamera << "n" <<
      nomCamera.right(nomCamera.indexOf("/dev/video")) << "index" <<
      nomCamera.indexOf("/dev/video");
00287         numero = n.toInt(&ok);
00288         if(ok)
00289             return numero;
00290     }
00291     return numero;
00292 }
```

6.2.3.4 finVideo

```
void Camera::finVideo ( )  [signal]
```

Envoie un signal lorsque la vidéo est interrompu.

Référencé par run().

6.2.3.5 getChoixResolution()

```
int Camera::getChoixResolution ( )
```

Récupère le choix de la resolution active.

Renvoie

le choix de la resolution active

Définition à la ligne 129 du fichier camera.cpp.

Références choixResolution.

Référencé par IHMReglageVideo : :chargerListeResolutionDisponible().

```
00130 {
00131     return choixResolution;
00132 }
```

6.2.3.6 getListeResolutionsCamera()

```
QList< QSize > Camera::getListeResolutionsCamera ( )
```

Retourne la liste des résolutions supportées par la caméra.

Renvoie

la liste des résolutions supportées par la caméra

Définition à la ligne 182 du fichier camera.cpp.

Références listeResolutionsCamera.

```
00183 {
00184     return listeResolutionsCamera;
00185 }
```

6.2.3.7 getNbCameras()

```
int Camera::getNbCameras ( )  [static]
```

Retourne le nombre de caméras connectés.

Renvoie

le nombre de caméras connectés

Définition à la ligne 271 du fichier camera.cpp.

Référencé par Camera(), et Rov : :demarrerCampagne().

```
00272 {
00273     qDebug() << Q_FUNC_INFO  << "Caméra(s) disponible(s)" << QCameraInfo::availableCameras().count();
00274     return QCameraInfo::availableCameras().count();
00275 }
```

6.2.3.8 getNom()

```
QString Camera::getNom ( ) const
```

Retourne le nom de la caméra.

Renvoie

nom de la caméra

Définition à la ligne 103 du fichier camera.cpp.

Références nomCamera.

Référencé par IHMReglageVideo : :chargerListeCameraDisponible().

```
00104 {
00105     return nomCamera;
00106 }
```

6.2.3.9 getResolution()

```
QSize Camera::getResolution ( )
```

Récupère la resolution active.

Renvoie

la resolution active

Définition à la ligne 122 du fichier camera.cpp.

Références choixResolution, HAUTEUR_DEFAUT, LARGEUR_DEFAUT, et listeResolutionsCamera.

Référencé par IHMReglageVideo : :chargerListeResolutionDisponible().

```
00123 {
00124     if(choixResolution != -1)
00125         return listeResolutionsCamera.at(choixResolution);
00126     return QSize(LARGEUR_DEFAUT,HAUTEUR_DEFAUT);
00127 }
```

6.2.3.10 lireListeResolutionsCamera()

```
QList< QSize > Camera::lireListeResolutionsCamera (
            QCameraInfo & cameraInfo )  [static]
```

Retourne la liste des résolutions supportés par la caméra passé en parametre.

Renvoie

la liste des résolutions supportés par la caméra passé en parametre

Paramètres

cameraInfo	

Définition à la ligne 305 du fichier camera.cpp.

Référencé par IHMReglageVideo : :chargerListeResolutionDisponible().

```
00306 {
00307     QList<QSize> listeResolutions;
00308     listeResolutions.clear();
00309     if(QCameraInfo::availableCameras().count() > 0)
00310     {
00311         QCamera *camera = new QCamera(cameraInfo);
00312         QMediaRecorder *mediaRecorder = new QMediaRecorder(camera);
00313         camera->load();
00314         qDebug() << Q_FUNC_INFO << mediaRecorder->supportedResolutions().size();
00315         if(mediaRecorder->supportedResolutions().size() > 0)
00316         {
00317             foreach (const QSize &resolution, mediaRecorder->supportedResolutions())
00318             {
00319                 qDebug() << Q_FUNC_INFO << resolution.width() << "x" << resolution.height();
00320                 listeResolutions.push_back(resolution);
00321             }
00322         }
00323         delete mediaRecorder;
00324         delete camera;
00325     }
00326     return listeResolutions;
00327 }
```

6.2.3.11 nouvelleImage

```
void Camera::nouvelleImage (
            QPixmap image )  [signal]
```

Envoie un signal lorsque une nouvelle image du flux vidéo est disponible.

Paramètres

image	

Référencé par run().

6.2.3.12 recupererListeResolutionsCamera() [1/4]

```
void Camera::recupererListeResolutionsCamera ( )
```

Récupère la liste des résolutions supportées par la caméra sélectionnée.

Définition à la ligne 134 du fichier camera.cpp.

Références nomCamera.

Référencé par Camera(), et recupererListeResolutionsCamera().

```
00135 {
00136     QCameraInfo cameraInfo(nomCamera.toLatin1());
00137     recupererListeResolutionsCamera(cameraInfo);
00138 }
```

6.2.3.13 recupererListeResolutionsCamera() [2/4]

```
void Camera::recupererListeResolutionsCamera (
            int numero )
```

Récupère la liste des résolutions supportées par la caméra à partir de son numéro.

Paramètres

numero	

Définition à la ligne 140 du fichier camera.cpp.

Références creerNomCamera(), et recupererListeResolutionsCamera().

```
00141 {
00142     QString nom = Camera::creerNomCamera(numero);
00143     QCameraInfo cameraInfo(nom.toLatin1());
00144     recupererListeResolutionsCamera(cameraInfo);
00145 }
```

6.2.3.14 recupererListeResolutionsCamera() [3/4]

```
void Camera::recupererListeResolutionsCamera (
            QString nomCamera )
```

Récupère la liste des résolutions supportées par la caméra à partir de son nom.

Paramètres

nomCamera	

Définition à la ligne 147 du fichier camera.cpp.

Références recupererListeResolutionsCamera().

```
00148 {
00149     QCameraInfo cameraInfo(nomCamera.toLatin1());
00150     recupererListeResolutionsCamera(cameraInfo);
00151 }
```

6.2.3.15 recupererListeResolutionsCamera() [4/4]

```
void Camera::recupererListeResolutionsCamera (
            QCameraInfo & cameraInfo )
```

Récupère la liste des résolutions supporté par la caméra.

Paramètres

cameraInfo	

Définition à la ligne 153 du fichier camera.cpp.

Références HAUTEUR_DEFAUT, LARGEUR_DEFAUT, et listeResolutionsCamera.

```
00154 {
00155     #ifndef SANS_DETECTION
00156     listeResolutionsCamera.clear();
00157     if(QCameraInfo::availableCameras().count() > 0)
00158     {
00159         QCamera *camera = new QCamera(cameraInfo, this);
00160         QMediaRecorder *mediaRecorder = new QMediaRecorder(camera, this);
00161         camera->load();
00162         qDebug() << Q_FUNC_INFO << this << mediaRecorder->supportedResolutions().size();
00163         if(mediaRecorder->supportedResolutions().size() > 0)
00164         {
00165             foreach (const QSize &resolution, mediaRecorder->supportedResolutions())
00166             {
00167                 qDebug() << Q_FUNC_INFO << resolution.width() << "x" << resolution.height();
00168                 listeResolutionsCamera.push_back(resolution);
00169             }
00170         }
00171         delete mediaRecorder;
00172         delete camera;
00173     }
00174     #else
00175     Q_UNUSED(cameraInfo);
00176     listeResolutionsCamera.clear();
00177     QSize resolutionDefaut(LARGEUR_DEFAUT, HAUTEUR_DEFAUT);
00178     listeResolutionsCamera.push_back(resolutionDefaut);
00179     #endif
00180 }
```

6.2.3.16 run()

```
void Camera::run ( )
```

Démarre une nouveau thread afin de capturer le flux video et l'envoyer à l'IHM.

Définition à la ligne 68 du fichier camera.cpp.

Références acquerirImageVideo(), changementProprietes, finVideo(), nomCamera, nouvelleImage(), numero, et setProprietes().

```
00069 {
00070     qDebug() << Q_FUNC_INFO << "start" << "numero" << numero << "nomCamera" <<
      nomCamera;
00071     this->setPriority(QThread::NormalPriority);
00072
00073     if(numero < 0)
00074     {
00075         qDebug() << Q_FUNC_INFO  << "Erreur numero" << numero << "nomCamera" <<
      nomCamera;
00076         return;
00077     }
00078
00079     cv::VideoCapture camera(numero);
00080     cv::Mat frame;
00081
00082     setProprietes(camera);
00083
00084     while(camera.isOpened() && !isInterruptionRequested())
00085     {
00086         acquerirImageVideo(camera, frame);
00087         if(frame.empty())
00088             continue;
00089
00090         QPixmap pixmap = QPixmap::fromImage(QImage(frame.data, frame.cols, frame.rows, frame.step,
      QImage::Format_RGB888).rgbSwapped());
00091         emit nouvelleImage(pixmap);
00092
00093         if(changementProprietes)
00094             setProprietes(camera);
00095     }
00096
00097     camera.release();
00098     qDebug() << Q_FUNC_INFO << "stop" << "numero" << numero << "nomCamera" <<
      nomCamera;
00099
00100     emit finVideo();
00101 }
```

6.2.3.17 setContraste

```
void Camera::setContraste (
            int contraste )  [slot]
```

Modifie le contraste de la caméra.

Paramètres

contraste	

Définition à la ligne 259 du fichier camera.cpp.

Références changementProprietes.

```
00260 {
00261     this->contraste = double(contraste)/100;
00262     changementProprietes = true;
00263 }
```

6.2.3.18 setLuminosite

```
void Camera::setLuminosite (
            int luminosite )  [slot]
```

Modifie la luminosite de la caméra.

Paramètres

luminosite	

Définition à la ligne 253 du fichier camera.cpp.

Références changementProprietes.

```
00254 {
00255     this->luminosite = double(luminosite)/100;
00256     changementProprietes = true;
00257 }
```

6.2.3.19 setProprietes()

```
void Camera::setProprietes (
            cv::VideoCapture & camera )
```

Après l'acquisition d'une nouvelle frame modifie les propriété de la caméra si ceux-ci ont été modifié par l'IHM.

Paramètres

camera	

Définition à la ligne 108 du fichier camera.cpp.

Références changementProprietes, contraste, hauteur, largeur, luminosite, et saturation.

Référencé par run().

```
00109 {
00110     qDebug() << Q_FUNC_INFO << "largeur" << largeur << "hauteur" <<
      hauteur;
00111     camera.set(CV_CAP_PROP_FRAME_WIDTH, largeur);
00112     camera.set(CV_CAP_PROP_FRAME_HEIGHT, hauteur);
00113
00114     qDebug() << Q_FUNC_INFO << "luminosite" << luminosite << "contraste" <<
      contraste << "saturation" << saturation;
00115     camera.set(CV_CAP_PROP_BRIGHTNESS, luminosite);
00116     camera.set(CV_CAP_PROP_CONTRAST, contraste);
00117     camera.set(CV_CAP_PROP_SATURATION, saturation);
00118
00119     changementProprietes = false;
00120 }
```

6.2.3.20 setResolution [1/3]

```
void Camera::setResolution (
            int largeur,
            int hauteur )  [slot]
```

Modifie la résolution (largeur x hauteur)

Paramètres

largeur	
hauteur	

Définition à la ligne 187 du fichier camera.cpp.

Références changementProprietes, choixResolution, hauteur, largeur, et listeResolutionsCamera.

Référencé par Camera().

```
00188 {
00189     qDebug() << Q_FUNC_INFO << "largeur" << largeur << "hauteur" <<
      hauteur;
00190     QSize size(largeur, hauteur);
00191     int i = listeResolutionsCamera.indexOf(size);
00192     if (i != -1)
00193     {
00194         choixResolution = i;
00195         this->largeur = largeur;
00196         this->hauteur = hauteur;
00197         changementProprietes = true;
00198     }
00199     else
00200     {
00201         size = listeResolutionsCamera.last();
00202         choixResolution = listeResolutionsCamera.indexOf(size);;
00203         this->largeur = size.width();
00204         this->hauteur = size.height();
00205         changementProprietes = true;
00206     }
00207     qDebug() << Q_FUNC_INFO << "largeur" << this->largeur << "hauteur" << this->hauteur << "
      choixResolution" << choixResolution;
00208 }
```

6.2.3.21 setResolution [2/3]

```
void Camera::setResolution (
            QSize resolution )  [slot]
```

Modifie la résolution (largeur x hauteur)

Paramètres

resolution	(QSize)

Définition à la ligne 210 du fichier camera.cpp.

Références changementProprietes, choixResolution, hauteur, largeur, et listeResolutionsCamera.

```
00211 {
00212     qDebug() << Q_FUNC_INFO << "largeur" << resolution.width() << "hauteur" << resolution.height();
00213     int i = listeResolutionsCamera.indexOf(resolution);
00214     if (i != -1)
00215     {
00216         choixResolution = i;
00217         this->largeur = resolution.width();
00218         this->hauteur = resolution.height();
00219         changementProprietes = true;
00220     }
00221     else
00222     {
00223         QSize size = listeResolutionsCamera.last();
00224         choixResolution = listeResolutionsCamera.indexOf(size);;
00225         this->largeur = size.width();
00226         this->hauteur = size.height();
00227         changementProprietes = true;
00228     }
00229     qDebug() << Q_FUNC_INFO << "largeur" << this->largeur << "hauteur" << this->
      hauteur << "choixResolution" << choixResolution;
00230 }
```

6.2.3.22 setResolution [3/3]

```
void Camera::setResolution (
            int choix )  [slot]
```

Modifie la résolution (index dans la liste)

Paramètres

choix	

Définition à la ligne 232 du fichier camera.cpp.

Références changementProprietes, choixResolution, hauteur, largeur, et listeResolutionsCamera.

```
00233 {
00234     qDebug() << Q_FUNC_INFO << "choix" << choix;
00235     if(choix < listeResolutionsCamera.size())
00236     {
00237         choixResolution = choix;
00238         this->largeur = listeResolutionsCamera.at(choix).width();
00239         this->hauteur = listeResolutionsCamera.at(choix).height();
00240         changementProprietes = true;
00241     }
00242     else
```

```
00243     {
00244         QSize size = listeResolutionsCamera.last();
00245         choixResolution = listeResolutionsCamera.indexOf(size);;
00246         this->largeur = size.width();
00247         this->hauteur = size.height();
00248         changementProprietes = true;
00249     }
00250     qDebug() << Q_FUNC_INFO << "largeur" << this->largeur << "hauteur" << this->
      hauteur << "choixResolution" << choixResolution;
00251 }
```

6.2.3.23 setSaturation

```
void Camera::setSaturation (
             int saturation )  [slot]
```

Modifie la saturation de la caméra.

Paramètres

saturation	

Définition à la ligne 265 du fichier camera.cpp.

Références changementProprietes.

```
00266 {
00267     this->saturation = double(saturation)/100;
00268     changementProprietes = true;
00269 }
```

6.2.4 Documentation des données membres

6.2.4.1 changementProprietes

```
bool Camera::changementProprietes  [private]
```

Attribut désignant si une propriete de la caméra doit être modifiée.

Définition à la ligne 70 du fichier camera.h.

Référencé par run(), setContraste(), setLuminosite(), setProprietes(), setResolution(), et setSaturation().

6.2.4.2 choixResolution

```
int Camera::choixResolution  [private]
```

Choix dans la liste contenant les résolutions supportés par la caméra.

Définition à la ligne 72 du fichier camera.h.

Référencé par getChoixResolution(), getResolution(), et setResolution().

6.2.4.3 contraste

```
double Camera::contraste  [private]
```

Attribut contenant le constraste de la vidéo.

Définition à la ligne 68 du fichier camera.h.

Référencé par Camera(), et setProprietes().

6.2.4.4 hauteur

```
int Camera::hauteur  [private]
```

Attribut contenant la hauteur (heigth) en pixels de la vidéo.

Définition à la ligne 66 du fichier camera.h.

Référencé par Camera(), setProprietes(), et setResolution().

6.2.4.5 largeur

```
int Camera::largeur  [private]
```

Attribut contenant la largeur (width) en pixels de la vidéo.

Définition à la ligne 65 du fichier camera.h.

Référencé par Camera(), setProprietes(), et setResolution().

6.2.4.6 listeResolutionsCamera

```
QList<QSize> Camera::listeResolutionsCamera  [private]
```

Liste contenant les résolutions supportés par la caméra.

Définition à la ligne 71 du fichier camera.h.

Référencé par getListeResolutionsCamera(), getResolution(), recupererListeResolutionsCamera(), et setResolution().

6.2.4.7 luminosite

```
double Camera::luminosite  [private]
```

Attribut contenant la luminosite de la vidéo.

Définition à la ligne 67 du fichier camera.h.

Référencé par Camera(), et setProprietes().

6.2.4.8 nomCamera

```
QString Camera::nomCamera  [private]
```

Attribut contenant le nom de la caméra sélectionnée.

Définition à la ligne 63 du fichier camera.h.

Référencé par Camera(), getNom(), recupererListeResolutionsCamera(), et run().

6.2.4.9 numero

```
int Camera::numero  [private]
```

Attribut contenant le numéro de la caméra sélectionnée.

Définition à la ligne 64 du fichier camera.h.

Référencé par Camera(), extraireNumero(), et run().

6.2.4.10 rov

```
Rov* Camera::rov  [private]
```

Objet rov permettant de récuperer les dernière mesures issues des capteurs.

Définition à la ligne 62 du fichier camera.h.

6.2.4.11 saturation

```
double Camera::saturation  [private]
```

Attribut contenant la saturation de la vidéo.

Définition à la ligne 69 du fichier camera.h.

Référencé par Camera(), et setProprietes().

La documentation de cette classe a été générée à partir des fichiers suivants :

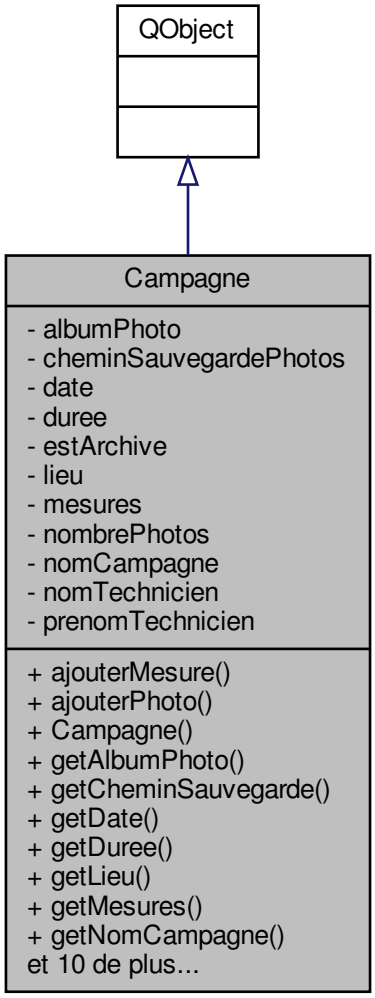
- camera.h
- camera.cpp

6.3 Référence de la classe Campagne

Class contenant les informations de la campagne en cours.

`#include "campagne.h"`

Graphe de collaboration de Campagne :



Fonctions membres publiques

- void ajouterMesure (Mesure &mesure)
  Ajoute une mesure dans le conteneur de mesure.
- void ajouterPhoto (Photo &photo)
  Ajoute une photo dans l'album photo.
- Campagne (QString nomCampagne, QString lieu, QString nomTechnicien, QString prenomTechnicien, QDateTime date, QObject *parent=nullptr, int duree=0)
  Constructeur de la classe Campagne.
- QVector< Photo > getAlbumPhoto () const
  Retourne l'album photo de la campagne.
- QString getCheminSauvegarde () const
  Retourne le chemin du dossier de sauvegarde des photos.

- QDateTime getDate () const
  Retourne la date de la campagne.
- int getDuree () const
  Retourne la durée de la campagne.
- QString getLieu () const
  Retourne le lieu de la campagne.
- QVector< Mesure > getMesures () const
  Retourne les mesures de la campagne.
- QString getNomCampagne () const
  Retourne le nom de la campagne.
- QString getNomTechnicien () const
  Retourne le nom du technicien.
- QString getPrenomTechnicien () const
  Retourne le prenom du technicien.
- int incrementeNombrePhoto ()
  Incrémente le nombre de photo prises durant la campagne et retourne son nombre.
- void modifierArchivePhoto (int numeroPhoto)
  Modifie l'état d'archive de la photo correspondant au numéro passé en paramètre.
- void setCheminSauvegarde (QString chemin)
  Modifie le chemin de sauvegarde des photos.
- void setDuree (int duree)
  Modifie la duree de la campagne.
- void setNombrePhotos (int nombre)
  Modifie le nombre de photos prises durant la campagne.
- void supprimerMesures ()
  Supprime les mesure du conteneur de Mesure, une fois celles-ci archivés dans la BDD.
- void supprimerPhotos ()
  Supprime les photo du conteneur de Photo, une fois celles-ci archivés dans la BDD.
- ~Campagne ()
  Destructeur de la classe Campagne.

Attributs privés

- QVector< Photo > albumPhoto
  Conteneur des photos prises durant la campagne.
- QString cheminSauvegardePhotos
  Attribut contenant le chemin de sauvegarde des photos.
- QDateTime date
  Attribut contenant la date de la campagne.
- int duree
  Attribut contenant la durée de la campagne en millisecondes.
- bool estArchive
  Attribut booléen afin de savoir si la campagne est toujours en cours.
- QString lieu
  Attribut contenant le lieu de la campagne.
- QVector< Mesure > mesures
  Conteneur des mesures enregistrés durant la campagne.
- int nombrePhotos
  Attribut contenant le nombre de photos prise durant la campagne.
- QString nomCampagne
  Attribut contenant le nom de la campagne.
- QString nomTechnicien
  Attribut contenant le nom du technicien.
- QString prenomTechnicien
  Attribut contenant le nom du technicien.

6.3.1 Description détaillée

Class contenant les informations de la campagne en cours.

Définition à la ligne 34 du fichier campagne.h.

6.3.2 Documentation des constructeurs et destructeur

6.3.2.1 Campagne()

```
Campagne::Campagne (
            QString nomCampagne,
            QString lieu,
            QString nomTechnicien,
            QString prenomTechnicien,
            QDateTime date,
            QObject * parent = nullptr,
            int duree = 0 )
```

Constructeur de la classe Campagne.

Paramètres

nomCampagne	
lieu	
nomTechnicien	
prenomTechnicien	
date	
parent	
duree	

Définition à la ligne 9 du fichier campagne.cpp.

```
00009
                                        : QObject (parent), nomCampagne(
      nomCampagne), nomTechnicien(nomTechnicien),
      prenomTechnicien(prenomTechnicien), lieu(
      lieu), date(date), duree(duree), estArchive(false),
      albumPhoto(), mesures(), nombrePhotos(0)
00010 {
00011     qDebug() << Q_FUNC_INFO;
00012 }
```

6.3.2.2 ~Campagne()

```
Campagne::~Campagne ( )
```

Destructeur de la classe Campagne.

Définition à la ligne 14 du fichier campagne.cpp.

```
00015 {
00016     qDebug() << Q_FUNC_INFO;
00017 }
```

6.3.3 Documentation des fonctions membres

6.3.3.1 ajouterMesure()

```
void Campagne::ajouterMesure (
            Mesure & mesure )
```

Ajoute une mesure dans le conteneur de mesure.

Paramètres

mesure	

Définition à la ligne 90 du fichier campagne.cpp.

Références mesures.

Référencé par Rov : :decoderTrameCapteur().

```
00091 {
00092     mesures.push_back(mesure);
00093 }
```

6.3.3.2 ajouterPhoto()

```
void Campagne::ajouterPhoto (
             Photo & photo )
```

Ajoute une photo dans l'album photo.

Paramètres

photo	

Définition à la ligne 80 du fichier campagne.cpp.

Références albumPhoto.

Référencé par IHMRov : :capturerImage(), et IHMAccueil : :chargerCampagnes().

```
00081 {
00082     albumPhoto.push_back(photo);
00083 }
```

6.3.3.3 getAlbumPhoto()

```
QVector< Photo > Campagne::getAlbumPhoto ( ) const
```

Retourne l'album photo de la campagne.

Renvoie

album photo de la campagne sous forme d'un QVector de photo

Définition à la ligne 70 du fichier campagne.cpp.

Références albumPhoto.

Référencé par IHMRov : :chargerPhotos(), IHMAccueil : :modifierCampagneBDD(), et IHMAlbumPhoto : :selectionnerPhoto().

```
00071 {
00072     return albumPhoto;
00073 }
```

6.3.3.4 getCheminSauvegarde()

```
QString Campagne::getCheminSauvegarde ( ) const
```

Retourne le chemin du dossier de sauvegarde des photos.

Renvoie

chemin du dossier de sauvegarde des photos sous forme d'un QString

Définition à la ligne 55 du fichier campagne.cpp.

Références cheminSauvegardePhotos.

Référencé par IHMAccueil : :ajouterCampagne(), IHMRov : :capturerImage(), et IHMAccueil : :enregistrerCampagneBDD().

```
00056 {
00057     return cheminSauvegardePhotos;
00058 }
```

6.3.3.5 getDate()

```
QDateTime Campagne::getDate ( ) const
```

Retourne la date de la campagne.

Renvoie

date de la campagne sous forme de QDateTime

Définition à la ligne 39 du fichier campagne.cpp.

Références date.

Référencé par IHMAccueil : :enregistrerCampagneBDD(), et IHMRov : :setCampagne().

```
00040 {
00041     return date;
00042 }
```

6.3.3.6 getDuree()

```
int Campagne::getDuree ( ) const
```

Retourne la durée de la campagne.

Renvoie

durée de la campagne sous forme d'un int

Définition à la ligne 44 du fichier campagne.cpp.

Références duree.

Référencé par IHMAccueil : :enregistrerCampagneBDD(), Rov : :getTempsCampagne(), et IHMAccueil : :modifierCampagneBDD().

```
00045 {
00046     return duree;
00047 }
```

6.3.3.7 getLieu()

```
QString Campagne::getLieu ( ) const
```

Retourne le lieu de la campagne.

Renvoie

lieu de la campagne sous forme de QString

Définition à la ligne 34 du fichier campagne.cpp.

Références lieu.

Référencé par IHMAccueil : :enregistrerCampagneBDD().

```
00035 {
00036     return lieu;
00037 }
```

6.3.3.8 getMesures()

```
QVector< Mesure > Campagne::getMesures ( ) const
```

Retourne les mesures de la campagne.

Renvoie

Mesures de la campagne sous forme d'un QVector de Mesure

Définition à la ligne 75 du fichier campagne.cpp.

Références mesures.

```
00076 {
00077     return mesures;
00078 }
```

6.3.3.9 getNomCampagne()

```
QString Campagne::getNomCampagne ( ) const
```

Retourne le nom de la campagne.

Renvoie

nom de la campagne sous forme de QString

Définition à la ligne 19 du fichier campagne.cpp.

Références nomCampagne.

Référencé par IHMAccueil : :ajouterCampagne(), IHMAccueil : :ajouterPhotoBDD(), IHMRov : :capturerImage(), IHMAccueil↩ : :enregistrerCampagneBDD(), IHMAccueil : :modifierCampagneBDD(), et IHMRov : :setCampagne().

```
00020 {
00021     return nomCampagne;
00022 }
```

6.3.3.10 getNomTechnicien()

```
QString Campagne::getNomTechnicien ( ) const
```

Retourne le nom du technicien.

Renvoie

nom du technicien sous forme de QString

Définition à la ligne 24 du fichier campagne.cpp.

Références nomTechnicien.

Référencé par IHMAccueil : :enregistrerCampagneBDD().

```
00025 {
00026     return nomTechnicien;
00027 }
```

6.3.3.11 getPrenomTechnicien()

```
QString Campagne::getPrenomTechnicien ( ) const
```

Retourne le prenom du technicien.

Renvoie

prenom du technicien sous forme de QString

Définition à la ligne 29 du fichier campagne.cpp.

Références prenomTechnicien.

Référencé par IHMAccueil : :enregistrerCampagneBDD().

```
00030 {
00031     return prenomTechnicien;
00032 }
```

6.3.3.12 incrementeNombrePhoto()

```
int Campagne::incrementeNombrePhoto ( )
```

Incrémente le nombre de photo prises durant la campagne et retourne son nombre.

Renvoie

nombre de photo prise durant la campagne sous forme d'un int

Définition à la ligne 105 du fichier campagne.cpp.

Références nombrePhotos.

Référencé par IHMRov : :capturerImage().

```
00106 {
00107     nombrePhotos++;
00108     return nombrePhotos;
00109 }
```

6.3.3.13 modifierArchivePhoto()

```
void Campagne::modifierArchivePhoto (
            int numeroPhoto )
```

Modifie l'état d'archive de la photo correspondant au numéro passé en paramètre.

Paramètres

numeroPhoto	

Définition à la ligne 85 du fichier campagne.cpp.

Références albumPhoto.

Référencé par IHMAlbumPhoto : :selectionnerPhoto().

```
00086 {
00087     albumPhoto[numeroPhoto].aGarder = !(albumPhoto[numeroPhoto].aGarder);
00088 }
```

6.3.3.14 setCheminSauvegarde()

```
void Campagne::setCheminSauvegarde (
            QString chemin )
```

Modifie le chemin de sauvegarde des photos.

Paramètres

chemin	

Définition à la ligne 60 du fichier campagne.cpp.

Références cheminSauvegardePhotos.

Référencé par IHMAccueil : :chargerCampagnes(), et IHMCreationCampagne : :validerCampagne().

```
00061 {
00062     cheminSauvegardePhotos = chemin;
00063 }
```

6.3.3.15 setDuree()

```
void Campagne::setDuree (
            int duree )
```

Modifie la duree de la campagne.

Paramètres

duree	

Définition à la ligne 49 du fichier campagne.cpp.

Références duree.

Référencé par Rov : :arreterCampagne().

```
00050 {
00051     this->duree = this->duree + duree;
00052 }
```

6.3.3.16 setNombrePhotos()

```
void Campagne::setNombrePhotos (
            int nombre )
```

Modifie le nombre de photos prises durant la campagne.

Paramètres

nombre	

Définition à la ligne 65 du fichier campagne.cpp.

Références nombrePhotos.

Référencé par IHMAccueil : :chargerCampagnes().

```
00066 {
00067     nombrePhotos = nombre;
00068 }
```

6.3.3.17 supprimerMesures()

```
void Campagne::supprimerMesures ( )
```

Supprime les mesure du conteneur de Mesure, une fois celles-ci archivés dans la BDD.

Définition à la ligne 95 du fichier campagne.cpp.

Références mesures.

Référencé par IHMAccueil : :modifierCampagneBDD().

```
00096 {
00097     mesures.clear();
00098 }
```

6.3.3.18 supprimerPhotos()

```
void Campagne::supprimerPhotos ( )
```

Supprime les photo du conteneur de Photo, une fois celles-ci archivés dans la BDD.

Définition à la ligne 100 du fichier campagne.cpp.

Références albumPhoto.

```
00101 {
00102     albumPhoto.clear();
00103 }
```

6.3.4 Documentation des données membres

6.3.4.1 albumPhoto

```
QVector<Photo> Campagne::albumPhoto  [private]
```

Conteneur des photos prises durant la campagne.

Définition à la ligne 46 du fichier campagne.h.

Référencé par ajouterPhoto(), getAlbumPhoto(), modifierArchivePhoto(), et supprimerPhotos().

6.3.4.2 cheminSauvegardePhotos

```
QString Campagne::cheminSauvegardePhotos  [private]
```

Attribut contenant le chemin de sauvegarde des photos.

Définition à la ligne 43 du fichier campagne.h.

Référencé par getCheminSauvegarde(), et setCheminSauvegarde().

6.3.4.3 date

```
QDateTime Campagne::date  [private]
```

Attribut contenant la date de la campagne.

Définition à la ligne 42 du fichier campagne.h.

Référencé par getDate().

6.3.4.4 duree

```
int Campagne::duree  [private]
```

Attribut contenant la durée de la campagne en millisecondes.

Définition à la ligne 44 du fichier campagne.h.

Référencé par getDuree(), et setDuree().

6.3.4.5 estArchive

```
bool Campagne::estArchive  [private]
```

Attribut booléen afin de savoir si la campagne est toujours en cours.

Définition à la ligne 45 du fichier campagne.h.

6.3.4.6 lieu

```
QString Campagne::lieu  [private]
```

Attribut contenant le lieu de la campagne.

Définition à la ligne 41 du fichier campagne.h.

Référencé par getLieu().

6.3.4.7 mesures

```
QVector<Mesure> Campagne::mesures  [private]
```

Conteneur des mesures enregistrés durant la campagne.

Définition à la ligne 47 du fichier campagne.h.

Référencé par ajouterMesure(), getMesures(), et supprimerMesures().

6.3.4.8 nombrePhotos

```
int Campagne::nombrePhotos  [private]
```

Attribut contenant le nombre de photos prise durant la campagne.

Définition à la ligne 48 du fichier campagne.h.

Référencé par incrementeNombrePhoto(), et setNombrePhotos().

6.3.4.9 nomCampagne

```
QString Campagne::nomCampagne  [private]
```

Attribut contenant le nom de la campagne.

Définition à la ligne 38 du fichier campagne.h.

Référencé par getNomCampagne().

6.3.4.10 nomTechnicien

```
QString Campagne::nomTechnicien  [private]
```

Attribut contenant le nom du technicien.

Définition à la ligne 39 du fichier campagne.h.

Référencé par getNomTechnicien().

6.3.4.11 prenomTechnicien

```
QString Campagne::prenomTechnicien  [private]
```

Attribut contenant le nom du technicien.

Définition à la ligne 40 du fichier campagne.h.

Référencé par getPrenomTechnicien().

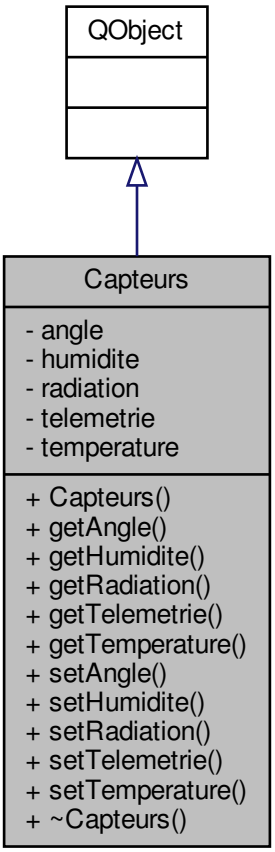
La documentation de cette classe a été générée à partir des fichiers suivants :
- campagne.h
- campagne.cpp

6.4 Référence de la classe Capteurs

Classe contenant les dernières informations issues des capteurs du rov.

```
#include "capteurs.h"
```

Graphe de collaboration de Capteurs :



Fonctions membres publiques

- Capteurs (QObject *parent=nullptr)
  Constructeur de la classe Capteurs.
- QString getAngle () const
  Récupère la dernière information issue du capteur de distance.
- QString getHumidite () const
  Récupère la dernière information issue du capteur de humidité
- QString getRadiation () const
  Récupère la dernière information issue du capteur de radiation.
- QString getTelemetrie () const
  Récupère la dernière information issue du capteur télémétrique.
- QString getTemperature () const
  Récupère la dernière information issue du capteur de température.
- void setAngle (QString angle)
  Modifie la dernière information issue du capteur de distance.
- void setHumidite (QString humidite)
  Modifie la dernière information issue du capteur de humidité
- void setRadiation (QString radiation)
  Modifie la dernière information issue du capteur de radiation.
- void setTelemetrie (QString telemetrie)
  Modifie la dernière information issue du capteur télémétrique.
- void setTemperature (QString temperature)
  Modifie la dernière information issue du capteur de température.
- ~Capteurs ()
  Destructeur de la classe Capteurs.

Attributs privés

- QString angle
  Dernière données capteurs de distance.
- QString humidite
  Dernière données de température.
- QString radiation
  Dernière données de radiation.
- QString telemetrie
  Dernière données télémétriques.
- QString temperature
  Dernière données de température.

6.4.1 Description détaillée

Classe contenant les dernières informations issues des capteurs du rov.

Définition à la ligne 18 du fichier capteurs.h.

6.4.2 Documentation des constructeurs et destructeur

6.4.2.1 Capteurs()

```
Capteurs::Capteurs (
            QObject * parent = nullptr )
```

Constructeur de la classe Capteurs.

Paramètres

parent	

Définition à la ligne 9 du fichier capteurs.cpp.

```
00009                                   : QObject(parent), telemetrie("--,--"),
      angle(""), temperature("--,--"), humidite("--,--"),
      radiation("--,--")
00010 {
00011     qDebug() << Q_FUNC_INFO;
00012 }
```

6.4.2.2 ~Capteurs()

```
Capteurs::~Capteurs ( )
```

Destructeur de la classe Capteurs.

Définition à la ligne 14 du fichier capteurs.cpp.

```
00015 {
00016     qDebug() << Q_FUNC_INFO;
00017 }
```

6.4.3 Documentation des fonctions membres

6.4.3.1 getAngle()

```
QString Capteurs::getAngle ( ) const
```

Récupère la dernière information issue du capteur de distance.

Renvoie

la dernière information issue du capteur de distance

Définition à la ligne 44 du fichier capteurs.cpp.

Références angle.

Référencé par IHMRov : :actualiserInformations(), IHMRov : :calculCoordonneesX(), et IHMRov : :calculCoordonneesY().

```
00045 {
00046     return angle;
00047 }
```

6.4.3.2 getHumidite()

```
QString Capteurs::getHumidite ( ) const
```

Récupère la dernière information issue du capteur de humidité

Renvoie

la dernière information issue du capteur de humidité

Définition à la ligne 59 du fichier capteurs.cpp.

Références humidite.

Référencé par IHMRov : :actualiserInformations().

```
00060 {
00061     return humidite;
00062 }
```

6.4.3.3 getRadiation()

```
QString Capteurs::getRadiation ( ) const
```

Récupère la dernière information issue du capteur de radiation.

Renvoie

la dernière information issue du capteur de radiation

Définition à la ligne 64 du fichier capteurs.cpp.

Références radiation.

Référencé par IHMRov : :actualiserInformations(), et IHMRov : :actualiserInformationsSeuils().

```
00065 {
00066     return radiation;
00067 }
```

6.4.3.4 getTelemetrie()

```
QString Capteurs::getTelemetrie ( ) const
```

Récupère la dernière information issue du capteur télémétrique.

Renvoie

la dernière information issue du capteur télémétrique

Définition à la ligne 49 du fichier capteurs.cpp.

Références telemetrie.

Référencé par IHMRov : :actualiserInformations(), IHMRov : :calculCoordonneesX(), et IHMRov : :calculCoordonneesY().

```
00050 {
00051     return telemetrie;
00052 }
```

6.4.3.5 getTemperature()

```
QString Capteurs::getTemperature ( ) const
```

Récupère la dernière information issue du capteur de température.

Renvoie

la dernière information issue du capteur de température

Définition à la ligne 54 du fichier capteurs.cpp.

Références temperature.

Référencé par IHMRov : :actualiserInformations(), et IHMRov : :actualiserInformationsSeuils().

```
00055 {
00056     return temperature;
00057 }
```

6.4.3.6 setAngle()

```
void Capteurs::setAngle (
            QString angle )
```

Modifie la dernière information issue du capteur de distance.

Paramètres

angle	

Définition à la ligne 39 du fichier capteurs.cpp.

Références angle.

Référencé par Rov : :decoderTrameTelemetrie().

```
00040 {
00041     this->angle = angle;
00042 }
```

6.4.3.7 setHumidite()

```
void Capteurs::setHumidite (
            QString humidite )
```

Modifie la dernière information issue du capteur de humidité

Paramètres

humidite	

Définition à la ligne 29 du fichier capteurs.cpp.

Références humidite.

Référencé par Rov : :decoderTrameCapteur().

```
00030 {
00031     this->humidite = humidite;
00032 }
```

6.4.3.8 setRadiation()

```
void Capteurs::setRadiation (
            QString radiation )
```

Modifie la dernière information issue du capteur de radiation.

Paramètres

radiation	

Définition à la ligne 34 du fichier capteurs.cpp.

Références radiation.

Référencé par Rov : :decoderTrameCapteur().

```
00035 {
00036     this->radiation = radiation;
00037 }
```

6.4.3.9 setTelemetrie()

```
void Capteurs::setTelemetrie (
            QString telemetrie )
```

Modifie la dernière information issue du capteur télémétrique.

Paramètres

telemetrie	

Définition à la ligne 19 du fichier capteurs.cpp.

Références telemetrie.

Référencé par Rov : :decoderTrameTelemetrie().

```
00020 {
00021     this->telemetrie = telemetrie;
00022 }
```

6.4.3.10 setTemperature()

```
void Capteurs::setTemperature (
            QString temperature )
```

Modifie la dernière information issue du capteur de température.

Paramètres

temperature	

Définition à la ligne 24 du fichier capteurs.cpp.

Références temperature.

Référencé par Rov : :decoderTrameCapteur().

```
00025 {
00026     this->temperature = temperature;
00027 }
```

6.4.4 Documentation des données membres

6.4.4.1 angle

```
QString Capteurs::angle  [private]
```

Dernière données capteurs de distance.

Définition à la ligne 23 du fichier capteurs.h.

Référencé par getAngle(), et setAngle().

6.4.4.2 humidite

```
QString Capteurs::humidite  [private]
```

Dernière données de température.

Définition à la ligne 25 du fichier capteurs.h.

Référencé par getHumidite(), et setHumidite().

6.4.4.3 radiation

```
QString Capteurs::radiation  [private]
```

Dernière données de radiation.

Définition à la ligne 26 du fichier capteurs.h.

Référencé par getRadiation(), et setRadiation().

6.4.4.4 telemetrie

```
QString Capteurs::telemetrie  [private]
```

Dernière données télémétriques.

Définition à la ligne 22 du fichier capteurs.h.

Référencé par getTelemetrie(), et setTelemetrie().

6.4.4.5 temperature

```
QString Capteurs::temperature  [private]
```

Dernière données de température.

Définition à la ligne 24 du fichier capteurs.h.

Référencé par getTemperature(), et setTemperature().

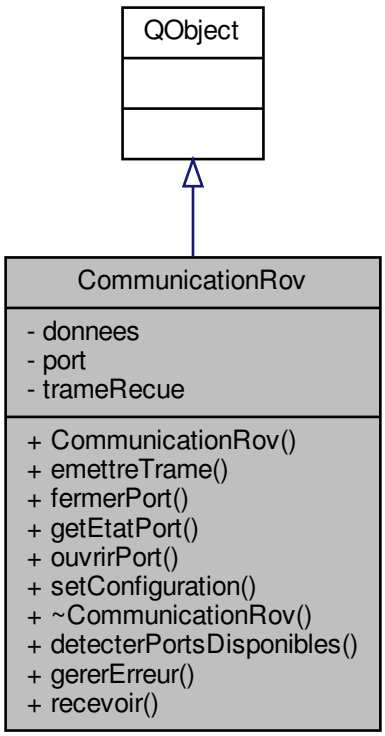
La documentation de cette classe a été générée à partir des fichiers suivants :
- capteurs.h
- capteurs.cpp

6.5 Référence de la classe CommunicationRov

Class permettant de mettre en place une communication avec le rov.

```
#include "communicationRov.h"
```

Graphe de collaboration de CommunicationRov :



Connecteurs publics

- void gererErreur (QSerialPort : :SerialPortError)
- void recevoir ()
  Récupere la trame disponible sur le port à la réception du signal ReadyRead et émet un signal nouvelleTrame.

Signaux

- void etatPortModifie (bool etat, QString information)
  Envoie un signal informant que l'état du port a été modifié
- void nouvelleTrame (QString trame)
  Envoie un signal informant qu'une nouvelle trame est disponible.

Fonctions membres publiques

- CommunicationRov (QObject *parent=nullptr)
  constructeur de la classe CommunicationRov
- int emettreTrame (QString trame)
  Emet la trame vers le robot.
- void fermerPort ()
  Permet de fermer le port série virtuel.
- bool getEtatPort ()
  retourne l'etat du port série
- bool ouvrirPort ()
  Permet d'ouvrir le port série virtuel.
- void setConfiguration (Configuration maConfiguration)
  Affecte a l'objet CommunicationRov une configuration du port série virtuel.
- ~CommunicationRov ()
  Destructeur de la classe CommunicationRov.

Fonctions membres publiques statiques

- static QStringList detecterPortsDisponibles ()
  retourne la liste des ports disponibles

Attributs privés

- QByteArray donnees
  Tableau contenant les données bruts envoyé depuis la liason série.
- QSerialPort * port
  accède a la configuration de la liaison série
- QString trameRecue
  Derniere trameRecue.

6.5.1 Description détaillée

Class permettant de mettre en place une communication avec le rov.

Définition à la ligne 38 du fichier communicationrov.h.

6.5.2 Documentation des constructeurs et destructeur

6.5.2.1 CommunicationRov()

```
CommunicationRov::CommunicationRov (
            QObject * parent = nullptr )
```

constructeur de la classe CommunicationRov

Paramètres

parent	

Définition à la ligne 9 du fichier communicationrov.cpp.

Références gererErreur(), et port.

```
00009                                                 : QObject(parent)
00010 {
00011     qDebug() << Q_FUNC_INFO;
00012     port = new QSerialPort(this);
00013
00014     connect(port, SIGNAL(errorOccurred(QSerialPort::SerialPortError)), this, SLOT(
      gererErreur(QSerialPort::SerialPortError)));
00015 }
```

6.5.2.2 ~CommunicationRov()

```
CommunicationRov::~CommunicationRov ( )
```

Destructeur de la classe CommunicationRov.

Définition à la ligne 17 du fichier communicationrov.cpp.

Références port.

```
00018 {
00019     if(port->isOpen())
00020         port->close();
00021     qDebug() << Q_FUNC_INFO;
00022 }
```

6.5.3 Documentation des fonctions membres

6.5.3.1 detecterPortsDisponibles()

```
QStringList CommunicationRov::detecterPortsDisponibles ( )  [static]
```

retourne la liste des ports disponibles

Renvoie

la liste des ports disponibles

Définition à la ligne 98 du fichier communicationrov.cpp.

Référencé par IHMConfiguration : :actualisePortsDisponibles(), et IHMConfiguration : :configurerWidgets().

```
00099 {
00100     QStringList listePortsDetectes;
00101
00102     foreach(const QSerialPortInfo &info, QSerialPortInfo::availablePorts())
00103     {
00104         listePortsDetectes.push_back(info.portName());
00105     }
00106
00107     return listePortsDetectes;
00108 }
```

6.5.3.2 emettreTrame()

```
int CommunicationRov::emettreTrame (
            QString trame )
```

Emet la trame vers le robot.

Paramètres

trame	

Renvoie

un entier correspondant au nombres d'octets envoyé, retourne -1 si la diffusion à échoué

Définition à la ligne 68 du fichier communicationrov.cpp.

Références port.

Référencé par Rov : :creerTrameCamera(), Rov : :creerTrameDeplacement(), Rov : :creerTrameOrdre(), Rov : :creerTramePilotage(), et Rov : :creerTramePince().

```
00069 {
00070     int nombresOctets = -1;
00071
00072     if (port == NULL || !port->isOpen())
00073     {
00074        return -1;
00075     }
00076
00077     nombresOctets = port->write(trame.toLatin1());
00078     return nombresOctets;
00079 }
```

6.5.3.3 etatPortModifie

```
void CommunicationRov::etatPortModifie (
             bool etat,
             QString information )  [signal]
```

Envoie un signal informant que l'état du port a été modifié

Paramètres

etat	
information	

Référencé par fermerPort(), et ouvrirPort().

6.5.3.4 fermerPort()

```
void CommunicationRov::fermerPort ( )
```

Permet de fermer le port série virtuel.

Définition à la ligne 50 du fichier communicationrov.cpp.

Références etatPortModifie(), port, et recevoir().

Référencé par gererErreur(), et IHMConfiguration : :modifierEtatPort().

```
00051 {
00052     port->close();
00053     disconnect(port, SIGNAL(readyRead()), this, SLOT(recevoir()));
00054     qDebug() << Q_FUNC_INFO << "Port fermé" << !port->isOpen();
00055     emit etatPortModifie(false, port->portName());
00056 }
```

6.5.3.5 gererErreur

```
void CommunicationRov::gererErreur (
            QSerialPort::SerialPortError erreur )  [slot]
```

Définition à la ligne 115 du fichier communicationrov.cpp.

Références fermerPort().

Référencé par CommunicationRov().

```
00116 {
00117     switch(erreur)
00118     {
00119         case QSerialPort::ResourceError : fermerPort();
00120             break;
00121     }
00122 }
```

6.5.3.6 getEtatPort()

```
bool CommunicationRov::getEtatPort ( )
```

retourne l'etat du port série

Renvoie

l'etat du port série

Définition à la ligne 110 du fichier communicationrov.cpp.

Références port.

Référencé par IHMConfiguration : :modifieEtatBoutons().

```
00111 {
00112     return port->isOpen();
00113 }
```

6.5.3.7 nouvelleTrame

```
void CommunicationRov::nouvelleTrame (
            QString trame )  [signal]
```

Envoie un signal informant qu'une nouvelle trame est disponible.

Paramètres

trame	

Référencé par recevoir().

6.5.3.8 ouvrirPort()

```
bool CommunicationRov::ouvrirPort ( )
```

Permet d'ouvrir le port série virtuel.

Définition à la ligne 24 du fichier communicationrov.cpp.

Références etatPortModifie(), port, et recevoir().

Référencé par IHMConfiguration : :IHMConfiguration(), et IHMConfiguration : :modifierEtatPort().

```
00025 {
00026     if(port->open(QIODevice::ReadWrite))
00027     {
00028         qDebug() << Q_FUNC_INFO << "Port ouvert" << port->isOpen();
00029         if(port->isOpen())
00030         {
00031             emit etatPortModifie(true, port->portName());
00032             connect(port, SIGNAL(readyRead()), this, SLOT(recevoir()));
00033             return true;
00034         }
00035         else
00036             return false;
00037     }
00038     #ifdef MODE_CONNECTE
00039     else
00040     {
00041         if(!port->isOpen())
00042         {
00043             QMessageBox::critical(nullptr, "Communication rov", QString::fromUtf8("Erreur ouverture du port
      série !"));
00044             return false;
00045         }
00046     }
00047     #endif
00048 }
```

6.5.3.9 recevoir

```
void CommunicationRov::recevoir ( )  [slot]
```

Récupere la trame disponible sur le port à la réception du signal ReadyRead et émet un signal nouvelleTrame.

Définition à la ligne 81 du fichier communicationrov.cpp.

Références donnees, nouvelleTrame(), port, et trameRecue.

Référencé par fermerPort(), et ouvrirPort().

```
00082 {
00083     while(port->bytesAvailable())
00084     {
00085         donnees += port->readAll();
00086         //qDebug() << Q_FUNC_INFO << "bytesAvailable" << port->bytesAvailable() << "donnees" << donnees;
00087     }
00088
00089     if(donnees.startsWith("$") && donnees.endsWith("\r\n"))
00090     {
00091         trameRecue = QString(donnees.data());
00092         //qDebug() << Q_FUNC_INFO << "trameRecue" << trameRecue;
00093         emit nouvelleTrame(trameRecue);
00094         donnees.clear();
00095     }
00096 }
```

6.5.3.10 setConfiguration()

```
void CommunicationRov::setConfiguration (
            Configuration maConfiguration )
```

Affecte a l'objet CommunicationRov une configuration du port série virtuel.

Paramètres

maConfiguration	

Définition à la ligne 58 du fichier communicationrov.cpp.

Références Configuration : :bitsDonnees, Configuration : :bitStop, Configuration : :debit, Configuration : :port, et port.

Référencé par Rov : :modifierConfiguration().

```
00059 {
00060     port->setPortName(maConfiguration.port);
00061     port->setBaudRate(maConfiguration.debit);
00062     port->setDataBits((QSerialPort::DataBits)maConfiguration.bitsDonnees);
00063     port->setStopBits((QSerialPort::StopBits)maConfiguration.bitStop);
00064     port->setParity(QSerialPort::NoParity);
00065     port->setFlowControl(QSerialPort::NoFlowControl);
00066 }
```

6.5.4 Documentation des données membres

6.5.4.1 donnees

```
QByteArray CommunicationRov::donnees  [private]
```

Tableau contenant les données bruts envoyé depuis la liason série.

Définition à la ligne 43 du fichier communicationrov.h.

Référencé par recevoir().

6.5.4.2 port

```
QSerialPort* CommunicationRov::port  [private]
```

accède a la configuration de la liaison série

Définition à la ligne 42 du fichier communicationrov.h.

Référencé par CommunicationRov(), emettreTrame(), fermerPort(), getEtatPort(), ouvrirPort(), recevoir(), setConfiguration(), et ~CommunicationRov().

6.5.4.3 trameRecue

```
QString CommunicationRov::trameRecue  [private]
```

Derniere trameRecue.

Définition à la ligne 44 du fichier communicationrov.h.

Référencé par recevoir().

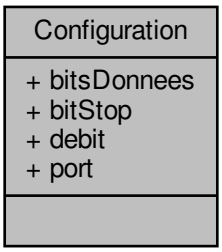
La documentation de cette classe a été générée à partir des fichiers suivants :
- communicationrov.h
- communicationrov.cpp

6.6 Référence de la structure Configuration

structure permettant de configurer une communication

`#include <communicationrov.h>`

Graphe de collaboration de Configuration :



Attributs publics

- int bitsDonnees
  Attribut définissant le nombre de bits de données de la communication.
- int bitStop
  Attribut définissant le nombre de bits de stop de la communication.
- int debit
  Attribut définissant la vitesse en bits/s de la communication.
- QString port
  Attribut définissant le nom d'un port.

6.6.1 Description détaillée

structure permettant de configurer une communication

Définition à la ligne 24 du fichier communicationrov.h.

6.6.2 Documentation des données membres

6.6.2.1 bitsDonnees

`int Configuration::bitsDonnees`

Attribut définissant le nombre de bits de données de la communication.

Définition à la ligne 28 du fichier communicationrov.h.

Référencé par IHMConfiguration : :modifierConfiguration(), et CommunicationRov : :setConfiguration().

6.6.2.2 bitStop

`int Configuration::bitStop`

Attribut définissant le nombre de bits de stop de la communication.

Définition à la ligne 29 du fichier communicationrov.h.

Référencé par IHMConfiguration : :modifierConfiguration(), et CommunicationRov : :setConfiguration().

6.6.2.3 debit

`int Configuration::debit`

Attribut définissant la vitesse en bits/s de la communication.

Définition à la ligne 27 du fichier communicationrov.h.

Référencé par IHMConfiguration : :modifierConfiguration(), et CommunicationRov : :setConfiguration().

6.6.2.4 port

`QString Configuration::port`

Attribut définissant le nom d'un port.

Définition à la ligne 26 du fichier communicationrov.h.

Référencé par IHMConfiguration : :modifierConfiguration(), et CommunicationRov : :setConfiguration().

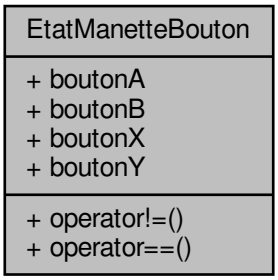
La documentation de cette structure a été générée à partir du fichier suivant :
— communicationrov.h

6.7 Référence de la structure EtatManetteBouton

Structure qui définit l'état de la manette en mode pilotage de la pince.

`#include <manette.h>`

Graphe de collaboration de EtatManetteBouton :



Fonctions membres publiques

- bool operator != (const EtatManetteBouton &maStructure) const
  Surcharge de l'opérateur != afin de comparer un état de manette prédéfini avec l'état actuel de la manette.
- bool operator== (const EtatManetteBouton &maStructure) const
  Surcharge de l'opérateur == afin de comparer un état de manette prédéfini avec l'état actuel de la manette.

Attributs publics

- bool boutonA
  Membre définissant l'état du bouton A.
- bool boutonB
  Membre définissant l'état du bouton B.
- bool boutonX
  Membre définissant l'état du bouton X.
- bool boutonY
  Membre définissant l'état du bouton Y.

6.7.1 Description détaillée

Structure qui définit l'état de la manette en mode pilotage de la pince.

Définition à la ligne 122 du fichier manette.h.

6.7.2 Documentation des fonctions membres

6.7.2.1 operator" !=()

```
bool EtatManetteBouton::operator!= (
            const EtatManetteBouton & maStructure ) const
```

Surcharge de l'opérateur != afin de comparer un état de manette prédéfini avec l'état actuel de la manette.

Paramètres

maStructure	

Renvoie

Si l'opération est possible ou pas

Définition à la ligne 608 du fichier manette.cpp.

```
00609 {
00610     return !(*this == maStructure);
00611 }
```

6.7.2.2 operator==()

```
bool EtatManetteBouton::operator== (
            const EtatManetteBouton & maStructure ) const
```

Surcharge de l'opérateur == afin de comparer un état de manette prédéfini avec l'état actuel de la manette.

Paramètres

maStructure	

Renvoie

Si l'opération est possible ou pas

Définition à la ligne 594 du fichier manette.cpp.

Références boutonA, boutonB, boutonX, et boutonY.

```
00595 {
00596     if(this->boutonA != maStructure.boutonA)
00597         return false;
00598     else if(this->boutonB != maStructure.boutonB)
00599         return false;
00600     else if(this->boutonX != maStructure.boutonX)
00601         return false;
00602     else if(this->boutonY != maStructure.boutonY)
00603         return false;
00604     else
00605         return true;
00606 }
```

6.7.3 Documentation des données membres

6.7.3.1 boutonA

```
bool EtatManetteBouton::boutonA
```

Membre définissant l'état du bouton A.

Définition à la ligne 124 du fichier manette.h.

Référencé par Manette : :changerBoutonA(), Manette : :initialiserEtatBouton(), et operator==().

6.7.3.2 boutonB

```
bool EtatManetteBouton::boutonB
```

Membre définissant l'état du bouton B.

Définition à la ligne 125 du fichier manette.h.

Référencé par Manette : :changerBoutonB(), Manette : :initialiserEtatBouton(), et operator==().

6.7.3.3 boutonX

```
bool EtatManetteBouton::boutonX
```

Membre définissant l'état du bouton X.

Définition à la ligne 126 du fichier manette.h.

Référencé par Manette : :changerBoutonX(), Manette : :initialiserEtatBouton(), et operator==().

6.7.3.4 boutonY

```
bool EtatManetteBouton::boutonY
```

Membre définissant l'état du bouton Y.

Définition à la ligne 127 du fichier manette.h.

Référencé par Manette : :changerBoutonY(), Manette : :initialiserEtatBouton(), et operator==().

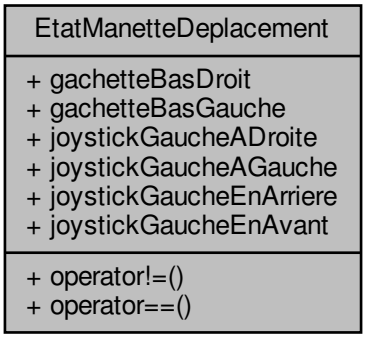
La documentation de cette structure a été générée à partir des fichiers suivants :
- manette.h
- manette.cpp

6.8 Référence de la structure EtatManetteDeplacement

Structure qui définit l'état de la manette en mode déplacement du robot.

```
#include <manette.h>
```

Graphe de collaboration de EtatManetteDeplacement :



Fonctions membres publiques

- bool operator != (const EtatManetteDeplacement &maStructure) const
  Surcharge de l'opérateur != afin de comparer un état de manette prédéfini avec l'état actuel de la manette.
- bool operator== (const EtatManetteDeplacement &maStructure) const
  Surcharge de l'opérateur == afin de comparer un état de manette prédéfini avec l'état actuel de la manette.

Attributs publics

- bool gachetteBasDroit
  Membre définissant l'état du bouton R2.
- bool gachetteBasGauche
  Membre définissant l'état du bouton L2.
- bool joystickGaucheADroite
  Membre définissant l'état du joystick sur l'axe X.
- bool joystickGaucheAGauche
  Membre définissant l'état du joystick sur l'axe X.
- bool joystickGaucheEnArriere
  Membre définissant l'état du joystick sur l'axe Y.
- bool joystickGaucheEnAvant
  Membre définissant l'état du joystick sur l'axe Y.

6.8.1 Description détaillée

Structure qui définit l'état de la manette en mode déplacement du robot.

Définition à la ligne 178 du fichier manette.h.

6.8.2 Documentation des fonctions membres

6.8.2.1 operator"!=()

```
bool EtatManetteDeplacement::operator!= (
            const EtatManetteDeplacement & maStructure ) const
```

Surcharge de l'opérateur != afin de comparer un état de manette prédéfini avec l'état actuel de la manette.

Paramètres

maStructure	

Renvoie

Si l'opération est possible ou pas

Définition à la ligne 654 du fichier manette.cpp.

```
00655 {
00656     return !(*this == maStructure);
00657 }
```

6.8.2.2 operator==()

```
bool EtatManetteDeplacement::operator== (
            const EtatManetteDeplacement & maStructure ) const
```

Surcharge de l'opérateur == afin de comparer un état de manette prédéfini avec l'état actuel de la manette.

Paramètres

maStructure	

Renvoie

Si l'opération est possible ou pas

Définition à la ligne 636 du fichier manette.cpp.

Références gachetteBasDroit, gachetteBasGauche, joystickGaucheADroite, joystickGaucheAGauche, joystickGaucheEnArriere, et joystickGaucheEnAvant.

```
00637 {
00638     if(this->gachetteBasDroit != maStructure.gachetteBasDroit)
00639         return false;
00640     else if(this->gachetteBasGauche != maStructure.
      gachetteBasGauche)
00641         return false;
00642     else if(this->joystickGaucheADroite != maStructure.
      joystickGaucheADroite)
00643         return false;
00644     else if(this->joystickGaucheAGauche != maStructure.
      joystickGaucheAGauche)
00645         return false;
00646     else if(this->joystickGaucheEnArriere != maStructure.
      joystickGaucheEnArriere)
00647         return false;
00648     else if(this->joystickGaucheEnAvant != maStructure.
      joystickGaucheEnAvant)
00649         return false;
00650     else
00651         return true;
00652 }
```

6.8.3 Documentation des données membres

6.8.3.1 gachetteBasDroit

```
bool EtatManetteDeplacement::gachetteBasDroit
```

Membre définissant l'état du bouton R2.

Définition à la ligne 185 du fichier manette.h.

Référencé par Manette : :changerGachetteBasDroit(), Manette : :initialisationStructureRotationADroite(), Manette : :initialisation↩ StructureRotationAGauche(), Manette : :initialisationStructuresEnArriere(), Manette : :initialisationStructuresEnAvant(), Manette↩ : :initialisationStructuresRotationADroiteDoucement(), Manette : :initialisationStructuresRotationAGaucheDoucement(), Manette↩ : :initialisationStructuresVirageArriereADroiteDoucement(), Manette : :initialisationStructuresVirageArriereAGaucheDoucement(), Manette : :initialisationStructuresVirageAvantADroiteDoucement(), Manette : :initialisationStructuresVirageAvantAGauche↩ Doucement(), Manette : :initialisationStructureVirageArriereADroite(), Manette : :initialisationStructureVirageArriereAGauche(), Manette : :initialisationStructureVirageAvantADroite(), Manette : :initialisationStructureVirageAvantAGauche(), Manette : :initialiser↩ Etats(), et operator==().

6.8.3.2 gachetteBasGauche

```
bool EtatManetteDeplacement::gachetteBasGauche
```

Membre définissant l'état du bouton L2.

Définition à la ligne 184 du fichier manette.h.

Référencé par Manette : :changerGachetteBasGauche(), Manette : :initialisationStructureRotationADroite(), Manette : :initialisation↩ StructureRotationAGauche(), Manette : :initialisationStructuresEnArriere(), Manette : :initialisationStructuresEnAvant(), Manette↩ : :initialisationStructuresRotationADroiteDoucement(), Manette : :initialisationStructuresRotationAGaucheDoucement(), Manette↩ : :initialisationStructuresVirageArriereADroiteDoucement(), Manette : :initialisationStructuresVirageArriereAGaucheDoucement(), Manette : :initialisationStructuresVirageAvantADroiteDoucement(), Manette : :initialisationStructuresVirageAvantAGauche↩ Doucement(), Manette : :initialisationStructureVirageArriereADroite(), Manette : :initialisationStructureVirageArriereAGauche(), Manette : :initialisationStructureVirageAvantADroite(), Manette : :initialisationStructureVirageAvantAGauche(), Manette : :initialiser↩ Etats(), et operator==().

6.8.3.3 joystickGaucheADroite

```
bool EtatManetteDeplacement::joystickGaucheADroite
```

Membre définissant l'état du joystick sur l'axe X.

Définition à la ligne 183 du fichier manette.h.

Référencé par Manette : :changerAxeXJoystickGauche(), Manette : :initialisationStructureRotationADroite(), Manette : :initialisation↩ StructureRotationAGauche(), Manette : :initialisationStructuresEnArriere(), Manette : :initialisationStructuresEnAvant(), Manette↩ : :initialisationStructuresRotationADroiteDoucement(), Manette : :initialisationStructuresRotationAGaucheDoucement(), Manette↩ : :initialisationStructuresVirageArriereADroiteDoucement(), Manette : :initialisationStructuresVirageArriereAGaucheDoucement(), Manette : :initialisationStructuresVirageAvantADroiteDoucement(), Manette : :initialisationStructuresVirageAvantAGauche↩ Doucement(), Manette : :initialisationStructureVirageArriereADroite(), Manette : :initialisationStructureVirageArriereAGauche(), Manette : :initialisationStructureVirageAvantADroite(), Manette : :initialisationStructureVirageAvantAGauche(), Manette : :initialiser↩ Etats(), et operator==().

6.8.3.4 joystickGaucheAGauche

```
bool EtatManetteDeplacement::joystickGaucheAGauche
```

Membre définissant l'état du joystick sur l'axe X.

Définition à la ligne 182 du fichier manette.h.

Référencé par Manette : :changerAxeXJoystickGauche(), Manette : :initialisationStructureRotationADroite(), Manette : :initialisation↩ StructureRotationAGauche(), Manette : :initialisationStructuresEnArriere(), Manette : :initialisationStructuresEnAvant(), Manette↩ : :initialisationStructuresRotationADroiteDoucement(), Manette : :initialisationStructuresRotationAGaucheDoucement(), Manette↩ : :initialisationStructuresVirageArriereADroiteDoucement(), Manette : :initialisationStructuresVirageArriereAGaucheDoucement(), Manette : :initialisationStructuresVirageAvantADroiteDoucement(), Manette : :initialisationStructuresVirageAvantAGauche↩ Doucement(), Manette : :initialisationStructureVirageArriereADroite(), Manette : :initialisationStructureVirageArriereAGauche(), Manette : :initialisationStructureVirageAvantADroite(), Manette : :initialisationStructureVirageAvantAGauche(), Manette : :initialiser↩ Etats(), et operator==().

6.8.3.5 joystickGaucheEnArriere

```
bool EtatManetteDeplacement::joystickGaucheEnArriere
```

Membre définissant l'état du joystick sur l'axe Y.

Définition à la ligne 181 du fichier manette.h.

Référencé par Manette : :changerAxeYJoystickGauche(), Manette : :initialisationStructureRotationADroite(), Manette : :initialisation↩ StructureRotationAGauche(), Manette : :initialisationStructuresEnArriere(), Manette : :initialisationStructuresEnAvant(), Manette↩ : :initialisationStructuresRotationADroiteDoucement(), Manette : :initialisationStructuresRotationAGaucheDoucement(), Manette↩ : :initialisationStructuresVirageArriereADroiteDoucement(), Manette : :initialisationStructuresVirageArriereAGaucheDoucement(), Manette : :initialisationStructuresVirageAvantADroiteDoucement(), Manette : :initialisationStructuresVirageAvantAGauche↩ Doucement(), Manette : :initialisationStructureVirageArriereADroite(), Manette : :initialisationStructureVirageArriereAGauche(), Manette : :initialisationStructureVirageAvantADroite(), Manette : :initialisationStructureVirageAvantAGauche(), Manette : :initialiser↩ Etats(), et operator==().

6.8.3.6 joystickGaucheEnAvant

```
bool EtatManetteDeplacement::joystickGaucheEnAvant
```

Membre définissant l'état du joystick sur l'axe Y.

Définition à la ligne 180 du fichier manette.h.

Référencé par Manette : :changerAxeYJoystickGauche(), Manette : :initialisationStructureRotationADroite(), Manette : :initialisationStructureRotationAGauche(), Manette : :initialisationStructuresEnArriere(), Manette : :initialisationStructuresEnAvant(), Manette : :initialisationStructuresRotationADroiteDoucement(), Manette : :initialisationStructuresRotationAGaucheDoucement(), Manette : :initialisationStructuresVirageArriereADroiteDoucement(), Manette : :initialisationStructuresVirageArriereAGaucheDoucement(), Manette : :initialisationStructuresVirageAvantADroiteDoucement(), Manette : :initialisationStructuresVirageAvantAGaucheDoucement(), Manette : :initialisationStructureVirageArriereADroite(), Manette : :initialisationStructureVirageArriereAGauche(), Manette : :initialisationStructureVirageAvantADroite(), Manette : :initialisationStructureVirageAvantAGauche(), Manette : :initialiserEtats(), et operator==().

La documentation de cette structure a été générée à partir des fichiers suivants :

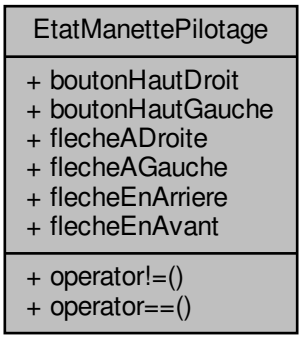
- manette.h
- manette.cpp

6.9 Référence de la structure EtatManettePilotage

Structure qui définit l'état de la manette en mode pilotage de la pince.

```
#include <manette.h>
```

Graphe de collaboration de EtatManettePilotage :



Fonctions membres publiques

- bool operator != (const EtatManettePilotage &maStructure) const
  Surcharge de l'opérateur != afin de comparer un état de manette prédéfini avec l'état actuel de la manette.
- bool operator== (const EtatManettePilotage &maStructure) const
  Surcharge de l'opérateur == afin de comparer un état de manette prédéfini avec l'état actuel de la manette.

Attributs publics

- bool boutonHautDroit
  Membre définissant l'état du bouton R1.
- bool boutonHautGauche
  Membre définissant l'état du bouton L1.
- bool flecheADroite
  Membre définissant l'état de la flèche de droite.
- bool flecheAGauche
  Membre définissant l'état de la flèche de gauche.
- bool flecheEnArriere
  Membre définissant l'état de la flèche du bas.
- bool flecheEnAvant
  Membre définissant l'état de la flèche du haut.

6.9.1 Description détaillée

Structure qui définit l'état de la manette en mode pilotage de la pince.

Définition à la ligne 150 du fichier manette.h.

6.9.2 Documentation des fonctions membres

6.9.2.1 operator"!=()

```
bool EtatManettePilotage::operator!= (
            const EtatManettePilotage & maStructure ) const
```

Surcharge de l'opérateur != afin de comparer un état de manette prédéfini avec l'état actuel de la manette.

Paramètres

maStructure	

Renvoie

Si l'opération est possible ou pas

Définition à la ligne 631 du fichier manette.cpp.

```
00632 {
00633     return !(*this == maStructure);
00634 }
```

6.9.2.2 operator==()

```
bool EtatManettePilotage::operator== (
            const EtatManettePilotage & maStructure ) const
```

Surcharge de l'opérateur == afin de comparer un état de manette prédéfini avec l'état actuel de la manette.

Paramètres

maStructure	

Renvoie

Si l'opération est possible ou pas

Définition à la ligne 613 du fichier manette.cpp.

Références boutonHautDroit, boutonHautGauche, flecheADroite, flecheAGauche, flecheEnArriere, et flecheEnAvant.

```
00614 {
00615     if(this->flecheEnAvant != maStructure.flecheEnAvant)
00616         return false;
00617     else if(this->flecheEnArriere != maStructure.flecheEnArriere)
00618         return false;
00619     else if(this->flecheAGauche != maStructure.flecheAGauche)
00620         return false;
00621     else if(this->flecheADroite != maStructure.flecheADroite)
00622         return false;
00623     else if(this->boutonHautGauche != maStructure.
      boutonHautGauche)
00624         return false;
00625     else if(this->boutonHautDroit != maStructure.boutonHautDroit)
00626         return false;
00627     else
00628         return true;
00629 }
```

6.9.3 Documentation des données membres

6.9.3.1 boutonHautDroit

```
bool EtatManettePilotage::boutonHautDroit
```

Membre définissant l'état du bouton R1.

Définition à la ligne 157 du fichier manette.h.

Référencé par Manette : :changerBoutonHautDroit(), Manette : :initialiserEtats(), Manette : :initialiserTypesPilotage(), et operator==().

6.9.3.2 boutonHautGauche

```
bool EtatManettePilotage::boutonHautGauche
```

Membre définissant l'état du bouton L1.

Définition à la ligne 156 du fichier manette.h.

Référencé par Manette : :changerBoutonHautGauche(), Manette : :initialiserEtats(), Manette : :initialiserTypesPilotage(), et operator==().

6.9.3.3 flecheADroite

`bool EtatManettePilotage::flecheADroite`

Membre définissant l'état de la flèche de droite.

Définition à la ligne 155 du fichier manette.h.

Référencé par Manette : :changerFlecheADroite(), Manette : :initialiserEtats(), Manette : :initialiserTypesPilotage(), et operator==().

6.9.3.4 flecheAGauche

`bool EtatManettePilotage::flecheAGauche`

Membre définissant l'état de la flèche de gauche.

Définition à la ligne 154 du fichier manette.h.

Référencé par Manette : :changerFlecheAGauche(), Manette : :initialiserEtats(), Manette : :initialiserTypesPilotage(), et operator==().

6.9.3.5 flecheEnArriere

`bool EtatManettePilotage::flecheEnArriere`

Membre définissant l'état de la flèche du bas.

Définition à la ligne 153 du fichier manette.h.

Référencé par Manette : :changerFlecheEnArriere(), Manette : :initialiserEtats(), Manette : :initialiserTypesPilotage(), et operator==().

6.9.3.6 flecheEnAvant

`bool EtatManettePilotage::flecheEnAvant`

Membre définissant l'état de la flèche du haut.

Définition à la ligne 152 du fichier manette.h.

Référencé par Manette : :changerFlecheEnAvant(), Manette : :initialiserEtats(), Manette : :initialiserTypesPilotage(), et operator==().

La documentation de cette structure a été générée à partir des fichiers suivants :

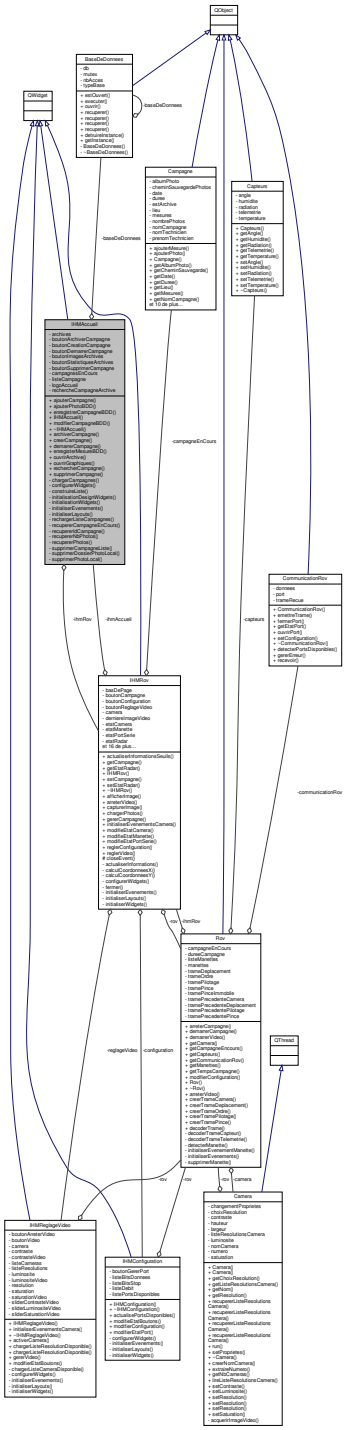
- manette.h
- manette.cpp

6.10 Référence de la classe IHMAccueil

Class permettant de créer une nouvelle campagne, reprendre une campagne mise en pause, archiver une campagne, supprimer une campagne, accéder à la base de données et configurer le matériel.

`#include "ihmaccueil.h"`

Graphe de collaboration de IHMAccueil :



Connecteurs publics

— void archiverCampagne ()

Permet d'archiver la campagne selectionner.
- void creerCampagne ()
  Permet de créer une nouvelle campagne.
- void demarrerCampagne ()
  Permet de démarrer ou reprendre une campagne non archivé
- void enregisterMesureBDD (QString temperature, QString humidite, QString radiation)
  enregistre les mesures recues dans la base de données
- void ouvrirArchive ()
  Ouvre le dossier des archvies correspondante à la mission saisie.
- void ouvrirGraphiques ()
  Ouvre l'ihm des graphiques correspondant à la misssion saisie.
- void rechercherCampagne (QString texte)
  Fait une recherche des noms dans la base de données correspondants au texte.
- void supprimerCampagne ()
  Permet d'archiver la campagne dans la base de données.

Fonctions membres publiques

- void ajouterCampagne (Campagne ∗campagne, bool verification=false)
  Ajoute une nouvelle campagne dans la liste des campagne non archivés.
- void ajouterPhotoBDD (Photo &photo, Campagne ∗campagne)
  Ajoute la photo prise dans la BDD associé a la campagne.
- void enregistrerCampagneBDD (Campagne ∗campagne)
  Enregistre les informations de la campagne dans la BDD.
- IHMAccueil (QWidget ∗parent=nullptr)
  Constructeur de la classe IHMAccueil.
- void modifierCampagneBDD (Campagne ∗campagne)
  Met à jour les informations de la campagne lors de l'arret de celle-ci dans la BDD.
- ∼IHMAccueil ()
  Destructeur de la classe IHMAccueil.

Fonctions membres privées

- void chargerCampagnes ()
  Récupere la liste des noms de campagne non terminés et ajoute les nom de la liste des campagnes disponibles.
- void configurerWidgets ()
  Configure les widgets de l'IHM.
- void construireListe (QVector< QString > liste)
  Construit la liste déroulante des campagnes sélectionnées.
- void initialisationDesignWidgets ()
  Initialise les design des widgets de l'IHM.
- void initialisationWidgets ()
  Initialise les widgets de l'IHM.
- void initialiserEvenements ()
  Initialise les evenements de l'IHM.
- void initialiserLayouts ()
  Initialise les layouts de l'IHM.
- void rechargerListeCampagnes ()
  Recharge la liste des campagnes en cours.
- void recupererCampagneEnCours (bool &retourCampagne, QString &requeteInformationsCampagne, QVector< QStringList > &campagnesEnCours)
  Récupère les campagnes en cours dans la base de données, le paramètre campagnesEnCours passé en référence récupère les valeurs.
- QString recupererIdCampagne ()
  Recupere l'id de la campagne séléctionné dans la liste.
- void recupererNbPhotos (QString &nombrePhotos, QString &requeteNombrePhotos)
  Récupère le nombre de photos dans la base de données, le paramètre nombrePhotos passé en référence récupère la valeur.
- void recupererPhotos (bool &retourPhoto, QString &requeteInformationsPhotos, QVector< QStringList > &informationsPhotos)
  Récupère les photos associés a une campagne dans la base de données, le paramètre informationsPhotos passé en référence récupère les valeurs.
- void supprimerCampagneListe ()
  Permet de supprimer de lq liste la campagne selectionné
- void supprimerDossierPhotoLocal ()
  Supprime le dossier photo si il est vide.
- void supprimerPhotoLocal (QString requete)
  Selectionne les chemin d'accès des photo à supprimer dans la base de données et les supprime en local.

Attributs privés

- QLabel * archives
  Texte indiquant la zone de gestation des archives.
- BaseDeDonnees * baseDeDonnees
  Instance d'un objet BaseDeDonnees permettant d'acceder a la BDD.
- QPushButton * boutonArchiverCampagne
  Bouton permettant d'archiver la campagne sélectionner.
- QPushButton * boutonCreationCampagne
  Bouton permettant de créer une nouvelle campagne.
- QPushButton * boutonDemarrerCampagne
  Bouton permettant de démarrer ou reprendre la campagne seléctionner.
- QPushButton * boutonImagesArchives
  Bouton permettant d'accéder aux archives.
- QPushButton * boutonStatistiquesArchives
  Bouton permettant de configurer le matériel.
- QPushButton * boutonSupprimerCampagne
  Bouton permettant de supprimmer la campagne sélectionner.
- QVector< Campagne * > campagnesEnCours
  Conteneur des campagnes non archivés.
- IHMRov * ihmRov
  Instance d'un objet ihmRov.
- QComboBox * listeCampagne
  Liste des campagnes créer et non archivés.
- QLabel * logoAccueil
  Logo de l'IHM accueil.
- QLineEdit * rechercheCampagneArchive
  Zone de recherche des campagnes archivés.

6.10.1 Description détaillée

Class permettant de créer une nouvelle campagne, reprendre une campagne mise en pause, archiver une campagne, supprimer une campagne, accéder à la base de données et configurer le matériel.

Définition à la ligne 29 du fichier ihmaccueil.h.

6.10.2 Documentation des constructeurs et destructeur

6.10.2.1 IHMAccueil()

```
IHMAccueil::IHMAccueil (
            QWidget * parent = nullptr )  [explicit]
```

Constructeur de la classe IHMAccueil.

Paramètres

parent	

Définition à la ligne 15 du fichier ihmaccueil.cpp.

Références baseDeDonnees, chargerCampagnes(), configurerWidgets(), BaseDeDonnees : :getInstance(), ihmRov, initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), initialiserLayouts(), et rechargerListeCampagnes().

```
00015                                       : QWidget(parent), campagnesEnCours()
00016 {
00017     qDebug() << Q_FUNC_INFO;
```

```
00018     baseDeDonnees = BaseDeDonnees::getInstance();
00019
00020     initialisationWidgets();
00021     initialisationDesignWidgets();
00022     initialiserEvenements();
00023     initialiserLayouts();
00024     configurerWidgets();
00025
00026     chargerCampagnes();
00027     rechargerListeCampagnes();
00028
00029     ihmRov = new IHMRov(this);
00030     ihmRov->hide();
00031 }
```

6.10.2.2 ~IHMAccueil()

```
IHMAccueil::~IHMAccueil ( )
```

Destructeur de la classe IHMAccueil.

Définition à la ligne 33 du fichier ihmaccueil.cpp.

Références BaseDeDonnees : :detruireInstance().

```
00034 {
00035     BaseDeDonnees::detruireInstance();
00036     qDebug() << Q_FUNC_INFO;
00037 }
```

6.10.3 Documentation des fonctions membres

6.10.3.1 ajouterCampagne()

```
void IHMAccueil::ajouterCampagne (
            Campagne * campagne,
            bool verification = false )
```

Ajoute une nouvelle campagne dans la liste des campagne non archivés.

Paramètres

campagne	
verification	

Définition à la ligne 327 du fichier ihmaccueil.cpp.

Références campagnesEnCours, Campagne : :getCheminSauvegarde(), Campagne : :getNomCampagne(), et rechargerListe↩ Campagnes().

Référencé par chargerCampagnes(), et IHMCreationCampagne : :validerCampagne().

```
00328 {
00329     QDir dossierCampagne(campagne->getCheminSauvegarde());
00330     if(dossierCampagne.exists())
00331     {
```

```
00332         if(!dossierCampagne.mkdir(campagne->getNomCampagne()))
00333         {
00334             qDebug() << Q_FUNC_INFO << "Erreur : impossible de créer le dossier" << campagne->
      getCheminSauvegarde() << "campagne" << campagne->
      getNomCampagne();
00335             if(verification)
00336             {
00337                 QMessageBox::critical(this, "Erreur", "Erreur : impossible de créer le dossier " + campagne
      ->getCheminSauvegarde() + " !");
00338                 return;
00339             }
00340         }
00341     }
00342     campagnesEnCours.push_back(campagne);
00343     rechargerListeCampagnes();
00344 }
```

6.10.3.2 ajouterPhotoBDD()

```
void IHMAccueil::ajouterPhotoBDD (
            Photo & photo,
            Campagne * campagne )
```

Ajoute la photo prise dans la BDD associé a la campagne.

Paramètres

photo	
campagne	

Définition à la ligne 386 du fichier ihmaccueil.cpp.

Références baseDeDonnees, Photo : :cheminSauvegarde, Photo : :dateheure, BaseDeDonnees : :executer(), Campagne : :getNom↩ Campagne(), et BaseDeDonnees : :recuperer().

Référencé par IHMRov : :capturerImage().

```
00387 {
00388     QString idCampagne;
00389     QString requeteIdCampagne = "SELECT IdCampagne FROM campagne WHERE campagne.nom = '" + campagne->
      getNomCampagne() + "'";
00390     baseDeDonnees->recuperer(requeteIdCampagne, idCampagne);
00391
00392     QString requeteInsertion = "INSERT INTO photo (idCampagne, cheminImage, aGarder, dateHeure) VALUES ('"
      + idCampagne + "', '" + photo.cheminSauvegarde + "', '1', '" + photo.
      dateheure.toString() + "')";
00393     baseDeDonnees->executer(requeteInsertion);
00394 }
```

6.10.3.3 archiverCampagne

```
void IHMAccueil::archiverCampagne ( )  [slot]
```

Permet d'archiver la campagne selectionner.

Définition à la ligne 424 du fichier ihmaccueil.cpp.

Références baseDeDonnees, BaseDeDonnees : :executer(), listeCampagne, BaseDeDonnees : :ouvrir(), recupererIdCampagne(), supprimerCampagneListe(), et supprimerPhotoLocal().

Référencé par initialiserEvenements().

```
00425 {
00426     baseDeDonnees->ouvrir("campagnes.sqlite");
00427     bool retourRequeteArchiver;
00428     bool retourRequeteSuppressionPhoto;
00429     QString requeteArchiver, requeteSuppressionPhoto;
00430 
00431     supprimerPhotoLocal("SELECT photo.cheminImage FROM photo WHERE photo.IdCampagne = '"
       + recupererIdCampagne() + "' AND photo.aGarder = '0'");
00432 
00433     requeteArchiver = "UPDATE campagne SET enCours = '0' WHERE campagne.nom = '" +
      listeCampagne->currentText() + "'";
00434     requeteSuppressionPhoto = "DELETE FROM photo WHERE photo.IdCampagne = '" +
      recupererIdCampagne() + "' AND photo.aGarder = '0'";
00435 
00436     retourRequeteArchiver = baseDeDonnees->executer(requeteArchiver);
00437     retourRequeteSuppressionPhoto = baseDeDonnees->executer(requeteSuppressionPhoto);
00438 
00439     if(!retourRequeteArchiver && !retourRequeteSuppressionPhoto)
00440     {
00441         #ifdef DEBUG_BASEDEDONNEES
00442             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00443         #endif
00444     }
00445     else
00446         supprimerCampagneListe();
00447 }
```

6.10.3.4 chargerCampagnes()

```
void IHMAccueil::chargerCampagnes ( )  [private]
```

Récupere la liste des noms de campagne non terminés et ajoute les nom de la liste des campagnes disponibles.

Définition à la ligne 202 du fichier ihmaccueil.cpp.

Références Photo : :aGarder, ajouterCampagne(), Campagne : :ajouterPhoto(), baseDeDonnees, campagnesEnCours, Photo↩ : :cheminSauvegarde, Photo : :dateheure, Photo : :image, BaseDeDonnees : :ouvrir(), recupererCampagneEnCours(), recuperer↩ NbPhotos(), recupererPhotos(), Campagne : :setCheminSauvegarde(), et Campagne : :setNombrePhotos().

Référencé par IHMAccueil().

```
00203 {
00204     baseDeDonnees->ouvrir("campagnes.sqlite");
00205 
00206     bool retourCampagne, retourPhoto;
00207     QVector<QStringList> campagnesEnCours;
00208     QString nombrePhotos;
00209     QVector<QStringList> informationsPhotos;
00210 
00211     QString requeteInformationsCampagne = "SELECT campagne.idCampagne, campagne.nom, campagne.lieu,
       technicien.nom, technicien.prenom, campagne.date, campagne.duree, campagne.cheminSauvegarde FROM campagne INNER
       JOIN technicien ON campagne.idTechnicien = technicien.idTechnicien WHERE campagne.enCours = '1' ORDER BY
       campagne.date DESC";
00212 
00213     #ifdef DEBUG_BASEDEDONNEES
00214     qDebug() << Q_FUNC_INFO << QString::fromUtf8("requête : ") << requeteInformationsCampagne;
00215     #endif
00216 
00217     recupererCampagneEnCours(retourCampagne, requeteInformationsCampagne,
      campagnesEnCours);
00218     if(retourCampagne)
00219     {
00220         for(int i=0; i < campagnesEnCours.size(); i++)
00221         {
00222             QStringList informationsCampagne = campagnesEnCours.at(i);
00223             QString requeteNombrePhotos = "SELECT COUNT(IdPhoto) FROM photo INNER JOIN campagne ON
       photo.IdCampagne = campagne.IdCampagne WHERE campagne.IdCampagne = '" + informationsCampagne.at(0) + "' ";
00224             recupererNbPhotos(nombrePhotos, requeteNombrePhotos);
00225 
00226             #ifdef DEBUG_BASEDEDONNEES
00227             qDebug() << Q_FUNC_INFO << QString::fromUtf8("%0 %1 %2 %3 %4 %5 %6 %7").arg(
      informationsCampagne.at(0)).arg(informationsCampagne.at(1)).arg(informationsCampagne.at(2)).arg(informationsCampagne.at(3)).arg
      (informationsCampagne.at(4)).arg(informationsCampagne.at(5)).arg(informationsCampagne.at(6)).arg(
      informationsCampagne.at(7));
00228             #endif
00229             Campagne *campagne = new Campagne(informationsCampagne.at(1),
      informationsCampagne.at(2), informationsCampagne.at(3), informationsCampagne.at(4), QDateTime::fromString(
```

```
      informationsCampagne.at(5)), this, informationsCampagne.at(6).toInt());
00230
00231             campagne->setCheminSauvegarde(informationsCampagne.at(7));
00232             campagne->setNombrePhotos(nombrePhotos.toInt());
00233
00234             QString requeteInformationsPhotos ="SELECT cheminImage, aGarder, dateHeure FROM photo INNER
       JOIN campagne ON photo.IdCampagne = campagne.IdCampagne WHERE campagne.enCours = '1' AND photo.IdCampagne = '"
      + informationsCampagne.at(0) + "'";
00235             recupererPhotos(retourPhoto, requeteInformationsPhotos, informationsPhotos);
00236             if(retourPhoto)
00237             {
00238                 for(int i=0; i < informationsPhotos.size(); i++)
00239                 {
00240                     QStringList informationPhoto = informationsPhotos.at(i);
00241                     Photo photo;
00242                     photo.cheminSauvegarde = informationPhoto.at(0);
00243                     photo.aGarder = informationPhoto.at(1).toInt();
00244                     photo.image = QPixmap(informationPhoto.at(0));
00245                     photo.dateheure = QDateTime::fromString(informationPhoto.at(2));
00246
00247                     campagne->ajouterPhoto(photo);
00248                 }
00249             }
00250             ajouterCampagne(campagne);
00251             informationsPhotos.clear();
00252         }
00253     }
00254     else
00255     {
00256        #ifdef DEBUG_BASEDEDONNEES
00257        qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00258        #endif
00259     }
00260 }
```

6.10.3.5 configurerWidgets()

```
void IHMAccueil::configurerWidgets ( )  [private]
```

Configure les widgets de l'IHM.

Définition à la ligne 128 du fichier ihmaccueil.cpp.

Références listeCampagne, logoAccueil, et rechercheCampagneArchive.

Référencé par IHMAccueil().

```
00129 {
00130     qDebug() << Q_FUNC_INFO << qApp->applicationDirPath() + "/images/Robot.png";
00131     logoAccueil->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/Robot.png"));
00132     listeCampagne->setEditable(false);
00133     rechercheCampagneArchive->setPlaceholderText("Nom de la campagne");
00134     rechercheCampagneArchive->setTextMargins(10,0,0,0);
00135 }
```

6.10.3.6 construireListe()

```
void IHMAccueil::construireListe (
             QVector< QString > liste )  [private]
```

Construit la liste déroulante des campagnes sélectionnées.

Paramètres

liste	des campagnes sélectionnées

Définition à la ligne 196 du fichier ihmaccueil.cpp.

Références rechercheCampagneArchive.

Référencé par rechercherCampagne().

```
00197 {
00198     QCompleter *completeur = new QCompleter(liste.toList(),this);
00199     rechercheCampagneArchive->setCompleter(completeur);
00200 }
```

6.10.3.7 creerCampagne

```
void IHMAccueil::creerCampagne ( )  [slot]
```

Permet de créer une nouvelle campagne.

Définition à la ligne 412 du fichier ihmaccueil.cpp.

Références baseDeDonnees, et BaseDeDonnees : :recuperer().

Référencé par initialiserEvenements().

```
00413 {
00414     QVector<QStringList> listeTechniciens;
00415     QString requete = "SELECT technicien.nom, technicien.prenom FROM technicien";
00416
00417     baseDeDonnees->recuperer(requete, listeTechniciens);
00418
00419     IHMCreationCampagne *ihmCreationCampagne = new
      IHMCreationCampagne(this, listeTechniciens);
00420     ihmCreationCampagne->setFixedSize(416,278);
00421     ihmCreationCampagne->exec();
00422 }
```

6.10.3.8 demarrerCampagne

```
void IHMAccueil::demarrerCampagne ( )  [slot]
```

Permet de démarrer ou reprendre une campagne non archivé

Définition à la ligne 396 du fichier ihmaccueil.cpp.

Références campagnesEnCours, IHMRov : :gererCampagne(), ihmRov, listeCampagne, et IHMRov : :setCampagne().

Référencé par initialiserEvenements().

```
00397 {
00398     for(QVector<Campagne*>::iterator it = campagnesEnCours.begin(); it !=
      campagnesEnCours.end(); it++)
00399     {
00400         if((*it)->getNomCampagne() == listeCampagne->currentText())
00401         {
00402             qDebug() << Q_FUNC_INFO << listeCampagne->currentText();
00403             ihmRov->setCampagne(*it);
00404             ihmRov->showMinimized();
00405             ihmRov->gererCampagne();
00406             this->setVisible(false);
00407             break;
00408         }
00409     }
00410 }
```

6.10.3.9 enregisterMesureBDD

```
void IHMAccueil::enregisterMesureBDD (
            QString temperature,
            QString humidite,
            QString radiation )  [slot]
```

enregistre les mesures recues dans la base de données

Paramètres

temperature	
humidite	
radiation	

Définition à la ligne 478 du fichier ihmaccueil.cpp.

Références baseDeDonnees, BaseDeDonnees : :executer(), BaseDeDonnees : :ouvrir(), et recupererIdCampagne().

```
00479 {
00480     baseDeDonnees->ouvrir("campagnes.sqlite");
00481     bool retourRequeteEnregistrementMesure;
00482
00483     QString requeteEnregistrementMesure = "INSERT INTO mesure (idCampagne, heure, temperature, radiation,
       humidite) VALUES (" + recupererIdCampagne() + ",'" + QDateTime::currentDateTime().toString
      () + "'," + temperature + "," + radiation + "," + humidite + ")";
00484
00485     retourRequeteEnregistrementMesure = baseDeDonnees->executer(
      requeteEnregistrementMesure);
00486
00487     if(!retourRequeteEnregistrementMesure)
00488     {
00489         #ifdef DEBUG_BASEDEDONNEES
00490             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00491         #endif
00492     }
00493 }
```

6.10.3.10 enregistrerCampagneBDD()

```
void IHMAccueil::enregistrerCampagneBDD (
             Campagne * campagne )
```

Enregistre les informations de la campagne dans la BDD.

Paramètres

campagne	

Définition à la ligne 346 du fichier ihmaccueil.cpp.

Références baseDeDonnees, BaseDeDonnees : :executer(), Campagne : :getCheminSauvegarde(), Campagne : :getDate(), Campagne : :getDuree(), Campagne : :getLieu(), Campagne : :getNomCampagne(), Campagne : :getNomTechnicien(), Campagne : :getPrenomTechnicien(), BaseDeDonnees : :ouvrir(), et BaseDeDonnees : :recuperer().

Référencé par IHMCreationCampagne : :validerCampagne().

```
00347 {
00348     #ifdef DEBUG_BASEDEDONNEES
00349     qDebug() << Q_FUNC_INFO << campagne->getNomCampagne();
00350     #endif
00351
00352     baseDeDonnees->ouvrir("campagnes.sqlite");
00353
00354     QString idTechnicien;
00355     QString requeteId = "SELECT technicien.IdTechnicien FROM technicien WHERE technicien.nom = '" +
      campagne->getNomTechnicien() + "' AND technicien.prenom = '" + campagne->
      getPrenomTechnicien() + "'";
00356     bool retour = baseDeDonnees->recuperer(requeteId, idTechnicien);
00357     if(!retour)
00358     {
00359         QString requeteInformation = "INSERT INTO technicien (nom, prenom) VALUES ('" + campagne->
      getNomTechnicien() + "','" + campagne->getPrenomTechnicien() + "')";
```

```
00360         baseDeDonnees->executer(requeteInformation);
00361         baseDeDonnees->recuperer(requeteId, idTechnicien);
00362     }
00363
00364     QString requeteInsertionCampagne = "INSERT INTO campagne (idTechnicien, nom, lieu, date, duree,
       enCours, cheminSauvegarde) VALUES (’" + idTechnicien + "’, ’" + campagne->getNomCampagne() + "’, ’" +
       campagne->getLieu() + "’, ’" + campagne->getDate().toString() + "’, ’" + QString::number(
      campagne->getDuree()) + "’, ’1’, ’" + campagne->getCheminSauvegarde() + "’)";
00365     baseDeDonnees->executer(requeteInsertionCampagne);
00366 }
```

6.10.3.11 initialisationDesignWidgets()

```
void IHMAccueil::initialisationDesignWidgets ( )  [private]
```

Initialise les design des widgets de l'IHM.

Définition à la ligne 53 du fichier ihmaccueil.cpp.

Références archives, boutonArchiverCampagne, boutonCreationCampagne, boutonDemarrerCampagne, boutonImagesArchives, boutonStatistiquesArchives, boutonSupprimerCampagne, listeCampagne, et rechercheCampagneArchive.

Référencé par IHMAccueil().

```
00054 {
00055     QFont policeBouton("", 15, 75, false);
00056     QFont policeTexte("", 13, 75, false);
00057
00058     archives->setFixedSize(80,50);
00059     archives->setFont(policeTexte);
00060
00061     rechercheCampagneArchive->setFixedSize(200,40);
00062     rechercheCampagneArchive->setFont(policeTexte);
00063     rechercheCampagneArchive->setStyleSheet("QLineEdit {border-image:
       url(design/QLine_200x40.png)}" "QLineEdit:hover {border-image: url(design/QLine_200x40_survole.png)}");
00064
00065     boutonImagesArchives->setFixedSize(157,50);
00066     boutonImagesArchives->setFont(policeBouton);
00067     boutonImagesArchives->setStyleSheet("QPushButton {border-image:
       url(design/bouton_157x50.png)}" "QPushButton:hover {border-image: url(design/bouton_157x50_survole.png)}");
00068
00069     boutonStatistiquesArchives->setFixedSize(157,50);
00070     boutonStatistiquesArchives->setFont(policeBouton);
00071     boutonStatistiquesArchives->setStyleSheet("QPushButton {border-image:
       url(design/bouton_157x50.png)}" "QPushButton:hover {border-image: url(design/bouton_157x50_survole.png)}");
00072
00073     boutonCreationCampagne->setFixedSize(302,50);
00074     boutonCreationCampagne->setFont(policeBouton);
00075     boutonCreationCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_302x50.png)}" "QPushButton:hover {border-image: url(design/bouton_302x50_survole.png)}");
00076
00077     boutonDemarrerCampagne->setFixedSize(302,50);
00078     boutonDemarrerCampagne->setFont(policeBouton);
00079     boutonDemarrerCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_302x50.png)}" "QPushButton:hover {border-image: url(design/bouton_302x50_survole.png)}");
00080
00081     boutonArchiverCampagne->setFixedSize(199,50);
00082     boutonArchiverCampagne->setFont(policeBouton);
00083     boutonArchiverCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_199x50.png)}" "QPushButton:hover {border-image: url(design/bouton_199x50_survole.png)}");
00084
00085     boutonSupprimerCampagne->setFixedSize(199,50);
00086     boutonSupprimerCampagne->setFont(policeBouton);
00087     boutonSupprimerCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_199x50.png)}" "QPushButton:hover {border-image: url(design/bouton_199x50_survole.png)}");
00088
00089     listeCampagne->setFixedSize(199,50);
00090     listeCampagne->setFont(policeTexte);
00091     listeCampagne->setStyleSheet("QComboBox {border-image: url(design/combobox_199x50.png)}" "
      QComboBox:hover {border-image: url(design/combobox_199x50_survole.png)}" "QComboBox::drop-down
       {border-image: url(rien.png)}" "QComboBox {padding: 0 0 0 15px}");
00092
00093
00094 }
```

6.10.3.12 initialisationWidgets()

```
void IHMAccueil::initialisationWidgets ( )  [private]
```

Initialise les widgets de l'IHM.

Définition à la ligne 39 du fichier ihmaccueil.cpp.

Références archives, boutonArchiverCampagne, boutonCreationCampagne, boutonDemarrerCampagne, boutonImagesArchives, boutonStatistiquesArchives, boutonSupprimerCampagne, listeCampagne, logoAccueil, et rechercheCampagneArchive.

Référencé par IHMAccueil().

```
00040 {
00041     archives = new QLabel("Archives :", this);
00042     rechercheCampagneArchive = new QLineEdit(this);
00043     boutonImagesArchives = new QPushButton("Photos", this);
00044     boutonStatistiquesArchives = new QPushButton("Graphiques", this);
00045     boutonCreationCampagne = new QPushButton("Créer campagne", this);
00046     boutonDemarrerCampagne = new QPushButton("Démarrer", this);
00047     boutonArchiverCampagne = new QPushButton("Archiver", this);
00048     boutonSupprimerCampagne = new QPushButton("Supprimer", this);
00049     listeCampagne = new QComboBox(this);
00050     logoAccueil = new QLabel(this);
00051 }
```

6.10.3.13 initialiserEvenements()

```
void IHMAccueil::initialiserEvenements ( )  [private]
```

Initialise les evenements de l'IHM.

Définition à la ligne 137 du fichier ihmaccueil.cpp.

Références archiverCampagne(), boutonArchiverCampagne, boutonCreationCampagne, boutonDemarrerCampagne, boutonImagesArchives, boutonStatistiquesArchives, boutonSupprimerCampagne, creerCampagne(), demarrerCampagne(), ouvrirArchive(), ouvrirGraphiques(), rechercheCampagneArchive, rechercherCampagne(), et supprimerCampagne().

Référencé par IHMAccueil().

```
00138 {
00139     connect(rechercheCampagneArchive, SIGNAL(textChanged(QString)), this, SLOT(
      rechercherCampagne(QString)));
00140     connect(boutonDemarrerCampagne, SIGNAL(clicked()), this, SLOT(
      demarrerCampagne()));
00141     connect(boutonCreationCampagne, SIGNAL(clicked()), this, SLOT(
      creerCampagne()));
00142     connect(boutonSupprimerCampagne, SIGNAL(clicked()), this, SLOT(
      supprimerCampagne()));
00143     connect(boutonArchiverCampagne, SIGNAL(clicked()), this, SLOT(
      archiverCampagne()));
00144     connect(boutonImagesArchives, SIGNAL(clicked()), this, SLOT(
      ouvrirArchive()));
00145     connect(boutonStatistiquesArchives, SIGNAL(clicked()), this, SLOT(
      ouvrirGraphiques()));
00146 }
```

6.10.3.14 initialiserLayouts()

```
void IHMAccueil::initialiserLayouts ( )  [private]
```

Initialise les layouts de l'IHM.

Définition à la ligne 96 du fichier ihmaccueil.cpp.

Références archives, boutonArchiverCampagne, boutonCreationCampagne, boutonDemarrerCampagne, boutonImagesArchives, boutonStatistiquesArchives, boutonSupprimerCampagne, listeCampagne, logoAccueil, NOM_FENETRE_ACCUEIL, et rechercheCampagneArchive.

Référencé par IHMAccueil().

```
00097 {
00098     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00099     QHBoxLayout *layoutSuperieur = new QHBoxLayout;
00100     QVBoxLayout *layoutCentral = new QVBoxLayout;
00101     QHBoxLayout *layoutConfigurationCampagne = new QHBoxLayout;
00102     QHBoxLayout *layoutCampagne = new QHBoxLayout;
00103
00104     layoutPrincipal->addLayout(layoutSuperieur);
00105     layoutSuperieur->addWidget(archives);
00106     layoutSuperieur->addWidget(rechercheCampagneArchive);
00107     layoutSuperieur->addWidget(boutonImagesArchives);
00108     layoutSuperieur->addWidget(boutonStatistiquesArchives);
00109
00110     layoutPrincipal->addLayout(layoutCentral);
00111     layoutCentral->setAlignment(Qt::AlignCenter);
00112     layoutCentral->addWidget(logoAccueil);
00113
00114     layoutPrincipal->addLayout(layoutConfigurationCampagne);
00115     layoutConfigurationCampagne->addWidget(listeCampagne);
00116     layoutConfigurationCampagne->addWidget(boutonArchiverCampagne);
00117     layoutConfigurationCampagne->addWidget(boutonSupprimerCampagne);
00118
00119     layoutPrincipal->addLayout(layoutCampagne);
00120     layoutCampagne->addWidget(boutonCreationCampagne);
00121     layoutCampagne->addWidget(boutonDemarrerCampagne);
00122
00123     setLayout(layoutPrincipal);
00124     setWindowTitle(NOM_FENETRE_ACCUEIL);
00125     setStyleSheet("background:#C1BEBE;");
00126 }
```

6.10.3.15 modifierCampagneBDD()

```
void IHMAccueil::modifierCampagneBDD (
            Campagne * campagne )
```

Met à jour les informations de la campagne lors de l'arret de celle-ci dans la BDD.

Paramètres

campagne	

Définition à la ligne 368 du fichier ihmaccueil.cpp.

Références baseDeDonnees, BaseDeDonnees : :executer(), Campagne : :getAlbumPhoto(), Campagne : :getDuree(), Campagne : :getNomCampagne(), BaseDeDonnees : :recuperer(), et Campagne : :supprimerMesures().

Référencé par IHMRov : :fermer().

```
00369 {
00370     QString idCampagne, requeteUpdateDuree, requeteIdCampagne, requeteInsertionMesures,
      requeteModifierPhotos;
00371
00372     requeteUpdateDuree = "UPDATE campagne SET duree = '"+ QString::number(campagne->
      getDuree()) + "' WHERE campagne.nom = '" + campagne->getNomCampagne() + "'";
00373     baseDeDonnees->executer(requeteUpdateDuree);
00374
00375     requeteIdCampagne = "SELECT campagne.idCampagne FROM campagne WHERE campagne.nom = '" + campagne->
      getNomCampagne() + "'";
00376     baseDeDonnees->recuperer(requeteIdCampagne, idCampagne);
00377
00378     for(int i =0; i < campagne->getAlbumPhoto().size(); ++i)
00379     {
00380         requeteModifierPhotos = "UPDATE photo set aGarder = '" + QString::number(campagne->
      getAlbumPhoto()[i].aGarder) + "' WHERE cheminImage = '" + campagne->
      getAlbumPhoto()[i].cheminSauvegarde + "'";
00381         baseDeDonnees->executer(requeteModifierPhotos);
00382     }
00383     campagne->supprimerMesures();
00384 }
```

6.10.3.16 ouvrirArchive

```
void IHMAccueil::ouvrirArchive ( )  [slot]
```

Ouvre le dossier des archvies correspondante à la mission saisie.

Définition à la ligne 522 du fichier ihmaccueil.cpp.

Références baseDeDonnees, BaseDeDonnees : :ouvrir(), rechercheCampagneArchive, et BaseDeDonnees : :recuperer().

Référencé par initialiserEvenements().

```
00523 {
00524     QStringList cheminSauvegarde;
00525     QString requete = "SELECT campagne.cheminSauvegarde, campagne.nom FROM campagne WHERE campagne.nom = '"
       + rechercheCampagneArchive->text() + "'";
00526
00527     baseDeDonnees->ouvrir("campagnes.sqlite");
00528     bool retour = baseDeDonnees->recuperer(requete, cheminSauvegarde);
00529     if(retour)
00530     {
00531         if(!(QDesktopServices::openUrl(QUrl(cheminSauvegarde.at(0) + "/" + cheminSauvegarde.at(1),
      QUrl::TolerantMode))))
00532             QMessageBox::critical(this, "Erreur", "Erreur : Chemin introuvable ");
00533     }
00534     else
00535     {
00536         QMessageBox::critical(this, "Erreur", "Erreur : Nom de campagne introuvable ");
00537     }
00538 }
```

6.10.3.17 ouvrirGraphiques

```
void IHMAccueil::ouvrirGraphiques ( )  [slot]
```

Ouvre l'ihm des graphiques correspondant à la misssion saisie.

Définition à la ligne 540 du fichier ihmaccueil.cpp.

Références baseDeDonnees, BaseDeDonnees : :ouvrir(), rechercheCampagneArchive, et BaseDeDonnees : :recuperer().

Référencé par initialiserEvenements().

```
00541 {
00542     QVector<QStringList> mesures;
00543     QString requete = "SELECT mesure.heure, mesure.radiation, mesure.temperature, mesure.humidite FROM
       mesure INNER JOIN campagne ON campagne.IdCampagne = mesure.IdCampagne WHERE campagne.nom = '" +
      rechercheCampagneArchive->text() + "'";
00544
00545     baseDeDonnees->ouvrir("campagnes.sqlite");
00546     bool retour = baseDeDonnees->recuperer(requete, mesures);
00547     if(retour)
00548     {
00549         if(!mesures.isEmpty())
00550         {
00551             IHMGraphiques *ihmGraphique = new IHMGraphiques(mesures);
00552             ihmGraphique->show();
00553         }
00554         else
00555         {
00556             QMessageBox::critical(this, "Erreur", "Erreur : Aucune données trouvées !");
00557         }
00558     }
00559 }
```

6.10.3.18 rechargerListeCampagnes()

```
void IHMAccueil::rechargerListeCampagnes ( )  [private]
```

Recharge la liste des campagnes en cours.

Définition à la ligne 148 du fichier ihmaccueil.cpp.

Références boutonArchiverCampagne, boutonDemarrerCampagne, boutonSupprimerCampagne, campagnesEnCours, et listeCampagne.

Référencé par ajouterCampagne(), IHMAccueil(), et supprimerCampagneListe().

```
00149 {
00150     listeCampagne->clear();
00151
00152     for(QVector<Campagne*>::iterator it = campagnesEnCours.begin(); it !=
      campagnesEnCours.end(); it++)
00153     {
00154         listeCampagne->addItem((*it)->getNomCampagne());
00155     }
00156
00157     if(listeCampagne->currentText().isEmpty())
00158     {
00159         boutonDemarrerCampagne->setDisabled(true);
00160         boutonDemarrerCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_302x50_grise.png)}");
00161
00162         boutonArchiverCampagne->setDisabled(true);
00163         boutonArchiverCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_199x50_grise.png)}");
00164
00165         boutonSupprimerCampagne->setDisabled(true);
00166         boutonSupprimerCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_199x50_grise.png)}");
00167     }
00168     else
00169     {
00170         boutonDemarrerCampagne->setEnabled(true);
00171         boutonDemarrerCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_302x50.png)}" "QPushButton:hover {border-image: url(design/bouton_302x50_survole.png)}");
00172
00173         boutonArchiverCampagne->setEnabled(true);
00174         boutonArchiverCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_199x50.png)}" "QPushButton:hover {border-image: url(design/bouton_199x50_survole.png)}");
00175
00176         boutonSupprimerCampagne->setEnabled(true);
00177         boutonSupprimerCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_199x50.png)}" "QPushButton:hover {border-image: url(design/bouton_199x50_survole.png)}");
00178     }
00179 }
```

6.10.3.19 rechercherCampagne

```
void IHMAccueil::rechercherCampagne (
            QString texte )  [slot]
```

Fait une recherche des noms dans la base de données correspondants au texte.

Paramètres

texte	

Définition à la ligne 495 du fichier ihmaccueil.cpp.

Références baseDeDonnees, construireListe(), BaseDeDonnees : :ouvrir(), et BaseDeDonnees : :recuperer().

Référencé par initialiserEvenements().

```
00496 {
00497     #ifdef DEBUG_BASEDEDONNEES
00498         qDebug() << Q_FUNC_INFO;
00499     #endif
00500
00501     baseDeDonnees->ouvrir("campagnes.sqlite");
00502     QVector<QString> listeCampagnesRecherchees;
00503
00504     bool retourRequeteRechercheCampagne;
00505
00506     QString requeteRechercheCampagne = "SELECT campagne.nom FROM campagne WHERE campagne.nom LIKE '" +
      texte + "%' AND campagne.enCours = '0'";
00507
00508     retourRequeteRechercheCampagne = baseDeDonnees->recuperer(
      requeteRechercheCampagne, listeCampagnesRecherchees);
00509
00510     if(!retourRequeteRechercheCampagne)
00511     {
00512         #ifdef DEBUG_BASEDEDONNEES
00513             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00514         #endif
00515     }
00516     else
00517     {
00518         construireListe(listeCampagnesRecherchees);
00519     }
00520 }
```

6.10.3.20 recupererCampagneEnCours()

```
void IHMAccueil::recupererCampagneEnCours (
            bool & retourCampagne,
            QString & requeteInformationsCampagne,
            QVector< QStringList > & campagnesEnCours )  [private]
```

Récupère les campagnes en cours dans la base de données, le paramètre campagnesEnCours passé en référence récupère les valeurs.

Paramètres

retourCampagne	
requeteInformationsCampagne	
campagnesEnCours	

Définition à la ligne 186 du fichier ihmaccueil.cpp.

Références baseDeDonnees, et BaseDeDonnees : :recuperer().

Référencé par chargerCampagnes().

```
00187 {
00188     retourCampagne = baseDeDonnees->recuperer(requeteInformationsCampagne,
      campagnesEnCours);
00189 }
```

6.10.3.21 recupererIdCampagne()

```
QString IHMAccueil::recupererIdCampagne ( )  [private]
```

Recupere l'id de la campagne séléctionné dans la liste.

Renvoie

L'id sous forme de QString

Définition à la ligne 278 du fichier ihmaccueil.cpp.

Références baseDeDonnees, listeCampagne, et BaseDeDonnees : :recuperer().

Référencé par archiverCampagne(), enregisterMesureBDD(), et supprimerCampagne().

```
00279 {
00280     bool retourRequeteIdCampagne;
00281     QString idCampagne;
00282     QString requeteIdCampagne = "SELECT idCampagne FROM campagne WHERE campagne.nom = '" +
      listeCampagne->currentText() + "'";
00283
00284     retourRequeteIdCampagne = baseDeDonnees->recuperer(requeteIdCampagne, idCampagne)
     ;
00285
00286     if(!retourRequeteIdCampagne)
00287     {
00288         #ifdef DEBUG_BASEDEDONNEES
00289             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00290         #endif
00291         return "0";
00292     }
00293     else
00294         return idCampagne;
00295 }
```

6.10.3.22 recupererNbPhotos()

```
void IHMAccueil::recupererNbPhotos (
            QString & nombrePhotos,
            QString & requeteNombrePhotos )  [private]
```

Récupère le nombre de photos dans la base de données, le paramètre nombrePhotos passé en référence récupère la valeur.

Paramètres

nombrePhotos	
requeteNombrePhotos	

Définition à la ligne 181 du fichier ihmaccueil.cpp.

Références baseDeDonnees, et BaseDeDonnees : :recuperer().

Référencé par chargerCampagnes().

```
00182 {
00183     baseDeDonnees->recuperer(requeteNombrePhotos, nombrePhotos);
00184 }
```

6.10.3.23 recupererPhotos()

```
void IHMAccueil::recupererPhotos (
            bool & retourPhoto,
            QString & requeteInformationsPhotos,
            QVector< QStringList > & informationsPhotos )  [private]
```

Récupère les photos associés a une campagne dans la base de données, le paramètre informationsPhotos passé en référence récupère les valeurs.

Paramètres

retourPhoto	
requeteInformationsPhotos	
informationsPhotos	

Définition à la ligne 191 du fichier ihmaccueil.cpp.

Références baseDeDonnees, et BaseDeDonnees : :recuperer().

Référencé par chargerCampagnes().

```
00192 {
00193     retourPhoto = baseDeDonnees->recuperer(requeteInformationsPhotos,
      informationsPhotos);
00194 }
```

6.10.3.24 supprimerCampagne

```
void IHMAccueil::supprimerCampagne ( )  [slot]
```

Permet d'archiver la campagne dans la base de données.

Définition à la ligne 449 du fichier ihmaccueil.cpp.

Références baseDeDonnees, BaseDeDonnees : :executer(), BaseDeDonnees : :ouvrir(), recupererIdCampagne(), supprimer↩CampagneListe(), supprimerDossierPhotoLocal(), et supprimerPhotoLocal().

Référencé par initialiserEvenements().

```
00450 {
00451     baseDeDonnees->ouvrir("campagnes.sqlite");
00452     bool retourRequeteSuppressionMesures;
00453     bool retourRequeteSuppressionPhotos;
00454     bool retourRequeteSuppressionCampagne;
00455
00456     supprimerPhotoLocal("SELECT photo.cheminImage FROM photo WHERE photo.IdCampagne = '"
       + recupererIdCampagne() + "'");
00457
00458     supprimerDossierPhotoLocal();
00459
00460     QString requeteSuppressionMesures = "DELETE FROM mesure WHERE mesure.IdCampagne = '" +
      recupererIdCampagne() + "'";
00461     QString requeteSuppressionPhotos = "DELETE FROM photo WHERE photo.IdCampagne = '" +
      recupererIdCampagne() + "'";
00462     QString requeteSuppressionCampagne = "DELETE FROM campagne WHERE campagne.IdCampagne = '" +
      recupererIdCampagne() + "'";
00463
00464     retourRequeteSuppressionMesures = baseDeDonnees->executer(
      requeteSuppressionMesures);
00465     retourRequeteSuppressionPhotos = baseDeDonnees->executer(requeteSuppressionPhotos)
      ;
00466     retourRequeteSuppressionCampagne = baseDeDonnees->executer(
```

```
    requeteSuppressionCampagne);
00467
00468     if(!retourRequeteSuppressionMesures && !retourRequeteSuppressionPhotos && !
    retourRequeteSuppressionCampagne)
00469     {
00470         #ifdef DEBUG_BASEDEDONNEES
00471             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00472         #endif
00473     }
00474     else
00475         supprimerCampagneListe();
00476 }
```

6.10.3.25 supprimerCampagneListe()

`void IHMAccueil::supprimerCampagneListe ( )  [private]`

Permet de supprimer de lq liste la campagne selectionné

Définition à la ligne 262 du fichier ihmaccueil.cpp.

Références campagnesEnCours, listeCampagne, et rechargerListeCampagnes().

Référencé par archiverCampagne(), et supprimerCampagne().

```
00263 {
00264     int nbCampagnes = campagnesEnCours.size();
00265     int n = 0;
00266     for(QVector<Campagne*>::iterator it = campagnesEnCours.begin(); n < nbCampagnes; it++)
00267     {
00268         if((*it)->getNomCampagne() == listeCampagne->currentText())
00269         {
00270             delete (*it);
00271             campagnesEnCours.erase(it);
00272         }
00273         n++;
00274     }
00275     rechargerListeCampagnes();
00276 }
```

6.10.3.26 supprimerDossierPhotoLocal()

`void IHMAccueil::supprimerDossierPhotoLocal ( )  [private]`

Supprime le dossier photo si il est vide.

Définition à la ligne 307 du fichier ihmaccueil.cpp.

Références campagnesEnCours, et listeCampagne.

Référencé par supprimerCampagne().

```
00308 {
00309     QDir qDir;
00310
00311     for(QVector<Campagne*>::Iterator it = campagnesEnCours.begin(); it !=
    campagnesEnCours.end(); ++it)
00312     {
00313         if((*it)->getNomCampagne() == listeCampagne->currentText())
00314         {
00315             if(qDir.exists((*it)->getCheminSauvegarde() + "/" + (*it)->getNomCampagne()))
00316             {
00317                 if(!qDir.rmdir((*it)->getCheminSauvegarde() + "/" + (*it)->getNomCampagne()))
00318                 {
00319                     qDebug() << Q_FUNC_INFO << "Erreur : impossible de supprimer le dossier" << (*it)->
    getCheminSauvegarde() + "/" + (*it)->getNomCampagne();
00320                     QMessageBox::critical(this, "Erreur", "Erreur : impossible de supprimer le dossier " +
    (*it)->getCheminSauvegarde() + "/" + (*it)->getNomCampagne() + " !");
00321                 }
00322             }
00323         }
00324     }
00325 }
```

6.10.3.27 supprimerPhotoLocal()

```
void IHMAccueil::supprimerPhotoLocal (
            QString requete )  [private]
```

Selectionne les chemin d'accès des photo à supprimer dans la base de données et les supprime en local.

Paramètres

requete	

Définition à la ligne 297 du fichier ihmaccueil.cpp.

Références baseDeDonnees, et BaseDeDonnees : :recuperer().

Référencé par archiverCampagne(), et supprimerCampagne().

```
00298 {
00299     QVector<QString> photoASupprimer;
00300     baseDeDonnees->recuperer(requete, photoASupprimer);
00301     for(QVector<QString>::Iterator it = photoASupprimer.begin(); it != photoASupprimer.end(); ++it)
00302     {
00303        QFile::remove((*it));
00304     }
00305 }
```

6.10.4 Documentation des données membres

6.10.4.1 archives

```
QLabel* IHMAccueil::archives  [private]
```

Texte indiquant la zone de gestation des archives.

Définition à la ligne 43 du fichier ihmaccueil.h.

Référencé par initialisationDesignWidgets(), initialisationWidgets(), et initialiserLayouts().

6.10.4.2 baseDeDonnees

```
BaseDeDonnees* IHMAccueil::baseDeDonnees  [private]
```

Instance d'un objet BaseDeDonnees permettant d'acceder a la BDD.

Définition à la ligne 45 du fichier ihmaccueil.h.

Référencé par ajouterPhotoBDD(), archiverCampagne(), chargerCampagnes(), creerCampagne(), enregisterMesureBDD(), enregistrerCampagneBDD(), IHMAccueil(), modifierCampagneBDD(), ouvrirArchive(), ouvrirGraphiques(), rechercherCampagne(), recupererCampagneEnCours(), recupererIdCampagne(), recupererNbPhotos(), recupererPhotos(), supprimerCampagne(), et supprimerPhotoLocal().

6.10.4.3 boutonArchiverCampagne

```
QPushButton* IHMAccueil::boutonArchiverCampagne  [private]
```

Bouton permettant d'archiver la campagne sélectionner.

Définition à la ligne 38 du fichier ihmaccueil.h.

Référencé par initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), initialiserLayouts(), et rechargerListe↩Campagnes().

6.10.4.4 boutonCreationCampagne

```
QPushButton* IHMAccueil::boutonCreationCampagne  [private]
```

Bouton permettant de créer une nouvelle campagne.

Définition à la ligne 36 du fichier ihmaccueil.h.

Référencé par initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), et initialiserLayouts().

6.10.4.5 boutonDemarrerCampagne

```
QPushButton* IHMAccueil::boutonDemarrerCampagne  [private]
```

Bouton permettant de démarrer ou reprendre la campagne seléctionner.

Définition à la ligne 37 du fichier ihmaccueil.h.

Référencé par initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), initialiserLayouts(), et rechargerListe↩Campagnes().

6.10.4.6 boutonImagesArchives

```
QPushButton* IHMAccueil::boutonImagesArchives  [private]
```

Bouton permettant d'accéder aux archives.

Définition à la ligne 34 du fichier ihmaccueil.h.

Référencé par initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), et initialiserLayouts().

6.10.4.7 boutonStatistiquesArchives

```
QPushButton* IHMAccueil::boutonStatistiquesArchives  [private]
```

Bouton permettant de configurer le matériel.

Définition à la ligne 35 du fichier ihmaccueil.h.

Référencé par initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), et initialiserLayouts().

6.10.4.8 boutonSupprimerCampagne

```
QPushButton* IHMAccueil::boutonSupprimerCampagne  [private]
```

Bouton permettant de supprimmer la campagne sélectionner.

Définition à la ligne 39 du fichier ihmaccueil.h.

Référencé par initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), initialiserLayouts(), et rechargerListe↩Campagnes().

6.10.4.9 campagnesEnCours

```
QVector<Campagne*> IHMAccueil::campagnesEnCours  [private]
```

Conteneur des campagnes non archivés.

Définition à la ligne 44 du fichier ihmaccueil.h.

Référencé par ajouterCampagne(), chargerCampagnes(), demarrerCampagne(), rechargerListeCampagnes(), supprimer↩CampagneListe(), et supprimerDossierPhotoLocal().

6.10.4.10 ihmRov

```
IHMRov* IHMAccueil::ihmRov  [private]
```

Instance d'un objet ihmRov.

Définition à la ligne 46 du fichier ihmaccueil.h.

Référencé par demarrerCampagne(), et IHMAccueil().

6.10.4.11 listeCampagne

```
QComboBox* IHMAccueil::listeCampagne  [private]
```

Liste des campagnes créer et non archivés.

Définition à la ligne 40 du fichier ihmaccueil.h.

Référencé par archiverCampagne(), configurerWidgets(), demarrerCampagne(), initialisationDesignWidgets(), initialisationWidgets(), initialiserLayouts(), rechargerListeCampagnes(), recupererIdCampagne(), supprimerCampagneListe(), et supprimerDossierPhoto↩Local().

6.10.4.12 logoAccueil

```
QLabel* IHMAccueil::logoAccueil  [private]
```

Logo de l'IHM accueil.

Définition à la ligne 42 du fichier ihmaccueil.h.

Référencé par configurerWidgets(), initialisationWidgets(), et initialiserLayouts().

6.10.4.13 rechercheCampagneArchive

```
QLineEdit* IHMAccueil::rechercheCampagneArchive  [private]
```

Zone de recherche des campagnes archivés.

Définition à la ligne 41 du fichier ihmaccueil.h.

Référencé par configurerWidgets(), construireListe(), initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), initialiserLayouts(), ouvrirArchive(), et ouvrirGraphiques().

La documentation de cette classe a été générée à partir des fichiers suivants :

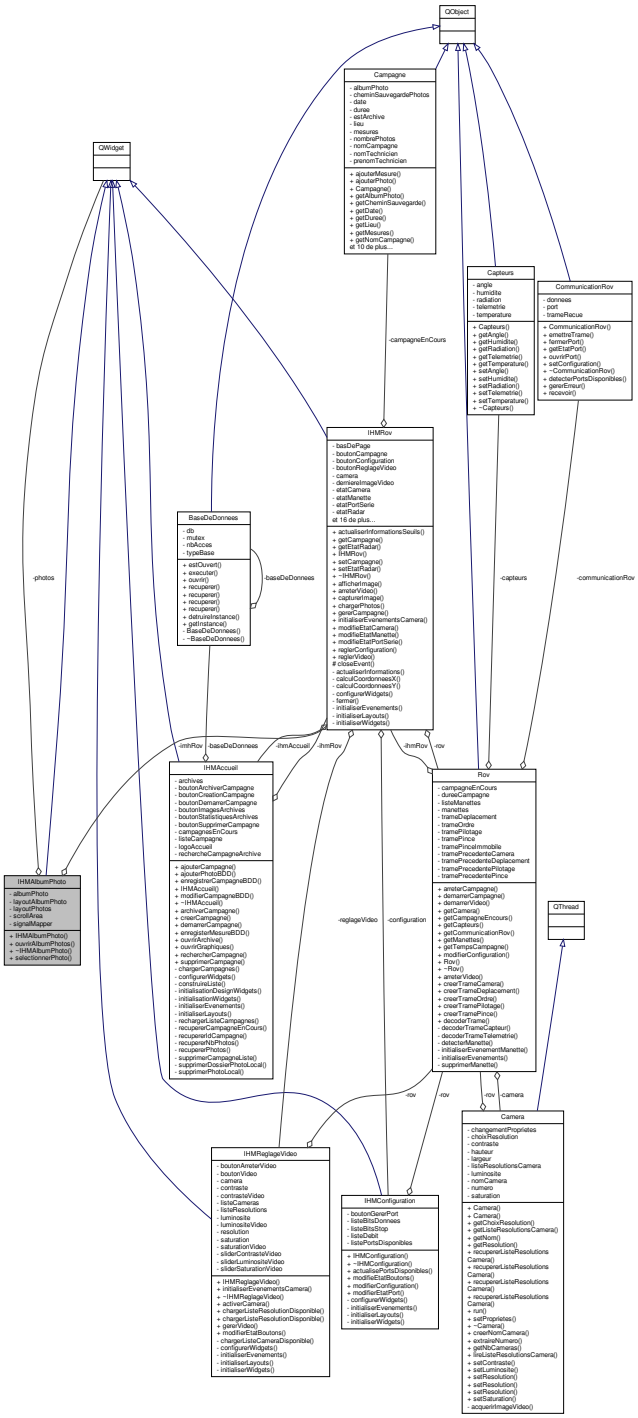
— ihmaccueil.h
— ihmaccueil.cpp

6.11 Référence de la classe IHMAlbumPhoto

Class permettant de visualiser les photos en cours de campagne.

```
#include "ihmAlbumphoto.h"
```

Graphe de collaboration de IHMAlbumPhoto :



Connecteurs publics

— void selectionnerPhoto (int numero)
Permet de selectionner la photo indiquer par le signalMapper.

Fonctions membres publiques

— IHMAlbumPhoto (IHMRov ∗ihmRov, QWidget ∗parent=nullptr)
Constructeur de la classe AlbumPhoto.

— void ouvrirAlbumPhotos (QVector< Photo > albumPhoto)
Ouvre une nouvelle fenetre contenant la liste des photos prises en cours de mission.
— ~IHMAlbumPhoto ()
Destructeur de la classe AlbumPhoto.

Attributs privés

— QVector< Photo > albumPhoto
Conteneur de photo.
— IHMRov * imhRov
Association avec l'IHMRov.
— QVBoxLayout * layoutAlbumPhoto
Layout s'agrandissant selon l'ajout de nouvelle photos.
— QHBoxLayout * layoutPhotos
Layout permettant d'accueillir les différentes photos.
— QWidget * photos
Emplacement permettant d'accueillir les différentes photos.
— QScrollArea * scrollArea
Permet une defilement pour visualiser l'ensemble des photos prises durant la campagne.
— QSignalMapper * signalMapper
Objet de type QSignalMapper, permet d'associer chaque photo de l'IHMAlbumPhoto à un signal.

6.11.1 Description détaillée

Class permettant de visualiser les photos en cours de campagne.

Définition à la ligne 35 du fichier ihmalbumphoto.h.

6.11.2 Documentation des constructeurs et destructeur

6.11.2.1 IHMAlbumPhoto()

```
IHMAlbumPhoto::IHMAlbumPhoto (
            IHMRov * ihmRov,
            QWidget * parent = nullptr )
```

Constructeur de la classe AlbumPhoto.

Paramètres

ihmRov	
parent	

Définition à la ligne 11 du fichier ihmalbumphoto.cpp.

Références layoutAlbumPhoto, layoutPhotos, photos, et scrollArea.

```
00011                                                   : QWidget(parent),
      imhRov(ihmRov)
00012 {
00013     qDebug() << Q_FUNC_INFO;
00014     photos = new QWidget();
00015     layoutPhotos = new QHBoxLayout;
00016     layoutAlbumPhoto = new QVBoxLayout;
00017     scrollArea = new QScrollArea();
00018
```

```
00019     layoutPhotos->setAlignment(Qt::AlignCenter);
00020     layoutAlbumPhoto->setAlignment(Qt::AlignCenter);
00021
00022     photos->setLayout(layoutAlbumPhoto);
00023     scrollArea->setWidgetResizable(true);
00024     scrollArea->setFrameStyle(QFrame::Panel);
00025     scrollArea->setWidget(photos);
00026     layoutPhotos->addWidget(scrollArea);
00027
00028     setLayout(layoutPhotos);
00029
00030     int width = qApp->desktop()->availableGeometry().width();
00031     int height = qApp->desktop()->availableGeometry().height();
00032     resize(width, height);
00033     setStyleSheet("background:#202020;color:white;");
00034 }
```

6.11.2.2 ∼IHMAlbumPhoto()

```
IHMAlbumPhoto::∼IHMAlbumPhoto ( )
```

Destructeur de la classe AlbumPhoto.

Définition à la ligne 36 du fichier ihmalbumphoto.cpp.

```
00037 {
00038     qDebug() << Q_FUNC_INFO;
00039 }
```

6.11.3 Documentation des fonctions membres

6.11.3.1 ouvrirAlbumPhotos()

```
void IHMAlbumPhoto::ouvrirAlbumPhotos (
             QVector< Photo > albumPhoto )
```

Ouvre une nouvelle fenetre contenant la liste des photos prises en cours de mission.

Définition à la ligne 41 du fichier ihmalbumphoto.cpp.

Références albumPhoto, layoutAlbumPhoto, selectionnerPhoto(), et signalMapper.

Référencé par IHMRov : :chargerPhotos().

```
00042 {
00043     if(albumPhoto.isEmpty())
00044     {
00045         QMessageBox::critical(this, "Erreur", "Listes photos vide !");
00046         return;
00047     }
00048
00049     QFont police("", 15, 50, false);
00050
00051     this->albumPhoto = albumPhoto;
00052     signalMapper = new QSignalMapper(this);
00053     int numeroPhoto = 0;
00054     for(QVector<Photo>::iterator it = albumPhoto.begin(); it !=
      albumPhoto.end(); ++it, numeroPhoto++)
00055     {
00056         QVBoxLayout *layoutPhoto = new QVBoxLayout;
00057         QHBoxLayout *layoutInformationsPhotos = new QHBoxLayout;
00058         QHBoxLayout *layoutPhotoAGArder = new QHBoxLayout;
00059
00060         QLabel *photo = new QLabel(this);
00061         photo->setPixmap((*it).image);
```

```
00062
00063         QLabel *dateHeure = new QLabel((*it).dateheure.toString(), this);
00064         QLabel *chemin = new QLabel((*it).cheminSauvegarde,this);
00065         QCheckBox *photoGarde = new QCheckBox(this);
00066         QWidget *information = new QWidget(this);
00067         QLabel *photoAGarder = new QLabel("Photo à garder :", this);
00068
00069         layoutPhotoAGArder->setAlignment(Qt::AlignRight);
00070         information->setFixedWidth((*it).image.width());
00071
00072         if(albumPhoto[numeroPhoto].aGarder)
00073             photoGarde->setChecked(true);
00074         else
00075             photoGarde->setChecked(false);
00076
00077         connect(photoGarde, SIGNAL(clicked()), signalMapper, SLOT(map()));
00078         signalMapper->setMapping(photoGarde, numeroPhoto);
00079
00080         layoutAlbumPhoto->addLayout(layoutPhoto);
00081         layoutPhoto->addWidget(information);
00082         information->setLayout(layoutInformationsPhotos);
00083         layoutPhoto->addWidget(photo);
00084         layoutInformationsPhotos->addWidget(dateHeure);
00085         layoutInformationsPhotos->addWidget(chemin);
00086         layoutInformationsPhotos->addLayout(layoutPhotoAGArder);
00087         layoutPhotoAGArder->addWidget(photoAGarder);
00088         layoutPhotoAGArder->addWidget(photoGarde);
00089
00090         dateHeure->setFont(police);
00091         chemin->setFont(police);
00092         photoAGarder->setFont(police);
00093         information->setStyleSheet("background-color: white");
00094         dateHeure->setStyleSheet("color: black");
00095         chemin->setStyleSheet("color: black");
00096         photoAGarder->setStyleSheet("color: black");
00097         photoGarde->setStyleSheet("color: black");
00098     }
00099     connect(signalMapper, SIGNAL(mapped(int)), this, SLOT(
      selectionnerPhoto(int)));
00100
00101     this->show();
00102 }
```

6.11.3.2 selectionnerPhoto

```
void IHMAlbumPhoto::selectionnerPhoto (
              int numero )  [slot]
```

Permet de selectionner la photo indiquer par le signalMapper.

Paramètres

numero	

Définition à la ligne 104 du fichier ihmalbumphoto.cpp.

Références albumPhoto, Campagne : :getAlbumPhoto(), IHMRov : :getCampagne(), imhRov, et Campagne : :modifierArchivePhoto().

Référencé par ouvrirAlbumPhotos().

```
00105 {
00106     if(numeroPhoto < albumPhoto.size())
00107     {
00108         imhRov->getCampagne()->modifierArchivePhoto(numeroPhoto);
00109         qDebug() << Q_FUNC_INFO << "numeroPhoto" << numeroPhoto << "A garder" <<
      imhRov->getCampagne()->getAlbumPhoto()[numeroPhoto].aGarder;
00110     }
00111 }
```

6.11.4 Documentation des données membres

6.11.4.1 albumPhoto

```
QVector<Photo> IHMAlbumPhoto::albumPhoto  [private]
```

Conteneur de photo.

Définition à la ligne 44 du fichier ihmalbumphoto.h.

Référencé par ouvrirAlbumPhotos(), et selectionnerPhoto().

6.11.4.2 imhRov

```
IHMRov* IHMAlbumPhoto::imhRov  [private]
```

Association avec l'IHMRov.

Définition à la ligne 45 du fichier ihmalbumphoto.h.

Référencé par selectionnerPhoto().

6.11.4.3 layoutAlbumPhoto

```
QVBoxLayout* IHMAlbumPhoto::layoutAlbumPhoto  [private]
```

Layout s'agrandissant selon l'ajout de nouvelle photos.

Définition à la ligne 41 du fichier ihmalbumphoto.h.

Référencé par IHMAlbumPhoto(), et ouvrirAlbumPhotos().

6.11.4.4 layoutPhotos

```
QHBoxLayout* IHMAlbumPhoto::layoutPhotos  [private]
```

Layout permettant d'accueillir les différentes photos.

Définition à la ligne 40 du fichier ihmalbumphoto.h.

Référencé par IHMAlbumPhoto().

6.11.4.5 photos

```
QWidget* IHMAlbumPhoto::photos  [private]
```

Emplacement permettant d'accueillir les différentes photos.

Définition à la ligne 39 du fichier ihmalbumphoto.h.

Référencé par IHMAlbumPhoto().

6.11.4.6 scrollArea

```
QScrollArea* IHMAlbumPhoto::scrollArea  [private]
```

Permet une defilement pour visualiser l'ensemble des photos prises durant la campagne.

Définition à la ligne 42 du fichier ihmalbumphoto.h.

Référencé par IHMAlbumPhoto().

6.11.4.7 signalMapper

```
QSignalMapper* IHMAlbumPhoto::signalMapper  [private]
```

Objet de type QSignalMapper, permet d'associer chaque photo de l'IHMAlbumPhoto à un signal.

Définition à la ligne 43 du fichier ihmalbumphoto.h.

Référencé par ouvrirAlbumPhotos().

La documentation de cette classe a été générée à partir des fichiers suivants :

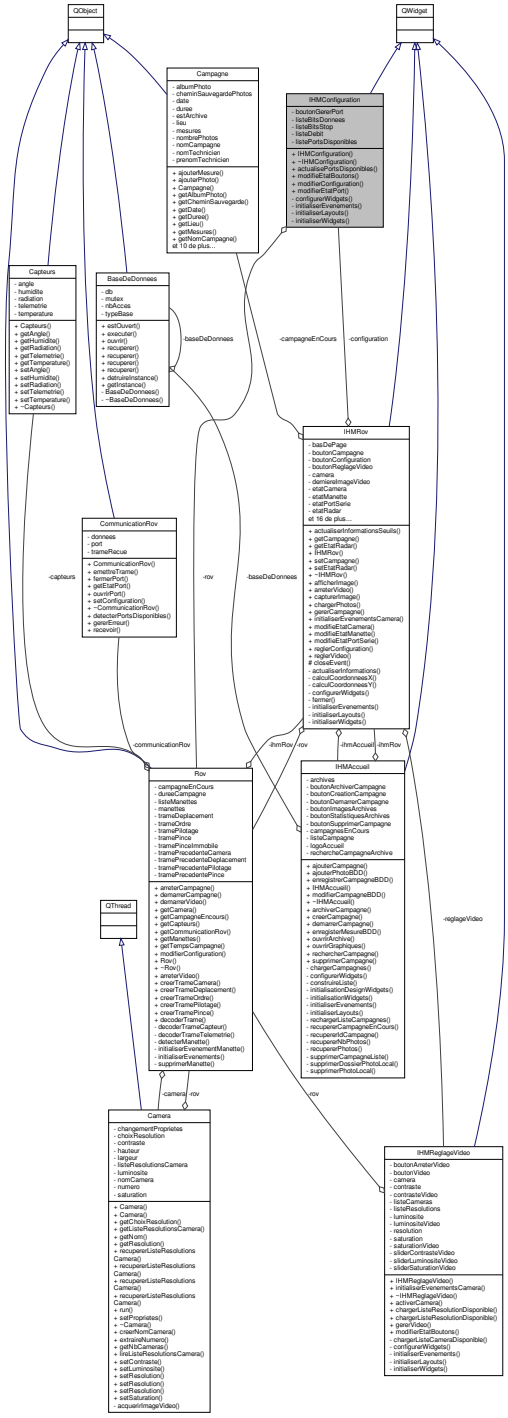
- ihmalbumphoto.h
- ihmalbumphoto.cpp

6.12 Référence de la classe IHMConfiguration

Class permettant de configurer la communication avec le rov.

```
#include "ihmconfiguration.h"
```

Graphe de collaboration de IHMConfiguration :



Connecteurs publics

— void actualisePortsDisponibles ()
Actualise la liste des ports disponible.
— void modifieEtatBoutons ()
Modifie l'état des boutons en fonction du port.
— void modifierConfiguration ()
Envoie au rov la nouvelle configuration de la communication.
— void modifierEtatPort ()
Modifie l'état du port.

Fonctions membres publiques

- IHMConfiguration (Rov *rov, QWidget *parent=nullptr)
  Constructeur de la classe IHMConfiguration.
- ~IHMConfiguration ()
  Destructeur de la classe IHMConfiguration.

Fonctions membres privées

- void configurerWidgets ()
  Configure l'état des widgets à la création de l'IHM.
- void initialiserEvenements ()
  Initialise les événements de l'IHM.
- void initialiserLayouts ()
  Initialise les layouts de l'IHM.
- void initialiserWidgets ()
  Initialise les widgets de l'IHM.

Attributs privés

- QPushButton * boutonGererPort
  Bouton permettant de gerer le port sélectionné
- QComboBox * listeBitsDonnees
  Liste permettant de configurer le nombre de bits de données de la communication.
- QComboBox * listeBitsStop
  Liste permettant de configurer le nombre de bits de stop de la communication.
- QComboBox * listeDebit
  Liste permettant de configurer le debit de la communication.
- QComboBox * listePortsDisponibles
  Liste des ports détéctés.
- Rov * rov
  Objet rov permettant de mofidier les reglage de la communication.

6.12.1 Description détaillée

Class permettant de configurer la communication avec le rov.

Définition à la ligne 20 du fichier ihmconfiguration.h.

6.12.2 Documentation des constructeurs et destructeur

6.12.2.1 IHMConfiguration()

```
IHMConfiguration::IHMConfiguration (
            Rov * rov,
            QWidget * parent = nullptr )
```

Constructeur de la classe IHMConfiguration.

Paramètres

rov	
parent	

Définition à la ligne 9 du fichier ihmconfiguration.cpp.

Références configurerWidgets(), Rov : :getCommunicationRov(), initialiserEvenements(), initialiserLayouts(), initialiserWidgets(), modifieEtatBoutons(), modifierConfiguration(), et CommunicationRov : :ouvrirPort().

```
00009                                                       : QWidget(parent),
      rov(rov)
00010 {
00011     qDebug() << Q_FUNC_INFO;
00012
00013     initialiserWidgets();
00014     configurerWidgets();
00015     initialiserLayouts();
00016     initialiserEvenements();
00017
00018     modifierConfiguration();
00019     rov->getCommunicationRov()->ouvrirPort();
00020     modifieEtatBoutons();
00021 }
```

6.12.2.2 ~IHMConfiguration()

```
IHMConfiguration::~IHMConfiguration ( )
```

Destructeur de la classe IHMConfiguration.

Définition à la ligne 23 du fichier ihmconfiguration.cpp.

```
00024 {
00025     qDebug() << Q_FUNC_INFO;
00026 }
```

6.12.3 Documentation des fonctions membres

6.12.3.1 actualisePortsDisponibles

```
void IHMConfiguration::actualisePortsDisponibles ( )  [slot]
```

Actualise la liste des ports disponible.

Définition à la ligne 139 du fichier ihmconfiguration.cpp.

Références CommunicationRov : :detecterPortsDisponibles(), et listePortsDisponibles.

Référencé par IHMRov : :reglerConfiguration().

```
00140 {
00141     listePortsDisponibles->clear();
00142     listePortsDisponibles->addItems(
      CommunicationRov::detecterPortsDisponibles());
00143     if(listePortsDisponibles->currentText() == "")
00144         listePortsDisponibles->addItem("Aucun port détécté");
00145 }
```

6.12.3.2 configurerWidgets()

```
void IHMConfiguration::configurerWidgets ( )  [private]
```

Configure l'état des widgets à la création de l'IHM.

Définition à la ligne 37 du fichier ihmconfiguration.cpp.

Références CommunicationRov : :detecterPortsDisponibles(), listeBitsDonnees, listeBitsStop, listeDebit, et listePortsDisponibles.

Référencé par IHMConfiguration().

```
00038 {
00039     listePortsDisponibles->addItems(
      CommunicationRov::detecterPortsDisponibles());
00040     if(listePortsDisponibles->currentText() == "")
00041         listePortsDisponibles->addItem("Aucun port détécté");
00042     listeDebit->addItem("9600");
00043     listeDebit->addItem("115200");
00044
00045     listeBitsDonnees->addItem("7");
00046     listeBitsDonnees->addItem("8");
00047     listeBitsDonnees->setCurrentIndex(1);
00048
00049     listeBitsStop->addItem("1");
00050     listeBitsStop->addItem("2");
00051 }
```

6.12.3.3 initialiserEvenements()

```
void IHMConfiguration::initialiserEvenements ( )  [private]
```

Initialise les événements de l'IHM.

Définition à la ligne 75 du fichier ihmconfiguration.cpp.

Références boutonGererPort, listeBitsDonnees, listeBitsStop, listeDebit, listePortsDisponibles, modifierConfiguration(), et modifier↩ EtatPort().

Référencé par IHMConfiguration().

```
00076 {
00077     connect(listePortsDisponibles, SIGNAL(currentIndexChanged(int)), this, SLOT(
      modifierConfiguration()));
00078     connect(listeDebit, SIGNAL(currentIndexChanged(int)), this, SLOT(
      modifierConfiguration()));
00079     connect(listeBitsDonnees, SIGNAL(currentIndexChanged(int)), this, SLOT(
      modifierConfiguration()));
00080     connect(listeBitsStop, SIGNAL(currentIndexChanged(int)), this, SLOT(
      modifierConfiguration()));
00081     connect(boutonGererPort, SIGNAL(clicked()), this, SLOT(
      modifierEtatPort()));
00082 }
```

6.12.3.4 initialiserLayouts()

```
void IHMConfiguration::initialiserLayouts ( )  [private]
```

Initialise les layouts de l'IHM.

Définition à la ligne 53 du fichier ihmconfiguration.cpp.

Références boutonGererPort, listeBitsDonnees, listeBitsStop, listeDebit, et listePortsDisponibles.

Référencé par IHMConfiguration().

```
00054 {
00055     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00056     QHBoxLayout *layoutInformation = new QHBoxLayout;
00057     QFormLayout *layoutConfiguration = new QFormLayout;
00058     QVBoxLayout *layoutCommande = new QVBoxLayout;
00059
00060     layoutCommande->setAlignment(Qt::AlignTop);
00061
00062     layoutPrincipal->addLayout(layoutInformation);
00063     layoutInformation->addLayout(layoutConfiguration);
00064     layoutInformation->addLayout(layoutCommande);
00065     layoutConfiguration->addRow("Port:", listePortsDisponibles);
00066     layoutConfiguration->addRow("Débit:", listeDebit);
00067     layoutConfiguration->addRow("Bits de données:", listeBitsDonnees);
00068     layoutConfiguration->addRow("Bits de stop:", listeBitsStop);
00069     layoutCommande->addWidget(boutonGererPort);
00070
00071     setLayout(layoutPrincipal);
00072     setStyleSheet("background:#202020;color:white;");
00073 }
```

6.12.3.5 initialiserWidgets()

```
void IHMConfiguration::initialiserWidgets ( )  [private]
```

Initialise les widgets de l'IHM.

Définition à la ligne 28 du fichier ihmconfiguration.cpp.

Références boutonGererPort, listeBitsDonnees, listeBitsStop, listeDebit, et listePortsDisponibles.

Référencé par IHMConfiguration().

```
00029 {
00030     listePortsDisponibles = new QComboBox(this);
00031     listeDebit = new QComboBox(this);
00032     listeBitsDonnees = new QComboBox(this);
00033     listeBitsStop = new QComboBox(this);
00034     boutonGererPort = new QPushButton("Fermer", this);
00035 }
```

6.12.3.6 modifieEtatBoutons

```
void IHMConfiguration::modifieEtatBoutons ( )  [slot]
```

Modifie l'état des boutons en fonction du port.

Définition à la ligne 84 du fichier ihmconfiguration.cpp.

Références boutonGererPort, Rov : :getCommunicationRov(), CommunicationRov : :getEtatPort(), listeBitsDonnees, listeBitsStop, listeDebit, listePortsDisponibles, et rov.

Référencé par IHMConfiguration(), et IHMRov : :modifieEtatPortSerie().

```
00085 {
00086     if(rov->getCommunicationRov()->getEtatPort())
00087     {
00088         listePortsDisponibles->setDisabled(true);
00089         listeDebit->setDisabled(true);
00090         listeBitsDonnees->setDisabled(true);
00091         listeBitsStop->setDisabled(true);
00092     }
00093     else
00094     {
00095         boutonGererPort->setText("Ouvrir");
00096         listePortsDisponibles->setEnabled(true);
00097         listeDebit->setEnabled(true);
00098         listeBitsDonnees->setEnabled(true);
00099         listeBitsStop->setEnabled(true);
00100     }
00101 }
```

6.12.3.7 modifierConfiguration

```
void IHMConfiguration::modifierConfiguration ( )  [slot]
```

Envoie au rov la nouvelle configuration de la communication.

Définition à la ligne 103 du fichier ihmconfiguration.cpp.

Références Configuration : :bitsDonnees, Configuration : :bitStop, Configuration : :debit, listeBitsDonnees, listeBitsStop, listeDebit, listePortsDisponibles, Rov : :modifierConfiguration(), Configuration : :port, et rov.

Référencé par IHMConfiguration(), et initialiserEvenements().

```
00104 {
00105     Configuration configuration;
00106
00107     configuration.port = listePortsDisponibles->currentText();
00108     configuration.debit = listeDebit->currentText().toInt();
00109     configuration.bitsDonnees = listeBitsDonnees->currentText().toInt();
00110     configuration.bitStop = listeBitsStop->currentText().toInt();
00111
00112     rov->modifierConfiguration(configuration);
00113 }
```

6.12.3.8 modifierEtatPort

```
void IHMConfiguration::modifierEtatPort ( )  [slot]
```

Modifie l'état du port.

Définition à la ligne 115 du fichier ihmconfiguration.cpp.

Références boutonGererPort, CommunicationRov : :fermerPort(), Rov : :getCommunicationRov(), listeBitsDonnees, listeBitsStop, listeDebit, listePortsDisponibles, CommunicationRov : :ouvrirPort(), et rov.

Référencé par initialiserEvenements().

```
00116 {
00117     if(boutonGererPort->text() == "Ouvrir")
00118     {
00119         if(rov->getCommunicationRov()->ouvrirPort())
00120         {
00121             boutonGererPort->setText("Fermer");
00122             listePortsDisponibles->setDisabled(true);
00123             listeDebit->setDisabled(true);
00124             listeBitsDonnees->setDisabled(true);
00125             listeBitsStop->setDisabled(true);
00126         }
00127     }
00128     else
00129     {
00130         rov->getCommunicationRov()->fermerPort();
00131         boutonGererPort->setText("Ouvrir");
00132         listePortsDisponibles->setEnabled(true);
00133         listeDebit->setEnabled(true);
00134         listeBitsDonnees->setEnabled(true);
00135         listeBitsStop->setEnabled(true);
00136     }
00137 }
```

6.12.4 Documentation des données membres

6.12.4.1 boutonGererPort

```
QPushButton* IHMConfiguration::boutonGererPort  [private]
```

Bouton permettant de gerer le port sélectionné

Définition à la ligne 29 du fichier ihmconfiguration.h.

Référencé par initialiserEvenements(), initialiserLayouts(), initialiserWidgets(), modifieEtatBoutons(), et modifierEtatPort().

6.12.4.2 listeBitsDonnees

```
QComboBox* IHMConfiguration::listeBitsDonnees  [private]
```

Liste permettant de configurer le nombre de bits de données de la communication.

Définition à la ligne 27 du fichier ihmconfiguration.h.

Référencé par configurerWidgets(), initialiserEvenements(), initialiserLayouts(), initialiserWidgets(), modifieEtatBoutons(), modifier↩Configuration(), et modifierEtatPort().

6.12.4.3 listeBitsStop

```
QComboBox* IHMConfiguration::listeBitsStop  [private]
```

Liste permettant de configurer le nombre de bits de stop de la communication.

Définition à la ligne 28 du fichier ihmconfiguration.h.

Référencé par configurerWidgets(), initialiserEvenements(), initialiserLayouts(), initialiserWidgets(), modifieEtatBoutons(), modifier↩Configuration(), et modifierEtatPort().

6.12.4.4 listeDebit

```
QComboBox* IHMConfiguration::listeDebit  [private]
```

Liste permettant de configurer le debit de la communication.

Définition à la ligne 26 du fichier ihmconfiguration.h.

Référencé par configurerWidgets(), initialiserEvenements(), initialiserLayouts(), initialiserWidgets(), modifieEtatBoutons(), modifier↩Configuration(), et modifierEtatPort().

6.12.4.5 listePortsDisponibles

```
QComboBox* IHMConfiguration::listePortsDisponibles  [private]
```

Liste des ports détéctés.

Définition à la ligne 25 du fichier ihmconfiguration.h.

Référencé par actualisePortsDisponibles(), configurerWidgets(), initialiserEvenements(), initialiserLayouts(), initialiserWidgets(), modifieEtatBoutons(), modifierConfiguration(), et modifierEtatPort().

6.12.4.6 rov

```
Rov* IHMConfiguration::rov  [private]
```

Objet rov permettant de mofidier les reglage de la communication.

Définition à la ligne 24 du fichier ihmconfiguration.h.

Référencé par modifieEtatBoutons(), modifierConfiguration(), et modifierEtatPort().

La documentation de cette classe a été générée à partir des fichiers suivants :

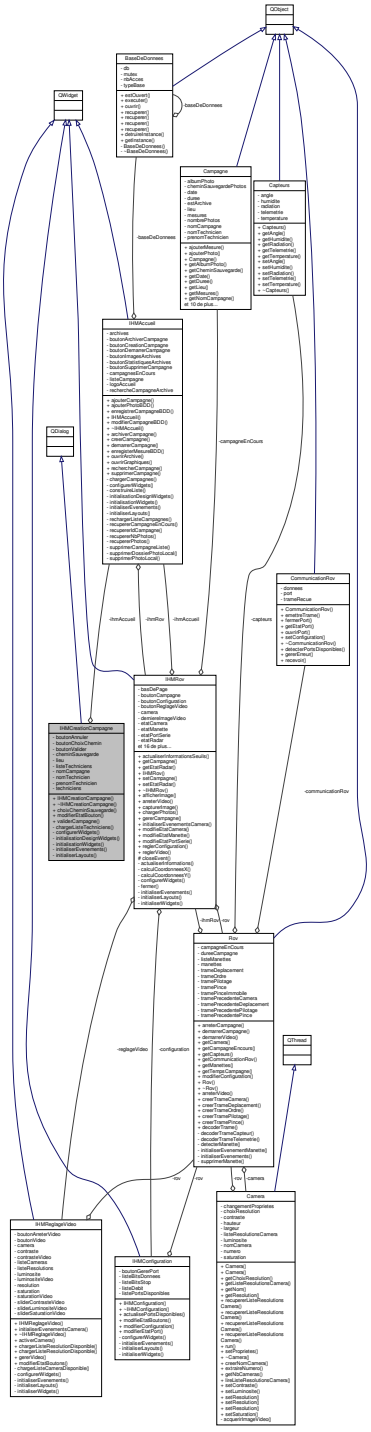
- ihmconfiguration.h
- ihmconfiguration.cpp

6.13 Référence de la classe IHMCreationCampagne

Class permettant de créer une nouvelle campagne.

`#include "ihmcreationcampagne.h"`

Graphe de collaboration de IHMCreationCampagne :



Connecteurs publics

— void choixCheminSauvegarde ()

Permet de choisir le chemin de sauvegarde des photos.

— void modifierEtatBouton (int index)
Modifie l'état des boutons de la boite de dialogue "création d'une nouvelle campagne" si un technicien connue est choisi les ligne permettant de rentrer un nouveau technicien deviennent non-éditables.

— void validerCampagne ()
Créer un nouvel objet Campagne et l'ajoute dans la liste des campagnes disponibles.

Fonctions membres publiques

— IHMCreationCampagne (IHMAccueil *ihmAccueil, QVector< QStringList > &listeTechniciens)
Constructeur de la classe IHMCreationCampagne.

— ~IHMCreationCampagne ()
Destructeur de la classe IHMCreationCampagne.

Fonctions membres privées

— void chargerListeTechniciens ()
Charge la liste des techniciens connus de la bdd dans la liste déroulante.

— void configurerWidgets ()
Configure les différents widgets.

— void initialisationDesignWidgets ()
Initialise le design des widgets de l'IHM.

— void initialisationWidgets ()
Initialise les widgets de l'IHM.

— void initialiserEvenements ()
Initialise les événements de l'IHM.

— void initialiserLayouts ()
Initialise les layouts de l'IMH.

Attributs privés

— QPushButton * boutonAnnuler
Bouton permettant d'annuler la création de la campagne.

— QPushButton * boutonChoixChemin
Bouton permettant de choisir le chemin de sauvegarde des photos.

— QPushButton * boutonValider
Bouton permettant de valider la création de la campagne.

— QLineEdit * cheminSauvegarde
Ligne editable permettant de choisir le chemin de sauvegarde des photos.

— IHMAccueil * ihmAccueil
Association avec la classe IHMAccueil.

— QLineEdit * lieu
Ligne editable permettant de rentrer le lieu de la campagne à créer.

— QVector< QStringList > listeTechniciens
Conteneur de liste des informations des techniciens présent dans la base de données.

— QLineEdit * nomCampagne
Ligne editable permettant de rentrer le nom de la campagne à créer.

— QLineEdit * nomTechnicien
Ligne editable permettant de rentrer le nom du technicien à créer.

— QLineEdit * prenomTechnicien
Ligne editable permettant de rentrer le prenom du technicien à créer.

— QComboBox * techniciens
Liste déroulantes contenant tous les technicien connus.

6.13.1 Description détaillée

Class permettant de créer une nouvelle campagne.

Définition à la ligne 22 du fichier ihmcreationcampagne.h.

6.13.2 Documentation des constructeurs et destructeur

6.13.2.1 IHMCreationCampagne()

```
IHMCreationCampagne::IHMCreationCampagne (
            IHMAccueil * ihmAccueil,
            QVector< QStringList > & listeTechniciens )  [explicit]
```

Constructeur de la classe IHMCreationCampagne.

Paramètres

ihmAccueil	
listeTechniciens	

Définition à la ligne 11 du fichier ihmcreationcampagne.cpp.

Références chargerListeTechniciens(), configurerWidgets(), initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), et initialiserLayouts().

```
00011                                                                                                                  :
      QDialog(ihmAccueil, Qt::Dialog), ihmAccueil(ihmAccueil),
      listeTechniciens(listeTechniciens)
00012 {
00013     qDebug() << Q_FUNC_INFO;
00014
00015     initialisationWidgets();
00016     initialisationDesignWidgets();
00017     initialiserEvenements();
00018     initialiserLayouts();
00019     configurerWidgets();
00020     chargerListeTechniciens();
00021 }
```

6.13.2.2 ~IHMCreationCampagne()

```
IHMCreationCampagne::~IHMCreationCampagne ( )
```

Destructeur de la classe IHMCreationCampagne.

Définition à la ligne 23 du fichier ihmcreationcampagne.cpp.

```
00024 {
00025     qDebug() << Q_FUNC_INFO;
00026 }
```

6.13.3 Documentation des fonctions membres

6.13.3.1 chargerListeTechniciens()

```
void IHMCreationCampagne::chargerListeTechniciens ( )  [private]
```

Charge la liste des techniciens connus de la bdd dans la liste déroulante.

Définition à la ligne 130 du fichier ihmcreationcampagne.cpp.

Références listeTechniciens, et techniciens.

Référencé par IHMCreationCampagne().

```
00131 {
00132     techniciens->addItem("Ajouter technicien");
00133 
00134     for(QVector<QStringList>::iterator it = listeTechniciens.begin(); it !=
      listeTechniciens.end(); ++it)
00135     {
00136         techniciens->addItem((*it).at(0) + " " + (*it).at(1));
00137     }
00138 }
```

6.13.3.2 choixCheminSauvegarde

```
void IHMCreationCampagne::choixCheminSauvegarde ( )  [slot]
```

Permet de choisir le chemin de sauvegarde des photos.

Définition à la ligne 177 du fichier ihmcreationcampagne.cpp.

Références cheminSauvegarde.

Référencé par initialiserEvenements().

```
00178 {
00179     cheminSauvegarde->setText(QFileDialog::getExistingDirectory(this, "Choix du chemin de
       sauvegarde des photos"));
00180 }
```

6.13.3.3 configurerWidgets()

```
void IHMCreationCampagne::configurerWidgets ( )  [private]
```

Configure les différents widgets.

Définition à la ligne 121 du fichier ihmcreationcampagne.cpp.

Références cheminSauvegarde, lieu, nomCampagne, nomTechnicien, et prenomTechnicien.

Référencé par IHMCreationCampagne().

```
00122 {
00123     nomCampagne->setTextMargins(10,0,0,0);
00124     nomTechnicien->setTextMargins(10,0,0,0);
00125     prenomTechnicien->setTextMargins(10,0,0,0);
00126     lieu->setTextMargins(10,0,0,0);
00127     cheminSauvegarde->setTextMargins(10,0,0,0);
00128 }
```

6.13.3.4 initialisationDesignWidgets()

```
void IHMCreationCampagne::initialisationDesignWidgets ( )  [private]
```

Initialise le design des widgets de l'IHM.

Définition à la ligne 41 du fichier ihmcreationcampagne.cpp.

Références boutonAnnuler, boutonChoixChemin, boutonValider, cheminSauvegarde, lieu, nomCampagne, nomTechnicien, prenomTechnicien, et techniciens.

Référencé par IHMCreationCampagne().

```
00042 {
00043     QFont policeBouton("", 13, 75, false);
00044     QFont policeText("", 13, 0, false);
00045
00046     nomCampagne->setFixedSize(194,30);
00047     nomCampagne->setFont(policeText);
00048     nomCampagne->setStyleSheet("QLineEdit {border-image: url(design/QLine_194x30.png)}" "
      QLineEdit:hover {border-image: url(design/QLine_194x30_survole.png)}");
00049
00050     nomTechnicien->setFixedSize(194,30);
00051     nomTechnicien->setFont(policeText);
00052     nomTechnicien->setStyleSheet("QLineEdit {border-image: url(design/QLine_194x30.png)}" "
      QLineEdit:hover {border-image: url(design/QLine_194x30_survole.png)}" "QLineEdit:disable {border-image:
       url(design/QLine_194x30_grise.png)}");
00053
00054     prenomTechnicien->setFixedSize(194,30);
00055     prenomTechnicien->setFont(policeText);
00056     prenomTechnicien->setStyleSheet("QLineEdit {border-image: url(design/QLine_194x30.png)}
      " "QLineEdit:hover {border-image: url(design/QLine_194x30_survole.png)}");
00057
00058     lieu->setFixedSize(194,30);
00059     lieu->setFont(policeText);
00060     lieu->setStyleSheet("QLineEdit {border-image: url(design/QLine_194x30.png)}" "QLineEdit:hover
       {border-image: url(design/QLine_194x30_survole.png)}");
00061
00062     cheminSauvegarde->setFixedSize(108,30);
00063     cheminSauvegarde->setFont(policeText);
00064     cheminSauvegarde->setStyleSheet("QLineEdit {border-image: url(design/QLine_108x30.png)}
      " "QLineEdit:hover {border-image: url(design/QLine_108x30_survole.png)}" );
00065
00066     boutonValider->setFixedSize(194,40);
00067     boutonValider->setFont(policeBouton);
00068     boutonValider->setStyleSheet("QPushButton {border-image: url(design/bouton_194x40.png)}" "
      QPushButton:hover {border-image: url(design/bouton_194x40_survole.png)}");
00069
00070     boutonAnnuler->setFixedSize(194,40);
00071     boutonAnnuler->setFont(policeBouton);
00072     boutonAnnuler->setStyleSheet("QPushButton {border-image: url(design/bouton_194x40.png)}" "
      QPushButton:hover {border-image: url(design/bouton_194x40_survole.png)}");
00073
00074     boutonChoixChemin->setFixedSize(80,30);
00075     boutonChoixChemin->setFont(policeText);
00076     boutonChoixChemin->setStyleSheet("QPushButton {border-image:
       url(design/bouton_80x30.png)}" "QPushButton:hover {border-image: url(design/bouton_80x30_survole.png)}");
00077
00078     techniciens->setMinimumSize(194,30);
00079     techniciens->setFont(policeText);
00080     techniciens->setStyleSheet("QComboBox {border-image: url(design/combobox_194x30.png)}" "
      QComboBox:hover {border-image: url(design/combobox_194x30_survole.png)}" "QComboBox::drop-down {border-image:
       url(rien.png)}" "QComboBox {padding: 0 0 0 10px}");
00081
00082     qDebug() << "nomCampagne" << nomCampagne->size();
00083
00084 }
```

6.13.3.5 initialisationWidgets()

```
void IHMCreationCampagne::initialisationWidgets ( )  [private]
```

Initialise les widgets de l'IHM.

Définition à la ligne 28 du fichier ihmcreationcampagne.cpp.

Références boutonAnnuler, boutonChoixChemin, boutonValider, cheminSauvegarde, lieu, nomCampagne, nomTechnicien, prenomTechnicien, et techniciens.

Référencé par IHMCreationCampagne().

```
00029 {
00030     nomCampagne = new QLineEdit(this);
00031     nomTechnicien = new QLineEdit(this);
00032     prenomTechnicien = new QLineEdit(this);
00033     lieu = new QLineEdit(this);
00034     cheminSauvegarde = new QLineEdit(this);
00035     boutonValider = new QPushButton("Valider", this);
00036     boutonAnnuler = new QPushButton("Annuler", this);
00037     boutonChoixChemin = new QPushButton("...", this);
00038     techniciens = new QComboBox(this);
00039 }
```

6.13.3.6 initialiserEvenements()

```
void IHMCreationCampagne::initialiserEvenements ( )  [private]
```

Initialise les événements de l'IHM.

Définition à la ligne 86 du fichier ihmcreationcampagne.cpp.

Références boutonAnnuler, boutonChoixChemin, boutonValider, choixCheminSauvegarde(), modifierEtatBouton(), techniciens, et validerCampagne().

Référencé par IHMCreationCampagne().

```
00087 {
00088     connect(boutonValider, SIGNAL(clicked()), this, SLOT(
      validerCampagne()));
00089     connect(boutonAnnuler, SIGNAL(clicked()), this, SLOT(close()));
00090     connect(boutonChoixChemin, SIGNAL(clicked()), this, SLOT(
      choixCheminSauvegarde()));
00091     connect(techniciens, SIGNAL(currentIndexChanged(int)), this, SLOT(
      modifierEtatBouton(int)));
00092 }
```

6.13.3.7 initialiserLayouts()

```
void IHMCreationCampagne::initialiserLayouts ( )  [private]
```

Initialise les layouts de l'IMH.

Définition à la ligne 94 du fichier ihmcreationcampagne.cpp.

Références boutonAnnuler, boutonChoixChemin, boutonValider, cheminSauvegarde, lieu, nomCampagne, nomTechnicien, prenomTechnicien, et techniciens.

Référencé par IHMCreationCampagne().

```
00095 {
00096     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00097     QFormLayout *layoutFormulaireCampagne = new QFormLayout;
00098     QHBoxLayout *layoutValidation = new QHBoxLayout;
00099     QHBoxLayout *layoutChoixChemin = new QHBoxLayout;
00100
00101     layoutPrincipal->addLayout(layoutFormulaireCampagne);
00102     layoutPrincipal->addLayout(layoutValidation);
00103     layoutChoixChemin->addWidget(cheminSauvegarde);
00104     layoutChoixChemin->addWidget(boutonChoixChemin);
00105
00106     layoutFormulaireCampagne->addRow("Nom campagne : ", nomCampagne);
00107     layoutFormulaireCampagne->addRow("Techniciens : ", techniciens);
00108     layoutFormulaireCampagne->addRow("Nom technicien : ", nomTechnicien);
00109     layoutFormulaireCampagne->addRow("Prenom technicien : ", prenomTechnicien);
00110     layoutFormulaireCampagne->addRow("Lieu campagne : ", lieu);
00111     layoutFormulaireCampagne->addRow("Chemin sauvegarde photos : ", layoutChoixChemin);
00112
00113     layoutValidation->addWidget(boutonValider);
00114     layoutValidation->addWidget(boutonAnnuler);
00115
00116     layoutValidation->setAlignment(Qt::AlignBottom);
00117
00118     setLayout(layoutPrincipal);
00119 }
```

6.13.3.8 modifierEtatBouton

```
void IHMCreationCampagne::modifierEtatBouton (
            int index )  [slot]
```

Modifie l'état des boutons de la boite de dialogue "création d'une nouvelle campagne" si un technicien connue est choisi les ligne permettant de rentrer un nouveau technicien deviennent non-éditables.

Paramètres

index	

Définition à la ligne 155 du fichier ihmcreationcampagne.cpp.

Références listeTechniciens, nomTechnicien, et prenomTechnicien.

Référencé par initialiserEvenements().

```
00156 {
00157     if(index > 0)
00158     {
00159         nomTechnicien->setDisabled(true);
00160         nomTechnicien->setText(listeTechniciens[index - 1].at(0));
00161         nomTechnicien->setStyleSheet("QLineEdit {border-image:
      url(design/QLine_194x30_grise.png)}");
```

```
00162         prenomTechnicien->setDisabled(true);
00163         prenomTechnicien->setText(listeTechniciens[index - 1].at(1));
00164         prenomTechnicien->setStyleSheet("QLineEdit {border-image:
       url(design/QLine_194x30_grise.png)}");
00165     }
00166     else
00167     {
00168         nomTechnicien->clear();
00169         prenomTechnicien->clear();
00170         nomTechnicien->setEnabled(true);
00171         nomTechnicien->setStyleSheet("QLineEdit {border-image: url(design/QLine_194x30.png)}")
    ;
00172         prenomTechnicien->setEnabled(true);
00173         prenomTechnicien->setStyleSheet("QLineEdit {border-image:
       url(design/QLine_194x30.png)}");
00174     }
00175 }
```

6.13.3.9 validerCampagne

```
void IHMCreationCampagne::validerCampagne ( )  [slot]
```

Créer un nouvel objet Campagne et l'ajoute dans la liste des campagnes disponibles.

Définition à la ligne 140 du fichier ihmcreationcampagne.cpp.

Références IHMAccueil : :ajouterCampagne(), cheminSauvegarde, IHMAccueil : :enregistrerCampagneBDD(), ihmAccueil, lieu, nomCampagne, nomTechnicien, prenomTechnicien, et Campagne : :setCheminSauvegarde().

Référencé par initialiserEvenements().

```
00141 {
00142     if(nomCampagne->text().isEmpty() || nomTechnicien->text().isEmpty() ||
      prenomTechnicien->text().isEmpty() || lieu->text().isEmpty() ||
      cheminSauvegarde->text().isEmpty())
00143     {
00144         QMessageBox::critical(nullptr, "Création campagne", "Informations incomplètes !");
00145         return;
00146     }
00147
00148     Campagne *campagne = new Campagne(nomCampagne->text(),
      lieu->text(), nomTechnicien->text(), prenomTechnicien->text(),
      QDateTime::currentDateTime(), this);
00149     campagne->setCheminSauvegarde(cheminSauvegarde->text());
00150     ihmAccueil->ajouterCampagne(campagne, true);
00151     ihmAccueil->enregistrerCampagneBDD(campagne);
00152     close();
00153 }
```

6.13.4 Documentation des données membres

6.13.4.1 boutonAnnuler

```
QPushButton* IHMCreationCampagne::boutonAnnuler  [private]
```

Bouton permettant d'annuler la création de la campagne.

Définition à la ligne 33 du fichier ihmcreationcampagne.h.

Référencé par initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), et initialiserLayouts().

6.13.4.2 boutonChoixChemin

```
QPushButton* IHMCreationCampagne::boutonChoixChemin  [private]
```

Bouton permettant de choisir le chemin de sauvegarde des photos.

Définition à la ligne 34 du fichier ihmcreationcampagne.h.

Référencé par initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), et initialiserLayouts().

6.13.4.3 boutonValider

```
QPushButton* IHMCreationCampagne::boutonValider  [private]
```

Bouton permettant de valider la création de la campagne.

Définition à la ligne 32 du fichier ihmcreationcampagne.h.

Référencé par initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), et initialiserLayouts().

6.13.4.4 cheminSauvegarde

```
QLineEdit* IHMCreationCampagne::cheminSauvegarde  [private]
```

Ligne editable permettant de choisir le chemin de sauvegarde des photos.

Définition à la ligne 31 du fichier ihmcreationcampagne.h.

Référencé par choixCheminSauvegarde(), configurerWidgets(), initialisationDesignWidgets(), initialisationWidgets(), initialiser↩Layouts(), et validerCampagne().

6.13.4.5 ihmAccueil

```
IHMAccueil* IHMCreationCampagne::ihmAccueil  [private]
```

Association avec la classe IHMAccueil.

Définition à la ligne 26 du fichier ihmcreationcampagne.h.

Référencé par validerCampagne().

6.13.4.6 lieu

```
QLineEdit* IHMCreationCampagne::lieu  [private]
```

Ligne editable permettant de rentrer le lieu de la campagne à créer.

Définition à la ligne 30 du fichier ihmcreationcampagne.h.

Référencé par configurerWidgets(), initialisationDesignWidgets(), initialisationWidgets(), initialiserLayouts(), et validerCampagne().

6.13.4.7 listeTechniciens

```
QVector<QStringList> IHMCreationCampagne::listeTechniciens  [private]
```

Conteneur de liste des informations des techniciens présent dans la base de données.

Définition à la ligne 36 du fichier ihmcreationcampagne.h.

Référencé par chargerListeTechniciens(), et modifierEtatBouton().

6.13.4.8 nomCampagne

```
QLineEdit* IHMCreationCampagne::nomCampagne  [private]
```

Ligne editable permettant de rentrer le nom de la campagne à créer.

Définition à la ligne 27 du fichier ihmcreationcampagne.h.

Référencé par configurerWidgets(), initialisationDesignWidgets(), initialisationWidgets(), initialiserLayouts(), et validerCampagne().

6.13.4.9 nomTechnicien

```
QLineEdit* IHMCreationCampagne::nomTechnicien  [private]
```

Ligne editable permettant de rentrer le nom du technicien à créer.

Définition à la ligne 28 du fichier ihmcreationcampagne.h.

Référencé par configurerWidgets(), initialisationDesignWidgets(), initialisationWidgets(), initialiserLayouts(), modifierEtatBouton(), et validerCampagne().

6.13.4.10 prenomTechnicien

```
QLineEdit* IHMCreationCampagne::prenomTechnicien  [private]
```

Ligne editable permettant de rentrer le prenom du technicien à créer.

Définition à la ligne 29 du fichier ihmcreationcampagne.h.

Référencé par configurerWidgets(), initialisationDesignWidgets(), initialisationWidgets(), initialiserLayouts(), modifierEtatBouton(), et validerCampagne().

6.13.4.11 techniciens

```
QComboBox* IHMCreationCampagne::techniciens  [private]
```

Liste déroulantes contenant tous les technicien connus.

Définition à la ligne 35 du fichier ihmcreationcampagne.h.

Référencé par chargerListeTechniciens(), initialisationDesignWidgets(), initialisationWidgets(), initialiserEvenements(), et initialiserLayouts().

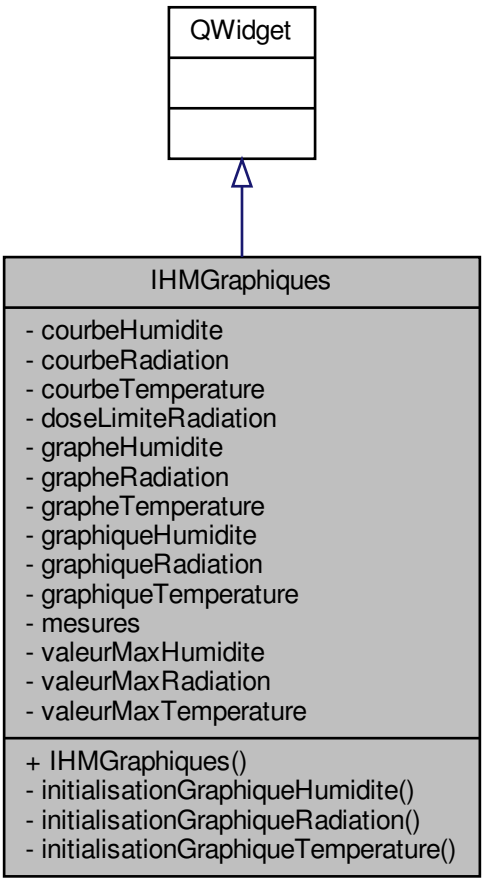
La documentation de cette classe a été générée à partir des fichiers suivants :
- ihmcreationcampagne.h
- ihmcreationcampagne.cpp

6.14 Référence de la classe IHMGraphiques

Class permettant de visualiser les graphiques des campagnes archivés.

```
#include "ihmgraphiques.h"
```

Graphe de collaboration de IHMGraphiques :



Fonctions membres publiques

— IHMGraphiques (QVector< QStringList > mesures, QWidget *parent=nullptr)
Constructeur de la classe IHMGraphiques.

Fonctions membres privées

— void initialisationGraphiqueHumidite ()
— void initialisationGraphiqueRadiation ()
— void initialisationGraphiqueTemperature ()

Attributs privés

- QLineSeries * courbeHumidite
  Les données d'humidite sous forme de courbe.
- QLineSeries * courbeRadiation
  Les données de radiation sous forme de courbe.
- QLineSeries * courbeTemperature
  Les données de temperature sous forme de courbe.
- QLineSeries * doseLimiteRadiation
  Le sueil de radiation acceptable.
- QChart * grapheHumidite
  la représentation du graphe humidite
- QChart * grapheRadiation
  la représentation du graphe radiation
- QChart * grapheTemperature
  la représentation du graphe temperature
- QChartView * graphiqueHumidite
  widget pour afficher le graphe humidite
- QChartView * graphiqueRadiation
  widget pour afficher le graphe radiation
- QChartView * graphiqueTemperature
  widget pour afficher le graphe temperature
- QVector< QStringList > mesures
  Conteneur des mesures de la base de données pour une campagne donnée.
- float valeurMaxHumidite
  La valeur max de l'humidité du graphique Humidité
- float valeurMaxRadiation
  La valeur max de la radiation du graphique Radiation.
- float valeurMaxTemperature
  La valeur max de la température du graphique Température.

6.14.1 Description détaillée

Class permettant de visualiser les graphiques des campagnes archivés.

Définition à la ligne 20 du fichier ihmgraphiques.h.

6.14.2 Documentation des constructeurs et destructeur

6.14.2.1 IHMGraphiques()

```
IHMGraphiques::IHMGraphiques (
            QVector< QStringList > mesures,
            QWidget * parent = nullptr )
```

Constructeur de la classe IHMGraphiques.

Paramètres

mesures	
parent	

Définition à la ligne 9 du fichier ihmgraphiques.cpp.

Références graphiqueHumidite, graphiqueRadiation, graphiqueTemperature, initialisationGraphiqueHumidite(), initialisationGraphiqueRadiation(), et initialisationGraphiqueTemperature().

```
00009                                                                    :
      QWidget(parent), mesures(mesures), valeurMaxRadiation(0.0),
      valeurMaxTemperature(0.0), valeurMaxHumidite(0.0)
00010 {
00011     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00012     QHBoxLayout *layoutGrapheRadiation = new QHBoxLayout;
00013     QHBoxLayout *layoutGrapheHumidite = new QHBoxLayout;
00014     QHBoxLayout *layoutGrapheTemperature = new QHBoxLayout;
00015
00016     initialisationGraphiqueRadiation();
00017     initialisationGraphiqueHumidite();
00018     initialisationGraphiqueTemperature();
00019
00020     resize(640, 480);
00021     setStyleSheet("background:#202020");
00022
00023     layoutPrincipal->addLayout(layoutGrapheRadiation);
00024     layoutPrincipal->addLayout(layoutGrapheHumidite);
00025     layoutPrincipal->addLayout(layoutGrapheTemperature);
00026     layoutGrapheRadiation->addWidget(graphiqueRadiation);
00027     layoutGrapheHumidite->addWidget(graphiqueHumidite);
00028     layoutGrapheTemperature->addWidget(graphiqueTemperature);
00029     setLayout(layoutPrincipal);
00030     showMaximized();
00031 }
```

6.14.3 Documentation des fonctions membres

6.14.3.1 initialisationGraphiqueHumidite()

```
void IHMGraphiques::initialisationGraphiqueHumidite ( )  [private]
```

Définition à la ligne 101 du fichier ihmgraphiques.cpp.

Références courbeHumidite, ESPACE_LISIBILITE, grapheHumidite, graphiqueHumidite, mesures, et valeurMaxHumidite.

Référencé par IHMGraphiques().

```
00102 {
00103     courbeHumidite = new QLineSeries();
00104
00105     for(QVector<QStringList>::iterator it = mesures.begin(); it !=
      mesures.end(); it++)
00106     {
00107         courbeHumidite->append(QDateTime::fromString((*it).at(0)).toMSecsSinceEpoch(), (*it).
      at(3).toFloat());
00108         if((*it).at(1).toFloat() > valeurMaxHumidite)
00109             valeurMaxHumidite = (*it).at(3).toFloat();
00110     }
00111
00112     courbeHumidite->setName(QString::fromUtf8("<font color=\"#FFFFFF\">Humidité</font>"));
00113     QPen pen;
00114     pen.setColor(QColor(Qt::darkGreen));
00115     pen.setWidth(2);
00116     courbeHumidite->setPen(pen);
00117
00118     grapheHumidite = new QChart();
00119     grapheHumidite->setBackgroundVisible(false);
00120     grapheHumidite->addSeries(courbeHumidite);
00121     grapheHumidite->setBackgroundBrush(QColor(0xFFFFFF));
00122
00123     QDateTimeAxis *axeXHumidite = new QDateTimeAxis;
00124     axeXHumidite->setTickCount(10);
00125     axeXHumidite->setFormat("hh:mm:ss");
00126     axeXHumidite->setTitleText("Heure");
00127     axeXHumidite->setLabelsColor(0xFFFFFF);
00128     axeXHumidite->setTitleBrush(QColor(0xFFFFFF));
00129     axeXHumidite->setMin(QDateTime::fromString(mesures[0].at(0)));
00130     axeXHumidite->setMax(QDateTime::fromString(mesures[mesures.size()-1].at(0)));
00131     grapheHumidite->addAxis(axeXHumidite, Qt::AlignBottom);
00132     courbeHumidite->attachAxis(axeXHumidite);
00133
00134     QValueAxis *axeYHumidite = new QValueAxis;
00135     axeYHumidite->setRange(0, 100);
00136     axeYHumidite->setLabelFormat("%d");
00137     axeYHumidite->setTitleText(QString::fromUtf8("Humidité en %"));
00138     axeYHumidite->setLabelsColor(0xFFFFFF);
```

```
00139     axeYHumidite->setTitleBrush(QColor(0xFFFFFF));
00140     grapheHumidite->addAxis(axeYHumidite, Qt::AlignLeft);
00141     courbeHumidite->setPointsVisible(true);
00142     courbeHumidite->setPointLabelsFormat("@yPoint"
      ESPACE_LISIBILITE);
00143     courbeHumidite->setPointLabelsVisible(true);
00144     courbeHumidite->attachAxis(axeYHumidite);
00145
00146     courbeHumidite->setPointLabelsColor(0xFFFFFF);
00147
00148     graphiqueHumidite = new QChartView(grapheHumidite);
00149     graphiqueHumidite->setRenderHint(QPainter::Antialiasing);
00150 }
```

6.14.3.2 initialisationGraphiqueRadiation()

```
void IHMGraphiques::initialisationGraphiqueRadiation ( )  [private]
```

Définition à la ligne 33 du fichier ihmgraphiques.cpp.

Références courbeRadiation, doseLimiteRadiation, ESPACE_LISIBILITE, grapheRadiation, graphiqueRadiation, mesures, et valeurMaxRadiation.

Référencé par IHMGraphiques().

```
00034 {
00035     courbeRadiation = new QLineSeries();
00036     doseLimiteRadiation = new QLineSeries();
00037
00038     for(QVector<QStringList>::iterator it = mesures.begin(); it !=
      mesures.end(); it++)
00039     {
00040         courbeRadiation->append(QDateTime::fromString((*it).at(0)).toMSecsSinceEpoch(),
      (*it).at(1).toFloat());
00041         if((*it).at(1).toFloat() > valeurMaxRadiation)
00042             valeurMaxRadiation = (*it).at(1).toFloat();
00043     }
00044
00045     doseLimiteRadiation->append(QDateTime::fromString(mesures[0].at(0)).
      toMSecsSinceEpoch(), 0.1);
00046     doseLimiteRadiation->append(QDateTime::fromString(mesures[
      mesures.size() - 1].at(0)).toMSecsSinceEpoch(), 0.1);
00047
00048     courbeRadiation->setName(QString::fromUtf8("<font color=\"#FFFFFF\">Radiation</font>"));
00049     doseLimiteRadiation->setName(QString::fromUtf8("<font color=\"#FFFFFF\">Dose
       limite</font>"));
00050     QPen pen;
00051     pen.setColor(QColor(Qt::darkGreen));
00052     pen.setWidth(2);
00053     courbeRadiation->setPen(pen);
00054     pen.setColor(QColor(Qt::darkRed));
00055     pen.setStyle(Qt::DashLine);
00056     doseLimiteRadiation->setPen(pen);
00057
00058     grapheRadiation = new QChart();
00059     grapheRadiation->setBackgroundVisible(false);
00060     grapheRadiation->addSeries(courbeRadiation);
00061     grapheRadiation->addSeries(doseLimiteRadiation);
00062     grapheRadiation->setBackgroundBrush(QColor(0xFFFFFF));
00063
00064     QDateTimeAxis *axeXRadiation = new QDateTimeAxis;
00065     axeXRadiation->setTickCount(10);
00066     axeXRadiation->setFormat("hh:mm:ss");
00067     axeXRadiation->setTitleText("Heure");
00068     axeXRadiation->setLabelsColor(0xFFFFFF);
00069     axeXRadiation->setTitleBrush(QColor(0xFFFFFF));
00070     axeXRadiation->setMin(QDateTime::fromString(mesures[0].at(0)));
00071     axeXRadiation->setMax(QDateTime::fromString(mesures[mesures.size()-1].at(0)));
00072     grapheRadiation->addAxis(axeXRadiation, Qt::AlignBottom);
00073     courbeRadiation->attachAxis(axeXRadiation);
00074     doseLimiteRadiation->attachAxis(axeXRadiation);
00075
00076     QValueAxis *axeYRadiation = new QValueAxis;
00077     axeYRadiation->setRange(0, int(valeurMaxRadiation) + 0.2);
00078     axeYRadiation->setLabelFormat("%.2f");
00079     axeYRadiation->setTitleText(QString::fromUtf8("Radiation en µsV/h"));
00080     axeYRadiation->setLabelsColor(0xFFFFFF);
00081     axeYRadiation->setTitleBrush(QColor(0xFFFFFF));
00082     grapheRadiation->addAxis(axeYRadiation, Qt::AlignLeft);
```

```
00083
00084     doseLimiteRadiation->setPointsVisible(true);
00085     doseLimiteRadiation->setPointLabelsFormat("@yPoint"
      ESPACE_LISIBILITE);
00086     doseLimiteRadiation->setPointLabelsVisible(true);
00087     doseLimiteRadiation->attachAxis(axeYRadiation);
00088
00089     courbeRadiation->setPointsVisible(true);
00090     courbeRadiation->setPointLabelsFormat("@yPoint"
      ESPACE_LISIBILITE);
00091     courbeRadiation->setPointLabelsVisible(true);
00092     courbeRadiation->attachAxis(axeYRadiation);
00093
00094     courbeRadiation->setPointLabelsColor(0xFFFFFF);
00095     doseLimiteRadiation->setPointLabelsColor(0xFFFFFF);
00096
00097     graphiqueRadiation = new QChartView(grapheRadiation);
00098     graphiqueRadiation->setRenderHint(QPainter::Antialiasing);
00099 }
```

6.14.3.3 initialisationGraphiqueTemperature()

```
void IHMGraphiques::initialisationGraphiqueTemperature ( )  [private]
```

Définition à la ligne 152 du fichier ihmgraphiques.cpp.

Références courbeTemperature, ESPACE_LISIBILITE, grapheTemperature, graphiqueTemperature, mesures, et valeurMax↩Temperature.

Référencé par IHMGraphiques().

```
00153 {
00154     courbeTemperature = new QLineSeries();
00155
00156     for(QVector<QStringList>::iterator it = mesures.begin(); it !=
      mesures.end(); it++)
00157     {
00158         courbeTemperature->append(QDateTime::fromString((*it).at(0)).toMSecsSinceEpoch(),
      (*it).at(2).toFloat());
00159         if((*it).at(2).toFloat() > valeurMaxTemperature)
00160             valeurMaxTemperature = (*it).at(2).toFloat();
00161     }
00162
00163     courbeTemperature->setName(QString::fromUtf8("<font color=\"#FFFFFF\"
      >Température</font>"));
00164     QPen pen;
00165     pen.setColor(QColor(Qt::darkGreen));
00166     pen.setWidth(2);
00167     courbeTemperature->setPen(pen);
00168
00169     grapheTemperature = new QChart();
00170     grapheTemperature->setBackgroundVisible(false);
00171     grapheTemperature->addSeries(courbeTemperature);
00172     grapheTemperature->setBackgroundBrush(QColor(0xFFFFFF));
00173
00174     QDateTimeAxis *axeXTemperature = new QDateTimeAxis;
00175     axeXTemperature->setTickCount(10);
00176     axeXTemperature->setFormat("hh:mm:ss");
00177     axeXTemperature->setTitleText("Heure");
00178     axeXTemperature->setLabelsColor(0xFFFFFF);
00179     axeXTemperature->setTitleBrush(QColor(0xFFFFFF));
00180     axeXTemperature->setMin(QDateTime::fromString(mesures[0].at(0)));
00181     axeXTemperature->setMax(QDateTime::fromString(mesures[mesures.size()-1].at(0)));
00182     grapheTemperature->addAxis(axeXTemperature, Qt::AlignBottom);
00183     courbeTemperature->attachAxis(axeXTemperature);
00184
00185     QValueAxis *axeYTemperature = new QValueAxis;
00186     axeYTemperature->setRange(0, int(valeurMaxTemperature) + 5);
00187     axeYTemperature->setLabelFormat("%.1f");
00188     axeYTemperature->setTitleText(QString::fromUtf8("Temperature en °C"));
00189     axeYTemperature->setLabelsColor(0xFFFFFF);
00190     axeYTemperature->setTitleBrush(QColor(0xFFFFFF));
00191     grapheTemperature->addAxis(axeYTemperature, Qt::AlignLeft);
00192     courbeTemperature->setPointsVisible(true);
00193     courbeTemperature->setPointLabelsFormat("@yPoint"
      ESPACE_LISIBILITE);
00194     courbeTemperature->setPointLabelsVisible(true);
00195     courbeTemperature->attachAxis(axeYTemperature);
00196
00197     courbeTemperature->setPointLabelsColor(0xFFFFFF);
00198
00199     graphiqueTemperature = new QChartView(grapheTemperature);
00200     graphiqueTemperature->setRenderHint(QPainter::Antialiasing);
00201 }
```

6.14.4 Documentation des données membres

6.14.4.1 courbeHumidite

```
QLineSeries* IHMGraphiques::courbeHumidite  [private]
```

Les données d'humidite sous forme de courbe.

Définition à la ligne 32 du fichier ihmgraphiques.h.

Référencé par initialisationGraphiqueHumidite().

6.14.4.2 courbeRadiation

```
QLineSeries* IHMGraphiques::courbeRadiation  [private]
```

Les données de radiation sous forme de courbe.

Définition à la ligne 27 du fichier ihmgraphiques.h.

Référencé par initialisationGraphiqueRadiation().

6.14.4.3 courbeTemperature

```
QLineSeries* IHMGraphiques::courbeTemperature  [private]
```

Les données de temperature sous forme de courbe.

Définition à la ligne 36 du fichier ihmgraphiques.h.

Référencé par initialisationGraphiqueTemperature().

6.14.4.4 doseLimiteRadiation

```
QLineSeries* IHMGraphiques::doseLimiteRadiation  [private]
```

Le sueil de radiation acceptable.

Définition à la ligne 28 du fichier ihmgraphiques.h.

Référencé par initialisationGraphiqueRadiation().

6.14.4.5 grapheHumidite

```
QChart* IHMGraphiques::grapheHumidite  [private]
```

la représentation du graphe humidite

Définition à la ligne 31 du fichier ihmgraphiques.h.

Référencé par initialisationGraphiqueHumidite().

6.14.4.6 grapheRadiation

```
QChart* IHMGraphiques::grapheRadiation  [private]
```

la représentation du graphe radiation

Définition à la ligne 26 du fichier ihmgraphiques.h.

Référencé par initialisationGraphiqueRadiation().

6.14.4.7 grapheTemperature

```
QChart* IHMGraphiques::grapheTemperature  [private]
```

la représentation du graphe temperature

Définition à la ligne 35 du fichier ihmgraphiques.h.

Référencé par initialisationGraphiqueTemperature().

6.14.4.8 graphiqueHumidite

```
QChartView* IHMGraphiques::graphiqueHumidite  [private]
```

widget pour afficher le graphe humidite

Définition à la ligne 30 du fichier ihmgraphiques.h.

Référencé par IHMGraphiques(), et initialisationGraphiqueHumidite().

6.14.4.9 graphiqueRadiation

```
QChartView* IHMGraphiques::graphiqueRadiation  [private]
```

widget pour afficher le graphe radiation

Définition à la ligne 25 du fichier ihmgraphiques.h.

Référencé par IHMGraphiques(), et initialisationGraphiqueRadiation().

6.14.4.10 graphiqueTemperature

```
QChartView* IHMGraphiques::graphiqueTemperature  [private]
```

widget pour afficher le graphe temperature

Définition à la ligne 34 du fichier ihmgraphiques.h.

Référencé par IHMGraphiques(), et initialisationGraphiqueTemperature().

6.14.4.11 mesures

```
QVector<QStringList> IHMGraphiques::mesures  [private]
```

Conteneur des mesures de la base de données pour une campagne donnée.

Définition à la ligne 24 du fichier ihmgraphiques.h.

Référencé par initialisationGraphiqueHumidite(), initialisationGraphiqueRadiation(), et initialisationGraphiqueTemperature().

6.14.4.12 valeurMaxHumidite

```
float IHMGraphiques::valeurMaxHumidite  [private]
```

La valeur max de l'humidité du graphique Humidité

Définition à la ligne 40 du fichier ihmgraphiques.h.

Référencé par initialisationGraphiqueHumidite().

6.14.4.13 valeurMaxRadiation

```
float IHMGraphiques::valeurMaxRadiation  [private]
```

La valeur max de la radiation du graphique Radiation.

Définition à la ligne 38 du fichier ihmgraphiques.h.

Référencé par initialisationGraphiqueRadiation().

6.14.4.14 valeurMaxTemperature

```
float IHMGraphiques::valeurMaxTemperature  [private]
```

La valeur max de la température du graphique Température.

Définition à la ligne 39 du fichier ihmgraphiques.h.

Référencé par initialisationGraphiqueTemperature().

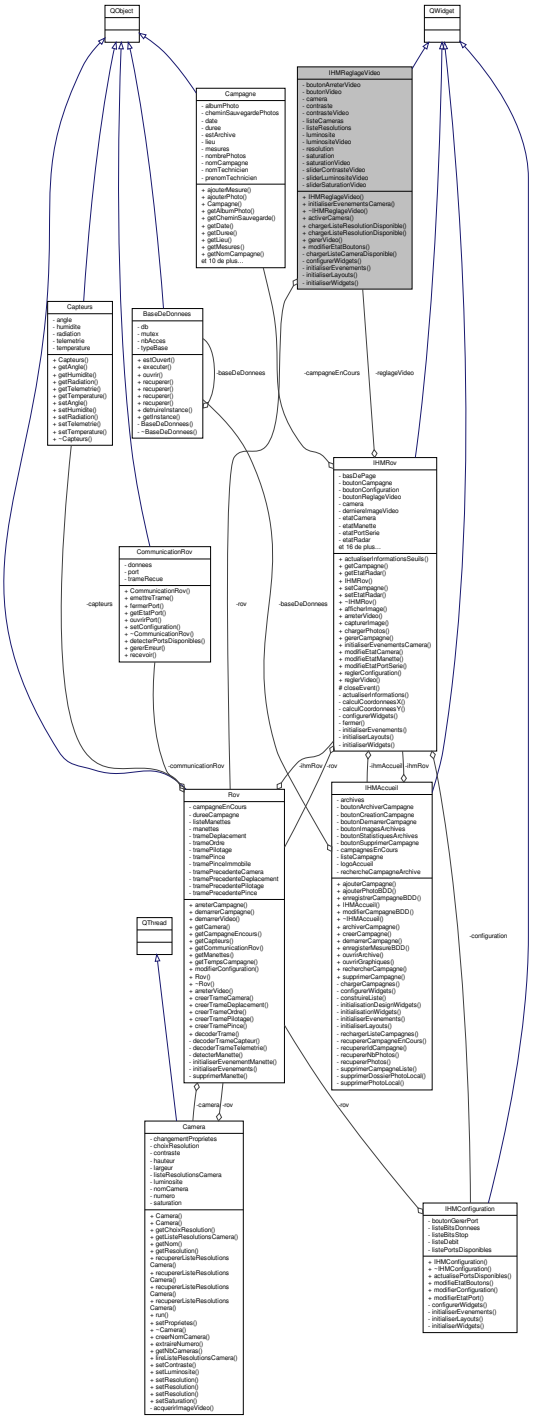
La documentation de cette classe a été générée à partir des fichiers suivants :
- ihmgraphiques.h
- ihmgraphiques.cpp

6.15 Référence de la classe IHMReglageVideo

Classe permettant de regler l'affichage du flux video.

`#include "ihmReglageVideo.h"`

Graphe de collaboration de IHMReglageVideo :



Connecteurs publics

— void activerCamera ()

Active la caméra.
- void chargerListeResolutionDisponible (int index)
  Charge les résolutions pour une caméra sélectionnée.
- void chargerListeResolutionDisponible (QString nom)
  Charge les résolutions pour une caméra sélectionnée.
- void gererVideo ()
  Modifie l'etat de la vidéo en fonction de l'état actuel.
- void modifierEtatBoutons ()
  Modifie l'etat des boutons lors du démarrage du flux vidéo.

Fonctions membres publiques

- IHMReglageVideo (Rov *rov, QWidget *parent=nullptr)
  Constructeur de la classe ReglageVideo.
- void initialiserEvenementsCamera ()
  Initialise les événements liés à la caméra.
- ~IHMReglageVideo ()
  Destructeur de la classe ReglageVideo.

Fonctions membres privées

- void chargerListeCameraDisponible ()
  Charge la liste des caméras disponibles dans la liste déroulante.
- void configurerWidgets ()
  Configure l'etat des widgets à la création de l'IHM.
- void initialiserEvenements ()
  Initialise les événements de l'IHM.
- void initialiserLayouts ()
  Initialise les layout de l'IHM.
- void initialiserWidgets ()
  Initialise les widgets de l'IHM.

Attributs privés

- QPushButton * boutonArreterVideo
  Bouton permettant d'arrêter le flux vidéo de la caméra selectionner.
- QPushButton * boutonVideo
  Bouton permettant de démarrer le flux vidéo de la caméra selectionner.
- QLabel * camera
  Texte informant de l'élément à selectionner (caméra)
- QLabel * contraste
  Texte informant le reglage à modifier.
- QSpinBox * contrasteVideo
  Zone de saisie permettant de modifier le contraste du flux vidéo.
- QComboBox * listeCameras
  Liste déroulante déstiné à accueillir la liste des caméra disponible.
- QComboBox * listeResolutions
  Liste déroulante déstiné à accueillir la liste des résolutions disponible.
- QLabel * luminosite
  Texte informant le reglage à modifier.
- QSpinBox * luminositeVideo
  Zone de saisie permettant de modifier la luminosite du flux vidéo.
- QLabel * resolution
  Texte informant de l'élément à selectionner (résolution)
- Rov * rov
  Objet rov permettant de mofidier les reglage du flux vidéo.
- QLabel * saturation
  Texte informant le reglage à modifier.
- QSpinBox * saturationVideo
  Zone de saisie permettant de modifier la saturation du flux vidéo.
- QSlider * sliderContrasteVideo
  Slider permettant de modifier le contraste du flux vidéo.
- QSlider * sliderLuminositeVideo
  Slider permettant de modifier la luminosite du flux vidéo.
- QSlider * sliderSaturationVideo
  Slider permettant de modifier la saturation du flux vidéo.

6.15.1 Description détaillée

Classe permettant de regler l'affichage du flux video.

Définition à la ligne 23 du fichier ihmreglagevideo.h.

6.15.2 Documentation des constructeurs et destructeur

6.15.2.1 IHMReglageVideo()

```
IHMReglageVideo::IHMReglageVideo (
            Rov * rov,
            QWidget * parent = nullptr )
```

Constructeur de la classe ReglageVideo.

Paramètres

rov	
parent	

Définition à la ligne 9 du fichier ihmreglagevideo.cpp.

Références configurerWidgets(), initialiserEvenements(), initialiserLayouts(), et initialiserWidgets().

```
00009                                                   : QWidget(parent),
      rov(rov)
00010 {
00011     qDebug() << Q_FUNC_INFO;
00012     initialiserWidgets();
00013     configurerWidgets();
00014     initialiserLayouts();
00015     initialiserEvenements();
00016 }
```

6.15.2.2 ~IHMReglageVideo()

```
IHMReglageVideo::~IHMReglageVideo ( )
```

Destructeur de la classe ReglageVideo.

Définition à la ligne 18 du fichier ihmreglagevideo.cpp.

```
00019 {
00020     qDebug() << Q_FUNC_INFO;
00021 }
```

6.15.3 Documentation des fonctions membres

6.15.3.1 activerCamera

```
void IHMReglageVideo::activerCamera ( )  [slot]
```

Active la caméra.

Définition à la ligne 214 du fichier ihmreglagevideo.cpp.

Références Rov : :demarrerVideo(), listeCameras, listeResolutions, et rov.

Référencé par gererVideo().

```
00215 {
00216     qDebug() << Q_FUNC_INFO << listeCameras->currentText() <<
      listeResolutions->currentText() << listeResolutions->currentIndex();
00217     rov->demarrerVideo(listeCameras->currentText(),
      listeResolutions->currentIndex());
00218     listeCameras->setEnabled(false);
00219 }
```

6.15.3.2 chargerListeCameraDisponible()

```
void IHMReglageVideo::chargerListeCameraDisponible ( )  [private]
```

Charge la liste des caméras disponibles dans la liste déroulante.

Définition à la ligne 143 du fichier ihmreglagevideo.cpp.

Références chargerListeResolutionDisponible(), Rov : :getCamera(), Camera : :getNom(), listeCameras, et rov.

Référencé par initialiserEvenementsCamera().

```
00144 {
00145     int nbCameras = QCameraInfo::availableCameras().count();
00146     qDebug() << Q_FUNC_INFO << "Caméra(s) disponible(s)" << QCameraInfo::availableCameras().count();
00147     if (nbCameras > 0)
00148     {
00149        #ifndef SANS_DETECTION
00150        QList<QCameraInfo> cameras = QCameraInfo::availableCameras();
00151        listeCameras->clear();
00152        int choix = -1, i = 0;
00153        foreach (const QCameraInfo &cameraInfo, cameras)
00154        {
00155            listeCameras->addItem(cameraInfo.deviceName());
00156            qDebug() << Q_FUNC_INFO << "Device" << cameraInfo.deviceName();
00157            qDebug() << Q_FUNC_INFO << "Description" << cameraInfo.description();
00158            if(rov->getCamera() != nullptr)
00159            {
00160                if(cameraInfo.deviceName() == rov->getCamera()->
      getNom())
00161                {
00162                    choix = i;
00163                }
00164            }
00165            i++;
00166        }
00167        #else
00168        int choix = 0;
00169        qDebug() << Q_FUNC_INFO << "Device" << rov->getCamera()->
      getNom();
00170        listeCameras->addItem(rov->getCamera()->getNom());
00171        #endif
00172
00173        if(choix != -1)
00174        {
00175            if(rov->getCamera() != nullptr)
00176                chargerListeResolutionDisponible(
      rov->getCamera()->getNom());
00177            listeCameras->setCurrentIndex(choix);
00178        }
00179     }
00180
00181 }
```

6.15.3.3 chargerListeResolutionDisponible [1/2]

```
void IHMReglageVideo::chargerListeResolutionDisponible (
            int index )  [slot]
```

Charge les résolutions pour une caméra sélectionnée.

Paramètres

index	

Définition à la ligne 183 du fichier ihmreglagevideo.cpp.

Références listeCameras.

Référencé par chargerListeCameraDisponible(), initialiserEvenementsCamera(), et modifierEtatBoutons().

```
00184 {
00185     if(index < 0)
00186         return;
00187     chargerListeResolutionDisponible(
      listeCameras->currentText());
00188 }
```

6.15.3.4 chargerListeResolutionDisponible [2/2]

```
void IHMReglageVideo::chargerListeResolutionDisponible (
             QString nom )  [slot]
```

Charge les résolutions pour une caméra sélectionnée.

Paramètres

nom	

Définition à la ligne 190 du fichier ihmreglagevideo.cpp.

Références Rov : :getCamera(), Camera : :getChoixResolution(), Camera : :getResolution(), Camera : :lireListeResolutionsCamera(), listeResolutions, resolution, et rov.

```
00191 {
00192     #ifndef SANS_DETECTION
00193     QCameraInfo cameraInfo(nom.toLatin1());
00194     QList<QSize> liste = Camera::lireListeResolutionsCamera(cameraInfo);
00195     listeResolutions->clear();
00196     foreach (const QSize &resolution, liste)
00197     {
00198         qDebug() << Q_FUNC_INFO << resolution.width() << "x" << resolution.height();
00199         listeResolutions->addItem(QString::number(resolution.width()) + QString("x") +
      QString::number(resolution.height()));
00200     }
00201     #else
00202     listeResolutions->clear();
00203     QSize resolutionDefaut = rov->getCamera()->getResolution();
00204     qDebug() << Q_FUNC_INFO << resolutionDefaut.width() << "x" << resolutionDefaut.height();
00205     listeResolutions->addItem(QString::number(resolutionDefaut.width()) + QString("x") +
      QString::number(resolutionDefaut.height()));
00206     #endif
00207     if(rov->getCamera() != nullptr)
00208     {
00209         qDebug() << Q_FUNC_INFO << "choixResolution" << rov->getCamera()->
      getChoixResolution();
00210         listeResolutions->setCurrentIndex(rov->getCamera()->
      getChoixResolution());
00211     }
00212 }
```

6.15.3.5 configurerWidgets()

```
void IHMReglageVideo::configurerWidgets ( )  [private]
```

Configure l'etat des widgets à la création de l'IHM.

Définition à la ligne 46 du fichier ihmreglagevideo.cpp.

Références contrasteVideo, listeCameras, luminositeVideo, saturationVideo, sliderContrasteVideo, sliderLuminositeVideo, et sliderSaturationVideo.

Référencé par IHMReglageVideo().

```
00047 {
00048     sliderLuminositeVideo->setValue(50);
00049     sliderContrasteVideo->setValue(50);
00050     sliderSaturationVideo->setValue(50);
00051
00052     luminositeVideo->setValue(50);
00053     contrasteVideo->setValue(50);
00054     saturationVideo->setValue(50);
00055
00056     listeCameras->setEnabled(false);
00057 }
```

6.15.3.6 gererVideo

```
void IHMReglageVideo::gererVideo ( )  [slot]
```

Modifie l'etat de la vidéo en fonction de l'état actuel.

Définition à la ligne 114 du fichier ihmreglagevideo.cpp.

Références activerCamera(), Rov : :arreterVideo(), boutonVideo, et rov.

Référencé par initialiserEvenements().

```
00115 {
00116     if(boutonVideo->text() == "Arrêter")
00117     {
00118         rov->arreterVideo();
00119         boutonVideo->setText("Démarrer");
00120     }
00121     else
00122     {
00123         activerCamera();
00124         boutonVideo->setText("Arrêter");
00125     }
00126 }
```

6.15.3.7 initialiserEvenements()

```
void IHMReglageVideo::initialiserEvenements ( )  [private]
```

Initialise les événements de l'IHM.

Définition à la ligne 103 du fichier ihmreglagevideo.cpp.

Références boutonVideo, contrasteVideo, gererVideo(), luminositeVideo, saturationVideo, sliderContrasteVideo, sliderLuminositeVideo, et sliderSaturationVideo.

Référencé par IHMReglageVideo().

```
00104 {
00105     connect(sliderLuminositeVideo, SIGNAL(valueChanged(int)),
      luminositeVideo, SLOT(setValue(int)));
00106     connect(sliderContrasteVideo, SIGNAL(valueChanged(int)),
      contrasteVideo, SLOT(setValue(int)));
00107     connect(sliderSaturationVideo, SIGNAL(valueChanged(int)),
      saturationVideo, SLOT(setValue(int)));
00108     connect(luminositeVideo, SIGNAL(valueChanged(int)),
      sliderLuminositeVideo, SLOT(setValue(int)));
00109     connect(contrasteVideo, SIGNAL(valueChanged(int)),
      sliderContrasteVideo, SLOT(setValue(int)));
00110     connect(saturationVideo, SIGNAL(valueChanged(int)),
      sliderSaturationVideo, SLOT(setValue(int)));
00111     connect(boutonVideo, SIGNAL(clicked()), this, SLOT(gererVideo()));
00112 }
```

6.15.3.8 initialiserEvenementsCamera()

```
void IHMReglageVideo::initialiserEvenementsCamera ( )
```

Initialise les événements liés à la caméra.

Définition à la ligne 128 du fichier ihmreglagevideo.cpp.

Références chargerListeCameraDisponible(), chargerListeResolutionDisponible(), contrasteVideo, Rov : :getCamera(), listeCameras, listeResolutions, luminositeVideo, rov, et saturationVideo.

Référencé par IHMRov : :initialiserEvenementsCamera().

```
00129 {
00130     if(rov->getCamera() != nullptr)
00131     {
00132         if(rov->getCamera()->isRunning())
00133             return;
00134         chargerListeCameraDisponible();
00135         connect(luminositeVideo, SIGNAL(valueChanged(int)), rov->
      getCamera(), SLOT(setLuminosite(int)));
00136         connect(contrasteVideo, SIGNAL(valueChanged(int)), rov->
      getCamera(), SLOT(setContraste(int)));
00137         connect(saturationVideo, SIGNAL(valueChanged(int)), rov->
      getCamera(), SLOT(setSaturation(int)));
00138         connect(listeCameras, SIGNAL(currentIndexChanged(int)), this, SLOT(
      chargerListeResolutionDisponible(int)));
00139         connect(listeResolutions, SIGNAL(currentIndexChanged(int)),
      rov->getCamera(), SLOT(setResolution(int)));
00140     }
00141 }
```

6.15.3.9 initialiserLayouts()

```
void IHMReglageVideo::initialiserLayouts ( )  [private]
```

Initialise les layout de l'IHM.

Définition à la ligne 59 du fichier ihmreglagevideo.cpp.

Références boutonVideo, camera, contraste, contrasteVideo, listeCameras, listeResolutions, luminosite, luminositeVideo, NOM_FENETRE_REGLAGEVIDEO, resolution, saturation, saturationVideo, sliderContrasteVideo, sliderLuminositeVideo, et sliderSaturationVideo.

Référencé par IHMReglageVideo().

```
00060 {
00061     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00062     QHBoxLayout *layoutReglageVideo = new QHBoxLayout;
00063     QVBoxLayout *layoutConfigurationLuminosite = new QVBoxLayout;
00064     QVBoxLayout *layoutConfigurationContraste = new QVBoxLayout;
00065     QVBoxLayout *layoutConfigurationSaturation = new QVBoxLayout;
00066     QHBoxLayout *layoutCamera = new QHBoxLayout;
00067     QHBoxLayout *layoutBoutonCamera = new QHBoxLayout;
00068
00069     layoutReglageVideo->setAlignment(Qt::AlignLeft);
00070     layoutConfigurationContraste->setAlignment(Qt::AlignTop);
00071     layoutConfigurationLuminosite->setAlignment(Qt::AlignTop);
00072     layoutConfigurationSaturation->setAlignment(Qt::AlignTop);
00073     layoutBoutonCamera->setAlignment(Qt::AlignLeft);
00074
00075     layoutPrincipal->addLayout(layoutReglageVideo);
00076     layoutPrincipal->addLayout(layoutCamera);
00077     layoutPrincipal->addLayout(layoutBoutonCamera);
00078
00079     layoutReglageVideo->addLayout(layoutConfigurationLuminosite);
00080     layoutReglageVideo->addLayout(layoutConfigurationContraste);
00081     layoutReglageVideo->addLayout(layoutConfigurationSaturation);
00082
00083     layoutConfigurationLuminosite->addWidget(luminosite);
00084     layoutConfigurationLuminosite->addWidget(sliderLuminositeVideo);
00085     layoutConfigurationLuminosite->addWidget(luminositeVideo);
00086     layoutConfigurationContraste->addWidget(contraste);
00087     layoutConfigurationContraste->addWidget(sliderContrasteVideo);
00088     layoutConfigurationContraste->addWidget(contrasteVideo);
00089     layoutConfigurationSaturation->addWidget(saturation);
00090     layoutConfigurationSaturation->addWidget(sliderSaturationVideo);
00091     layoutConfigurationSaturation->addWidget(saturationVideo);
00092     layoutCamera->addWidget(camera);
00093     layoutCamera->addWidget(listeCameras);
00094     layoutCamera->addWidget(resolution);
00095     layoutCamera->addWidget(listeResolutions);
00096     layoutBoutonCamera->addWidget(boutonVideo);
00097
00098     setLayout(layoutPrincipal);
00099     setWindowTitle(NOM_FENETRE_REGLAGEVIDEO);
00100     setStyleSheet("background:#202020;color:white;");
00101 }
```

6.15.3.10 initialiserWidgets()

```
void IHMReglageVideo::initialiserWidgets ( )  [private]
```

Initialise les widgets de l'IHM.

Définition à la ligne 23 du fichier ihmreglagevideo.cpp.

Références boutonVideo, camera, contraste, contrasteVideo, listeCameras, listeResolutions, luminosite, luminositeVideo, resolution, saturation, saturationVideo, sliderContrasteVideo, sliderLuminositeVideo, et sliderSaturationVideo.

Référencé par IHMReglageVideo().

```
00024 {
00025     sliderLuminositeVideo = new QSlider(Qt::Horizontal, this);
00026     sliderContrasteVideo = new QSlider(Qt::Horizontal, this);
00027     sliderSaturationVideo = new QSlider(Qt::Horizontal, this);
00028
00029     luminosite = new QLabel("Luminosité", this);
00030     contraste = new QLabel("Contraste", this);
00031     saturation = new QLabel("Saturation", this);
00032
00033     luminositeVideo = new QSpinBox(this);
00034     contrasteVideo = new QSpinBox(this);
00035     saturationVideo = new QSpinBox(this);
00036
00037     camera = new QLabel("Camera(s): ", this);
00038     resolution = new QLabel("Résolution: ", this);
00039
00040     listeCameras = new QComboBox(this);
00041     listeResolutions = new QComboBox(this);
00042
00043     boutonVideo = new QPushButton("Arrêter", this);
00044 }
```

6.15.3.11 modifierEtatBoutons

`void IHMReglageVideo::modifierEtatBoutons ( )  [slot]`

Modifie l'etat des boutons lors du démarrage du flux vidéo.

Définition à la ligne 221 du fichier ihmreglagevideo.cpp.

Références chargerListeResolutionDisponible(), contrasteVideo, Rov : :getCamera(), listeCameras, listeResolutions, luminositeVideo, rov, et saturationVideo.

Référencé par IHMRov : :arreterVideo().

```
00222 {
00223     if(rov->getCamera() != nullptr)
00224     {
00225         disconnect(luminositeVideo, SIGNAL(valueChanged(int)),
      rov->getCamera(), SLOT(setLuminosite(int)));
00226         disconnect(contrasteVideo, SIGNAL(valueChanged(int)), rov->
      getCamera(), SLOT(setContraste(int)));
00227         disconnect(saturationVideo, SIGNAL(valueChanged(int)),
      rov->getCamera(), SLOT(setSaturation(int)));
00228         disconnect(listeCameras, SIGNAL(currentIndexChanged(int)), this, SLOT(
      chargerListeResolutionDisponible(int)));
00229         disconnect(listeResolutions, SIGNAL(currentIndexChanged(int)),
      rov->getCamera(), SLOT(setResolution(int)));
00230     }
00231     listeCameras->setEnabled(true);
00232 }
```

6.15.4 Documentation des données membres

6.15.4.1 boutonArreterVideo

`QPushButton* IHMReglageVideo::boutonArreterVideo  [private]`

Bouton permettant d'arrêter le flux vidéo de la caméra selectionner.

Définition à la ligne 42 du fichier ihmreglagevideo.h.

6.15.4.2 boutonVideo

```
QPushButton* IHMReglageVideo::boutonVideo  [private]
```

Bouton permettant de démarrer le flux vidéo de la caméra selectionner.

Définition à la ligne 41 du fichier ihmreglagevideo.h.

Référencé par gererVideo(), initialiserEvenements(), initialiserLayouts(), et initialiserWidgets().

6.15.4.3 camera

```
QLabel* IHMReglageVideo::camera  [private]
```

Texte informant de l'élément à selectionner (caméra)

Définition à la ligne 37 du fichier ihmreglagevideo.h.

Référencé par initialiserLayouts(), et initialiserWidgets().

6.15.4.4 contraste

```
QLabel* IHMReglageVideo::contraste  [private]
```

Texte informant le reglage à modifier.

Définition à la ligne 32 du fichier ihmreglagevideo.h.

Référencé par initialiserLayouts(), et initialiserWidgets().

6.15.4.5 contrasteVideo

```
QSpinBox* IHMReglageVideo::contrasteVideo  [private]
```

Zone de saisie permettant de modifier le contraste du flux vidéo.

Définition à la ligne 35 du fichier ihmreglagevideo.h.

Référencé par configurerWidgets(), initialiserEvenements(), initialiserEvenementsCamera(), initialiserLayouts(), initialiserWidgets(), et modifierEtatBoutons().

6.15.4.6 listeCameras

```
QComboBox* IHMReglageVideo::listeCameras  [private]
```

Liste déroulante déstiné à accueillir la liste des caméra disponible.

Définition à la ligne 38 du fichier ihmreglagevideo.h.

Référencé par activerCamera(), chargerListeCameraDisponible(), chargerListeResolutionDisponible(), configurerWidgets(), initialiserEvenementsCamera(), initialiserLayouts(), initialiserWidgets(), et modifierEtatBoutons().

6.15.4.7 listeResolutions

```
QComboBox* IHMReglageVideo::listeResolutions  [private]
```

Liste déroulante déstiné à accueillir la liste des résolutions disponible.

Définition à la ligne 40 du fichier ihmreglagevideo.h.

Référencé par activerCamera(), chargerListeResolutionDisponible(), initialiserEvenementsCamera(), initialiserLayouts(), initialiserWidgets(), et modifierEtatBoutons().

6.15.4.8 luminosite

```
QLabel* IHMReglageVideo::luminosite  [private]
```

Texte informant le reglage à modifier.

Définition à la ligne 31 du fichier ihmreglagevideo.h.

Référencé par initialiserLayouts(), et initialiserWidgets().

6.15.4.9 luminositeVideo

```
QSpinBox* IHMReglageVideo::luminositeVideo  [private]
```

Zone de saisie permettant de modifier la luminosite du flux vidéo.

Définition à la ligne 34 du fichier ihmreglagevideo.h.

Référencé par configurerWidgets(), initialiserEvenements(), initialiserEvenementsCamera(), initialiserLayouts(), initialiserWidgets(), et modifierEtatBoutons().

6.15.4.10 resolution

```
QLabel* IHMReglageVideo::resolution  [private]
```

Texte informant de l'élément à selectionner (résolution)

Définition à la ligne 39 du fichier ihmreglagevideo.h.

Référencé par chargerListeResolutionDisponible(), initialiserLayouts(), et initialiserWidgets().

6.15.4.11 rov

```
Rov* IHMReglageVideo::rov  [private]
```

Objet rov permettant de mofidier les reglage du flux vidéo.

Définition à la ligne 27 du fichier ihmreglagevideo.h.

Référencé par activerCamera(), chargerListeCameraDisponible(), chargerListeResolutionDisponible(), gererVideo(), initialiserEvenementsCamera(), et modifierEtatBoutons().

6.15.4.12 saturation

```
QLabel* IHMReglageVideo::saturation  [private]
```

Texte informant le reglage à modifier.

Définition à la ligne 33 du fichier ihmreglagevideo.h.

Référencé par initialiserLayouts(), et initialiserWidgets().

6.15.4.13 saturationVideo

```
QSpinBox* IHMReglageVideo::saturationVideo  [private]
```

Zone de saisie permettant de modifier la saturation du flux vidéo.

Définition à la ligne 36 du fichier ihmreglagevideo.h.

Référencé par configurerWidgets(), initialiserEvenements(), initialiserEvenementsCamera(), initialiserLayouts(), initialiserWidgets(), et modifierEtatBoutons().

6.15.4.14 sliderContrasteVideo

```
QSlider* IHMReglageVideo::sliderContrasteVideo  [private]
```

Slider permettant de modifier le contraste du flux vidéo.

Définition à la ligne 29 du fichier ihmreglagevideo.h.

Référencé par configurerWidgets(), initialiserEvenements(), initialiserLayouts(), et initialiserWidgets().

6.15.4.15 sliderLuminositeVideo

```
QSlider* IHMReglageVideo::sliderLuminositeVideo  [private]
```

Slider permettant de modifier la luminosite du flux vidéo.

Définition à la ligne 28 du fichier ihmreglagevideo.h.

Référencé par configurerWidgets(), initialiserEvenements(), initialiserLayouts(), et initialiserWidgets().

6.15.4.16 sliderSaturationVideo

```
QSlider* IHMReglageVideo::sliderSaturationVideo  [private]
```

Slider permettant de modifier la saturation du flux vidéo.

Définition à la ligne 30 du fichier ihmreglagevideo.h.

Référencé par configurerWidgets(), initialiserEvenements(), initialiserLayouts(), et initialiserWidgets().

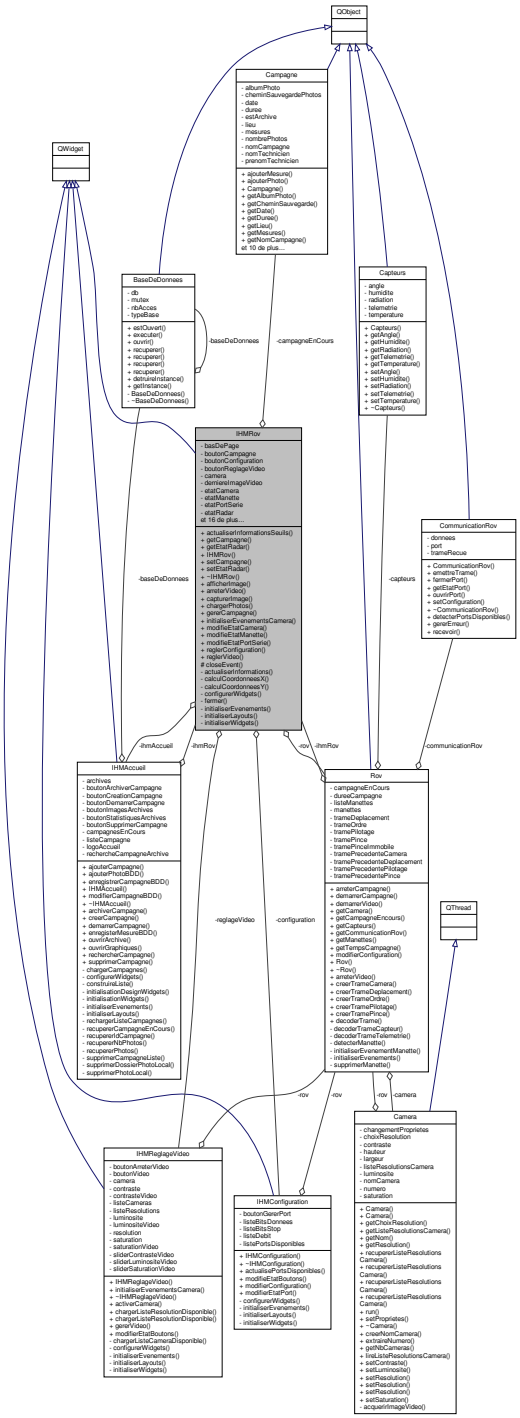
La documentation de cette classe a été générée à partir des fichiers suivants :
- ihmreglagevideo.h
- ihmreglagevideo.cpp

6.16 Référence de la classe IHMRov

IHM permettant d'obtenir le flux vidéo en direct placé sur le robot et d'obtenir les informations relatifs à ses capteurs.

`#include "ihmrov.h"`

Graphe de collaboration de IHMRov :



Connecteurs publics

— void afficherImage (QPixmap image)

Affiche la nouvelle image du flux vidéo dans l'ihm.
- void arreterVideo ()
  Déconnecte les événements liés à la caméra et modifie l'état des boutons de l'IHM.
- void capturerImage (bool etat=false)
  Enregistre la dernière image du flux vidéo.
- void chargerPhotos ()
  Charge les photos disponible dans le conteneur albumPhoto de la classe campagne pour les afficher dans l'IHMAlbumPhoto.
- void gererCampagne ()
  Arrête la campagne en cours.
- void initialiserEvenementsCamera ()
  Initialise les événements liés à la caméra.
- void modifieEtatCamera (bool etat, QString information)
  Modifie l'affichage de l'état de la caméra.
- void modifieEtatManette (bool etat)
  Modifie l'affichage de l'état de la manette.
- void modifieEtatPortSerie (bool etat, QString information)
  Modifie l'affichage de l'état du port série.
- void reglerConfiguration ()
  Ouvre une nouvelle fenetre permettant de régler la communication.
- void reglerVideo ()
  Ouvre une nouvelle fenetre permettant de régler l'affichage vidéo.

Fonctions membres publiques

- void actualiserInformationsSeuils ()
  Actualise les informations affichés des indicateur de dépassement des seuils acceptable.
- Campagne * getCampagne ()
  Retourne l'objet campagne en cours.
- bool getEtatRadar ()
  Donne l'etat de etatRadar.
- IHMRov (IHMAccueil *ihmAccueil, QWidget *parent=nullptr)
  Constructeur de la classe IHMRov.
- void setCampagne (Campagne *campagne)
  Associe une campagne a la campagne en cours du rov.
- void setEtatRadar (bool etatRadar)
  Dertermine l'etat de etatRadar.
- ~IHMRov ()
  Destructeur de la classe IHMRov.

Fonctions membres protégées

- void closeEvent (QCloseEvent *event)
  Gêre l'état de la campagne lors de la fermeture forcé de la fenêtre ihmRov.

Fonctions membres privées

- void actualiserInformations (QPixmap &image)
  Actualise les informations incrusté dans l'image (heure, données capteur, durée missions)
- double calculCoordonneesX (QPixmap &image)
  Calcule les coordonnées x de l'obstacle pour le radar.
- double calculCoordonneesY (QPixmap &image)
  Calcule les coordonnées y de l'obstacle pour le radar.
- void configurerWidgets ()
  Configure l'état des widgets à la création de l'IHM.
- void fermer ()
  Arrête la campagne et ferme l'ihmRov.
- void initialiserEvenements ()
  Initialise les événements de l'IHM.
- void initialiserLayouts ()
  Initialise les layouts de l'IHM.
- void initialiserWidgets ()
  Initialise les widgets de l'IHM.

Attributs privés

- QLabel * basDePage
  Emplacement permettant de créer un espace en bas de la page.
- QPushButton * boutonCampagne
  Bouton permettant de mettre en pause la campagne en cours.
- QPushButton * boutonConfiguration
  Bouton permettant d'accéder à la configuration de la communication.
- QPushButton * boutonReglageVideo
  Bouton permettant d'accéder aux reglage de la vidéo.
- QLabel * camera
  Emplacement permettant de définir le type de matériel.
- Campagne * campagneEnCours
  Instance d'un objet Campagne possédant les informations de la campagne en cours.
- IHMConfiguration * configuration
  Instance d'un objet IHMConfiguration permettant de modifier les reglages de la communication.
- QPixmap derniereImageVideo
  Dernière image reçue du flux vidéo.
- QLabel * etatCamera
  Emplacement permettant de visualiser l'état de la caméra.
- QLabel * etatManette
  Emplacement permettant de visualiser l'état de la manette.
- QLabel * etatPortSerie
  Emplacement permettant de visualiser l'état du port série.
- bool etatRadar
  Dertermine si on affiche un radar.
- QLabel * fluxVideo
  Emplacement permettant d'accueiller le flux vidéo.
- QLabel * hautDePage
  Emplacement permettant de créer un espace en haut de la page.
- IHMAccueil * ihmAccueil
  Relation entre l'ihmAccueil et l'ihmRov.
- QwtThermo * indicateurRadiation
  Indicateur permettant de visualiser l'etat de la radiation actuel avec indication de dépassement de seuil.
- QwtThermo * indicateurTemperature
  Indicateur permettant de visualiser l'etat de la temperature actuel avec indication de dépassement de seuil.
- QLabel * logoEtatCamera
  Emplacement permettant de visualiser l'état de la caméra à l'aide d'un logo.
- QLabel * logoEtatManette
  Emplacement permettant de visualiser l'état de la manette à l'aide d'un logo.
- QLabel * logoEtatPortSerie
  Emplacement permettant de visualiser l'état du port série à l'aide d'un logo.
- QLabel * manette
  Emplacement permettant de définir le type de matériel.
- QPushButton * photosEnCours
  Bouton permettant d'accéder aux photo prise en cours de campagne.
- QVector< QPoint > pointsRadar
  Conteneur des points du radar.
- QLabel * portSerie
  Emplacement permettant de définir le type de matériel.
- QLabel * radiation
  Emplacement permettant de définir le type de seuil.
- IHMReglageVideo * reglageVideo
  Instance d'un objet reglageVidéo permettant de modifier les reglages du flux vidéo.
- Rov * rov
  Instance d'un objet rov possédant le controle sur les autres classes.
- QLabel * temperature
  Emplacement permettant de définir le type de seuil.
- QPushButton * testCapturePhoto
  Bouton de simulation de prise de photo.
- QGroupBox * zoneEtatMateriel
  Zone regroupant les informations sur l'état du matériel.
- QGroupBox * zoneInformationSeuils
  Zone regroupant les informations sur l'état des seuils de dépassement.

6.16.1 Description détaillée

IHM permettant d'obtenir le flux vidéo en direct placé sur le robot et d'obtenir les informations relatifs à ses capteurs.

Définition à la ligne 81 du fichier ihmrov.h.

6.16.2 Documentation des constructeurs et destructeur

6.16.2.1 IHMRov()

```
IHMRov::IHMRov (
            IHMAccueil * ihmAccueil,
            QWidget * parent = nullptr )
```

Constructeur de la classe IHMRov.

Paramètres

ihmAccueil	
parent	

Définition à la ligne 16 du fichier ihmrov.cpp.

Références configuration, configurerWidgets(), Rov : :getManettes(), initialiserEvenements(), initialiserLayouts(), initialiserWidgets(), modifieEtatManette(), reglageVideo, et rov.

```
00016                                                   : QWidget(parent),
      campagneEnCours(nullptr), ihmAccueil(ihmAccueil),
      etatRadar(true)
00017 {
00018     qDebug() << Q_FUNC_INFO << this << "width" << qApp->desktop()->screen()->width() << "height" << qApp->
      desktop()->screen()->height();
00019     rov = new Rov(this);
00020     reglageVideo = nullptr;
00021     configuration = nullptr;
00022
00023     initialiserWidgets();
00024     configurerWidgets();
00025     initialiserLayouts();
00026     initialiserEvenements();
00027
00028     if(!rov->getManettes().isEmpty())
00029         modifieEtatManette(true);
00030 }
```

6.16.2.2 ~IHMRov()

```
IHMRov::~IHMRov ( )
```

Destructeur de la classe IHMRov.

Définition à la ligne 32 du fichier ihmrov.cpp.

```
00033 {
00034     qDebug() << Q_FUNC_INFO;
00035 }
```

6.16.3 Documentation des fonctions membres

6.16.3.1 actualiserInformations()

```
void IHMRov::actualiserInformations (
            QPixmap & image )  [private]
```

Actualise les informations incrusté dans l'image (heure, données capteur, durée missions)

Paramètres

image	

Définition à la ligne 251 du fichier ihmrov.cpp.

Références ANGLE_MAX_RADAR, ANGLE_MIN_RADAR, calculCoordonneesX(), calculCoordonneesY(), DISTANCE_MAX_RAD↩AR, Capteurs : :getAngle(), Rov : :getCapteurs(), getEtatRadar(), Capteurs : :getHumidite(), Capteurs : :getRadiation(), Capteurs↩: :getTelemetrie(), Capteurs : :getTemperature(), Rov : :getTempsCampagne(), pointsRadar, et rov.

Référencé par afficherImage().

```
00252 {
00253     QPainter p(&image);
00254     QPen pen;
00255     QPen penCadre;
00256     QFont fontHaut("Open Sans");
00257     QFont fontBas("Open Sans");
00258     int marge = 0;
00259     pen.setWidth(qApp->desktop()->screen()->width() * 0.005);
00260 
00261     //QFontDatabase fontDatabase;
00262     //qDebug() << fontDatabase.families();
00263 
00264     // règle la taille de police
00265     fontHaut.setPixelSize(image.height()*0.045); // 4.5 % de la hauteur de l'image en pixel
00266     fontBas.setPixelSize(fontHaut.pixelSize()*0.75); // 75% de la hauteur normale
00267 
00268     /*
00269     // pour le dessin du bandeau
00270     penCadre.setWidth(1);
00271     penCadre.setBrush(Qt::lightGray);
00272     penCadre.setCapStyle(Qt::RoundCap);
00273     penCadre.setJoinStyle(Qt::RoundJoin);
00274     p.setPen(penCadre);
00275     // dessine un bandeau en haut (hauteur 5% de la hauteur de l'image)
00276     p.drawLine( 0, (image.height()*0.05)+1, image.width(), (image.height()*0.05)+1 );
00277     // dessine un bandeau en bas (hauteur 5% de la hauteur de l'image)
00278     p.drawLine( 0, (image.height()*0.95)-1, image.width(), (image.height()*0.95)-1 );
00279     */
00280     QRect bandeauHaut( 0, 0, image.width(), image.height()*0.05 );
00281     QRect bandeauBas( 0, image.height()*0.95, image.width(), image.height()*0.05 );
00282     penCadre.setBrush(QBrush(QColor(255, 255, 255, 255)));
00283     p.setPen(penCadre);
00284     p.fillRect(bandeauHaut, QBrush(QColor(128, 128, 128, 64)));
00285     p.fillRect(bandeauBas, QBrush(QColor(128, 128, 128, 64)));
00286 
00287     // découpe le bandeau en trois cadres (largeurs : 25 % 50 % 25% de la largeur de l'image)
00288     QRect bandeauHautGauche( 0, 0, image.width()*0.25, image.height()*0.05 );
00289     QRect bandeauHautCentre( image.width()*0.25, 0, image.width()*0.5, image.height()*0.05 );
00290     QRect bandeauHautDroite( image.width()*0.75, 0, image.width()*0.25, image.height()*0.05 );
00291 
00292     // découpe le bandeau en trois cadres (largeurs : 25 % 25 % 25% 25% de la largeur de l'image)
00293     QRect bandeauBasGauche( 0, image.height()*0.95, image.width()*0.25, image.height()*0.05 );
00294     QRect bandeauBasCentreGauche( image.width()*0.25, image.height()*0.95, image.width()*0.25, image.height
      ()*0.05 );
00295     QRect bandeauBasCentreDroite( image.width()*0.50, image.height()*0.95, image.width()*0.25, image.height
      ()*0.05 );
00296     QRect bandeauRadar( image.width()*0.75, image.height() - image.width()*0.25, image.width()*0.25, image.
      width()*0.25);
00297     QRect bandeauBasDroite( image.width()*0.75, image.height()*0.95, image.width()*0.25, image.height()*0.0
      5 );
00298 
00299     /*
00300     // pour le dessin des cadres
00301     p.setPen(pen);
00302     // dessine les trois cadres du haut
00303     p.drawRect( bandeauHautGauche ); // haut gauche
00304     p.fillRect(bandeauHautGauche, QBrush(QColor(128, 128, 128, 64)));
00305     p.drawRect( bandeauHautCentre ); // haut centre
00306     p.fillRect(bandeauHautCentre, QBrush(QColor(128, 128, 128, 64)));
00307     p.drawRect( bandeauHautDroite ); // haut droite
00308     p.fillRect(bandeauHautDroite, QBrush(QColor(128, 128, 128, 64)));
00309     // dessine les quatres cadres du bas
00310     p.drawRect( bandeauBasGauche ); // bas gauche
00311     p.fillRect(bandeauBasGauche, QBrush(QColor(128, 128, 128, 64)));
00312     p.drawRect( bandeauBasCentreGauche ); // bas centre gauche
00313     p.fillRect(bandeauBasCentreGauche, QBrush(QColor(128, 128, 128, 64)));
00314     p.drawRect( bandeauBasCentreDroite ); // bas centre droite
00315     p.fillRect(bandeauBasCentreDroite, QBrush(QColor(128, 128, 128, 64)));
00316     p.drawRect( bandeauBasDroite ); // bas droite
00317     p.fillRect(bandeauBasDroite, QBrush(QColor(128, 128, 128, 64)));
```

```
00318     */
00319
00320     // pour le dessin des textes et images
00321     pen.setBrush(Qt::darkRed);
00322     p.setPen(pen);
00323     p.setFont(fontHaut);
00324
00325     marge = image.width()*0.0025;
00326     QRect cadreLogoHorloge( bandeauHautGauche.x(), bandeauHautGauche.y(), bandeauHautGauche.width()*0.1,
      bandeauHautGauche.height() ); // 10% du bandeau
00327     cadreLogoHorloge.adjust(marge, marge, -marge, -marge);
00328     QImage logoHeure(qApp->applicationDirPath() + "/images/logo_heure.png");
00329     p.drawImage(cadreLogoHorloge, logoHeure);
00330     p.drawText(bandeauHautGauche, Qt::AlignHCenter|Qt::AlignVCenter, QTime::currentTime().toString());
00331
00332     QRect cadreLogoDuree( bandeauHautDroite.x(), bandeauHautDroite.y(), bandeauHautDroite.width()*0.1,
      bandeauHautGauche.height() ); // 10% du bandeau
00333     cadreLogoDuree.adjust(marge, marge, -marge, -marge);
00334     QImage logoDuree(qApp->applicationDirPath() + "/images/logo_duree.png");
00335     p.drawImage(cadreLogoDuree, logoDuree);
00336     p.drawText(bandeauHautDroite, Qt::AlignHCenter|Qt::AlignVCenter, rov->
      getTempsCampagne());
00337
00338     p.setFont(fontBas);
00339     QRect cadreLogoTemperature( bandeauBasGauche.x(), bandeauBasGauche.y(), bandeauBasGauche.width()*0.1,
      bandeauBasGauche.height() ); // 10% du bandeau
00340     cadreLogoTemperature.adjust(marge, marge, -marge, -marge);
00341     QImage logoTemperature(qApp->applicationDirPath() + "/images/logo_temperature.png");
00342     p.drawImage(cadreLogoTemperature, logoTemperature);
00343     p.drawText(bandeauBasGauche, Qt::AlignHCenter|Qt::AlignVCenter, rov->
      getCapteurs()->getTemperature() + " °C");
00344
00345     QRect cadreLogoHumidite( bandeauBasCentreGauche.x(), bandeauBasCentreGauche.y(), bandeauBasCentreGauche
      .width()*0.1, bandeauBasCentreGauche.height() ); // 10% du bandeau
00346     cadreLogoHumidite.adjust(marge, marge, -marge, -marge);
00347     QImage logoHumditie(qApp->applicationDirPath() + "/images/logo_humidite.png");
00348     p.drawImage(cadreLogoHumidite, logoHumditie);
00349     p.drawText(bandeauBasCentreGauche, Qt::AlignHCenter|Qt::AlignVCenter, rov->
      getCapteurs()->getHumidite() + "%");
00350
00351     QRect cadreLogoRadiation( bandeauBasCentreDroite.x(), bandeauBasCentreDroite.y(),
      bandeauBasCentreDroite.width()*0.1, bandeauBasCentreDroite.height() ); // 10% du bandeau
00352     cadreLogoRadiation.adjust(marge, marge, -marge, -marge);
00353     QImage logoRadiation(qApp->applicationDirPath() + "/images/logo_radiation.png");
00354     p.drawImage(cadreLogoRadiation, logoRadiation);
00355     p.drawText(bandeauBasCentreDroite, Qt::AlignHCenter|Qt::AlignVCenter, rov->
      getCapteurs()->getRadiation() + " uS/h");
00356
00357     if(getEtatRadar())
00358     {
00359         pen.setBrush(Qt::green);
00360         p.setPen(pen);
00361         //p.drawPoint(image.width()*0.875, image.height() - image.width()*0.125); // placement du point 0
       sur le radar
00362         p.drawImage(bandeauRadar, QImage(qApp->applicationDirPath() + "/images/RADAR.png"));
00363
00364         if(rov->getCapteurs()->getTelemetrie().toInt() <=
      DISTANCE_MAX_RADAR && rov->getCapteurs()->
      getTelemetrie().toInt() >= 0 && rov->getCapteurs()->
      getAngle().toInt() >= 0 && rov->getCapteurs()->getAngle().toInt() <=
      ANGLE_MAX_RADAR && rov->getCapteurs()->getAngle() != "" &&
      rov->getCapteurs()->getTelemetrie() != "")
00365         {
00366             if(rov->getCapteurs()->getAngle().toInt() ==
      ANGLE_MAX_RADAR || rov->getCapteurs()->getAngle().toInt() ==
      ANGLE_MIN_RADAR)
00367             {
00368                 pointsRadar.clear();
00369             }
00370             pointsRadar.push_back(QPoint(calculCoordonneesX(image),
      calculCoordonneesY(image)));
00371             for (QVector<QPoint>::iterator it = pointsRadar.begin(); it !=
      pointsRadar.end(); ++it)
00372             {
00373                p.drawPoint((*it).x(),(*it).y());
00374             }
00375         }
00376     }
00377     else
00378     {
00379         QRect cadreLogoObstacle( bandeauBasDroite.x(), bandeauBasDroite.y(), bandeauBasDroite.width()*0.1,
      bandeauBasDroite.height() ); // 10% du bandeau
00380         cadreLogoObstacle.adjust(marge, marge, -marge, -marge);
00381         QImage logoObstacle(qApp->applicationDirPath() + "/images/logo_telemetrie.png");
00382         p.drawImage(cadreLogoObstacle, logoObstacle);
00383         p.drawText(bandeauBasDroite, Qt::AlignHCenter|Qt::AlignVCenter, "Obstacle : " +
      rov->getCapteurs()->getTelemetrie() + " cm");
00384
00385     }
00386     p.end();
00387 }
```

6.16.3.2 actualiserInformationsSeuils()

```
void IHMRov::actualiserInformationsSeuils ( )
```

Actualise les informations affichés des indicateur de dépassement des seuils acceptable.

Définition à la ligne 389 du fichier ihmrov.cpp.

Références Rov : :getCapteurs(), Capteurs : :getRadiation(), Capteurs : :getTemperature(), indicateurRadiation, indicateurTemperature, rov, SEUIL_RADIATION_ACCEPTABLE, SEUIL_TEMPERATURE_NEGATIVE_ACCEPTABLE, et SEUIL_TEMPERATURE_POSITIVE_ACCEPTABLE.

Référencé par Rov : :decoderTrameCapteur().

```
00390 {
00391     if(rov->getCapteurs()->getRadiation().toDouble() >
      SEUIL_RADIATION_ACCEPTABLE)
00392     {
00393         indicateurRadiation->setFillBrush(QBrush(QColor(Qt::GlobalColor::darkRed)));
00394     }
00395     else
00396     {
00397         indicateurRadiation->setFillBrush(QBrush(QColor(Qt::GlobalColor::darkGreen)));
00398     }
00399
00400     if(rov->getCapteurs()->getTemperature().toDouble() >
      SEUIL_TEMPERATURE_POSITIVE_ACCEPTABLE ||
      rov->getCapteurs()->getTemperature().toDouble() <
      SEUIL_TEMPERATURE_NEGATIVE_ACCEPTABLE)
00401     {
00402         indicateurTemperature->setFillBrush(QBrush(QColor(Qt::GlobalColor::darkRed)));
00403     }
00404     else
00405     {
00406         indicateurTemperature->setFillBrush(QBrush(QColor(Qt::GlobalColor::darkGreen))
      );
00407     }
00408
00409     indicateurRadiation->setValue(rov->getCapteurs()->
      getRadiation().toDouble());
00410     indicateurTemperature->setValue(rov->getCapteurs()->
      getTemperature().toDouble());
00411 }
```

6.16.3.3 afficherImage

```
void IHMRov::afficherImage (
             QPixmap image )  [slot]
```

Affiche la nouvelle image du flux vidéo dans l'ihm.

Paramètres

image	

Définition à la ligne 447 du fichier ihmrov.cpp.

Références actualiserInformations(), derniereImageVideo, et fluxVideo.

```
00448 {
00449     derniereImageVideo = image;
00450     actualiserInformations(image);
00451     fluxVideo->setPixmap(image);
00452 }
```

6.16.3.4 arreterVideo

```
void IHMRov::arreterVideo ( )  [slot]
```

Déconnecte les événements liés à la caméra et modifie l'état des boutons de l'IHM.

Définition à la ligne 528 du fichier ihmrov.cpp.

Références fluxVideo, IHMReglageVideo : :modifierEtatBoutons(), et reglageVideo.

Référencé par Rov : :arreterVideo().

```
00529 {
00530     reglageVideo->modifierEtatBoutons();
00531     fluxVideo->setPixmap(QPixmap(qApp->applicationDirPath() +  "/images/signal-interrompu.jpg"));
00532 }
```

6.16.3.5 calculCoordonneesX()

```
double IHMRov::calculCoordonneesX (
            QPixmap & image )  [private]
```

Calcule les coordonnées x de l'obstacle pour le radar.

Renvoie

entier correspondant à la valeur sur l'axe des x du radar d'un objet detecté

Paramètres

image	

Définition à la ligne 413 du fichier ihmrov.cpp.

Références DISTANCE_MAX_RADAR, Capteurs : :getAngle(), Rov : :getCapteurs(), Capteurs : :getTelemetrie(), et rov.

Référencé par actualiserInformations().

```
00414 {
00415     if(rov->getCapteurs()->getAngle().toDouble() > 90)
00416         return image.width()*(0.875 - 0.125*qSin(qDegreesToRadians(90.0 - (rov->
      getCapteurs()->getAngle().toDouble() - 2 * (rov->
      getCapteurs()->getAngle().toDouble() - 90))))* rov->
      getCapteurs()->getTelemetrie().toDouble()/
      DISTANCE_MAX_RADAR);
00417     else
00418         return image.width()*(0.875 + 0.125*qSin(qDegreesToRadians(90.0 - rov->
      getCapteurs()->getAngle().toDouble()))* rov->getCapteurs()->
      getTelemetrie().toDouble()/DISTANCE_MAX_RADAR);
00419 }
```

6.16.3.6 calculCoordonneesY()

```
double IHMRov::calculCoordonneesY (
            QPixmap & image )  [private]
```

Calcule les coordonnées y de l'obstacle pour le radar.

Renvoie

entier correspondant à la valeur sur l'axe des y du radar d'un objet detecté

Paramètres

image	

Définition à la ligne 421 du fichier ihmrov.cpp.

Références DISTANCE_MAX_RADAR, Capteurs : :getAngle(), Rov : :getCapteurs(), Capteurs : :getTelemetrie(), et rov.

Référencé par actualiserInformations().

```
00422 {
00423     return (image.height() - image.width()*0.125) - image.width()*0.125*qSin(qDegreesToRadians(
      rov->getCapteurs()->getAngle().toDouble()))*rov->
      getCapteurs()->getTelemetrie().toDouble()/
      DISTANCE_MAX_RADAR;
00424 }
```

6.16.3.7 capturerImage

```
void IHMRov::capturerImage (
            bool etat = false )  [slot]
```

Enregistre la dernière image du flux vidéo.

Paramètres

etat	

Définition à la ligne 473 du fichier ihmrov.cpp.

Références Photo : :aGarder, Campagne : :ajouterPhoto(), IHMAccueil : :ajouterPhotoBDD(), campagneEnCours, Photo : :chemin↩ Sauvegarde, Photo : :dateheure, derniereImageVideo, Campagne : :getCheminSauvegarde(), Campagne : :getNomCampagne(), ihmAccueil, Photo : :image, et Campagne : :incrementeNombrePhoto().

Référencé par initialiserEvenements(), et initialiserWidgets().

```
00474 {
00475     #ifndef PAS_DE_MANETTE
00476     if(etat)
00477     #endif
00478     //Q_UNUSED(etat)
00479     {
00480         Photo photo;
00481
00482         photo.image = dernieresImageVideo;
00483         photo.dateheure = QDateTime::currentDateTime();
```

```
00484         photo.aGarder = true;
00485         photo.cheminSauvegarde = campagneEnCours->
      getCheminSauvegarde() + "/" + campagneEnCours->
      getNomCampagne() + "/" + "Capture_" + QString::number(
      campagneEnCours->incrementeNombrePhoto());
00486
00487         campagneEnCours->ajouterPhoto(photo);
00488         qDebug() << Q_FUNC_INFO << "Photo capturée";
00489
00490         photo.image.save(photo.cheminSauvegarde, "PNG");
00491
00492         ihmAccueil->ajouterPhotoBDD(photo,
      campagneEnCours);
00493     }
00494 }
```

6.16.3.8 chargerPhotos

```
void IHMRov::chargerPhotos ( )  [slot]
```

Charge les photos disponible dans le conteneur albumPhoto de la classe campagne pour les afficher dans l'IHMAlbumPhoto.

Définition à la ligne 517 du fichier ihmrov.cpp.

Références campagneEnCours, Campagne : :getAlbumPhoto(), et IHMAlbumPhoto : :ouvrirAlbumPhotos().

Référencé par initialiserEvenements().

```
00518 {
00519     IHMAlbumPhoto *ihmAlbumPhoto = new IHMAlbumPhoto(this);
00520     ihmAlbumPhoto->ouvrirAlbumPhotos(campagneEnCours->
      getAlbumPhoto());
00521 }
```

6.16.3.9 closeEvent()

```
void IHMRov::closeEvent (
            QCloseEvent * event )  [protected]
```

Gère l'état de la campagne lors de la fermeture forcé de la fenêtre ihmRov.

Paramètres

event	

Définition à la ligne 534 du fichier ihmrov.cpp.

Références fermer(), Rov : :getCampagneEncours(), et rov.

```
00535 {
00536     qDebug() << Q_FUNC_INFO << rov->getCampagneEncours();
00537     if(rov->getCampagneEncours())
00538     {
00539         fermer();
00540         event->accept(); // -> close
00541     }
00542     else
00543     {
00544         event->ignore();
00545     }
00546 }
```

6.16.3.10 configurerWidgets()

```
void IHMRov::configurerWidgets ( )  [private]
```

Configure l'état des widgets à la création de l'IHM.

Définition à la ligne 114 du fichier ihmrov.cpp.

Références camera, etatCamera, etatManette, etatPortSerie, fluxVideo, indicateurRadiation, indicateurTemperature, logoEtatCamera, logoEtatManette, logoEtatPortSerie, manette, portSerie, radiation, RATIO, SEUIL_RADIATION_ACCEPTABLE, SEUIL_TEMPERATURE_NEGATIVE_ACCEPTABLE, SEUIL_TEMPERATURE_POSITIVE_ACCEPTABLE, temperature, zoneEtatMateriel, et zoneInformationSeuils.

Référencé par IHMRov().

```
00115 {
00116     int width = int(qApp->desktop()->screen()->width() * RATIO);
00117     int height = int(qApp->desktop()->screen()->height() * RATIO);
00118     fluxVideo->setFixedSize(width, height);
00119     fluxVideo->setScaledContents(true);
00120     fluxVideo->setAlignment(Qt::AlignHCenter | Qt::AlignVCenter);
00121     fluxVideo->setPixmap(QPixmap(qApp->applicationDirPath() +  "/images/signal-interrompu.jpg"));
00122
00123     logoEtatPortSerie->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/inactif.png
      ").scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00124     etatPortSerie->setText("Fermé");
00125
00126     logoEtatCamera->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/inactif.png").
      scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00127     etatCamera->setText("Eteinte");
00128
00129     logoEtatManette->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/inactif.png").
      scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00130     etatManette->setText("Déconnectée");
00131
00132     indicateurTemperature->setScale(QwtInterval(
      SEUIL_TEMPERATURE_NEGATIVE_ACCEPTABLE,
      SEUIL_TEMPERATURE_POSITIVE_ACCEPTABLE));
00133     indicateurRadiation->setScale(QwtInterval(0,
      SEUIL_RADIATION_ACCEPTABLE));
00134
00135     indicateurTemperature->setOrigin(0.);
00136     indicateurTemperature->setOriginMode(QwtThermo::OriginMode::OriginCustom);
00137
00138     zoneEtatMateriel->setStyleSheet("QGroupBox {background: white} QGroupBox{border: 2px
       solid black}");
00139     zoneInformationSeuils->setStyleSheet("QGroupBox {background: white}
       QGroupBox{border: 2px solid black}");
00140     portSerie->setStyleSheet("QLabel {background: white}");
00141     camera->setStyleSheet("QLabel {background: white}");
00142     manette->setStyleSheet("QLabel {background: white}");
00143     etatPortSerie->setStyleSheet("QLabel {background: white}");
00144     etatCamera->setStyleSheet("QLabel {background: white}");
00145     etatManette->setStyleSheet("QLabel {background: white}");
00146     logoEtatPortSerie->setStyleSheet("QLabel {background: white}");
00147     logoEtatCamera->setStyleSheet("QLabel {background: white}");
00148     logoEtatManette->setStyleSheet("QLabel {background: white}");
00149     indicateurRadiation->setStyleSheet("QwtThermo {background: white}");
00150     indicateurTemperature->setStyleSheet("QwtThermo {background: white}");
00151     temperature->setStyleSheet("QLabel {background: white}");
00152     radiation->setStyleSheet("QLabel {background: white}");
00153 }
```

6.16.3.11 fermer()

```
void IHMRov::fermer ( )  [private]
```

Arrête la campagne et ferme l'ihmRov.

Définition à la ligne 548 du fichier ihmrov.cpp.

Références Rov : :arreterCampagne(), boutonCampagne, campagneEnCours, configuration, ihmAccueil, IHMAccueil : :modifierCampagneBDD(), reglageVideo, et rov.

Référencé par closeEvent(), et gererCampagne().

```
00549 {
00550     rov->arreterCampagne();
00551     delete reglageVideo;
00552     delete configuration;
00553     reglageVideo = nullptr;
00554     configuration = nullptr;
00555     boutonCampagne->setText(QString::fromUtf8("Démarrer"));
00556     ihmAccueil->modifierCampagneBDD(campagneEnCours);
00557     this->setVisible(false);
00558     ihmAccueil->setVisible(true);
00559 }
```

6.16.3.12 gererCampagne

```
void IHMRov::gererCampagne ( )  [slot]
```

Arrête la campagne en cours.

Définition à la ligne 496 du fichier ihmrov.cpp.

Références boutonCampagne, configuration, Rov : :demarrerCampagne(), fermer(), reglageVideo, et rov.

Référencé par IHMAccueil : :demarrerCampagne(), et initialiserEvenements().

```
00497 {
00498     qDebug() << Q_FUNC_INFO << boutonCampagne->text();
00499     if(boutonCampagne->text() == QString::fromUtf8("Démarrer"))
00500     {
00501         if(reglageVideo == nullptr)
00502             reglageVideo = new IHMReglageVideo(rov);
00503         if(configuration == nullptr)
00504             configuration = new IHMConfiguration(
      rov);
00505         if(rov->demarrerCampagne())
00506         {
00507             boutonCampagne->setText(QString::fromUtf8("Arrêter"));
00508         }
00509     }
00510     else if(boutonCampagne->text() == QString::fromUtf8("Arrêter"))
00511     {
00512         fermer();
00513         //this->close();
00514     }
00515 }
```

6.16.3.13 getCampagne()

```
Campagne * IHMRov::getCampagne ( )
```

Retourne l'objet campagne en cours.

Renvoie

l'objet campagne en cours

Définition à la ligne 432 du fichier ihmrov.cpp.

Références campagneEnCours.

Référencé par Rov : :arreterCampagne(), Rov : :decoderTrameCapteur(), Rov : :getTempsCampagne(), et IHMAlbumPhoto↩ : :selectionnerPhoto().

```
00433 {
00434     return campagneEnCours;
00435 }
```

6.16.3.14 getEtatRadar()

```
bool IHMRov::getEtatRadar ( )
```

Donne l'etat de etatRadar.

Définition à la ligne 442 du fichier ihmrov.cpp.

Références etatRadar.

Référencé par actualiserInformations().

```
00443 {
00444     return etatRadar;
00445 }
```

6.16.3.15 initialiserEvenements()

```
void IHMRov::initialiserEvenements ( )  [private]
```

Initialise les événements de l'IHM.

Définition à la ligne 239 du fichier ihmrov.cpp.

Références boutonCampagne, boutonConfiguration, boutonReglageVideo, capturerImage(), chargerPhotos(), gererCampagne(), ihmAccueil, photosEnCours, reglerConfiguration(), reglerVideo(), rov, et testCapturePhoto.

Référencé par IHMRov().

```
00240 {
00241     connect(boutonReglageVideo, SIGNAL(clicked()), this, SLOT(
      reglerVideo()));
00242     connect(boutonConfiguration, SIGNAL(clicked()), this, SLOT(
      reglerConfiguration()));
00243     connect(photosEnCours, SIGNAL(clicked()), this, SLOT(
      chargerPhotos()));
00244     #ifdef PAS_DE_MANETTE
00245     connect(testCapturePhoto, SIGNAL(clicked(bool)), this, SLOT(
      capturerImage(bool)));
00246     #endif
00247     connect(boutonCampagne, SIGNAL(clicked()), this, SLOT(
      gererCampagne()));
00248     connect(rov, SIGNAL(enregistrerMesures(QString, QString, QString)),
      ihmAccueil, SLOT(enregisterMesureBDD(QString, QString, QString)));
00249 }
```

6.16.3.16 initialiserEvenementsCamera

```
void IHMRov::initialiserEvenementsCamera ( )  [slot]
```

Initialise les événements liés à la caméra.

Définition à la ligne 523 du fichier ihmrov.cpp.

Références IHMReglageVideo : :initialiserEvenementsCamera(), et reglageVideo.

Référencé par Rov : :demarrerVideo().

```
00524 {
00525     reglageVideo->initialiserEvenementsCamera();
00526 }
```

6.16.3.17 initialiserLayouts()

```
void IHMRov::initialiserLayouts ( )  [private]
```

Initialise les layouts de l'IHM.

Définition à la ligne 155 du fichier ihmrov.cpp.

Références basDePage, boutonCampagne, boutonConfiguration, boutonReglageVideo, camera, etatCamera, etatManette, etatPortSerie, fluxVideo, hautDePage, indicateurRadiation, indicateurTemperature, logoEtatCamera, logoEtatManette, logoEtatPortSerie, manette, photosEnCours, portSerie, radiation, temperature, testCapturePhoto, zoneEtatMateriel, et zoneInformationSeuils.

Référencé par IHMRov().

```
00156 {
00157     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00158     QHBoxLayout *layoutInformationRov = new QHBoxLayout;
00159     QVBoxLayout *layoutCamera = new QVBoxLayout;
00160     QVBoxLayout *layoutOptionVideo = new QVBoxLayout;
00161     QVBoxLayout *layoutGestionCampagne = new QVBoxLayout;
00162     QVBoxLayout *layoutReglageVideo = new QVBoxLayout;
00163     QVBoxLayout *layoutInformationMateriel = new QVBoxLayout;
00164     QHBoxLayout *layoutEtatPortSerie = new QHBoxLayout;
00165     QHBoxLayout *layoutEtatCamera = new QHBoxLayout;
00166     QHBoxLayout *layoutEtatManette = new QHBoxLayout;
00167     QHBoxLayout *layoutInformationSeuils = new QHBoxLayout;
00168     QVBoxLayout *layoutSeuilTemperature = new QVBoxLayout;
00169     QVBoxLayout *layoutSeuilRadiation = new QVBoxLayout;
00170
00171     layoutOptionVideo->setAlignment(Qt::AlignTop);
00172     layoutGestionCampagne->setAlignment(Qt::AlignBottom);
00173     layoutInformationMateriel->setAlignment(Qt::AlignTop);
00174     layoutCamera->addWidget(fluxVideo);
00175
00176     layoutOptionVideo->addWidget(boutonReglageVideo);
00177     layoutOptionVideo->addWidget(boutonConfiguration);
00178     layoutOptionVideo->addWidget(photosEnCours);
00179     #ifdef PAS_DE_MANETTE
00180     layoutOptionVideo->addWidget(testCapturePhoto);
00181     #endif
00182
00183     layoutInformationMateriel->addWidget(portSerie);
00184     layoutInformationMateriel->addLayout(layoutEtatPortSerie);
00185     layoutInformationMateriel->addWidget(camera);
00186     layoutInformationMateriel->addLayout(layoutEtatCamera);
00187     layoutInformationMateriel->addWidget(manette);
00188     layoutInformationMateriel->addLayout(layoutEtatManette);
00189
00190     layoutEtatPortSerie->setAlignment(Qt::AlignLeft);
00191     layoutEtatCamera->setAlignment(Qt::AlignLeft);
00192     layoutEtatManette->setAlignment(Qt::AlignLeft);
00193
00194     layoutEtatPortSerie->addWidget(logoEtatPortSerie);
00195     layoutEtatPortSerie->addWidget(etatPortSerie);
00196
00197     layoutEtatCamera->addWidget(logoEtatCamera);
00198     layoutEtatCamera->addWidget(etatCamera);
00199
00200     layoutEtatManette->addWidget(logoEtatManette);
00201     layoutEtatManette->addWidget(etatManette);
00202
00203     layoutSeuilTemperature->addWidget(temperature);
00204     layoutSeuilTemperature->addWidget(indicateurTemperature);
00205
00206     layoutSeuilRadiation->addWidget(radiation);
00207     layoutSeuilRadiation->addWidget(indicateurRadiation);
00208
00209     layoutInformationSeuils->addLayout(layoutSeuilTemperature);
00210     layoutInformationSeuils->addLayout(layoutSeuilRadiation);
00211
00212     layoutPrincipal->addWidget(hautDePage);
00213     layoutPrincipal->addLayout(layoutInformationRov);
00214     layoutInformationRov->addLayout(layoutCamera);
00215     layoutInformationRov->addStretch();
00216     layoutInformationRov->addLayout(layoutReglageVideo);
00217     layoutReglageVideo->addLayout(layoutOptionVideo);
00218
00219     //layoutReglageVideo->addLayout(layoutInformationMateriel);
00220     zoneEtatMateriel->setLayout(layoutInformationMateriel);
00221     layoutReglageVideo->addWidget(zoneEtatMateriel);
00222     //layoutReglageVideo->addLayout(layoutInformationSeuils);
00223     zoneInformationSeuils->setLayout(layoutInformationSeuils);
00224     layoutReglageVideo->addWidget(zoneInformationSeuils);
```

```
00225
00226     layoutReglageVideo->addLayout(layoutGestionCampagne);
00227     layoutGestionCampagne->addWidget(boutonCampagne);
00228     layoutPrincipal->addWidget(basDePage);
00229
00230     setLayout(layoutPrincipal);
00231     resize(width(), fluxVideo->maximumHeight());
00232     //setStyleSheet("background:#101010;");
00233     setStyleSheet("background:#C1BEBE;");
00234
00235     setWindowFlags(windowFlags() & ~Qt::WindowCloseButtonHint);
00236     showMinimized();
00237 }
```

6.16.3.18 initialiserWidgets()

```
void IHMRov::initialiserWidgets ( )  [private]
```

Initialise les widgets de l'IHM.

Définition à la ligne 37 du fichier ihmrov.cpp.

Références basDePage, boutonCampagne, boutonConfiguration, boutonReglageVideo, camera, capturerImage(), etatCamera, etatManette, etatPortSerie, fluxVideo, hautDePage, indicateurRadiation, indicateurTemperature, logoEtatCamera, logoEtatManette, logoEtatPortSerie, manette, photosEnCours, portSerie, radiation, temperature, testCapturePhoto, zoneEtatMateriel, et zone↩InformationSeuils.

Référencé par IHMRov().

```
00038 {
00039     fluxVideo = new QLabel("Aucune image détéctée",this);
00040     photosEnCours = new QPushButton("Album Photo", this);
00041     boutonReglageVideo = new QPushButton("Réglages Vidéo", this);
00042     boutonCampagne = new QPushButton(QString::fromUtf8("Démarrer"), this);
00043     boutonConfiguration = new QPushButton("Communication", this);
00044     hautDePage = new QLabel(this);
00045     basDePage = new QLabel(this);
00046     logoEtatPortSerie = new QLabel(this);
00047     logoEtatCamera = new QLabel(this);
00048     logoEtatManette = new QLabel(this);
00049     etatPortSerie = new QLabel(this);
00050     etatCamera = new QLabel(this);
00051     etatManette = new QLabel(this);
00052     portSerie = new QLabel("Port série :", this);
00053     camera = new QLabel("Caméra :", this);
00054     manette = new QLabel("Manette :", this);
00055     indicateurTemperature = new QwtThermo(this);
00056     indicateurRadiation = new QwtThermo(this);
00057     zoneEtatMateriel = new QGroupBox(this);
00058     zoneInformationSeuils = new QGroupBox(this);
00059     temperature = new QLabel("Température\n", this);
00060     radiation = new QLabel("Radiation\n", this);
00061
00062     QFont police1("", 15, 75, false);
00063     QFont police2("Cursive", 12, 40, true);
00064
00065     portSerie->setFont(police1);
00066     camera->setFont(police1);
00067     manette->setFont(police1);
00068     etatPortSerie->setFont(police2);
00069     etatCamera->setFont(police2);
00070     etatManette->setFont(police2);
00071     temperature->setFont(police2);
00072     radiation->setFont(police2);
00073
00074     fluxVideo->setFixedSize(230,50);
00075     fluxVideo->setFont(police1);
00076
00077     photosEnCours->setFixedSize(230,50);
00078     photosEnCours->setFont(police1);
00079     photosEnCours->setStyleSheet("QPushButton {border-image: url(design/bouton_230x50.png)}" "
      QPushButton:hover {border-image: url(design/bouton_230x50_survole.png)}");
00080     //photosEnCours->setStyleSheet("QPushButton {border-image: url(design/bouton_230x50_survole.png)}"
       "QPushButton:hover {border-image: url(design/bouton_230x50.png)}");
00081
00082     boutonReglageVideo->setFixedSize(230,50);
00083     boutonReglageVideo->setFont(police1);
```

```
00084     boutonReglageVideo->setStyleSheet("QPushButton {border-image:
      url(design/bouton_230x50.png)}" "QPushButton:hover {border-image: url(design/bouton_230x50_survole.png)}");
00085     //boutonReglageVideo->setStyleSheet("QPushButton {border-image: url(design/bouton_230x50_survole.png)}"
      "QPushButton:hover {border-image: url(design/bouton_230x50.png)}");
00086
00087     boutonCampagne->setFixedSize(230,50);
00088     boutonCampagne->setFont(police1);
00089     boutonCampagne->setStyleSheet("QPushButton {border-image: url(design/bouton_230x50.png)}"
      "QPushButton:hover {border-image: url(design/bouton_230x50_survole.png)}");
00090     //boutonCampagne->setStyleSheet("QPushButton {border-image: url(design/bouton_230x50_survole.png)}"
      "QPushButton:hover {border-image: url(design/bouton_230x50.png)}");
00091
00092     boutonConfiguration->setFixedSize(230,50);
00093     boutonConfiguration->setFont(police1);
00094     boutonConfiguration->setStyleSheet("QPushButton {border-image:
      url(design/bouton_230x50.png)}" "QPushButton:hover {border-image: url(design/bouton_230x50_survole.png)}");
00095     //boutonConfiguration->setStyleSheet("QPushButton {border-image:
      url(design/bouton_230x50_survole.png)}" "QPushButton:hover {border-image: url(design/bouton_230x50.png)}");
00096
00097     hautDePage->setMinimumHeight(1);
00098     //hautDePage->setStyleSheet("QLabel {border-image: url(design/fond_noir.png)}");
00099     basDePage->setMinimumHeight(1);
00100     //basDePage->setStyleSheet("QLabel {border-image: url(design/fond_noir.png)}");
00101
00102     #ifdef PAS_DE_MANETTE
00103     testCapturePhoto = new QPushButton("Capturer", this);
00104     testCapturePhoto->setFixedSize(230,50);
00105     //testCapturePhoto->setFont(police);
00106     testCapturePhoto->setStyleSheet("QPushButton {border-image:
      url(design/bouton_230x50.png)}" "QPushButton:hover {border-image: url(design/bouton_230x50_survole.png)}");
00107     QAction *actionCapturerPhoto = new QAction(this);
00108     actionCapturerPhoto->setShortcut(QKeySequence(Qt::Key_C));
00109     addAction(actionCapturerPhoto);
00110     connect(actionCapturerPhoto, SIGNAL(triggered()), this, SLOT(capturerImage()));
00111     #endif
00112 }
```

6.16.3.19 modifieEtatCamera

```
void IHMRov::modifieEtatCamera (
            bool etat,
            QString information )  [slot]
```

Modifie l'affichage de l'état de la caméra.

Paramètres

etat	

Définition à la ligne 576 du fichier ihmrov.cpp.

Références etatCamera, et logoEtatCamera.

Référencé par Rov : :arreterVideo(), et Rov : :demarrerVideo().

```
00577 {
00578     if(etat)
00579     {
00580         logoEtatCamera->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/actif.png").
      scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00581         etatCamera->setText(information);
00582     }
00583     else
00584     {
00585         logoEtatCamera->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/inactif.png")
      .scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00586         etatCamera->setText("Eteinte");
00587     }
00588 }
```

6.16.3.20 modifieEtatManette

```
void IHMRov::modifieEtatManette (
            bool etat )  [slot]
```

Modifie l'affichage de l'état de la manette.

Paramètres

etat	

Définition à la ligne 590 du fichier ihmrov.cpp.

Références etatManette, et logoEtatManette.

Référencé par IHMRov().

```
00591 {
00592     if(etat)
00593     {
00594         logoEtatManette->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/actif.png")
      .scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00595         etatManette->setText("Connectée");
00596     }
00597     else
00598     {
00599         logoEtatManette->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/inactif.png
      ").scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00600         etatManette->setText("Déconnectée");
00601     }
00602 }
```

6.16.3.21 modifieEtatPortSerie

```
void IHMRov::modifieEtatPortSerie (
            bool etat,
            QString information )  [slot]
```

Modifie l'affichage de l'état du port série.

Paramètres

etat	
information	

Définition à la ligne 561 du fichier ihmrov.cpp.

Références configuration, etatPortSerie, logoEtatPortSerie, et IHMConfiguration : :modifieEtatBoutons().

```
00562 {
00563     if(etat)
00564     {
00565         logoEtatPortSerie->setPixmap(QPixmap(qApp->applicationDirPath() + "
      /images/actif.png").scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00566         etatPortSerie->setText(information);
00567     }
00568     else
00569     {
00570         configuration->modifieEtatBoutons();
00571         logoEtatPortSerie->setPixmap(QPixmap(qApp->applicationDirPath() + "
```

```
      /images/inactif.png").scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00572         etatPortSerie->setText("Fermé");
00573     }
00574 }
```

6.16.3.22 reglerConfiguration

```
void IHMRov::reglerConfiguration ( )  [slot]
```

Ouvre une nouvelle fenetre permettant de régler la communication.

Définition à la ligne 463 du fichier ihmrov.cpp.

Références IHMConfiguration : :actualisePortsDisponibles(), et configuration.

Référencé par initialiserEvenements().

```
00464 {
00465     if(configuration != nullptr)
00466     {
00467         configuration->actualisePortsDisponibles();
00468         configuration->show();
00469         configuration->raise();
00470     }
00471 }
```

6.16.3.23 reglerVideo

```
void IHMRov::reglerVideo ( )  [slot]
```

Ouvre une nouvelle fenetre permettant de régler l'affichage vidéo.

Définition à la ligne 454 du fichier ihmrov.cpp.

Références reglageVideo.

Référencé par initialiserEvenements().

```
00455 {
00456     if(reglageVideo != nullptr)
00457     {
00458         reglageVideo->show();
00459         reglageVideo->raise();
00460     }
00461 }
```

6.16.3.24 setCampagne()

```
void IHMRov::setCampagne (
             Campagne * campagne )
```

Associe une campagne a la campagne en cours du rov.

Paramètres

campagne	

Définition à la ligne 426 du fichier ihmrov.cpp.

Références campagneEnCours, Campagne : :getDate(), Campagne : :getNomCampagne(), et NOM_FENETRE_ROV.

Référencé par IHMAccueil : :demarrerCampagne().

```
00427 {
00428     campagneEnCours = campagne;
00429     setWindowTitle(NOM_FENETRE_ROV  " " + campagne->
      getNomCampagne() + " " + campagne->getDate().toString());
00430 }
```

6.16.3.25 setEtatRadar()

```
void IHMRov::setEtatRadar (
            bool etatRadar )
```

Dertermine l'etat de etatRadar.

Paramètres

etatRadar	

Définition à la ligne 437 du fichier ihmrov.cpp.

Références etatRadar.

Référencé par Rov : :creerTrameDeplacement().

```
00438 {
00439     this->etatRadar = etatRadar;
00440 }
```

6.16.4 Documentation des données membres

6.16.4.1 basDePage

```
QLabel* IHMRov::basDePage  [private]
```

Emplacement permettant de créer un espace en bas de la page.

Définition à la ligne 100 du fichier ihmrov.h.

Référencé par initialiserLayouts(), et initialiserWidgets().

6.16.4.2 boutonCampagne

```
QPushButton* IHMRov::boutonCampagne  [private]
```

Bouton permettant de mettre en pause la campagne en cours.

Définition à la ligne 97 du fichier ihmrov.h.

Référencé par fermer(), gererCampagne(), initialiserEvenements(), initialiserLayouts(), et initialiserWidgets().

6.16.4.3 boutonConfiguration

```
QPushButton* IHMRov::boutonConfiguration  [private]
```

Bouton permettant d'accéder à la configuration de la communication.

Définition à la ligne 98 du fichier ihmrov.h.

Référencé par initialiserEvenements(), initialiserLayouts(), et initialiserWidgets().

6.16.4.4 boutonReglageVideo

```
QPushButton* IHMRov::boutonReglageVideo  [private]
```

Bouton permettant d'accéder aux reglage de la vidéo.

Définition à la ligne 92 du fichier ihmrov.h.

Référencé par initialiserEvenements(), initialiserLayouts(), et initialiserWidgets().

6.16.4.5 camera

```
QLabel* IHMRov::camera  [private]
```

Emplacement permettant de définir le type de matériel.

Définition à la ligne 108 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), et initialiserWidgets().

6.16.4.6 campagneEnCours

```
Campagne* IHMRov::campagneEnCours  [private]
```

Instance d'un objet Campagne possédant les informations de la campagne en cours.

Définition à la ligne 85 du fichier ihmrov.h.

Référencé par capturerImage(), chargerPhotos(), fermer(), getCampagne(), et setCampagne().

6.16.4.7 configuration

```
IHMConfiguration* IHMRov::configuration  [private]
```

Instance d'un objet IHMConfiguration permettant de modifier les reglages de la communication.

Définition à la ligne 89 du fichier ihmrov.h.

Référencé par fermer(), gererCampagne(), IHMRov(), modifieEtatPortSerie(), et reglerConfiguration().

6.16.4.8 derniereImageVideo

```
QPixmap IHMRov::dernieреImageVideo  [private]
```

Dernière image reçue du flux vidéo.

Définition à la ligne 93 du fichier ihmrov.h.

Référencé par afficherImage(), et capturerImage().

6.16.4.9 etatCamera

```
QLabel* IHMRov::etatCamera  [private]
```

Emplacement permettant de visualiser l'état de la caméra.

Définition à la ligne 105 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), initialiserWidgets(), et modifieEtatCamera().

6.16.4.10 etatManette

```
QLabel* IHMRov::etatManette  [private]
```

Emplacement permettant de visualiser l'état de la manette.

Définition à la ligne 106 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), initialiserWidgets(), et modifieEtatManette().

6.16.4.11 etatPortSerie

```
QLabel* IHMRov::etatPortSerie  [private]
```

Emplacement permettant de visualiser l'état du port série.

Définition à la ligne 104 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), initialiserWidgets(), et modifieEtatPortSerie().

6.16.4.12 etatRadar

```
bool IHMRov::etatRadar  [private]
```

Dertermine si on affiche un radar.

Définition à la ligne 110 du fichier ihmrov.h.

Référencé par getEtatRadar(), et setEtatRadar().

6.16.4.13 fluxVideo

```
QLabel* IHMRov::fluxVideo  [private]
```

Emplacement permettant d'accueiller le flux vidéo.

Définition à la ligne 90 du fichier ihmrov.h.

Référencé par afficherImage(), arreterVideo(), configurerWidgets(), initialiserLayouts(), et initialiserWidgets().

6.16.4.14 hautDePage

```
QLabel* IHMRov::hautDePage  [private]
```

Emplacement permettant de créer un espace en haut de la page.

Définition à la ligne 99 du fichier ihmrov.h.

Référencé par initialiserLayouts(), et initialiserWidgets().

6.16.4.15 ihmAccueil

```
IHMAccueil* IHMRov::ihmAccueil  [private]
```

Relation entre l'ihmAccueil et l'ihmRov.

Définition à la ligne 86 du fichier ihmrov.h.

Référencé par capturerImage(), fermer(), et initialiserEvenements().

6.16.4.16 indicateurRadiation

```
QwtThermo* IHMRov::indicateurRadiation  [private]
```

Indicateur permettant de visualiser l'etat de la radiation actuel avec indication de dépassement de seuil.

Définition à la ligne 112 du fichier ihmrov.h.

Référencé par actualiserInformationsSeuils(), configurerWidgets(), initialiserLayouts(), et initialiserWidgets().

6.16.4.17 indicateurTemperature

```
QwtThermo* IHMRov::indicateurTemperature  [private]
```

Indicateur permettant de visualiser l'etat de la temperature actuel avec indication de dépassement de seuil.

Définition à la ligne 111 du fichier ihmrov.h.

Référencé par actualiserInformationsSeuils(), configurerWidgets(), initialiserLayouts(), et initialiserWidgets().

6.16.4.18 logoEtatCamera

```
QLabel* IHMRov::logoEtatCamera  [private]
```

Emplacement permettant de visualiser l'état de la caméra à l'aide d'un logo.

Définition à la ligne 102 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), initialiserWidgets(), et modifieEtatCamera().

6.16.4.19 logoEtatManette

```
QLabel* IHMRov::logoEtatManette  [private]
```

Emplacement permettant de visualiser l'état de la manette à l'aide d'un logo.

Définition à la ligne 103 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), initialiserWidgets(), et modifieEtatManette().

6.16.4.20 logoEtatPortSerie

```
QLabel* IHMRov::logoEtatPortSerie  [private]
```

Emplacement permettant de visualiser l'état du port série à l'aide d'un logo.

Définition à la ligne 101 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), initialiserWidgets(), et modifieEtatPortSerie().

6.16.4.21 manette

```
QLabel* IHMRov::manette  [private]
```

Emplacement permettant de définir le type de matériel.

Définition à la ligne 109 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), et initialiserWidgets().

6.16.4.22 photosEnCours

```
QPushButton* IHMRov::photosEnCours  [private]
```

Bouton permettant d'accéder aux photo prise en cours de campagne.

Définition à la ligne 91 du fichier ihmrov.h.

Référencé par initialiserEvenements(), initialiserLayouts(), et initialiserWidgets().

6.16.4.23 pointsRadar

```
QVector<QPoint> IHMRov::pointsRadar  [private]
```

Conteneur des points du radar.

Définition à la ligne 117 du fichier ihmrov.h.

Référencé par actualiserInformations().

6.16.4.24 portSerie

```
QLabel* IHMRov::portSerie  [private]
```

Emplacement permettant de définir le type de matériel.

Définition à la ligne 107 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), et initialiserWidgets().

6.16.4.25 radiation

```
QLabel* IHMRov::radiation  [private]
```

Emplacement permettant de définir le type de seuil.

Définition à la ligne 114 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), et initialiserWidgets().

6.16.4.26 reglageVideo

```
IHMReglageVideo* IHMRov::reglageVideo  [private]
```

Instance d'un objet reglageVidéo permettant de modifier les reglages du flux vidéo.

Définition à la ligne 88 du fichier ihmrov.h.

Référencé par arreterVideo(), fermer(), gererCampagne(), IHMRov(), initialiserEvenementsCamera(), et reglerVideo().

6.16.4.27 rov

```
Rov* IHMRov::rov  [private]
```

Instance d'un objet rov possédant le controle sur les autres classes.

Définition à la ligne 87 du fichier ihmrov.h.

Référencé par actualiserInformations(), actualiserInformationsSeuils(), calculCoordonneesX(), calculCoordonneesY(), closeEvent(), fermer(), gererCampagne(), IHMRov(), et initialiserEvenements().

6.16.4.28 temperature

```
QLabel* IHMRov::temperature  [private]
```

Emplacement permettant de définir le type de seuil.

Définition à la ligne 113 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), et initialiserWidgets().

6.16.4.29 testCapturePhoto

```
QPushButton* IHMRov::testCapturePhoto  [private]
```

Bouton de simulation de prise de photo.

Définition à la ligne 95 du fichier ihmrov.h.

Référencé par initialiserEvenements(), initialiserLayouts(), et initialiserWidgets().

6.16.4.30 zoneEtatMateriel

```
QGroupBox* IHMRov::zoneEtatMateriel  [private]
```

Zone regroupant les informations sur l'état du matériel.

Définition à la ligne 115 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), et initialiserWidgets().

6.16.4.31 zoneInformationSeuils

```
QGroupBox* IHMRov::zoneInformationSeuils  [private]
```

Zone regroupant les informations sur l'état des seuils de dépassement.

Définition à la ligne 116 du fichier ihmrov.h.

Référencé par configurerWidgets(), initialiserLayouts(), et initialiserWidgets().

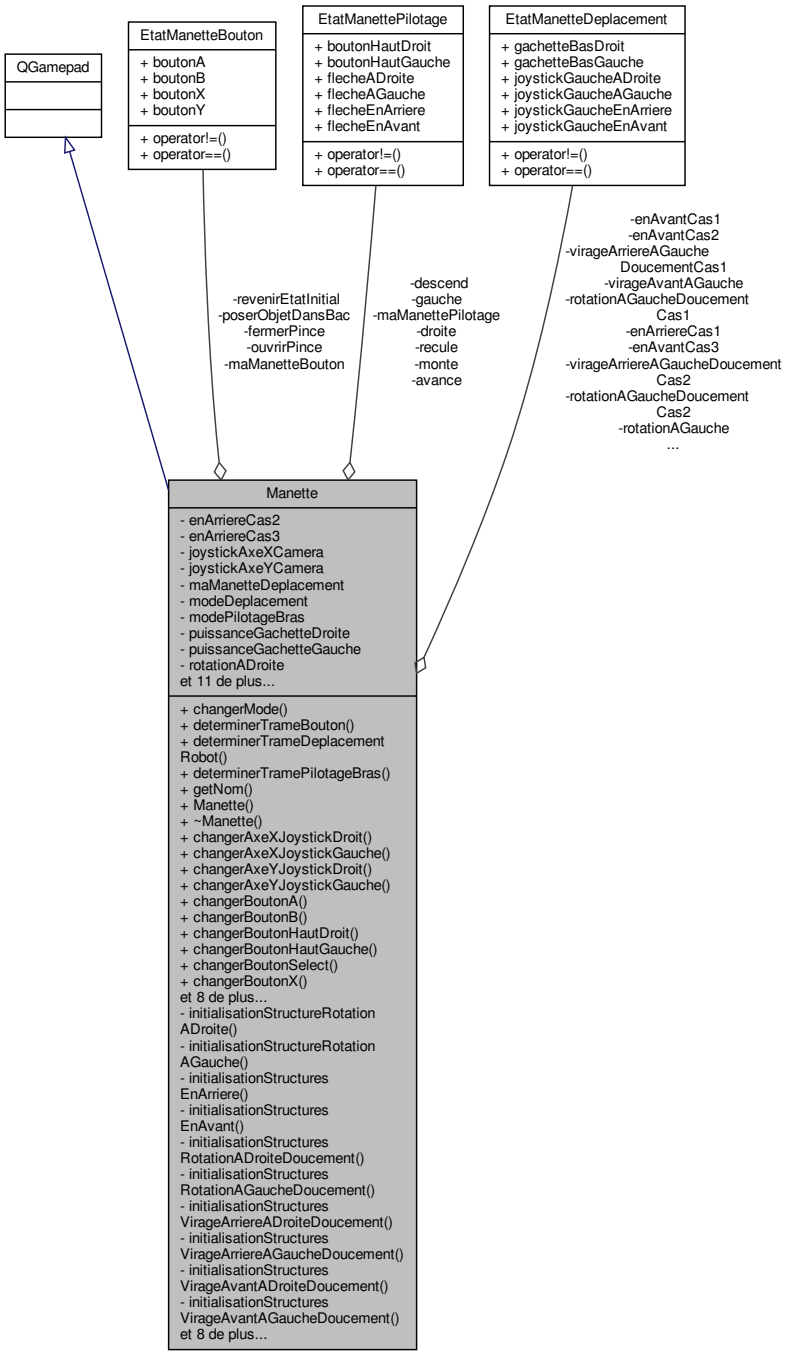
La documentation de cette classe a été générée à partir des fichiers suivants :
- ihmrov.h
- ihmrov.cpp

6.17 Référence de la classe Manette

Classe permettant une communication entre le rov et la manette.

`#include "manette.h"`

Graphe de collaboration de Manette :



Connecteurs publics

— void changerAxeXJoystickDroit (double valeur)

change l'etat de l'attribut joystickAxeXCamera en fonction du signe de "valeur"
- void changerAxeXJoystickGauche (double valeur)
  change l'etat des attributs joystickGaucheAGauche et joystickGaucheADroite en fonction du signe de "valeur"
- void changerAxeYJoystickDroit (double valeur)
  change l'etat de l'attribut joystickAxeYCamera en fonction du signe de "valeur"
- void changerAxeYJoystickGauche (double valeur)
  change l'etat des attributs joystickGaucheEnAvant et joystickGaucheEnArriere en fonction du signe de "valeur"
- void changerBoutonA (bool)
  change l'état du membre boutonA
- void changerBoutonB (bool)
  change l'état du membre boutonB
- void changerBoutonHautDroit (bool etat)
  change l'etat des attributs boutonHautDroit
- void changerBoutonHautGauche (bool etat)
  change l'etat des attributs boutonHautGauche
- void changerBoutonSelect (bool etat)
  change l'etat des attributs modeDeplacement modePilotageBras en fonction de "etat"
- void changerBoutonX (bool)
  change l'état du membre boutonX
- void changerBoutonY (bool)
  change l'état du membre boutonY
- void changerFlecheADroite (bool etat)
  change l'etat de l'attribut flecheADroite en fonction de "etat"
- void changerFlecheAGauche (bool etat)
  change l'etat de l'attribut flecheAGauche en fonction de "valeur"
- void changerFlecheEnArriere (bool etat)
  change l'etat de l'attribut flecheEnArriere en fonction de "valeur"
- void changerFlecheEnAvant (bool etat)
  change l'etat de l'attribut flecheEnAvant en fonction de "valeur"
- void changerGachetteBasDroit (double valeur)
  change l'etat des attributs gachetteBasDroit et puissance en fonction de "valeur"
- void changerGachetteBasGauche (double valeur)
  change l'etat des attributs gachetteBasGauche et puissance en fonction de "valeur"
- void fermerApplication (bool etat)
  ferme l'application

Signaux

- void creationTrameDeplacement (char deplacementAxeX, int puissance, char deplacementAxeY)
  Envoye des élément de la trame pour la création de la trame de déplacement.
- void creationTrameOrdre (QString ordre)
  Envoye des élément de la trame pour la création de la trame des ordres.
- void creationTramePilotage (QString direction)
  Envoye des élément de la trame pour la création de la trame de pilotage.
- void creationTramePince (QString mouvementPince)
  Envoye des élément de la trame pour la création de la trame d'ouverture et fermeture de la pince.
- void nouvelleTrameCamera (QString axeY, QString axeX)
  Nouvelle trame de commande de la caméra disponible.
- void prendrePhoto ()
  Envoi un signal indiquant que le bouton photo est pressé

Fonctions membres publiques

- void changerMode ()
  Change de mode en fonction du bouton SELECT.
- void determinerTrameBouton ()
  Verifie l'etat de la manette et creer la trame correspondante.
- void determinerTrameDeplacementRobot ()
  Verifie l'etat de la manette et creer la trame correspondante.
- void determinerTramePilotageBras ()
  Verifie l'etat de la manette et creer la trame correspondante.
- QString getNom ()
  Retourne le nom de la manette.
- Manette (int deviceId)
  Contructeur de la classe Manette.
- ~Manette ()
  Destructeur de la classe Manette.

Fonctions membres privées

- void initialisationStructureRotationADroite ()
  Initialise les états de la structures rotationADroite.
- void initialisationStructureRotationAGauche ()
  Initialise les états de la structures rotationAGauche.
- void initialisationStructuresEnArriere ()
  Initialise les états de la structures enArriere.
- void initialisationStructuresEnAvant ()
  Initialise les états de la structures enAvant.
- void initialisationStructuresRotationADroiteDoucement ()
  Initialise les états de la structures rotationADroiteDoucement.
- void initialisationStructuresRotationAGaucheDoucement ()
  Initialise les états de la structures rotationAGaucheDoucement.
- void initialisationStructuresVirageArriereADroiteDoucement ()
  Initialise les états de la structures virageArriereADroiteDoucement.
- void initialisationStructuresVirageArriereAGaucheDoucement ()
  Initialise les états de la structures virageArriereAGaucheDoucement.
- void initialisationStructuresVirageAvantADroiteDoucement ()
  Initialise les états de la structures virageAvantADroite.
- void initialisationStructuresVirageAvantAGaucheDoucement ()
  Initialise les états de la structures virageAvantAGaucheDoucement.
- void initialisationStructureVirageArriereADroite ()
  Initialise les états de la structures virageArriereADroite.
- void initialisationStructureVirageArriereAGauche ()
  Initialise les états de la structures virageArriereAGauche.
- void initialisationStructureVirageAvantADroite ()
  Initialise les états de la structures.
- void initialisationStructureVirageAvantAGauche ()
  Initialise les états de la structures virageAvantAGauche.
- void initialiserEtatBouton ()
  Initialise les états des structures EtatBouton.
- void initialiserEtats ()
  Initialise les états de la manette.
- void initialiserTypesDeplacement ()
  Initialise les états de la manette pour le déplacement.
- void initialiserTypesPilotage ()
  Initialise les états de la manette pour le pilotage.

Attributs privés

- EtatManettePilotage avance
  Structure définissant l'état des bouton de la manette pour avancer.
- EtatManettePilotage descend
  Structure définissant l'état des bouton de la manette pour descendre.
- EtatManettePilotage droite
  Structure définissant l'état des bouton de la manette pour aller à droite.
- EtatManetteDeplacement enArriereCas1
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer.
- EtatManetteDeplacement enArriereCas2
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer.
- EtatManetteDeplacement enArriereCas3
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer.
- EtatManetteDeplacement enAvantCas1
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer.
- EtatManetteDeplacement enAvantCas2
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer.
- EtatManetteDeplacement enAvantCas3
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer.
- EtatManetteBouton fermerPince
  Structure définissant l'etat des bouton A, B, X, Y de la manette pour fermer la pince.
- EtatManettePilotage gauche
  Structure définissant l'état des bouton de la manette pour aller à gauche.
- QString joystickAxeXCamera
  Attribut contenant l'etat actuel du joystick droite sur l'axe des x.
- QString joystickAxeYCamera
  Attribut contenant l'etat actuel du joystick droite sur l'axe des y.
- EtatManetteBouton maManetteBouton
  Structure définissant l'etat des bouton A, B, X, Y de la manette.
- EtatManetteDeplacement maManetteDeplacement

Structure définissant l'état des bouton de la manette en mode deplacement du robot.
- EtatManettePilotage maManettePilotage
  Structure définissant l'état des bouton de la manette en mode pilotage du bras.
- bool modeDeplacement
  Attribut définissant l'état du mode de déplacement.
- bool modePilotageBras
  Attribut définissant l'état du mode pilotage du bras articulé
- EtatManettePilotage monte
  Structure définissant l'état des bouton de la manette pour monter.
- EtatManetteBouton ouvrirPince
  Structure définissant l'etat des bouton A, B, X, Y de la manette pour ouvrir la pince.
- EtatManetteBouton poserObjetDansBac
  Structure définissant l'etat des bouton A, B, X, Y de la manette pour poser l'objet dans le bac.
- int puissanceGachetteDroite
  Attribut définissant la valeur de la puissance de la gachette droite.
- int puissanceGachetteGauche
  Attribut définissant la valeur de la puissance de la gachette gauche.
- EtatManettePilotage recule
  Structure définissant l'état des bouton de la manette pour reculer.
- EtatManetteBouton revenirEtatInitial
  Structure définissant l'etat des bouton A, B, X, Y de la manette pour revenir à l'etat initiale du bras.
- EtatManetteDeplacement rotationADroite
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à droite.
- EtatManetteDeplacement rotationADroiteDoucementCas1
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à droite doucement.
- EtatManetteDeplacement rotationADroiteDoucementCas2
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à droite doucement.
- EtatManetteDeplacement rotationAGauche
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à gauche.
- EtatManetteDeplacement rotationAGaucheDoucementCas1
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à gauche doucement.
- EtatManetteDeplacement rotationAGaucheDoucementCas2
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à gauche doucement.
- EtatManetteDeplacement virageArriereADroite
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à droite.
- EtatManetteDeplacement virageArriereADroiteDoucementCas1
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à droite doucement.
- EtatManetteDeplacement virageArriereADroiteDoucementCas2
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à droite doucement.
- EtatManetteDeplacement virageArriereAGauche
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à gauche.
- EtatManetteDeplacement virageArriereAGaucheDoucementCas1
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à gauche doucement.
- EtatManetteDeplacement virageArriereAGaucheDoucementCas2
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à gauche doucement.
- EtatManetteDeplacement virageAvantADroite
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à droite.
- EtatManetteDeplacement virageAvantADroiteDoucementCas1
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à droite doucement.
- EtatManetteDeplacement virageAvantADroiteDoucementCas2
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à droite doucement.
- EtatManetteDeplacement virageAvantAGauche
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à gauche.
- EtatManetteDeplacement virageAvantAGaucheDoucementCas1
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à gauche doucement.
- EtatManetteDeplacement virageAvantAGaucheDoucementCas2
  Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à gauche doucement.

6.17.1 Description détaillée

Classe permettant une communication entre le rov et la manette.

Définition à la ligne 210 du fichier manette.h.

6.17.2 Documentation des constructeurs et destructeur

6.17.2.1 Manette()

```
Manette::Manette (
            int deviceId )  [explicit]
```

Contructeur de la classe Manette.

Paramètres

deviceId	

Définition à la ligne 9 du fichier manette.cpp.

Références initialiserEtatBouton(), initialiserEtats(), initialiserTypesDeplacement(), et initialiserTypesPilotage().

```
00009                             : QGamepad(deviceID), modeDeplacement(true),
      modePilotageBras(false), puissanceGachetteDroite(0),
      puissanceGachetteGauche(0), joystickAxeXCamera("0"),
      joystickAxeYCamera("0")
00010 {
00011     qDebug() << Q_FUNC_INFO << this;
00012     initialiserEtats();
00013     initialiserEtatBouton();
00014     initialiserTypesPilotage();
00015     initialiserTypesDeplacement();
00016 }
```

6.17.2.2 ~Manette()

```
Manette::~Manette ( )
```

Destructeur de la classe Manette.

Définition à la ligne 18 du fichier manette.cpp.

```
00019 {
00020     qDebug() << Q_FUNC_INFO << this;
00021 }
```

6.17.3 Documentation des fonctions membres

6.17.3.1 changerAxeXJoystickDroit

```
void Manette::changerAxeXJoystickDroit (
            double valeur )  [slot]
```

change l'etat de l'attribut joystickAxeXCamera en fonction du signe de "valeur"

Paramètres

valeur	

Définition à la ligne 455 du fichier manette.cpp.

Références joystickAxeXCamera, joystickAxeYCamera, nouvelleTrameCamera(), et TAUX_VALIDITE_JOYSTICK.

```
00456 {
00457     if(valeur >= TAUX_VALIDITE_JOYSTICK)
00458         joystickAxeXCamera = "D";
00459     else if(valeur <= -TAUX_VALIDITE_JOYSTICK)
00460         joystickAxeXCamera = "G";
00461     else
00462         joystickAxeXCamera = "0";
00463 
00464     emit nouvelleTrameCamera(joystickAxeXCamera,
      joystickAxeYCamera);
00465 }
```

6.17.3.2 changerAxeXJoystickGauche

```
void Manette::changerAxeXJoystickGauche (
             double valeur )  [slot]
```

change l'etat des attributs joystickGaucheAGauche et joystickGaucheADroite en fonction du signe de "valeur"

Paramètres

valeur	

Définition à la ligne 421 du fichier manette.cpp.

Références determinerTrameDeplacementRobot(), EtatManetteDeplacement : :joystickGaucheADroite, EtatManetteDeplacement↩ : :joystickGaucheAGauche, maManetteDeplacement, modeDeplacement, et TAUX_VALIDITE_JOYSTICK.

```
00422 {
00423     if(modeDeplacement)
00424     {
00425         if(valeur <= -TAUX_VALIDITE_JOYSTICK)
00426             maManetteDeplacement.joystickGaucheAGauche = true;
00427         else if(valeur >= TAUX_VALIDITE_JOYSTICK)
00428             maManetteDeplacement.joystickGaucheADroite = true;
00429         else if(valeur > -TAUX_VALIDITE_JOYSTICK && valeur <
      TAUX_VALIDITE_JOYSTICK)
00430         {
00431             maManetteDeplacement.joystickGaucheADroite = false;
00432             maManetteDeplacement.joystickGaucheAGauche = false;
00433         }
00434         determinerTrameDeplacementRobot();
00435     }
00436 }
```

6.17.3.3 changerAxeYJoystickDroit

```
void Manette::changerAxeYJoystickDroit (
             double valeur )  [slot]
```

change l'etat de l'attribut joystickAxeYCamera en fonction du signe de "valeur"

Paramètres

valeur	

Définition à la ligne 467 du fichier manette.cpp.

Références joystickAxeXCamera, joystickAxeYCamera, nouvelleTrameCamera(), et TAUX_VALIDITE_JOYSTICK.

```
00468 {
00469     if(valeur >= TAUX_VALIDITE_JOYSTICK)
00470         joystickAxeYCamera = "B";
00471     else if(valeur <= -TAUX_VALIDITE_JOYSTICK)
00472         joystickAxeYCamera = "H";
00473     else
00474         joystickAxeYCamera = "0";
00475
00476     emit nouvelleTrameCamera(joystickAxeXCamera,
      joystickAxeYCamera);
00477 }
```

6.17.3.4 changerAxeYJoystickGauche

```
void Manette::changerAxeYJoystickGauche (
            double valeur )  [slot]
```

change l'etat des attributs joystickGaucheEnAvant et joystickGaucheEnArriere en fonction du signe de "valeur"

Paramètres

valeur	

Définition à la ligne 438 du fichier manette.cpp.

Références determinerTrameDeplacementRobot(), EtatManetteDeplacement : :joystickGaucheEnArriere, EtatManetteDeplacement↩ : :joystickGaucheEnAvant, maManetteDeplacement, modeDeplacement, et TAUX_VALIDITE_JOYSTICK.

```
00439 {
00440     if(modeDeplacement)
00441     {
00442         if(valeur >= TAUX_VALIDITE_JOYSTICK)
00443             maManetteDeplacement.joystickGaucheEnAvant = true;
00444         else if(valeur <= -TAUX_VALIDITE_JOYSTICK)
00445             maManetteDeplacement.joystickGaucheEnArriere = true;
00446         else if(valeur > -TAUX_VALIDITE_JOYSTICK && valeur <
      TAUX_VALIDITE_JOYSTICK)
00447         {
00448             maManetteDeplacement.joystickGaucheEnAvant = false;
00449             maManetteDeplacement.joystickGaucheEnArriere = false
      ;
00450         }
00451         determinerTrameDeplacementRobot();
00452     }
00453 }
```

6.17.3.5 changerBoutonA

```
void Manette::changerBoutonA (
            bool etat )  [slot]
```

change l'état du membre boutonA

Définition à la ligne 551 du fichier manette.cpp.

Références EtatManetteBouton : :boutonA, determinerTrameBouton(), maManetteBouton, et modePilotageBras.

```
00552 {
00553     maManetteBouton.boutonA = etat;
00554     if(modePilotageBras)
00555         determinerTrameBouton();
00556 }
```

6.17.3.6 changerBoutonB

```
void Manette::changerBoutonB (
            bool etat )  [slot]
```

change l'état du membre boutonB

Définition à la ligne 558 du fichier manette.cpp.

Références EtatManetteBouton : :boutonB, determinerTrameBouton(), maManetteBouton, et modePilotageBras.

```
00559 {
00560     maManetteBouton.boutonB = etat;
00561     if(modePilotageBras)
00562         determinerTrameBouton();
00563 }
```

6.17.3.7 changerBoutonHautDroit

```
void Manette::changerBoutonHautDroit (
            bool etat )  [slot]
```

change l'etat des attributs boutonHautDroit

Définition à la ligne 486 du fichier manette.cpp.

Références EtatManettePilotage : :boutonHautDroit, determinerTramePilotageBras(), maManettePilotage, et modePilotageBras.

```
00487 {
00488     maManettePilotage.boutonHautDroit = etat;
00489     if(modePilotageBras)
00490         determinerTramePilotageBras();
00491 }
```

6.17.3.8 changerBoutonHautGauche

```
void Manette::changerBoutonHautGauche (
            bool etat )  [slot]
```

change l'etat des attributs boutonHautGauche

Définition à la ligne 479 du fichier manette.cpp.

Références EtatManettePilotage : :boutonHautGauche, determinerTramePilotageBras(), maManettePilotage, et modePilotageBras.

```
00480 {
00481     maManettePilotage.boutonHautGauche = etat;
00482     if(modePilotageBras)
00483         determinerTramePilotageBras();
00484 }
```

6.17.3.9 changerBoutonSelect

```
void Manette::changerBoutonSelect (
            bool etat )  [slot]
```

change l'etat des attributs modeDeplacement modePilotageBras en fonction de "etat"

Paramètres

etat	

Définition à la ligne 579 du fichier manette.cpp.

Références changerMode().

```
00580 {
00581     if(etat)
00582     {
00583         changerMode();
00584     }
00585     qDebug() << Q_FUNC_INFO << "Button Select" << etat ;
00586 }
```

6.17.3.10 changerBoutonX

```
void Manette::changerBoutonX (
            bool etat )  [slot]
```

change l'état du membre boutonX

Définition à la ligne 565 du fichier manette.cpp.

Références EtatManetteBouton : :boutonX, determinerTrameBouton(), maManetteBouton, et modePilotageBras.

```
00566 {
00567     maManetteBouton.boutonX = etat;
00568     if(modePilotageBras)
00569         determinerTrameBouton();
00570 }
```

6.17.3.11 changerBoutonY

```
void Manette::changerBoutonY (
            bool etat )  [slot]
```

change l'état du membre boutonY

Définition à la ligne 572 du fichier manette.cpp.

Références EtatManetteBouton : :boutonY, determinerTrameBouton(), maManetteBouton, et modePilotageBras.

```
00573 {
00574     maManetteBouton.boutonY = etat;
00575     if(modePilotageBras)
00576         determinerTrameBouton();
00577 }
```

6.17.3.12 changerFlecheADroite

```
void Manette::changerFlecheADroite (
            bool etat )  [slot]
```

change l'etat de l'attribut flecheADroite en fonction de "etat"

Paramètres

etat	

Définition à la ligne 542 du fichier manette.cpp.

Références determinerTrameDeplacementRobot(), determinerTramePilotageBras(), EtatManettePilotage : :flecheADroite, maManettePilotage, et modeDeplacement.

```
00543 {
00544     maManettePilotage.flecheADroite = etat;
00545     if(modeDeplacement)
00546         determinerTrameDeplacementRobot();
00547     else
00548         determinerTramePilotageBras();
00549 }
```

6.17.3.13 changerFlecheAGauche

```
void Manette::changerFlecheAGauche (
            bool etat )  [slot]
```

change l'etat de l'attribut flecheAGauche en fonction de "valeur"

Paramètres

etat	

Définition à la ligne 533 du fichier manette.cpp.

Références determinerTrameDeplacementRobot(), determinerTramePilotageBras(), EtatManettePilotage : :flecheAGauche, maManettePilotage, et modeDeplacement.

```
00534 {
00535     maManettePilotage.flecheAGauche = etat;
00536     if(modeDeplacement)
00537         determinerTrameDeplacementRobot();
00538     else
00539         determinerTramePilotageBras();
00540 }
```

6.17.3.14 changerFlecheEnArriere

```
void Manette::changerFlecheEnArriere (
            bool etat )  [slot]
```

change l'etat de l'attribut flecheEnArriere en fonction de "valeur"

Paramètres

etat	

Définition à la ligne 524 du fichier manette.cpp.

Références determinerTrameDeplacementRobot(), determinerTramePilotageBras(), EtatManettePilotage : :flecheEnArriere, maManettePilotage, et modeDeplacement.

```
00525 {
00526     maManettePilotage.flecheEnArriere = etat;
00527     if(modeDeplacement)
00528         determinerTrameDeplacementRobot();
00529     else
00530         determinerTramePilotageBras();
00531 }
```

6.17.3.15 changerFlecheEnAvant

```
void Manette::changerFlecheEnAvant (
            bool etat )  [slot]
```

change l'etat de l'attribut flecheEnAvant en fonction de "valeur"

Paramètres

etat	

Définition à la ligne 515 du fichier manette.cpp.

Références determinerTrameDeplacementRobot(), determinerTramePilotageBras(), EtatManettePilotage : :flecheEnAvant, maManettePilotage, et modeDeplacement.

```
00516 {
00517     maManettePilotage.flecheEnAvant = etat;
00518     if(modeDeplacement)
00519         determinerTrameDeplacementRobot();
00520     else
00521         determinerTramePilotageBras();
00522 }
```

6.17.3.16 changerGachetteBasDroit

```
void Manette::changerGachetteBasDroit (
            double valeur )  [slot]
```

change l'etat des attributs gachetteBasDroit et puissance en fonction de "valeur"

Paramètres

valeur	

Définition à la ligne 504 du fichier manette.cpp.

Références determinerTrameDeplacementRobot(), EtatManetteDeplacement : :gachetteBasDroit, maManetteDeplacement, modeDeplacement, et puissanceGachetteDroite.

```
00505 {
00506     puissanceGachetteDroite = int(valeur*100);
00507     if (valeur > 0)
00508         maManetteDeplacement.gachetteBasDroit = true;
00509     else
00510         maManetteDeplacement.gachetteBasDroit = false;
00511     if(modeDeplacement)
00512         determinerTrameDeplacementRobot();
00513 }
```

6.17.3.17 changerGachetteBasGauche

```
void Manette::changerGachetteBasGauche (
             double valeur )  [slot]
```

change l'etat des attributs gachetteBasGauche et puissance en fonction de "valeur"

Paramètres

valeur	

Définition à la ligne 493 du fichier manette.cpp.

Références determinerTrameDeplacementRobot(), EtatManetteDeplacement : :gachetteBasGauche, maManetteDeplacement, modeDeplacement, et puissanceGachetteGauche.

```
00494 {
00495     puissanceGachetteGauche = int(valeur*100);
00496     if (valeur > 0)
00497         maManetteDeplacement.gachetteBasGauche = true;
00498     else
00499         maManetteDeplacement.gachetteBasGauche = false;
00500     if(modeDeplacement)
00501         determinerTrameDeplacementRobot();
00502 }
```

6.17.3.18 changerMode()

```
void Manette::changerMode ( )
```

Change de mode en fonction du bouton SELECT.

Définition à la ligne 407 du fichier manette.cpp.

Références modeDeplacement, et modePilotageBras.

Référencé par changerBoutonSelect().

```
00408 {
00409     if(modeDeplacement)
00410     {
00411         modePilotageBras = true;
00412         modeDeplacement = false;
00413     }
00414     else
00415     {
00416         modePilotageBras = false;
00417         modeDeplacement = true;
00418     }
00419 }
```

6.17.3.19 creationTrameDeplacement

```
void Manette::creationTrameDeplacement (
            char deplacementAxeX,
            int puissance,
            char deplacementAxeY )  [signal]
```

Envoye des élément de la trame pour la création de la trame de déplacement.

Paramètres

deplacementAxeX	
puissance	
deplacementAxeY	

Référencé par determinerTrameDeplacementRobot().

6.17.3.20 creationTrameOrdre

```
void Manette::creationTrameOrdre (
            QString ordre )  [signal]
```

Envoye des élément de la trame pour la création de la trame des ordres.

Paramètres

ordre	

Référencé par determinerTrameBouton().

6.17.3.21 creationTramePilotage

```
void Manette::creationTramePilotage (
            QString direction )  [signal]
```

Envoye des élément de la trame pour la création de la trame de pilotage.

Paramètres

direction	

Référencé par determinerTramePilotageBras().

6.17.3.22 creationTramePince

```
void Manette::creationTramePince (
            QString mouvementPince )  [signal]
```

Envoye des élément de la trame pour la création de la trame d'ouverture et fermeture de la pince.

Référencé par determinerTrameBouton().

6.17.3.23 determinerTrameBouton()

```
void Manette::determinerTrameBouton ( )
```

Verifie l'etat de la manette et creer la trame correspondante.

Définition à la ligne 393 du fichier manette.cpp.

Références creationTrameOrdre(), creationTramePince(), fermerPince, FERMETURE_PINCE, IMMOBILE, maManetteBouton, OUVERTURE_PINCE, ouvrirPince, POSER_OBJET_DANS_BAC, poserObjetDansBac, RETOUR_ETAT_INITIAL, et revenirEtatInitial.

Référencé par changerBoutonA(), changerBoutonB(), changerBoutonX(), et changerBoutonY().

```
00394 {
00395     if(maManetteBouton == ouvrirPince)
00396         emit creationTramePince(OUVERTURE_PINCE);
00397     else if(maManetteBouton == fermerPince)
00398         emit creationTramePince(FERMETURE_PINCE);
00399     else if(maManetteBouton == revenirEtatInitial)
00400         emit creationTrameOrdre(RETOUR_ETAT_INITIAL);
00401     else if(maManetteBouton == poserObjetDansBac)
00402         emit creationTrameOrdre(POSER_OBJET_DANS_BAC);
00403     else
00404         emit creationTramePince(IMMOBILE);
00405 }
```

6.17.3.24 determinerTrameDeplacementRobot()

```
void Manette::determinerTrameDeplacementRobot ( )
```

Verifie l'etat de la manette et creer la trame correspondante.

Définition à la ligne 341 du fichier manette.cpp.

Références creationTrameDeplacement(), enArriereCas1, enArriereCas2, enArriereCas3, enAvantCas1, enAvantCas2, enAvantCas3, maManetteDeplacement, puissanceGachetteDroite, puissanceGachetteGauche, REDUCTION_VITESSE, rotationADroite, rotationADroiteDoucementCas1, rotationADroiteDoucementCas2, rotationAGauche, rotationAGaucheDoucementCas1, rotationAGaucheDoucementCas2, virageArriereADroite, virageArriereADroiteDoucementCas1, virageArriereADroiteDoucementCas2, virageArriereAGauche, virageArriereAGaucheDoucementCas1, virageArriereAGaucheDoucementCas2, virageAvantADroite, virageAvantADroiteDoucementCas1, virageAvantADroiteDoucementCas2, virageAvantAGauche, virageAvantAGaucheDoucementCas1, et virageAvantAGaucheDoucementCas2.

Référencé par changerAxeXJoystickGauche(), changerAxeYJoystickGauche(), changerFlecheADroite(), changerFlecheAGauche(), changerFlecheEnArriere(), changerFlecheEnAvant(), changerGachetteBasDroit(), et changerGachetteBasGauche().

```
00342 {
00343     if(maManetteDeplacement == enAvantCas1 ||
      maManetteDeplacement == enAvantCas2 ||
      maManetteDeplacement == enAvantCas3)
00344         emit creationTrameDeplacement('A',
      puissanceGachetteDroite, '0');
00345     else if(maManetteDeplacement == enArriereCas1 ||
      maManetteDeplacement == enArriereCas2 ||
      maManetteDeplacement == enArriereCas3)
00346         emit creationTrameDeplacement('R',
      puissanceGachetteGauche, '0');
00347     else if(maManetteDeplacement == rotationAGauche)
00348         emit creationTrameDeplacement('0', 100, 'G');
00349     else if(maManetteDeplacement ==
      rotationAGaucheDoucementCas1 || maManetteDeplacement ==
      rotationAGaucheDoucementCas2)
00350         emit creationTrameDeplacement('0', int(100 *
      REDUCTION_VITESSE), 'G');
00351     else if(maManetteDeplacement == rotationADroite)
00352         emit creationTrameDeplacement('0', 100, 'D');
00353     else if(maManetteDeplacement ==
      rotationADroiteDoucementCas1 || maManetteDeplacement ==
      rotationADroiteDoucementCas2)
00354         emit creationTrameDeplacement('0', int(100 *
      REDUCTION_VITESSE), 'D');
00355     else if(maManetteDeplacement == virageAvantAGauche)
00356         emit creationTrameDeplacement('A',
      puissanceGachetteDroite, 'G');
00357     else if(maManetteDeplacement ==
      virageAvantAGaucheDoucementCas1 ||
      maManetteDeplacement == virageAvantAGaucheDoucementCas2)
00358         emit creationTrameDeplacement('A', int(
      puissanceGachetteDroite * REDUCTION_VITESSE), 'G');
00359     else if(maManetteDeplacement == virageAvantADroite)
00360         emit creationTrameDeplacement('A',
      puissanceGachetteDroite, 'D');
00361     else if(maManetteDeplacement ==
      virageAvantADroiteDoucementCas1 ||
      maManetteDeplacement == virageAvantADroiteDoucementCas2)
00362         emit creationTrameDeplacement('A', int(
      puissanceGachetteDroite * REDUCTION_VITESSE), 'D');
00363     else if(maManetteDeplacement == virageArriereAGauche)
00364         emit creationTrameDeplacement('R',
      puissanceGachetteGauche, 'G');
00365     else if(maManetteDeplacement ==
      virageArriereAGaucheDoucementCas1 ||
      maManetteDeplacement == virageArriereAGaucheDoucementCas2
      )
00366         emit creationTrameDeplacement('R', int(
      puissanceGachetteGauche * REDUCTION_VITESSE), 'G');
00367     else if(maManetteDeplacement == virageArriereADroite)
00368         emit creationTrameDeplacement('R',
      puissanceGachetteGauche, 'D');
00369     else if(maManetteDeplacement ==
      virageArriereADroiteDoucementCas1 ||
      maManetteDeplacement == virageArriereADroiteDoucementCas2
      )
00370         emit creationTrameDeplacement('R', int(
      puissanceGachetteGauche * REDUCTION_VITESSE), 'D');
00371     else
00372         emit creationTrameDeplacement('0', 0, '0');
00373 }
```

6.17.3.25 determinerTramePilotageBras()

```
void Manette::determinerTramePilotageBras ( )
```

Verifie l'etat de la manette et creer la trame correspondante.

Définition à la ligne 375 du fichier manette.cpp.

Références avance, AVANCER, creationTramePilotage(), descend, DESCENDRE, DROITE, droite, GAUCHE, gauche, IMMOBILE, maManettePilotage, monte, MONTER, recule, et RECULER.

Référencé par changerBoutonHautDroit(), changerBoutonHautGauche(), changerFlecheADroite(), changerFlecheAGauche(), changerFlecheEnArriere(), et changerFlecheEnAvant().

```
00376 {
00377     if(maManettePilotage == avance)
00378         emit creationTramePilotage(AVANCER);
00379     else if(maManettePilotage == recule)
00380         emit creationTramePilotage(RECULER);
00381     else if(maManettePilotage == gauche)
00382         emit creationTramePilotage(GAUCHE);
00383     else if(maManettePilotage == droite)
00384         emit creationTramePilotage(DROITE);
00385     else if(maManettePilotage == monte)
00386         emit creationTramePilotage(MONTER);
00387     else if(maManettePilotage == descend)
00388         emit creationTramePilotage(DESCENDRE);
00389     else
00390         emit creationTramePilotage(IMMOBILE);
00391 }
```

6.17.3.26 fermerApplication

```
void Manette::fermerApplication (
             bool etat )  [slot]
```

ferme l'application

Paramètres

etat	

Définition à la ligne 588 du fichier manette.cpp.

```
00589 {
00590     Q_UNUSED(etat);
00591     QCoreApplication::quit();
00592 }
```

6.17.3.27 getNom()

```
QString Manette::getNom ( )
```

Retourne le nom de la manette.

Renvoie

le nom de la manette

Définition à la ligne 659 du fichier manette.cpp.

```
00660 {
00661     return name();
00662 }
```

6.17.3.28 initialisationStructureRotationADroite()

```
void Manette::initialisationStructureRotationADroite ( )  [private]
```

Initialise les états de la structures rotationADroite.

Définition à la ligne 178 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, et rotationADroite.

Référencé par initialiserTypesDeplacement().

```
00179 {
00180     rotationADroite.joystickGaucheEnAvant = false;
00181     rotationADroite.joystickGaucheEnArriere = false;
00182     rotationADroite.joystickGaucheAGauche = false;
00183     rotationADroite.joystickGaucheADroite = true;
00184     rotationADroite.gachetteBasGauche = false;
00185     rotationADroite.gachetteBasDroit = false;
00186 }
```

6.17.3.29 initialisationStructureRotationAGauche()

```
void Manette::initialisationStructureRotationAGauche ( )  [private]
```

Initialise les états de la structures rotationAGauche.

Définition à la ligne 151 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, et rotationAGauche.

Référencé par initialiserTypesDeplacement().

```
00152 {
00153     rotationAGauche.joystickGaucheEnAvant = false;
00154     rotationAGauche.joystickGaucheEnArriere = false;
00155     rotationAGauche.joystickGaucheAGauche = true;
00156     rotationAGauche.joystickGaucheADroite = false;
00157     rotationAGauche.gachetteBasGauche = false;
00158     rotationAGauche.gachetteBasDroit = false;
00159 }
```

6.17.3.30 initialisationStructuresEnArriere()

```
void Manette::initialisationStructuresEnArriere ( )  [private]
```

Initialise les états de la structures enArriere.

Définition à la ligne 127 du fichier manette.cpp.

Références enArriereCas1, enArriereCas2, enArriereCas3, EtatManetteDeplacement : :gachetteBasDroit, EtatManette↩ Deplacement : :gachetteBasGauche, EtatManetteDeplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystick↩ GaucheAGauche, EtatManetteDeplacement : :joystickGaucheEnArriere, et EtatManetteDeplacement : :joystickGaucheEnAvant.

Référencé par initialiserTypesDeplacement().

```
00128 {
00129     enArriereCas1.joystickGaucheEnAvant = false;
00130     enArriereCas1.joystickGaucheEnArriere = false;
00131     enArriereCas1.joystickGaucheAGauche = false;
00132     enArriereCas1.joystickGaucheADroite = false;
00133     enArriereCas1.gachetteBasGauche = true;
00134     enArriereCas1.gachetteBasDroit = false;
00135
00136     enArriereCas2.joystickGaucheEnAvant = true;
00137     enArriereCas2.joystickGaucheEnArriere = false;
00138     enArriereCas2.joystickGaucheAGauche = false;
00139     enArriereCas2.joystickGaucheADroite = false;
00140     enArriereCas2.gachetteBasGauche = true;
00141     enArriereCas2.gachetteBasDroit = false;
00142
00143     enArriereCas3.joystickGaucheEnAvant = false;
00144     enArriereCas3.joystickGaucheEnArriere = true;
00145     enArriereCas3.joystickGaucheAGauche = false;
00146     enArriereCas3.joystickGaucheADroite = false;
00147     enArriereCas3.gachetteBasGauche = true;
00148     enArriereCas3.gachetteBasDroit = false;
00149 }
```

6.17.3.31 initialisationStructuresEnAvant()

```
void Manette::initialisationStructuresEnAvant ( )  [private]
```

Initialise les états de la structures enAvant.

Définition à la ligne 103 du fichier manette.cpp.

Références enAvantCas1, enAvantCas2, enAvantCas3, EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement↩ : :gachetteBasGauche, EtatManetteDeplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystickGaucheEnArriere, et EtatManetteDeplacement : :joystickGaucheEnAvant.

Référencé par initialiserTypesDeplacement().

```
00104 {
00105     enAvantCas1.joystickGaucheEnAvant = false;
00106     enAvantCas1.joystickGaucheEnArriere = false;
00107     enAvantCas1.joystickGaucheAGauche = false;
00108     enAvantCas1.joystickGaucheADroite = false;
00109     enAvantCas1.gachetteBasGauche = false;
00110     enAvantCas1.gachetteBasDroit = true;
00111
00112     enAvantCas2.joystickGaucheEnAvant = true;
00113     enAvantCas2.joystickGaucheEnArriere = false;
00114     enAvantCas2.joystickGaucheAGauche = false;
00115     enAvantCas2.joystickGaucheADroite = false;
00116     enAvantCas2.gachetteBasGauche = false;
00117     enAvantCas2.gachetteBasDroit = true;
00118
00119     enAvantCas3.joystickGaucheEnAvant = false;
00120     enAvantCas3.joystickGaucheEnArriere = true;
00121     enAvantCas3.joystickGaucheAGauche = false;
00122     enAvantCas3.joystickGaucheADroite = false;
00123     enAvantCas3.gachetteBasGauche = false;
00124     enAvantCas3.gachetteBasDroit = true;
00125 }
```

6.17.3.32 initialisationStructuresRotationADroiteDoucement()

```
void Manette::initialisationStructuresRotationADroiteDoucement ( )  [private]
```

Initialise les états de la structures rotationADroiteDoucement.

Définition à la ligne 188 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, rotationADroiteDoucementCas1, et rotationADroite↩ DoucementCas2.

Référencé par initialiserTypesDeplacement().

```
00189 {
00190     rotationADroiteDoucementCas1.
      joystickGaucheEnAvant = false;
00191     rotationADroiteDoucementCas1.
      joystickGaucheEnArriere = true;
00192     rotationADroiteDoucementCas1.
      joystickGaucheAGauche = false;
00193     rotationADroiteDoucementCas1.
      joystickGaucheADroite = true;
00194     rotationADroiteDoucementCas1.gachetteBasGauche = false;
00195     rotationADroiteDoucementCas1.gachetteBasDroit = false;
00196
00197     rotationADroiteDoucementCas2.
      joystickGaucheEnAvant = true;
00198     rotationADroiteDoucementCas2.
      joystickGaucheEnArriere = false;
00199     rotationADroiteDoucementCas2.
      joystickGaucheAGauche = false;
00200     rotationADroiteDoucementCas2.
      joystickGaucheADroite = true;
00201     rotationADroiteDoucementCas2.gachetteBasGauche = false;
00202     rotationADroiteDoucementCas2.gachetteBasDroit = false;
00203 }
```

6.17.3.33 initialisationStructuresRotationAGaucheDoucement()

```
void Manette::initialisationStructuresRotationAGaucheDoucement ( )  [private]
```

Initialise les états de la structures rotationAGaucheDoucement.

Définition à la ligne 161 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, rotationAGaucheDoucementCas1, et rotationAGauche↩ DoucementCas2.

Référencé par initialiserTypesDeplacement().

```
00162 {
00163     rotationAGaucheDoucementCas1.
      joystickGaucheEnAvant = false;
00164     rotationAGaucheDoucementCas1.
      joystickGaucheEnArriere = true;
00165     rotationAGaucheDoucementCas1.
      joystickGaucheAGauche = true;
00166     rotationAGaucheDoucementCas1.
      joystickGaucheADroite = false;
00167     rotationAGaucheDoucementCas1.gachetteBasGauche = false;
00168     rotationAGaucheDoucementCas1.gachetteBasDroit = false;
00169
00170     rotationAGaucheDoucementCas2.
      joystickGaucheEnAvant = true;
00171     rotationAGaucheDoucementCas2.
      joystickGaucheEnArriere = false;
00172     rotationAGaucheDoucementCas2.
      joystickGaucheAGauche = true;
00173     rotationAGaucheDoucementCas2.
      joystickGaucheADroite = false;
00174     rotationAGaucheDoucementCas2.gachetteBasGauche = false;
00175     rotationAGaucheDoucementCas2.gachetteBasDroit = false;
00176 }
```

6.17.3.34 initialisationStructuresVirageArriereADroiteDoucement()

```
void Manette::initialisationStructuresVirageArriereADroiteDoucement ( )  [private]
```

Initialise les états de la structures virageArriereADroiteDoucement.

Définition à la ligne 296 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, virageArriereADroiteDoucementCas1, et virageArriereA↩ DroiteDoucementCas2.

Référencé par initialiserTypesDeplacement().

```
00297 {
00298     virageArriereADroiteDoucementCas1.
      joystickGaucheEnAvant = true;
00299     virageArriereADroiteDoucementCas1.
      joystickGaucheEnArriere = false;
00300     virageArriereADroiteDoucementCas1.
      joystickGaucheAGauche = false;
00301     virageArriereADroiteDoucementCas1.
      joystickGaucheADroite = true;
00302     virageArriereADroiteDoucementCas1.
      gachetteBasGauche = true;
00303     virageArriereADroiteDoucementCas1.
      gachetteBasDroit = false;
00304
00305     virageArriereADroiteDoucementCas2.
      joystickGaucheEnAvant = false;
00306     virageArriereADroiteDoucementCas2.
      joystickGaucheEnArriere = true;
00307     virageArriereADroiteDoucementCas2.
      joystickGaucheAGauche = false;
00308     virageArriereADroiteDoucementCas2.
      joystickGaucheADroite = true;
00309     virageArriereADroiteDoucementCas2.
      gachetteBasGauche = true;
00310     virageArriereADroiteDoucementCas2.
      gachetteBasDroit = false;
00311 }
```

6.17.3.35 initialisationStructuresVirageArriereAGaucheDoucement()

```
void Manette::initialisationStructuresVirageArriereAGaucheDoucement ( )  [private]
```

Initialise les états de la structures virageArriereAGaucheDoucement.

Définition à la ligne 269 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, virageArriereAGaucheDoucementCas1, et virageArriereA↩ GaucheDoucementCas2.

Référencé par initialiserTypesDeplacement().

```
00270 {
00271     virageArriereAGaucheDoucementCas1.
      joystickGaucheEnAvant = true;
00272     virageArriereAGaucheDoucementCas1.
      joystickGaucheEnArriere = false;
00273     virageArriereAGaucheDoucementCas1.
      joystickGaucheAGauche = true;
00274     virageArriereAGaucheDoucementCas1.
      joystickGaucheADroite = false;
00275     virageArriereAGaucheDoucementCas1.
      gachetteBasGauche = true;
00276     virageArriereAGaucheDoucementCas1.
      gachetteBasDroit = false;
00277
00278     virageArriereAGaucheDoucementCas2.
      joystickGaucheEnAvant = false;
00279     virageArriereAGaucheDoucementCas2.
      joystickGaucheEnArriere = true;
00280     virageArriereAGaucheDoucementCas2.
      joystickGaucheAGauche = true;
00281     virageArriereAGaucheDoucementCas2.
      joystickGaucheADroite = false;
00282     virageArriereAGaucheDoucementCas2.
      gachetteBasGauche = true;
00283     virageArriereAGaucheDoucementCas2.
      gachetteBasDroit = false;
00284 }
```

6.17.3.36 initialisationStructuresVirageAvantADroiteDoucement()

```
void Manette::initialisationStructuresVirageAvantADroiteDoucement ( )  [private]
```

Initialise les états de la structures virageAvantADroite.

Définition à la ligne 242 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, virageAvantADroiteDoucementCas1, et virageAvantADroite↩ DoucementCas2.

Référencé par initialiserTypesDeplacement().

```
00243 {
00244     virageAvantADroiteDoucementCas1.
      joystickGaucheEnAvant = true;
00245     virageAvantADroiteDoucementCas1.
      joystickGaucheEnArriere = false;
00246     virageAvantADroiteDoucementCas1.
      joystickGaucheAGauche = false;
00247     virageAvantADroiteDoucementCas1.
      joystickGaucheADroite = true;
00248     virageAvantADroiteDoucementCas1.
      gachetteBasGauche = false;
00249     virageAvantADroiteDoucementCas1.
      gachetteBasDroit = true;
00250
00251     virageAvantADroiteDoucementCas2.
      joystickGaucheEnAvant = false;
00252     virageAvantADroiteDoucementCas2.
      joystickGaucheEnArriere = true;
00253     virageAvantADroiteDoucementCas2.
      joystickGaucheAGauche = false;
00254     virageAvantADroiteDoucementCas2.
      joystickGaucheADroite = true;
00255     virageAvantADroiteDoucementCas2.
      gachetteBasGauche = false;
00256     virageAvantADroiteDoucementCas2.
      gachetteBasDroit = true;
00257 }
```

6.17.3.37 initialisationStructuresVirageAvantAGaucheDoucement()

```
void Manette::initialisationStructuresVirageAvantAGaucheDoucement ( )  [private]
```

Initialise les états de la structures virageAvantAGaucheDoucement.

Définition à la ligne 215 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, virageAvantAGaucheDoucementCas1, et virageAvantA↩ GaucheDoucementCas2.

Référencé par initialiserTypesDeplacement().

```
00216 {
00217     virageAvantAGaucheDoucementCas1.
      joystickGaucheEnAvant = true;
00218     virageAvantAGaucheDoucementCas1.
      joystickGaucheEnArriere = false;
00219     virageAvantAGaucheDoucementCas1.
      joystickGaucheAGauche = true;
00220     virageAvantAGaucheDoucementCas1.
      joystickGaucheADroite = false;
00221     virageAvantAGaucheDoucementCas1.
      gachetteBasGauche = false;
00222     virageAvantAGaucheDoucementCas1.
      gachetteBasDroit = true;
00223
00224     virageAvantAGaucheDoucementCas2.
      joystickGaucheEnAvant = false;
00225     virageAvantAGaucheDoucementCas2.
      joystickGaucheEnArriere = true;
00226     virageAvantAGaucheDoucementCas2.
      joystickGaucheAGauche = true;
00227     virageAvantAGaucheDoucementCas2.
      joystickGaucheADroite = false;
00228     virageAvantAGaucheDoucementCas2.
      gachetteBasGauche = false;
00229     virageAvantAGaucheDoucementCas2.
      gachetteBasDroit = true;
00230 }
```

6.17.3.38 initialisationStructureVirageArriereADroite()

```
void Manette::initialisationStructureVirageArriereADroite ( )  [private]
```

Initialise les états de la structures virageArriereADroite.

Définition à la ligne 286 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, et virageArriereADroite.

Référencé par initialiserTypesDeplacement().

```
00287 {
00288     virageArriereADroite.joystickGaucheEnAvant = false;
00289     virageArriereADroite.joystickGaucheEnArriere = false;
00290     virageArriereADroite.joystickGaucheAGauche = false;
00291     virageArriereADroite.joystickGaucheADroite = true;
00292     virageArriereADroite.gachetteBasGauche = true;
00293     virageArriereADroite.gachetteBasDroit = false;
00294 }
```

6.17.3.39 initialisationStructureVirageArriereAGauche()

```
void Manette::initialisationStructureVirageArriereAGauche ( )  [private]
```

Initialise les états de la structures virageArriereAGauche.

Définition à la ligne 259 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, et virageArriereAGauche.

Référencé par initialiserTypesDeplacement().

```
00260 {
00261     virageArriereAGauche.joystickGaucheEnAvant = false;
00262     virageArriereAGauche.joystickGaucheEnArriere = false;
00263     virageArriereAGauche.joystickGaucheAGauche = true;
00264     virageArriereAGauche.joystickGaucheADroite = false;
00265     virageArriereAGauche.gachetteBasGauche = true;
00266     virageArriereAGauche.gachetteBasDroit = false;
00267 }
```

6.17.3.40 initialisationStructureVirageAvantADroite()

```
void Manette::initialisationStructureVirageAvantADroite ( )  [private]
```

Initialise les états de la structures.

Définition à la ligne 232 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, et virageAvantADroite.

Référencé par initialiserTypesDeplacement().

```
00233 {
00234     virageAvantADroite.joystickGaucheEnAvant = false;
00235     virageAvantADroite.joystickGaucheEnArriere = false;
00236     virageAvantADroite.joystickGaucheAGauche = false;
00237     virageAvantADroite.joystickGaucheADroite = true;
00238     virageAvantADroite.gachetteBasGauche = false;
00239     virageAvantADroite.gachetteBasDroit = true;
00240 }
```

6.17.3.41 initialisationStructureVirageAvantAGauche()

```
void Manette::initialisationStructureVirageAvantAGauche ( )  [private]
```

Initialise les états de la structures virageAvantAGauche.

Définition à la ligne 205 du fichier manette.cpp.

Références EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManette↩ Deplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystick↩ GaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, et virageAvantAGauche.

Référencé par initialiserTypesDeplacement().

```
00206 {
00207     virageAvantAGauche.joystickGaucheEnAvant = false;
00208     virageAvantAGauche.joystickGaucheEnArriere = false;
00209     virageAvantAGauche.joystickGaucheAGauche = true;
00210     virageAvantAGauche.joystickGaucheADroite = false;
00211     virageAvantAGauche.gachetteBasGauche = false;
00212     virageAvantAGauche.gachetteBasDroit = true;
00213 }
```

6.17.3.42 initialiserEtatBouton()

```
void Manette::initialiserEtatBouton ( )  [private]
```

Initialise les états des structures EtatBouton.

Définition à la ligne 313 du fichier manette.cpp.

Références EtatManetteBouton : :boutonA, EtatManetteBouton : :boutonB, EtatManetteBouton : :boutonX, EtatManetteBouton : :boutonY, fermerPince, maManetteBouton, ouvrirPince, poserObjetDansBac, et revenirEtatInitial.

Référencé par Manette().

```
00314 {
00315     maManetteBouton.boutonA = false;
00316     maManetteBouton.boutonB = false;
00317     maManetteBouton.boutonX = false;
00318     maManetteBouton.boutonY = false;
00319 
00320     ouvrirPince.boutonA = false;
00321     ouvrirPince.boutonB = false;
00322     ouvrirPince.boutonX = true;
00323     ouvrirPince.boutonY = false;
00324 
00325     fermerPince.boutonA = false;
00326     fermerPince.boutonB = true;
00327     fermerPince.boutonX = false;
00328     fermerPince.boutonY = false;
00329 
00330     poserObjetDansBac.boutonA = true;
00331     poserObjetDansBac.boutonB = false;
00332     poserObjetDansBac.boutonX = false;
00333     poserObjetDansBac.boutonY = false;
00334 
00335     revenirEtatInitial.boutonA = false;
00336     revenirEtatInitial.boutonB = false;
00337     revenirEtatInitial.boutonX = false;
00338     revenirEtatInitial.boutonY = true;
00339 }
```

6.17.3.43 initialiserEtats()

```
void Manette::initialiserEtats ( )  [private]
```

Initialise les états de la manette.

Définition à la ligne 23 du fichier manette.cpp.

Références EtatManettePilotage : :boutonHautDroit, EtatManettePilotage : :boutonHautGauche, EtatManettePilotage : :flecheADroite, EtatManettePilotage : :flecheAGauche, EtatManettePilotage : :flecheEnArriere, EtatManettePilotage : :flecheEnAvant, EtatManetteDeplacement : :gachetteBasDroit, EtatManetteDeplacement : :gachetteBasGauche, EtatManetteDeplacement : :joystickGaucheADroite, EtatManetteDeplacement : :joystickGaucheAGauche, EtatManetteDeplacement : :joystickGaucheEnArriere, EtatManetteDeplacement : :joystickGaucheEnAvant, maManetteDeplacement, et maManettePilotage.

Référencé par Manette().

```
00024 {
00025     maManettePilotage.flecheADroite = false;
00026     maManettePilotage.flecheAGauche = false;
00027     maManettePilotage.flecheEnArriere = false;
00028     maManettePilotage.flecheEnAvant = false;
00029     maManettePilotage.boutonHautDroit = false;
00030     maManettePilotage.boutonHautGauche = false;
00031 
00032     maManetteDeplacement.joystickGaucheADroite = false;
00033     maManetteDeplacement.joystickGaucheAGauche = false;
00034     maManetteDeplacement.joystickGaucheEnArriere = false;
00035     maManetteDeplacement.joystickGaucheEnAvant = false;
00036     maManetteDeplacement.gachetteBasDroit = false;
00037     maManetteDeplacement.gachetteBasGauche = false;
00038 }
```

6.17.3.44 initialiserTypesDeplacement()

```
void Manette::initialiserTypesDeplacement ( )  [private]
```

Initialise les états de la manette pour le déplacement.

Définition à la ligne 85 du fichier manette.cpp.

Références initialisationStructureRotationADroite(), initialisationStructureRotationAGauche(), initialisationStructuresEnArriere(), initialisationStructuresEnAvant(), initialisationStructuresRotationADroiteDoucement(), initialisationStructuresRotationAGauche↩ Doucement(), initialisationStructuresVirageArriereADroiteDoucement(), initialisationStructuresVirageArriereAGaucheDoucement(), initialisationStructuresVirageAvantADroiteDoucement(), initialisationStructuresVirageAvantAGaucheDoucement(), initialisation↩ StructureVirageArriereADroite(), initialisationStructureVirageArriereAGauche(), initialisationStructureVirageAvantADroite(), et initialisationStructureVirageAvantAGauche().

Référencé par Manette().

```
00086 {
00087     initialisationStructuresEnAvant();
00088     initialisationStructuresEnArriere();
00089     initialisationStructureRotationAGauche();
00090     initialisationStructuresRotationAGaucheDoucement();
00091     initialisationStructureRotationADroite();
00092     initialisationStructuresRotationADroiteDoucement();
00093     initialisationStructureVirageAvantAGauche();
00094     initialisationStructuresVirageAvantAGaucheDoucement(
      );
00095     initialisationStructureVirageAvantADroite();
00096     initialisationStructuresVirageAvantADroiteDoucement(
      );
00097     initialisationStructureVirageArriereAGauche();
00098     initialisationStructuresVirageArriereAGaucheDoucement
      ();
00099     initialisationStructureVirageArriereADroite();
00100     initialisationStructuresVirageArriereADroiteDoucement
      ();
00101 }
```

6.17.3.45 initialiserTypesPilotage()

```
void Manette::initialiserTypesPilotage ( )  [private]
```

Initialise les états de la manette pour le pilotage.

Définition à la ligne 40 du fichier manette.cpp.

Références avance, EtatManettePilotage : :boutonHautDroit, EtatManettePilotage : :boutonHautGauche, descend, droite, Etat↩ ManettePilotage : :flecheADroite, EtatManettePilotage : :flecheAGauche, EtatManettePilotage : :flecheEnArriere, EtatManette↩ Pilotage : :flecheEnAvant, gauche, monte, et recule.

Référencé par Manette().

```
00041 {
00042     avance.flecheADroite = false;
00043     avance.flecheAGauche = false;
00044     avance.flecheEnArriere = false;
00045     avance.flecheEnAvant = true;
00046     avance.boutonHautDroit = false;
00047     avance.boutonHautGauche = false;
00048
00049     recule.flecheADroite = false;
00050     recule.flecheAGauche = false;
00051     recule.flecheEnArriere = true;
00052     recule.flecheEnAvant = false;
00053     recule.boutonHautDroit = false;
00054     recule.boutonHautGauche = false;
00055
00056     gauche.flecheADroite = false;
```

```
00057     gauche.flecheAGauche = true;
00058     gauche.flecheEnArriere = false;
00059     gauche.flecheEnAvant = false;
00060     gauche.boutonHautDroit = false;
00061     gauche.boutonHautGauche = false;
00062
00063     droite.flecheADroite = true;
00064     droite.flecheAGauche = false;
00065     droite.flecheEnArriere = false;
00066     droite.flecheEnAvant = false;
00067     droite.boutonHautDroit = false;
00068     droite.boutonHautGauche = false;
00069
00070     monte.flecheADroite = false;
00071     monte.flecheAGauche = false;
00072     monte.flecheEnArriere = false;
00073     monte.flecheEnAvant = false;
00074     monte.boutonHautDroit = true;
00075     monte.boutonHautGauche = false;
00076
00077     descend.flecheADroite = false;
00078     descend.flecheAGauche = false;
00079     descend.flecheEnArriere = false;
00080     descend.flecheEnAvant = false;
00081     descend.boutonHautDroit = false;
00082     descend.boutonHautGauche = true;
00083 }
```

6.17.3.46 nouvelleTrameCamera

```
void Manette::nouvelleTrameCamera (
              QString axeY,
              QString axeX )  [signal]
```

Nouvelle trame de commande de la caméra disponible.

Paramètres

axeY	
axeX	

Référencé par changerAxeXJoystickDroit(), et changerAxeYJoystickDroit().

6.17.3.47 prendrePhoto

```
void Manette::prendrePhoto ( )  [signal]
```

Envoi un signal indiquant que le bouton photo est pressé

6.17.4 Documentation des données membres

6.17.4.1 avance

```
EtatManettePilotage Manette::avance  [private]
```

Structure définissant l'état des bouton de la manette pour avancer.

Définition à la ligne 247 du fichier manette.h.

Référencé par determinerTramePilotageBras(), et initialiserTypesPilotage().

6.17.4.2 descend

```
EtatManettePilotage Manette::descend  [private]
```

Structure définissant l'état des bouton de la manette pour descendre.

Définition à la ligne 252 du fichier manette.h.

Référencé par determinerTramePilotageBras(), et initialiserTypesPilotage().

6.17.4.3 droite

```
EtatManettePilotage Manette::droite  [private]
```

Structure définissant l'état des bouton de la manette pour aller à droite.

Définition à la ligne 250 du fichier manette.h.

Référencé par determinerTramePilotageBras(), et initialiserTypesPilotage().

6.17.4.4 enArriereCas1

```
EtatManetteDeplacement Manette::enArriereCas1  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer.

Définition à la ligne 224 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresEnArriere().

6.17.4.5 enArriereCas2

```
EtatManetteDeplacement Manette::enArriereCas2  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer.

Définition à la ligne 225 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresEnArriere().

6.17.4.6 enArriereCas3

```
EtatManetteDeplacement Manette::enArriereCas3  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer.

Définition à la ligne 226 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresEnArriere().

6.17.4.7 enAvantCas1

`EtatManetteDeplacement Manette::enAvantCas1 [private]`

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer.

Définition à la ligne 221 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresEnAvant().

6.17.4.8 enAvantCas2

`EtatManetteDeplacement Manette::enAvantCas2 [private]`

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer.

Définition à la ligne 222 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresEnAvant().

6.17.4.9 enAvantCas3

`EtatManetteDeplacement Manette::enAvantCas3 [private]`

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer.

Définition à la ligne 223 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresEnAvant().

6.17.4.10 fermerPince

`EtatManetteBouton Manette::fermerPince [private]`

Structure définissant l'etat des bouton A, B, X, Y de la manette pour fermer la pince.

Définition à la ligne 217 du fichier manette.h.

Référencé par determinerTrameBouton(), et initialiserEtatBouton().

6.17.4.11 gauche

`EtatManettePilotage Manette::gauche [private]`

Structure définissant l'état des bouton de la manette pour aller à gauche.

Définition à la ligne 249 du fichier manette.h.

Référencé par determinerTramePilotageBras(), et initialiserTypesPilotage().

6.17.4.12 joystickAxeXCamera

```
QString Manette::joystickAxeXCamera  [private]
```

Attribut contenant l'etat actuel du joystick droite sur l'axe des x.

Définition à la ligne 257 du fichier manette.h.

Référencé par changerAxeXJoystickDroit(), et changerAxeYJoystickDroit().

6.17.4.13 joystickAxeYCamera

```
QString Manette::joystickAxeYCamera  [private]
```

Attribut contenant l'etat actuel du joystick droite sur l'axe des y.

Définition à la ligne 258 du fichier manette.h.

Référencé par changerAxeXJoystickDroit(), et changerAxeYJoystickDroit().

6.17.4.14 maManetteBouton

```
EtatManetteBouton Manette::maManetteBouton  [private]
```

Structure définissant l'etat des bouton A, B, X, Y de la manette.

Définition à la ligne 215 du fichier manette.h.

Référencé par changerBoutonA(), changerBoutonB(), changerBoutonX(), changerBoutonY(), determinerTrameBouton(), et initialiserEtatBouton().

6.17.4.15 maManetteDeplacement

```
EtatManetteDeplacement Manette::maManetteDeplacement  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot.

Définition à la ligne 220 du fichier manette.h.

Référencé par changerAxeXJoystickGauche(), changerAxeYJoystickGauche(), changerGachetteBasDroit(), changerGachetteBasGauche(), determinerTrameDeplacementRobot(), et initialiserEtats().

6.17.4.16 maManettePilotage

```
EtatManettePilotage Manette::maManettePilotage  [private]
```

Structure définissant l'état des bouton de la manette en mode pilotage du bras.

Définition à la ligne 246 du fichier manette.h.

Référencé par changerBoutonHautDroit(), changerBoutonHautGauche(), changerFlecheADroite(), changerFlecheAGauche(), changerFlecheEnArriere(), changerFlecheEnAvant(), determinerTramePilotageBras(), et initialiserEtats().

6.17.4.17 modeDeplacement

```
bool Manette::modeDeplacement  [private]
```

Attribut définissant l'état du mode de déplacement.

Définition à la ligne 253 du fichier manette.h.

Référencé par changerAxeXJoystickGauche(), changerAxeYJoystickGauche(), changerFlecheADroite(), changerFlecheAGauche(), changerFlecheEnArriere(), changerFlecheEnAvant(), changerGachetteBasDroit(), changerGachetteBasGauche(), et changerMode().

6.17.4.18 modePilotageBras

```
bool Manette::modePilotageBras  [private]
```

Attribut définissant l'état du mode pilotage du bras articulé

Définition à la ligne 254 du fichier manette.h.

Référencé par changerBoutonA(), changerBoutonB(), changerBoutonHautDroit(), changerBoutonHautGauche(), changerBoutonX(), changerBoutonY(), et changerMode().

6.17.4.19 monte

```
EtatManettePilotage Manette::monte  [private]
```

Structure définissant l'état des bouton de la manette pour monter.

Définition à la ligne 251 du fichier manette.h.

Référencé par determinerTramePilotageBras(), et initialiserTypesPilotage().

6.17.4.20 ouvrirPince

```
EtatManetteBouton Manette::ouvrirPince  [private]
```

Structure définissant l'etat des bouton A, B, X, Y de la manette pour ouvrir la pince.

Définition à la ligne 216 du fichier manette.h.

Référencé par determinerTrameBouton(), et initialiserEtatBouton().

6.17.4.21 poserObjetDansBac

```
EtatManetteBouton Manette::poserObjetDansBac  [private]
```

Structure définissant l'etat des bouton A, B, X, Y de la manette pour poser l'objet dans le bac.

Définition à la ligne 218 du fichier manette.h.

Référencé par determinerTrameBouton(), et initialiserEtatBouton().

6.17.4.22 puissanceGachetteDroite

```
int Manette::puissanceGachetteDroite  [private]
```

Attribut définissant la valeur de la puissance de la gachette droite.

Définition à la ligne 255 du fichier manette.h.

Référencé par changerGachetteBasDroit(), et determinerTrameDeplacementRobot().

6.17.4.23 puissanceGachetteGauche

```
int Manette::puissanceGachetteGauche  [private]
```

Attribut définissant la valeur de la puissance de la gachette gauche.

Définition à la ligne 256 du fichier manette.h.

Référencé par changerGachetteBasGauche(), et determinerTrameDeplacementRobot().

6.17.4.24 recule

```
EtatManettePilotage Manette::recule  [private]
```

Structure définissant l'état des bouton de la manette pour reculer.

Définition à la ligne 248 du fichier manette.h.

Référencé par determinerTramePilotageBras(), et initialiserTypesPilotage().

6.17.4.25 revenirEtatInitial

```
EtatManetteBouton Manette::revenirEtatInitial  [private]
```

Structure définissant l'etat des bouton A, B, X, Y de la manette pour revenir à l'etat initiale du bras.

Définition à la ligne 219 du fichier manette.h.

Référencé par determinerTrameBouton(), et initialiserEtatBouton().

6.17.4.26 rotationADroite

```
EtatManetteDeplacement Manette::rotationADroite  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à droite.

Définition à la ligne 230 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructureRotationADroite().

6.17.4.27 rotationADroiteDoucementCas1

`EtatManetteDeplacement Manette::rotationADroiteDoucementCas1 [private]`

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à droite doucement.

Définition à la ligne 231 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresRotationADroiteDoucement().

6.17.4.28 rotationADroiteDoucementCas2

`EtatManetteDeplacement Manette::rotationADroiteDoucementCas2 [private]`

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à droite doucement.

Définition à la ligne 232 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresRotationADroiteDoucement().

6.17.4.29 rotationAGauche

`EtatManetteDeplacement Manette::rotationAGauche [private]`

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à gauche.

Définition à la ligne 227 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructureRotationAGauche().

6.17.4.30 rotationAGaucheDoucementCas1

`EtatManetteDeplacement Manette::rotationAGaucheDoucementCas1 [private]`

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à gauche doucement.

Définition à la ligne 228 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresRotationAGaucheDoucement().

6.17.4.31 rotationAGaucheDoucementCas2

`EtatManetteDeplacement Manette::rotationAGaucheDoucementCas2 [private]`

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour tourner à gauche doucement.

Définition à la ligne 229 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresRotationAGaucheDoucement().

6.17.4.32 virageArriereADroite

```
EtatManetteDeplacement Manette::virageArriereADroite  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à droite.

Définition à la ligne 242 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructureVirageArriereADroite().

6.17.4.33 virageArriereADroiteDoucementCas1

```
EtatManetteDeplacement Manette::virageArriereADroiteDoucementCas1  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à droite doucement.

Définition à la ligne 243 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresVirageArriereADroiteDoucement().

6.17.4.34 virageArriereADroiteDoucementCas2

```
EtatManetteDeplacement Manette::virageArriereADroiteDoucementCas2  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à droite doucement.

Définition à la ligne 244 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresVirageArriereADroiteDoucement().

6.17.4.35 virageArriereAGauche

```
EtatManetteDeplacement Manette::virageArriereAGauche  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à gauche.

Définition à la ligne 239 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructureVirageArriereAGauche().

6.17.4.36 virageArriereAGaucheDoucementCas1

```
EtatManetteDeplacement Manette::virageArriereAGaucheDoucementCas1  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à gauche doucement.

Définition à la ligne 240 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresVirageArriereAGaucheDoucement().

6.17.4.37 virageArriereAGaucheDoucementCas2

```
EtatManetteDeplacement Manette::virageArriereAGaucheDoucementCas2  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour reculer et tourner à gauche doucement.

Définition à la ligne 241 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresVirageArriereAGaucheDoucement().

6.17.4.38 virageAvantADroite

```
EtatManetteDeplacement Manette::virageAvantADroite  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à droite.

Définition à la ligne 236 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructureVirageAvantADroite().

6.17.4.39 virageAvantADroiteDoucementCas1

```
EtatManetteDeplacement Manette::virageAvantADroiteDoucementCas1  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à droite doucement.

Définition à la ligne 237 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresVirageAvantADroiteDoucement().

6.17.4.40 virageAvantADroiteDoucementCas2

```
EtatManetteDeplacement Manette::virageAvantADroiteDoucementCas2  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à droite doucement.

Définition à la ligne 238 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresVirageAvantADroiteDoucement().

6.17.4.41 virageAvantAGauche

```
EtatManetteDeplacement Manette::virageAvantAGauche  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à gauche.

Définition à la ligne 233 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructureVirageAvantAGauche().

6.17.4.42 virageAvantAGaucheDoucementCas1

```
EtatManetteDeplacement Manette::virageAvantAGaucheDoucementCas1  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à gauche doucement.

Définition à la ligne 234 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresVirageAvantAGaucheDoucement().

6.17.4.43 virageAvantAGaucheDoucementCas2

```
EtatManetteDeplacement Manette::virageAvantAGaucheDoucementCas2  [private]
```

Structure définissant l'état des bouton de la manette en mode deplacement du robot pour avancer et tourner à gauche doucement.

Définition à la ligne 235 du fichier manette.h.

Référencé par determinerTrameDeplacementRobot(), et initialisationStructuresVirageAvantAGaucheDoucement().

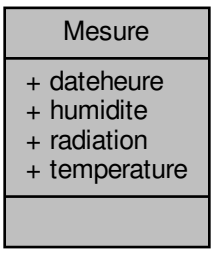
La documentation de cette classe a été générée à partir des fichiers suivants :
- manette.h
- manette.cpp

6.18 Référence de la structure Mesure

structure permettant de définir les propriété d'une mesure prise à une heure précise

```
#include <campagne.h>
```

Graphe de collaboration de Mesure :



Attributs publics

- QDateTime dateheure
  Date/Heure.
- QString humidite
  Donnée humidité
- QString radiation
  Donnée radiation.
- QString temperature
  Donnée temperature.

6.18.1 Description détaillée

structure permettant de définir les propriété d'une mesure prise à une heure précise

Définition à la ligne 21 du fichier campagne.h.

6.18.2 Documentation des données membres

6.18.2.1 dateheure

`QDateTime Mesure::dateheure`

Date/Heure.

Définition à la ligne 23 du fichier campagne.h.

Référencé par Rov : :decoderTrameCapteur().

6.18.2.2 humidite

`QString Mesure::humidite`

Donnée humidité

Définition à la ligne 24 du fichier campagne.h.

Référencé par Rov : :decoderTrameCapteur().

6.18.2.3 radiation

`QString Mesure::radiation`

Donnée radiation.

Définition à la ligne 26 du fichier campagne.h.

Référencé par Rov : :decoderTrameCapteur().

6.18.2.4 temperature

`QString Mesure::temperature`

Donnée temperature.

Définition à la ligne 25 du fichier campagne.h.

Référencé par Rov : :decoderTrameCapteur().

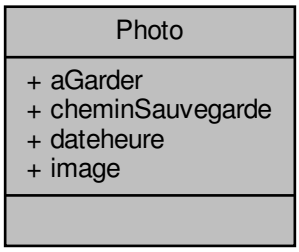
La documentation de cette structure a été générée à partir du fichier suivant :
— campagne.h

6.19 Référence de la structure Photo

structure contenant les informations d'une photo de campagne

`#include <ihmalbumphoto.h>`

Graphe de collaboration de Photo :



Attributs publics

- bool aGarder
  Booléen afin de savoir si la photo sera archivé ou non.
- QString cheminSauvegarde
  Chemin de sauvegarde de la photo.
- QDateTime dateheure
  Date/Heure de la photo.
- QPixmap image
  Image de la photo.

6.19.1 Description détaillée

structure contenant les informations d'une photo de campagne

Définition à la ligne 20 du fichier ihmalbumphoto.h.

6.19.2 Documentation des données membres

6.19.2.1 aGarder

`bool Photo::aGarder`

Booléen afin de savoir si la photo sera archivé ou non.

Définition à la ligne 24 du fichier ihmalbumphoto.h.

Référencé par IHMRov : :capturerImage(), et IHMAccueil : :chargerCampagnes().

6.19.2.2 cheminSauvegarde

`QString Photo::cheminSauvegarde`

Chemin de sauvegarde de la photo.

Définition à la ligne 25 du fichier ihmalbumphoto.h.

Référencé par IHMAccueil : :ajouterPhotoBDD(), IHMRov : :capturerImage(), et IHMAccueil : :chargerCampagnes().

6.19.2.3 dateheure

`QDateTime Photo::dateheure`

Date/Heure de la photo.

Définition à la ligne 23 du fichier ihmalbumphoto.h.

Référencé par IHMAccueil : :ajouterPhotoBDD(), IHMRov : :capturerImage(), et IHMAccueil : :chargerCampagnes().

6.19.2.4 image

`QPixmap Photo::image`

Image de la photo.

Définition à la ligne 22 du fichier ihmalbumphoto.h.

Référencé par IHMRov : :capturerImage(), et IHMAccueil : :chargerCampagnes().

La documentation de cette structure a été générée à partir du fichier suivant :
— ihmalbumphoto.h

6.20 Référence de la classe QDialog

La classe QDialog est la classe de base des fenêtres de dialogue.

Graphe de collaboration de QDialog :

6.20.1 Description détaillée

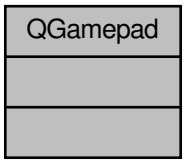
La classe QDialog est la classe de base des fenêtres de dialogue.

La documentation de cette classe a été générée à partir du fichier suivant :
— main.cpp

6.21 Référence de la classe QGamepad

La classe QGamepad est utilisée pour accéder à l'état actuel du matériel de la manette de jeu connecté à un système.

Graphe de collaboration de QGamepad :



6.21.1 Description détaillée

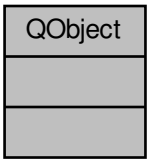
La classe QGamepad est utilisée pour accéder à l'état actuel du matériel de la manette de jeu connecté à un système.

La documentation de cette classe a été générée à partir du fichier suivant :
— main.cpp

6.22 Référence de la classe QObject

La classe QObject est la classe de base de tous les objets Qt. Elle permet à ces objets Qt de disposer entre autres du mécanisme de communication signal/slot.

Graphe de collaboration de QObject :



6.22.1 Description détaillée

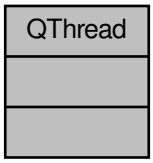
La classe QObject est la classe de base de tous les objets Qt. Elle permet à ces objets Qt de disposer entre autres du mécanisme de communication signal/slot.

La documentation de cette classe a été générée à partir du fichier suivant :
— main.cpp

6.23 Référence de la classe QThread

La classe QThread fournit un moyen indépendant de gérer les threads dans Qt.

Graphe de collaboration de QThread :



6.23.1 Description détaillée

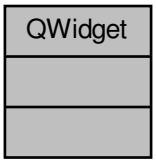
La classe QThread fournit un moyen indépendant de gérer les threads dans Qt.

La documentation de cette classe a été générée à partir du fichier suivant :
— main.cpp

6.24 Référence de la classe QWidget

La classe QWidget est la classe de base de tous les objets graphiques d'interface utilisateur.

Graphe de collaboration de QWidget :



6.24.1 Description détaillée

La classe QWidget est la classe de base de tous les objets graphiques d'interface utilisateur.

La documentation de cette classe a été générée à partir du fichier suivant :

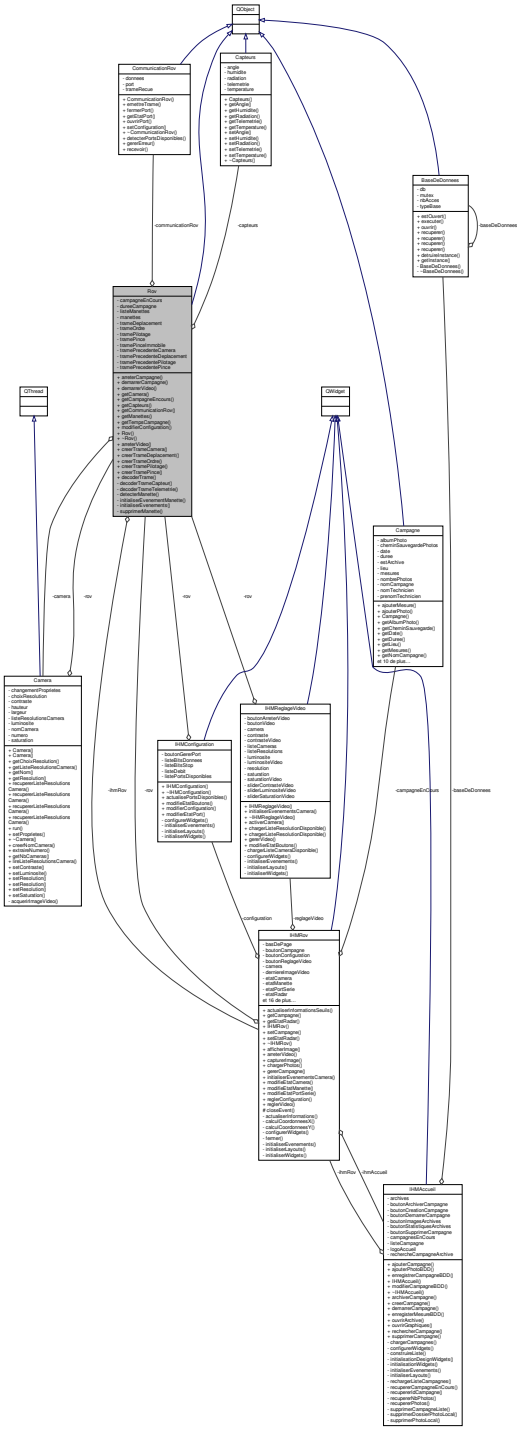
— main.cpp

6.25 Référence de la classe Rov

Classe controlant tout les traitements en provenance et en direction de la communication avec le rov.

```
#include "rov.h"
```

Graphe de collaboration de Rov :



Connecteurs publics

- void arreterVideo ()
  Arrete le flux vidéo.
- void creerTrameCamera (QString axeX, QString axeY)
  Crée les trames de la caméra.
- void creerTrameDeplacement (char deplacementAxeX, int puissance, char deplacementAxeY)
  Crée les trames de déplacement.
- void creerTrameOrdre (QString ordre)
  Crée les trames d'ordre.
- void creerTramePilotage (QString deplacement)
  Crée les trames de pilotage.
- void creerTramePince (QString mouvementPince)
  Crée les trames de la pince.
- void decoderTrame (QString trame)
  Décode la trame réçue par le port série selon le protocole établie.

Signaux

- void donneesTelemetrieDecode (QString donnes)
  signal contenant les nouvelle données de télémetrie (pour l'ihm)
- void enregistrerMesures (QString temperature, QString humidite, QString radiation)
  signal contenant les nouvelles mesures recus du robot (pour les enregistrer dans la BDD)
- void styleRadar (bool etat)
  signale le style de Radar à afficher

Fonctions membres publiques

- void arreterCampagne ()
  Arrête la campagne.
- bool demarrerCampagne ()
  Démarre la campagne.
- bool demarrerVideo (QString nomCamera, int choixResolution=-1)
  Démarre un nouveau flux vidéo.
- Camera * getCamera ()
  Retourne l'objet caméra créée par le rov.
- bool getCampagneEncours () const
  Retourne l'etat de la campagne.
- Capteurs * getCapteurs ()
  Retourne l'objet capteurs créée par le rov.
- CommunicationRov * getCommunicationRov ()
  Retourne l'objet communicationRov créée par le rov.
- QVector< Manette * > getManettes ()
  Retourne le conteneur de manettes créée par le rov.
- QString getTempsCampagne ()
  Retourne la durée de la campagne.
- void modifierConfiguration (Configuration &configuration)
  Modifie la configuration de la communication.
- Rov (IHMRov *ihmRov, QObject *parent=nullptr)
  Constructeur de la classe Rov.
- ~Rov ()
  Destructeur de la classe Rov.

Fonctions membres privées

- void decoderTrameCapteur (QString trameCapteur)
  Décode la trame capteur reçue selon le protocole.
- void decoderTrameTelemetrie (QString trameTelemetrie)
  Décode la trame télémetrie reçue selon le protocole.
- void detecterManette ()
  detecte les mannettes
- void initialiserEvenementManette (Manette *manette)
  Initialise les evenement de la manette.
- void initialiserEvenements ()
  Initialise le(s) Evenement(s)
- void supprimerManette ()
  Supprime les manettes du conteneur des manettes.

Attributs privés

- Camera * camera
  Instance d'un objet camera possedant les informations nécessaire à l'affichage du flux vidéo.
- bool campagneEnCours
  Etat de si une campagne est en cour.
- Capteurs * capteurs
  Instance d'un objet contenant les dernières informations issues des capteurs du rov.
- CommunicationRov * communicationRov
  Instance d'un objet permettant la récupération des trames envoyé par la liaison série.
- QTime dureeCampagne
  Timer lancer au début de la campagne.
- IHMRov * ihmRov
  Instance d'un objet ihmRov permettant la connexion entre les autre classe associé aux rov et l'ihm.
- QList< int > listeManettes
  Liste des mannettes connecté
- QVector< Manette * > manettes
  Conteneur des manettes.
- QString trameDeplacement
  Contenu trame déplacement.
- QString trameOrdre
  Contenu trame ordre.
- QString tramePilotage
  Contenu trame pilotage.
- QString tramePince
  Contenu trame pince.
- QString tramePinceImmobile
  Contenu trame pince à zéro.
- QString tramePrecedenteCamera
  Contenu précédente trame caméra.
- QString tramePrecedenteDeplacement
  Contenu précédente trame déplacement.
- QString tramePrecedentePilotage
  Contenu précédente trame pilotage.
- QString tramePrecedentePince
  Contenu précédente trame pince.

6.25.1 Description détaillée

Classe controlant tout les traitements en provenance et en direction de la communication avec le rov.

Définition à la ligne 91 du fichier rov.h.

6.25.2 Documentation des constructeurs et destructeur

6.25.2.1 Rov()

```
Rov::Rov (
            IHMRov * ihmRov,
            QObject * parent = nullptr )
```

Constructeur de la classe Rov.

Paramètres

ihmRov	
parent	

Définition à la ligne 11 du fichier rov.cpp.

Références capteurs, communicationRov, detecterManette(), et initialiserEvenements().

```
00011                                       : QObject(parent), ihmRov(ihmRov),
      camera(nullptr), trameDeplacement("$DEP;0;0;0\r\n"),
      tramePrecedenteDeplacement("$DEP;0;0;0\r\n"),
      tramePilotage("$BRAS;0\r\n"), tramePrecedentePilotage("$BRAS;0\r\n"),
      tramePince("$PINCE;0\r\n"), tramePrecedentePince("$PINCE;0\r\n"),
      tramePrecedenteCamera(TRAME_CAMERA_IMMOBILE),
00012 campagneEnCours(false)
00013 {
00014     qDebug() << Q_FUNC_INFO;
00015     communicationRov = new CommunicationRov(this);
00016     capteurs = new Capteurs(this);
00017
00018     initialiserEvenements();
00019     detecterManette();
00020 }
```

6.25.2.2 ~Rov()

```
Rov::~Rov ( )
```

Destructeur de la classe Rov.

Définition à la ligne 22 du fichier rov.cpp.

Références arreterCampagne(), et supprimerManette().

```
00023 {
00024     supprimerManette();
00025     arreterCampagne();
00026     qDebug() << Q_FUNC_INFO;
00027 }
```

6.25.3 Documentation des fonctions membres

6.25.3.1 arreterCampagne()

```
void Rov::arreterCampagne ( )
```

Arrête la campagne.

Définition à la ligne 140 du fichier rov.cpp.

Références arreterVideo(), campagneEnCours, dureeCampagne, IHMRov : :getCampagne(), ihmRov, et Campagne : :setDuree().

Référencé par IHMRov : :fermer(), et ~Rov().

```
00141 {
00142     qDebug() << Q_FUNC_INFO;
00143     ihmRov->getCampagne()->setDuree(dureeCampagne.elapsed());
00144     arreterVideo();
00145     campagneEnCours = false;
00146 }
```

6.25.3.2 arreterVideo

```
void Rov::arreterVideo ( )  [slot]
```

Arrete le flux vidéo.

Définition à la ligne 273 du fichier rov.cpp.

Références IHMRov : :arreterVideo(), camera, ihmRov, et IHMRov : :modifieEtatCamera().

Référencé par arreterCampagne(), et IHMReglageVideo : :gererVideo().

```
00274 {
00275     if(camera != nullptr)
00276     {
00277         qDebug() << Q_FUNC_INFO << "isRunning" << camera->isRunning();
00278         ihmRov->arreterVideo();
00279         disconnect(camera, SIGNAL(nouvelleImage(QPixmap)), ihmRov, SLOT(afficherImage(QPixmap))
      );
00280         camera->requestInterruption();
00281         camera->wait();
00282         delete camera;
00283         camera = nullptr;
00284         ihmRov->modifieEtatCamera(false, "");
00285     }
00286 }
```

6.25.3.3 creerTrameCamera

```
void Rov::creerTrameCamera (
             QString axeX,
             QString axeY )  [slot]
```

Crée les trames de la caméra.

Paramètres

axeX	
axeY	

Définition à la ligne 261 du fichier rov.cpp.

Références communicationRov, DEBUT_TRAME_CAMERA, CommunicationRov : :emettreTrame(), et tramePrecedenteCamera.

Référencé par initialiserEvenementManette().

```
00262 {
00263     QString trameCamera = DEBUT_TRAME_CAMERA ";" + axeX + ";" + axeY + "\r\n";
00264
00265     if(tramePrecedenteCamera != trameCamera)
00266     {
00267         tramePrecedenteCamera = trameCamera;
00268         communicationRov->emettreTrame(trameCamera);
00269         qDebug() << Q_FUNC_INFO << "trame camera" << trameCamera;
00270     }
00271 }
```

6.25.3.4 creerTrameDeplacement

```
void Rov::creerTrameDeplacement (
            char deplacementAxeX,
            int puissance,
            char deplacementAxeY )  [slot]
```

Crée les trames de déplacement.

Définition à la ligne 213 du fichier rov.cpp.

Références communicationRov, DEBUT_TRAME_DEPLACEMENT, CommunicationRov : :emettreTrame(), ihmRov, IHMRov : :setEtatRadar(), TRAME_DEPLACEMENT_IMMOBILE, trameDeplacement, et tramePrecedenteDeplacement.

Référencé par initialiserEvenementManette().

```
00214 {
00215     trameDeplacement = DEBUT_TRAME_DEPLACEMENT ";" + QString(
      deplacementAxeX) + ";" + QString::number(puissance) + ";" + QString(deplacementAxeY) + "\r\n";
00216
00217     if(tramePrecedenteDeplacement != trameDeplacement)
00218     {
00219         if(TRAME_DEPLACEMENT_IMMOBILE ==
      trameDeplacement)
00220             ihmRov->setEtatRadar(true);
00221         else
00222             ihmRov->setEtatRadar(false);
00223         qDebug() << Q_FUNC_INFO << "trameDeplacement :" << trameDeplacement;
00224         tramePrecedenteDeplacement = trameDeplacement;
00225         communicationRov->emettreTrame(trameDeplacement);
00226     }
00227 }
```

6.25.3.5 creerTrameOrdre

```
void Rov::creerTrameOrdre (
            QString ordre )  [slot]
```

Crée les trames d'ordre.

Paramètres

ordre	

Définition à la ligne 240 du fichier rov.cpp.

Références communicationRov, DEBUT_TRAME_ORDRE, CommunicationRov : :emettreTrame(), TRAME_PILOTAGE_IMMOBILE, TRAME_PINCE_IMMOBILE, trameOrdre, tramePince, et tramePrecedentePilotage.

Référencé par initialiserEvenementManette().

```
00241 {
00242     trameOrdre = DEBUT_TRAME_ORDRE ";" + ordre + "\r\n";
00243     if(tramePrecedentePilotage == TRAME_PILOTAGE_IMMOBILE &&
      tramePince == TRAME_PINCE_IMMOBILE)
00244     {
00245         qDebug() << Q_FUNC_INFO << "trameOrdre :" << trameOrdre;
00246         communicationRov->emettreTrame(trameOrdre);
00247     }
00248 }
```

6.25.3.6 creerTramePilotage

```
void Rov::creerTramePilotage (
            QString deplacement )  [slot]
```

Crée les trames de pilotage.

Paramètres

deplacement	

Définition à la ligne 229 du fichier rov.cpp.

Références communicationRov, DEBUT_TRAME_PILOTAGE, CommunicationRov : :emettreTrame(), TRAME_PINCE_IMMOBILE, tramePilotage, tramePince, et tramePrecedentePilotage.

Référencé par initialiserEvenementManette().

```
00230 {
00231     tramePilotage = DEBUT_TRAME_PILOTAGE ";" + deplacement + "\r\n";
00232     if(tramePrecedentePilotage != tramePilotage &&
      tramePince == TRAME_PINCE_IMMOBILE)
00233     {
00234         qDebug() << Q_FUNC_INFO << "tramePilotage :" << tramePilotage;
00235         tramePrecedentePilotage = tramePilotage;
00236         communicationRov->emettreTrame(tramePilotage);
00237     }
00238 }
```

6.25.3.7 creerTramePince

```
void Rov::creerTramePince (
            QString mouvementPince )  [slot]
```

Crée les trames de la pince.

Paramètres

mouvementPince	

Définition à la ligne 250 du fichier rov.cpp.

Références communicationRov, DEBUT_TRAME_PINCE, CommunicationRov : :emettreTrame(), TRAME_PILOTAGE_IMMOBILE, tramePince, tramePrecedentePilotage, et tramePrecedentePince.

Référencé par initialiserEvenementManette().

```
00251 {
00252     tramePince = DEBUT_TRAME_PINCE ";" + mouvementPince + "\r\n";
00253     if(tramePrecedentePilotage == TRAME_PILOTAGE_IMMOBILE &&
      tramePrecedentePince != tramePince)
00254     {
00255         qDebug() << Q_FUNC_INFO << "tramePince :" << tramePince;
00256         tramePrecedentePince = tramePince;
00257         communicationRov->emettreTrame(tramePince);
00258     }
00259 }
```

6.25.3.8 decoderTrame

```
void Rov::decoderTrame (
            QString trame )  [slot]
```

Décode la trame réçue par le port série selon le protocole établie.

Paramètres

trame	

Définition à la ligne 185 du fichier rov.cpp.

Références DEBUT_TRAME_CAPTEUR, DEBUT_TRAME_TELEMETRIE, decoderTrameCapteur(), et decoderTrameTelemetrie().

Référencé par initialiserEvenements().

```
00186 {
00187     QString trameTelemetrie, trameCapteur;
00188
00189     QStringList trames = trame.split("\r\n");
00190     qDebug() << Q_FUNC_INFO << "trame" << trames.size();
00191     for(int i = 0; i < trames.size(); i++)
00192     {
00193         if(!trames[i].isEmpty())
00194         {
00195             if(trames[i].startsWith(DEBUT_TRAME_TELEMETRIE))
00196             {
00197                 trameTelemetrie = trames[i];
00198                 decoderTrameTelemetrie(trameTelemetrie);
00199             }
00200             else if(trames[i].startsWith(DEBUT_TRAME_CAPTEUR))
00201             {
00202                 trameCapteur = trames[i];
00203                 decoderTrameCapteur(trameCapteur);
00204             }
00205         }
00206         else
00207         {
00208             qDebug() << Q_FUNC_INFO << "Trame inconnue !";
00209         }
00210     }
00211 }
```

6.25.3.9 decoderTrameCapteur()

```
void Rov::decoderTrameCapteur (
            QString trameCapteur )  [private]
```

Décode la trame capteur reçue selon le protocole.

Paramètres

trameCapteur	

Définition à la ligne 88 du fichier rov.cpp.

Références IHMRov : :actualiserInformationsSeuils(), Campagne : :ajouterMesure(), capteurs, Mesure : :dateheure, enregistrer↩Mesures(), IHMRov : :getCampagne(), Mesure : :humidite, ihmRov, Mesure : :radiation, Capteurs : :setHumidite(), Capteurs : :set↩Radiation(), Capteurs : :setTemperature(), et Mesure : :temperature.

Référencé par decoderTrame().

```
00089 {
00090     QString temperature, humidite, radiation;
00091
00092     temperature = QString::number(trameCapteur.section(";",1,1).toDouble()/10);
00093     humidite = trameCapteur.section(";",2,2);
00094     radiation = trameCapteur.section(";",3,3);
00095
00096     capteurs->setTemperature(temperature);
00097     capteurs->setHumidite(humidite);
00098     capteurs->setRadiation(radiation);
00099
00100     Mesure mesure;
00101     mesure.dateheure = QDateTime::currentDateTime();
00102     mesure.temperature = temperature;
00103     mesure.humidite = humidite;
00104     mesure.radiation = radiation;
00105
00106     QDateTime date;
00107
00108     ihmRov->getCampagne()->ajouterMesure(mesure);
00109     ihmRov->actualiserInformationsSeuils();
00110
00111     emit enregistrerMesures(temperature, humidite, radiation);
00112
00113 }
```

6.25.3.10 decoderTrameTelemetrie()

```
void Rov::decoderTrameTelemetrie (
            QString trameTelemetrie )  [private]
```

Décode la trame télémetrie reçue selon le protocole.

Paramètres

trameTelemetrie	

Définition à la ligne 82 du fichier rov.cpp.

Références capteurs, Capteurs : :setAngle(), et Capteurs : :setTelemetrie().

Référencé par decoderTrame().

```
00083 {
00084     capteurs->setTelemetrie(trameTelemetrie.section(";",1,1));
00085     capteurs->setAngle(trameTelemetrie.section(";",2,2));
00086 }
```

6.25.3.11 demarrerCampagne()

```
bool Rov::demarrerCampagne ( )
```

Démarre la campagne.

Définition à la ligne 127 du fichier rov.cpp.

Références CAMERA_DEFAUT, campagneEnCours, Camera : :creerNomCamera(), demarrerVideo(), dureeCampagne, et Camera↩ : :getNbCameras().

Référencé par IHMRov : :gererCampagne().

```
00128 {
00129     qDebug() << Q_FUNC_INFO;
00130     if(Camera::getNbCameras() > 0)
00131     {
00132         dureeCampagne.start();
00133         QString nom = Camera::creerNomCamera(CAMERA_DEFAUT);
00134         if(demarrerVideo(nom))
00135             campagneEnCours = true;
00136     }
00137     return campagneEnCours;
00138 }
```

6.25.3.12 demarrerVideo()

```
bool Rov::demarrerVideo (
            QString nomCamera,
            int choixResolution = -1 )
```

Démarre un nouveau flux vidéo.

Paramètres

nomCamera	
choixResolution	

Définition à la ligne 165 du fichier rov.cpp.

Références camera, ihmRov, IHMRov : :initialiserEvenementsCamera(), et IHMRov : :modifieEtatCamera().

Référencé par IHMReglageVideo : :activerCamera(), et demarrerCampagne().

```
00166 {
00167     qDebug() << Q_FUNC_INFO << "nomCamera" << nomCamera << "choixResolution" << choixResolution;
00168     if(camera == nullptr)
00169     {
00170         camera = new Camera(this, nomCamera, choixResolution);
00171         connect(camera, SIGNAL(nouvelleImage(QPixmap)), ihmRov, SLOT(afficherImage(QPixmap)));
00172         ihmRov->initialiserEvenementsCamera();
00173         camera->start();
00174         ihmRov->modifieEtatCamera(true, nomCamera);
00175         return true;
00176     }
00177     return false;
00178 }
```

6.25.3.13 detecterManette()

```
void Rov::detecterManette ( )  [private]
```

detecte les mannettes

Définition à la ligne 29 du fichier rov.cpp.

Références initialiserEvenementManette(), listeManettes, et manettes.

Référencé par Rov().

```
00030 {
00031     listeManettes = QGamepadManager::instance()->connectedGamepads();
00032
00033     if (listeManettes.isEmpty())
00034     {
00035         qDebug() << Q_FUNC_INFO << "Aucune manette détectée !";
00036     }
00037     else
00038     {
00039         qDebug() << Q_FUNC_INFO << "Nombre de manettes" << listeManettes.size();
00040     }
00041
00042     for (auto m : listeManettes)
00043     {
00044         qDebug() << Q_FUNC_INFO << "-> Manette" << m;
00045         Manette *manette = new Manette(m);
00046         manettes.push_back(manette);
00047         initialiserEvenementManette(manette);
00048     }
00049 }
```

6.25.3.14 donneesTelemetrieDecode

```
void Rov::donneesTelemetrieDecode (
            QString donnes )  [signal]
```

signal contenant les nouvelle données de télémetrie (pour l'ihm)

Paramètres

donnes	

6.25.3.15 enregistrerMesures

```
void Rov::enregistrerMesures (
            QString temperature,
            QString humidite,
            QString radiation )  [signal]
```

signal contenant les nouvelles mesures recus du robot (pour les enregistrer dans la BDD)

Paramètres

temperature	
humidite	
radiation	

Référencé par decoderTrameCapteur().

6.25.3.16 getCamera()

```
Camera * Rov::getCamera ( )
```

Retourne l'objet caméra créée par le rov.

Renvoie

L'objet caméra créée par le rov

Définition à la ligne 148 du fichier rov.cpp.

Références camera.

Référencé par IHMReglageVideo : :chargerListeCameraDisponible(), IHMReglageVideo : :chargerListeResolutionDisponible(), IHM↩ReglageVideo : :initialiserEvenementsCamera(), et IHMReglageVideo : :modifierEtatBoutons().

```
00149 {
00150     return camera;
00151 }
```

6.25.3.17 getCampagneEncours()

```
bool Rov::getCampagneEncours ( ) const
```

Retourne l'etat de la campagne.

Renvoie

l'etat de campagneEnCours (indiquant si la campagne est en cours ou pas)

Définition à la ligne 180 du fichier rov.cpp.

Références campagneEnCours.

Référencé par IHMRov : :closeEvent().

```
00181 {
00182     return campagneEnCours;
00183 }
```

6.25.3.18 getCapteurs()

```
Capteurs * Rov::getCapteurs ( )
```

Retourne l'objet capteurs créée par le rov.

Renvoie

L'objet capteurs créée par le rov

Définition à la ligne 153 du fichier rov.cpp.

Références capteurs.

Référencé par IHMRov : :actualiserInformations(), IHMRov : :actualiserInformationsSeuils(), IHMRov : :calculCoordonneesX(), et IH↩MRov : :calculCoordonneesY().

```
00154 {
00155     return capteurs;
00156 }
```

6.25.3.19 getCommunicationRov()

```
CommunicationRov * Rov::getCommunicationRov ( )
```

Retourne l'objet communicationRov créée par le rov.

Renvoie

L'objet communicationRov créée par le rov

Définition à la ligne 293 du fichier rov.cpp.

Références communicationRov.

Référencé par IHMConfiguration : :IHMConfiguration(), IHMConfiguration : :modifieEtatBoutons(), et IHMConfiguration : :modifier↩ EtatPort().

```
00294 {
00295     return communicationRov;
00296 }
```

6.25.3.20 getManettes()

```
QVector< Manette * > Rov::getManettes ( )
```

Retourne le conteneur de manettes créée par le rov.

Renvoie

le conteneur de manettes créée par le rov

Définition à la ligne 298 du fichier rov.cpp.

Références manettes.

Référencé par IHMRov : :IHMRov().

```
00299 {
00300     return manettes;
00301 }
```

6.25.3.21 getTempsCampagne()

```
QString Rov::getTempsCampagne ( )
```

Retourne la durée de la campagne.

Renvoie

La durée de la campagne

Définition à la ligne 158 du fichier rov.cpp.

Références dureeCampagne, IHMRov : :getCampagne(), Campagne : :getDuree(), et ihmRov.

Référencé par IHMRov : :actualiserInformations().

```
00159 {
00160     QTime duree(0,0);
00161     QTime dureeMission = duree.addMSecs(dureeCampagne.elapsed() +
      ihmRov->getCampagne()->getDuree());
00162     return dureeMission.toString("hh:mm:ss");
00163 }
```

6.25.3.22 initialiserEvenementManette()

```
void Rov::initialiserEvenementManette (
            Manette * manette )  [private]
```

Initialise les evenement de la manette.

Paramètres

manette	

Définition à la ligne 51 du fichier rov.cpp.

Références creerTrameCamera(), creerTrameDeplacement(), creerTrameOrdre(), creerTramePilotage(), creerTramePince(), et ihmRov.

Référencé par detecterManette().

```
00052 {
00053     connect(manette, SIGNAL(creationTrameDeplacement(char, int, char)), this, SLOT(
      creerTrameDeplacement(char, int, char)));
00054     connect(manette, SIGNAL(creationTramePilotage(QString)), this, SLOT(
      creerTramePilotage(QString)));
00055     connect(manette, SIGNAL(creationTrameOrdre(QString)), this, SLOT(
      creerTrameOrdre(QString)));
00056     connect(manette, SIGNAL(creationTramePince(QString)), this, SLOT(
      creerTramePince(QString)));
00057     connect(manette, SIGNAL(nouvelleTrameCamera(QString,QString)), this, SLOT(
      creerTrameCamera(QString,QString)));
00058
00059     connect(manette, SIGNAL(axisLeftXChanged(double)), manette, SLOT(changerAxeXJoystickGauche(double)));
00060     connect(manette, SIGNAL(axisLeftYChanged(double)), manette, SLOT(changerAxeYJoystickGauche(double)));
00061     connect(manette, SIGNAL(axisRightXChanged(double)), manette, SLOT(changerAxeXJoystickDroit(double)));
00062     connect(manette, SIGNAL(axisRightYChanged(double)), manette, SLOT(changerAxeYJoystickDroit(double)));
00063     connect(manette, SIGNAL(buttonL1Changed(bool)), manette, SLOT(changerBoutonHautGauche(bool)));
00064     connect(manette, SIGNAL(buttonR1Changed(bool)), manette, SLOT(changerBoutonHautDroit(bool)));
00065     connect(manette, SIGNAL(buttonL2Changed(double)), manette, SLOT(changerGachetteBasGauche(double)));
00066     connect(manette, SIGNAL(buttonR2Changed(double)), manette, SLOT(changerGachetteBasDroit(double)));
00067     connect(manette, SIGNAL(buttonUpChanged(bool)), manette, SLOT(changerFlecheEnAvant(bool)));
00068     connect(manette, SIGNAL(buttonDownChanged(bool)), manette, SLOT(changerFlecheEnArriere(bool)));
00069     connect(manette, SIGNAL(buttonLeftChanged(bool)), manette, SLOT(changerFlecheAGauche(bool)));
00070     connect(manette, SIGNAL(buttonRightChanged(bool)), manette, SLOT(changerFlecheADroite(bool)));
00071     connect(manette, SIGNAL(buttonAChanged(bool)), manette, SLOT(changerBoutonA(bool)));
00072     connect(manette, SIGNAL(buttonBChanged(bool)), manette, SLOT(changerBoutonB(bool)));
00073     connect(manette, SIGNAL(buttonXChanged(bool)), manette, SLOT(changerBoutonX(bool)));
00074     connect(manette, SIGNAL(buttonYChanged(bool)), manette, SLOT(changerBoutonY(bool)));
00075     connect(manette, SIGNAL(buttonSelectChanged(bool)), manette, SLOT(changerBoutonSelect(bool)));
00076     connect(manette, SIGNAL(buttonGuideChanged(bool)), manette, SLOT(fermerApplication(bool)));
00077     connect(manette, SIGNAL(buttonR3Changed(bool)), ihmRov, SLOT(capturerImage(bool)));
00078
00079     connect(manette, SIGNAL(connectedChanged(bool)), ihmRov, SLOT(modifieEtatManette(bool)));
00080 }
```

6.25.3.23 initialiserEvenements()

```
void Rov::initialiserEvenements ( )  [private]
```

Initialise le(s) Evenement(s)

Définition à la ligne 115 du fichier rov.cpp.

Références communicationRov, decoderTrame(), et ihmRov.

Référencé par Rov().

```
00116 {
00117     connect(communicationRov, SIGNAL(nouvelleTrame(QString)), this, SLOT(
      decoderTrame(QString)));
00118     connect(communicationRov, SIGNAL(etatPortModifie(bool, QString)),
      ihmRov, SLOT(modifieEtatPortSerie(bool, QString)));
00119 }
```

6.25.3.24 modifierConfiguration()

```
void Rov::modifierConfiguration (
            Configuration & configuration )
```

Modifie la configuration de la communication.

Paramètres

configuration	

Définition à la ligne 288 du fichier rov.cpp.

Références communicationRov, et CommunicationRov : :setConfiguration().

Référencé par IHMConfiguration : :modifierConfiguration().

```
00289 {
00290     communicationRov->setConfiguration(configuration);
00291 }
```

6.25.3.25 styleRadar

```
void Rov::styleRadar (
            bool etat )  [signal]
```

signale le style de Radar à afficher

Paramètres

etat	

6.25.3.26 supprimerManette()

```
void Rov::supprimerManette ( )  [private]
```

Supprime les manettes du conteneur des manettes.

Définition à la ligne 121 du fichier rov.cpp.

Références manettes.

Référencé par ∼Rov().

```
00122 {
00123     for(int i=0;i<manettes.size();i++)
00124         delete manettes[i];
00125 }
```

6.25.4 Documentation des données membres

6.25.4.1 camera

```
Camera* Rov::camera  [private]
```

Instance d'un objet camera possedant les informations nécessaire à l'affichage du flux vidéo.

Définition à la ligne 97 du fichier rov.h.

Référencé par arreterVideo(), demarrerVideo(), et getCamera().

6.25.4.2 campagneEnCours

```
bool Rov::campagneEnCours  [private]
```

Etat de si une campagne est en cour.

Définition à la ligne 111 du fichier rov.h.

Référencé par arreterCampagne(), demarrerCampagne(), et getCampagneEncours().

6.25.4.3 capteurs

```
Capteurs* Rov::capteurs  [private]
```

Instance d'un objet contenant les dernières informations issues des capteurs du rov.

Définition à la ligne 96 du fichier rov.h.

Référencé par decoderTrameCapteur(), decoderTrameTelemetrie(), getCapteurs(), et Rov().

6.25.4.4 communicationRov

```
CommunicationRov* Rov::communicationRov  [private]
```

Instance d'un objet permettant la récupération des trames envoyé par la liaison série.

Définition à la ligne 98 du fichier rov.h.

Référencé par creerTrameCamera(), creerTrameDeplacement(), creerTrameOrdre(), creerTramePilotage(), creerTramePince(), getCommunicationRov(), initialiserEvenements(), modifierConfiguration(), et Rov().

6.25.4.5 dureeCampagne

```
QTime Rov::dureeCampagne  [private]
```

Timer lancer au début de la campagne.

Définition à la ligne 110 du fichier rov.h.

Référencé par arreterCampagne(), demarrerCampagne(), et getTempsCampagne().

6.25.4.6 ihmRov

```
IHMRov* Rov::ihmRov  [private]
```

Instance d'un objet ihmRov permettant la connexion entre les autre classe associé aux rov et l'ihm.

Définition à la ligne 95 du fichier rov.h.

Référencé par arreterCampagne(), arreterVideo(), creerTrameDeplacement(), decoderTrameCapteur(), demarrerVideo(), getTemps↩Campagne(), initialiserEvenementManette(), et initialiserEvenements().

6.25.4.7 listeManettes

```
QList<int> Rov::listeManettes  [private]
```

Liste des mannettes connecté

Définition à la ligne 99 du fichier rov.h.

Référencé par detecterManette().

6.25.4.8 manettes

```
QVector<Manette*> Rov::manettes  [private]
```

Conteneur des manettes.

Définition à la ligne 100 du fichier rov.h.

Référencé par detecterManette(), getManettes(), et supprimerManette().

6.25.4.9 trameDeplacement

```
QString Rov::trameDeplacement  [private]
```

Contenu trame déplacement.

Définition à la ligne 101 du fichier rov.h.

Référencé par creerTrameDeplacement().

6.25.4.10 trameOrdre

```
QString Rov::trameOrdre  [private]
```

Contenu trame ordre.

Définition à la ligne 105 du fichier rov.h.

Référencé par creerTrameOrdre().

6.25.4.11 tramePilotage

```
QString Rov::tramePilotage  [private]
```

Contenu trame pilotage.

Définition à la ligne 103 du fichier rov.h.

Référencé par creerTramePilotage().

6.25.4.12 tramePince

```
QString Rov::tramePince  [private]
```

Contenu trame pince.

Définition à la ligne 106 du fichier rov.h.

Référencé par creerTrameOrdre(), creerTramePilotage(), et creerTramePince().

6.25.4.13 tramePinceImmobile

```
QString Rov::tramePinceImmobile  [private]
```

Contenu trame pince à zéro.

Définition à la ligne 109 du fichier rov.h.

6.25.4.14 tramePrecedenteCamera

```
QString Rov::tramePrecedenteCamera  [private]
```

Contenu précédente trame caméra.

Définition à la ligne 108 du fichier rov.h.

Référencé par creerTrameCamera().

6.25.4.15 tramePrecedenteDeplacement

```
QString Rov::tramePrecedenteDeplacement  [private]
```

Contenu précédente trame déplacement.

Définition à la ligne 102 du fichier rov.h.

Référencé par creerTrameDeplacement().

6.25.4.16 tramePrecedentePilotage

```
QString Rov::tramePrecedentePilotage  [private]
```

Contenu précédente trame pilotage.

Définition à la ligne 104 du fichier rov.h.

Référencé par creerTrameOrdre(), creerTramePilotage(), et creerTramePince().

6.25.4.17 tramePrecedentePince

```
QString Rov::tramePrecedentePince  [private]
```

Contenu précédente trame pince.

Définition à la ligne 107 du fichier rov.h.

Référencé par creerTramePince().

La documentation de cette classe a été générée à partir des fichiers suivants :

— rov.h
— rov.cpp

7 Documentation des fichiers

7.1 Référence du fichier basededonnees.cpp

Fichier qui contient la définition de la classe BaseDeDonnees.

```
#include "basededonnees.h"
#include <QDebug>
#include <QMessageBox>
```

7.1.1 Description détaillée

Fichier qui contient la définition de la classe BaseDeDonnees.

Définition dans le fichier basededonnees.cpp.

7.2 basededonnees.cpp

```
00001
00007 #include "basededonnees.h"
00008 #include <QDebug>
00009 #include <QMessageBox>
00010
00011 BaseDeDonnees* BaseDeDonnees::baseDeDonnees = nullptr;
00012 QString BaseDeDonnees::typeBase = "QSQLITE";
00013 int BaseDeDonnees::nbAcces = 0;
00014
00015 BaseDeDonnees::BaseDeDonnees(QString type)
00016 {
00017     #ifdef DEBUG_BASEDEDONNEES
00018     qDebug() << Q_FUNC_INFO << type;
00019     #endif
00020     db = QSqlDatabase::addDatabase(type);
00021     typeBase = type;
00022 }
00023
00024 BaseDeDonnees::~BaseDeDonnees()
00025 {
00026     #ifdef DEBUG_BASEDEDONNEES
00027     qDebug() << Q_FUNC_INFO;
00028     #endif
00029 }
00030
00031 BaseDeDonnees* BaseDeDonnees::getInstance(QString type)
00032 {
00033     if(baseDeDonnees == nullptr)
00034         baseDeDonnees = new BaseDeDonnees(type);
00035
00036     nbAcces++;
00037     #ifdef DEBUG_BASEDEDONNEES
00038     qDebug() << Q_FUNC_INFO << "nbAcces" << nbAcces;
00039     #endif
00040
00041     return baseDeDonnees;
00042 }
00043
00044 void BaseDeDonnees::detruireInstance()
00045 {
00046     if(baseDeDonnees != nullptr)
00047     {
00048         if(nbAcces > 0)
00049             nbAcces--;
00050
00051         #ifdef DEBUG_BASEDEDONNEES
00052         qDebug() << Q_FUNC_INFO << "nbAcces restants" << nbAcces;
00053         #endif
00054
00055         if(nbAcces == 0)
00056         {
00057             delete baseDeDonnees;
00058             baseDeDonnees = nullptr;
00059         }
00060     }
00061 }
00062
00063 bool BaseDeDonnees::ouvrir(QString fichierBase)
00064 {
00065     if(typeBase != "QSQLITE")
00066         return false;
00067     QMutexLocker verrou(&mutex);
00068     if(!db.isOpen())
00069     {
00070        db.setDatabaseName(fichierBase);
00071
00072        #ifdef DEBUG_BASEDEDONNEES
00073        qDebug() << Q_FUNC_INFO << db.databaseName();
00074         #endif
00075
00076        if(db.open())
00077        {
00078
00079            #ifdef DEBUG_BASEDEDONNEES
00080            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Ouverture réussie à %1").arg(
      db.databaseName());
00081            #endif
00082
00083            return true;
00084        }
00085        else
00086        {
00087            #ifdef DEBUG_BASEDEDONNEES
00088            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : impossible d'ouvrir la base de données !"
      );
00089            #endif
00090            QMessageBox::critical(nullptr, QString::fromUtf8("BaseDeDonnees"), QString::fromUtf8("Impossible
       d'ouvrir la base de données !"));
```

```
00091             return false;
00092         }
00093     }
00094     else
00095         return true;
00096 }
00097
00098 bool BaseDeDonnees::estOuvert()
00099 {
00100     QMutexLocker verrou(&mutex);
00101     return db.isOpen();
00102 }
00103
00104 bool BaseDeDonnees::executer(QString requete)
00105 {
00106     QMutexLocker verrou(&mutex);
00107     QSqlQuery r;
00108     bool retour;
00109
00110     if(db.isOpen())
00111     {
00112         if(requete.contains("UPDATE") || requete.contains("INSERT") || requete.contains("DELETE"))
00113         {
00114             retour = r.exec(requete);
00115             #ifdef DEBUG_BASEDEDONNEES
00116             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00117             #endif
00118             if(retour)
00119             {
00120                 return true;
00121             }
00122             else
00123             {
00124                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00125                 return false;
00126             }
00127         }
00128         else
00129         {
00130             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00131             return false;
00132         }
00133     }
00134     else
00135         return false;
00136
00137 }
00138
00139 bool BaseDeDonnees::recuperer(QString requete, QString &donnees)
00140 {
00141     QMutexLocker verrou(&mutex);
00142     QSqlQuery r;
00143     bool retour;
00144
00145     if(db.isOpen())
00146     {
00147         if(requete.contains("SELECT"))
00148         {
00149             retour = r.exec(requete);
00150             #ifdef DEBUG_BASEDEDONNEES
00151             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00152             #endif
00153             if(retour)
00154             {
00155                 r.first();
00156
00157                 if(!r.isValid())
00158                 {
00159                     #ifdef DEBUG_BASEDEDONNEES
00160                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Résultat non valide !");
00161                     #endif
00162                     return false;
00163                 }
00164
00165                 if(r.isNull(0))
00166                 {
00167                     #ifdef DEBUG_BASEDEDONNEES
00168                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Aucun résultat !");
00169                     #endif
00170                     return false;
00171                 }
00172                 donnees = r.value(0).toString();
00173                 #ifdef DEBUG_BASEDEDONNEES
00174                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00175                 #endif
00176                 return true;
00177             }
```

```
00178             else
00179             {
00180                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00181                 return false;
00182             }
00183         }
00184         else
00185         {
00186             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00187             return false;
00188         }
00189     }
00190     else
00191         return false;
00192 }
00193
00194 bool BaseDeDonnees::recuperer(QString requete, QStringList &donnees)
00195 {
00196     QMutexLocker verrou(&mutex);
00197     QSqlQuery r;
00198     bool retour;
00199
00200     if(db.isOpen())
00201     {
00202         if(requete.contains("SELECT"))
00203         {
00204             retour = r.exec(requete);
00205             #ifdef DEBUG_BASEDEDONNEES
00206             qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString, QStringList)> retour %1 pour
       la requete : %2").arg(QString::number(retour)).arg(requete);
00207             #endif
00208             if(retour)
00209             {
00210                 r.first();
00211
00212                 if(!r.isValid())
00213                 {
00214                     #ifdef DEBUG_BASEDEDONNEES
00215                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Résultat non valide !");
00216                     #endif
00217                     return false;
00218                 }
00219
00220                 for(int i=0;i<r.record().count();i++)
00221                     if(!r.isNull(i))
00222                         donnees << r.value(i).toString();
00223                 #ifdef DEBUG_BASEDEDONNEES
00224                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00225                 #endif
00226                 return true;
00227             }
00228             else
00229             {
00230                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00231                 return false;
00232             }
00233         }
00234         else
00235         {
00236             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00237             return false;
00238         }
00239     }
00240     else
00241         return false;
00242 }
00243
00244 bool BaseDeDonnees::recuperer(QString requete, QVector<QString> &donnees)
00245 {
00246     QMutexLocker verrou(&mutex);
00247     QSqlQuery r;
00248     bool retour;
00249     QString data;
00250
00251     if(db.isOpen())
00252     {
00253         if(requete.contains("SELECT"))
00254         {
00255             retour = r.exec(requete);
00256             #ifdef DEBUG_BASEDEDONNEES
00257             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00258             #endif
00259             if(retour)
00260             {
00261                 while ( r.next() )
00262                 {
```

```
00263                     data = r.value(0).toString();
00264
00265                     #ifdef DEBUG_BASEDEDONNEES
00266                     qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00267                     #endif
00268
00269                     donnees.push_back(data);
00270                 }
00271                 #ifdef DEBUG_BASEDEDONNEES
00272                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00273                 #endif
00274                 return true;
00275             }
00276             else
00277             {
00278                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
    lastError().text()).arg(requete);
00279                 return false;
00280             }
00281         }
00282         else
00283         {
00284             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
    );
00285             return false;
00286         }
00287     }
00288     else
00289         return false;
00290 }
00291
00292 bool BaseDeDonnees::recuperer(QString requete, QVector<QStringList> &donnees)
00293 {
00294     QMutexLocker verrou(&mutex);
00295     QSqlQuery r;
00296     bool retour;
00297     QStringList data;
00298
00299     if(db.isOpen())
00300     {
00301         if(requete.contains("SELECT"))
00302         {
00303             retour = r.exec(requete);
00304             #ifdef DEBUG_BASEDEDONNEES
00305             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
    QString::number(retour)).arg(requete);
00306             #endif
00307             if(retour)
00308             {
00309                 while ( r.next() )
00310                 {
00311                     for(int i=0;i<r.record().count();i++)
00312                         data << r.value(i).toString();
00313
00314                     #ifdef DEBUG_BASEDEDONNEES
00315                     qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00316                     for(int i=0;i<r.record().count();i++)
00317                         qDebug() << r.value(i).toString();
00318                     #endif
00319
00320                     donnees.push_back(data);
00321
00322                     data.clear();
00323                 }
00324                 #ifdef DEBUG_BASEDEDONNEES
00325                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00326                 #endif
00327                 return true;
00328             }
00329             else
00330             {
00331                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
    lastError().text()).arg(requete);
00332                 return false;
00333             }
00334         }
00335         else
00336         {
00337             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
    );
00338             return false;
00339         }
00340     }
00341     else
00342         return false;
00343 }
```

7.3 Référence du fichier basededonnees.h

Fichier qui contient la déclaration de la classe BaseDeDonnees.

```
#include <QObject>
#include <QtSql/QtSql>
#include <QSqlDatabase>
#include <QMutex>
#include <QString>
```

Classes

— class BaseDeDonnees
Class permettant de s'interfacer avec la base de données.

Macros

— #define DEBUG_BASEDEDONNEES

7.3.1 Description détaillée

Fichier qui contient la déclaration de la classe BaseDeDonnees.

Définition dans le fichier basededonnees.h.

7.3.2 Documentation des macros

7.3.2.1 DEBUG_BASEDEDONNEES

```
#define DEBUG_BASEDEDONNEES
```

Définition à la ligne 16 du fichier basededonnees.h.

7.4 basededonnees.h

```
00001
00007 #ifndef BASEDEDONNEE_H
00008 #define BASEDEDONNEE_H
00009
00010 #include <QObject>
00011 #include <QtSql/QtSql>
00012 #include <QSqlDatabase>
00013 #include <QMutex>
00014 #include <QString>
00015
00016 #define DEBUG_BASEDEDONNEES
00017
00023 class BaseDeDonnees : public QObject
00024 {
00025     Q_OBJECT
00026 private:
00027     static BaseDeDonnees*  baseDeDonnees;
00028     static QString         typeBase;
00029     static int             nbAcces;
00030     QSqlDatabase           db;
00031     QMutex                 mutex;
```

```
00032
00038     BaseDeDonnees(QString type);
00043     ~BaseDeDonnees();
00044
00045 public:
00052     static BaseDeDonnees* getInstance(QString type="QSQLITE");
00057     static void detruireInstance();
00064     bool ouvrir(QString fichierBase);
00070     bool estOuvert();
00071
00078     bool executer(QString requete);
00079
00087     bool recuperer(QString requete, QString &donnees);
00095     bool recuperer(QString requete, QStringList &donnees);
00103     bool recuperer(QString requete, QVector<QString> &donnees);
00111     bool recuperer(QString requete, QVector<QStringList> &donnees);
00112
00113 signals:
00114
00115 public slots:
00116 };
00117
00118 #endif // BASEDEDONNEE_H
```

7.5 Référence du fichier camera.cpp

Fichier qui contient la définition de la classe Camera.

```
#include "camera.h"
#include <QApplication>
#include <QDesktopWidget>
#include <QMediaRecorder>
```

7.5.1 Description détaillée

Fichier qui contient la définition de la classe Camera.

Définition dans le fichier camera.cpp.

7.6 camera.cpp

```
00001
00007 #include "camera.h"
00008 #include <QApplication>
00009 #include <QDesktopWidget>
00010 #include <QMediaRecorder>
00011
00012 Camera::Camera(Rov *rov, int numero, int choixResolution): rov(rov), numero(numero),
      largeur(LARGEUR_DEFAUT), hauteur(HAUTEUR_DEFAUT), luminosite(
      SEUIL_DEFAUT), contraste(SEUIL_DEFAUT), saturation(
      SEUIL_DEFAUT), changementProprietes(false), choixResolution(choixResolution)
00013 {
00014     #if CV_VERSION_MAJOR == 3
00015     qDebug() << Q_FUNC_INFO << "OpenCV" << CV_VERSION_MAJOR << CV_VERSION_MINOR;
00016     #else
00017     qDebug() << Q_FUNC_INFO << "OpenCV" << CV_MAJOR_VERSION << CV_MINOR_VERSION;
00018     #endif
00019
00020     Camera::getNbCameras();
00021
00022     nomCamera = Camera::creerNomCamera(numero);
00023     recupererListeResolutionsCamera();
00024     if(choixResolution == -1)
00025         setResolution(largeur, hauteur);
00026     else
00027         setResolution(choixResolution);
00028     qDebug() << Q_FUNC_INFO << this;
00029     qDebug() << Q_FUNC_INFO << "numero" << numero << "nomCamera" << nomCamera;
00030     qDebug() << Q_FUNC_INFO << "largeur" << largeur << "hauteur" <<
      hauteur;
00031     qDebug() << Q_FUNC_INFO << "luminosite" << luminosite << "contraste" <<
      contraste << "saturation" << saturation;
```

```
00032 }
00033
00034 Camera::Camera(Rov *rov, QString nomCamera, int
      choixResolution): rov(rov), nomCamera(nomCamera), largeur(
      LARGEUR_DEFAUT), hauteur(HAUTEUR_DEFAUT),
      luminosite(SEUIL_DEFAUT), contraste(SEUIL_DEFAUT),
      saturation(SEUIL_DEFAUT), changementProprietes(false),
      choixResolution(choixResolution)
00035 {
00036     #if CV_VERSION_MAJOR == 3
00037     qDebug() << Q_FUNC_INFO << "OpenCV" << CV_VERSION_MAJOR << CV_VERSION_MINOR;
00038     #else
00039     qDebug() << Q_FUNC_INFO << "OpenCV" << CV_MAJOR_VERSION << CV_MINOR_VERSION;
00040     #endif
00041
00042     Camera::getNbCameras();
00043
00044     numero = Camera::extraireNumero(nomCamera);
00045
00046     recupererListeResolutionsCamera();
00047     if(choixResolution == -1)
00048         setResolution(largeur, hauteur);
00049     else
00050         setResolution(choixResolution);
00051
00052     qDebug() << Q_FUNC_INFO << this;
00053     qDebug() << Q_FUNC_INFO << "numero" << numero << "nomCamera" <<
      nomCamera;
00054     qDebug() << Q_FUNC_INFO << "largeur" << largeur << "hauteur" <<
      hauteur;
00055     qDebug() << Q_FUNC_INFO << "luminosite" << luminosite << "contraste" <<
      contraste << "saturation" << saturation;
00056 }
00057
00058 Camera::~Camera()
00059 {
00060     qDebug() << Q_FUNC_INFO << this;
00061 }
00062
00063 void Camera::acquerirImageVideo(cv::VideoCapture &camera, cv::Mat &frame)
00064 {
00065     camera >> frame;
00066 }
00067
00068 void Camera::run()
00069 {
00070     qDebug() << Q_FUNC_INFO << "start" << "numero" << numero << "nomCamera" <<
      nomCamera;
00071     this->setPriority(QThread::NormalPriority);
00072
00073     if(numero < 0)
00074     {
00075         qDebug() << Q_FUNC_INFO  << "Erreur numero" << numero << "nomCamera" <<
      nomCamera;
00076         return;
00077     }
00078
00079     cv::VideoCapture camera(numero);
00080     cv::Mat frame;
00081
00082     setProprietes(camera);
00083
00084     while(camera.isOpened() && !isInterruptionRequested())
00085     {
00086         acquerirImageVideo(camera, frame);
00087         if(frame.empty())
00088             continue;
00089
00090         QPixmap pixmap = QPixmap::fromImage(QImage(frame.data, frame.cols, frame.rows, frame.step,
      QImage::Format_RGB888).rgbSwapped());
00091         emit nouvelleImage(pixmap);
00092
00093         if(changementProprietes)
00094             setProprietes(camera);
00095     }
00096
00097     camera.release();
00098     qDebug() << Q_FUNC_INFO << "stop" << "numero" << numero << "nomCamera" <<
      nomCamera;
00099
00100     emit finVideo();
00101 }
00102
00103 QString Camera::getNom() const
00104 {
00105     return nomCamera;
00106 }
00107
00108 void Camera::setProprietes(cv::VideoCapture &camera)
00109 {
00110     qDebug() << Q_FUNC_INFO << "largeur" << largeur << "hauteur" <<
```

```
    hauteur;
00111     camera.set(CV_CAP_PROP_FRAME_WIDTH, largeur);
00112     camera.set(CV_CAP_PROP_FRAME_HEIGHT, hauteur);
00113
00114     qDebug() << Q_FUNC_INFO << "luminosite" << luminosite << "contraste" <<
    contraste << "saturation" << saturation;
00115     camera.set(CV_CAP_PROP_BRIGHTNESS, luminosite);
00116     camera.set(CV_CAP_PROP_CONTRAST, contraste);
00117     camera.set(CV_CAP_PROP_SATURATION, saturation);
00118
00119     changementProprietes = false;
00120 }
00121
00122 QSize Camera::getResolution()
00123 {
00124     if(choixResolution != -1)
00125         return listeResolutionsCamera.at(choixResolution);
00126     return QSize(LARGEUR_DEFAUT,HAUTEUR_DEFAUT);
00127 }
00128
00129 int Camera::getChoixResolution()
00130 {
00131     return choixResolution;
00132 }
00133
00134 void Camera::recupererListeResolutionsCamera()
00135 {
00136     QCameraInfo cameraInfo(nomCamera.toLatin1());
00137     recupererListeResolutionsCamera(cameraInfo);
00138 }
00139
00140 void Camera::recupererListeResolutionsCamera(int
    numero)
00141 {
00142     QString nom = Camera::creerNomCamera(numero);
00143     QCameraInfo cameraInfo(nom.toLatin1());
00144     recupererListeResolutionsCamera(cameraInfo);
00145 }
00146
00147 void Camera::recupererListeResolutionsCamera(QString
    nomCamera)
00148 {
00149     QCameraInfo cameraInfo(nomCamera.toLatin1());
00150     recupererListeResolutionsCamera(cameraInfo);
00151 }
00152
00153 void Camera::recupererListeResolutionsCamera(QCameraInfo &cameraInfo
    )
00154 {
00155     #ifndef SANS_DETECTION
00156     listeResolutionsCamera.clear();
00157     if(QCameraInfo::availableCameras().count() > 0)
00158     {
00159         QCamera *camera = new QCamera(cameraInfo, this);
00160         QMediaRecorder *mediaRecorder = new QMediaRecorder(camera, this);
00161         camera->load();
00162         qDebug() << Q_FUNC_INFO << this << mediaRecorder->supportedResolutions().size();
00163         if(mediaRecorder->supportedResolutions().size() > 0)
00164         {
00165             foreach (const QSize &resolution, mediaRecorder->supportedResolutions())
00166             {
00167                 qDebug() << Q_FUNC_INFO << resolution.width() << "x" << resolution.height();
00168                 listeResolutionsCamera.push_back(resolution);
00169             }
00170         }
00171         delete mediaRecorder;
00172         delete camera;
00173     }
00174     #else
00175     Q_UNUSED(cameraInfo);
00176     listeResolutionsCamera.clear();
00177     QSize resolutionDefaut(LARGEUR_DEFAUT, HAUTEUR_DEFAUT);
00178     listeResolutionsCamera.push_back(resolutionDefaut);
00179     #endif
00180 }
00181
00182 QList<QSize> Camera::getListeResolutionsCamera()
00183 {
00184     return listeResolutionsCamera;
00185 }
00186
00187 void Camera::setResolution(int largeur, int hauteur)
00188 {
00189     qDebug() << Q_FUNC_INFO << "largeur" << largeur << "hauteur" << hauteur;
00190     QSize size(largeur, hauteur);
00191     int i = listeResolutionsCamera.indexOf(size);
00192     if (i != -1)
00193     {
00194         choixResolution = i;
00195         this->largeur = largeur;
00196         this->hauteur = hauteur;
```

```
00197         changementProprietes = true;
00198     }
00199     else
00200     {
00201         size = listeResolutionsCamera.last();
00202         choixResolution = listeResolutionsCamera.indexOf(size);;
00203         this->largeur = size.width();
00204         this->hauteur = size.height();
00205         changementProprietes = true;
00206     }
00207     qDebug() << Q_FUNC_INFO << "largeur" << this->largeur << "hauteur" << this->hauteur << "choixResolution
      " << choixResolution;
00208 }
00209
00210 void Camera::setResolution(QSize resolution)
00211 {
00212     qDebug() << Q_FUNC_INFO << "largeur" << resolution.width() << "hauteur" << resolution.height();
00213     int i = listeResolutionsCamera.indexOf(resolution);
00214     if (i != -1)
00215     {
00216         choixResolution = i;
00217         this->largeur = resolution.width();
00218         this->hauteur = resolution.height();
00219         changementProprietes = true;
00220     }
00221     else
00222     {
00223         QSize size = listeResolutionsCamera.last();
00224         choixResolution = listeResolutionsCamera.indexOf(size);;
00225         this->largeur = size.width();
00226         this->hauteur = size.height();
00227         changementProprietes = true;
00228     }
00229     qDebug() << Q_FUNC_INFO << "largeur" << this->largeur << "hauteur" << this->
      hauteur << "choixResolution" << choixResolution;
00230 }
00231
00232 void Camera::setResolution(int choix)
00233 {
00234     qDebug() << Q_FUNC_INFO << "choix" << choix;
00235     if(choix < listeResolutionsCamera.size())
00236     {
00237         choixResolution = choix;
00238         this->largeur = listeResolutionsCamera.at(choix).width();
00239         this->hauteur = listeResolutionsCamera.at(choix).height();
00240         changementProprietes = true;
00241     }
00242     else
00243     {
00244         QSize size = listeResolutionsCamera.last();
00245         choixResolution = listeResolutionsCamera.indexOf(size);;
00246         this->largeur = size.width();
00247         this->hauteur = size.height();
00248         changementProprietes = true;
00249     }
00250     qDebug() << Q_FUNC_INFO << "largeur" << this->largeur << "hauteur" << this->
      hauteur << "choixResolution" << choixResolution;
00251 }
00252
00253 void Camera::setLuminosite(int luminosite)
00254 {
00255     this->luminosite = double(luminosite)/100;
00256     changementProprietes = true;
00257 }
00258
00259 void Camera::setContraste(int contraste)
00260 {
00261     this->contraste = double(contraste)/100;
00262     changementProprietes = true;
00263 }
00264
00265 void Camera::setSaturation(int saturation)
00266 {
00267     this->saturation = double(saturation)/100;
00268     changementProprietes = true;
00269 }
00270
00271 int Camera::getNbCameras()
00272 {
00273     qDebug() << Q_FUNC_INFO  << "Caméra(s) disponible(s)" << QCameraInfo::availableCameras().count();
00274     return QCameraInfo::availableCameras().count();
00275 }
00276
00277 int Camera::extraireNumero(QString nomCamera)
00278 {
00279     int numero = -1;
00280     QString video = "/dev/video";
00281
00282     if(nomCamera.contains(video))
00283     {
00284         QString n = nomCamera.mid(video.length(), nomCamera.length());
```

```
00285         bool ok;
00286         qDebug() << Q_FUNC_INFO << "nom" << nomCamera << "n" << nomCamera.right(nomCamera.indexOf("
      /dev/video")) << "index" << nomCamera.indexOf("/dev/video");
00287         numero = n.toInt(&ok);
00288         if(ok)
00289             return numero;
00290     }
00291     return numero;
00292 }
00293
00294 QString Camera::creerNomCamera(int numero)
00295 {
00296     QString nom;
00297
00298     if(numero >= 0)
00299     {
00300         nom = QString("/dev/video") + QString::number(numero);
00301     }
00302     return nom;
00303 }
00304
00305 QList<QSize> Camera::lireListeResolutionsCamera(QCameraInfo &cameraInfo)
00306 {
00307     QList<QSize> listeResolutions;
00308     listeResolutions.clear();
00309     if(QCameraInfo::availableCameras().count() > 0)
00310     {
00311         QCamera *camera = new QCamera(cameraInfo);
00312         QMediaRecorder *mediaRecorder = new QMediaRecorder(camera);
00313         camera->load();
00314         qDebug() << Q_FUNC_INFO << mediaRecorder->supportedResolutions().size();
00315         if(mediaRecorder->supportedResolutions().size() > 0)
00316         {
00317             foreach (const QSize &resolution, mediaRecorder->supportedResolutions())
00318             {
00319                 qDebug() << Q_FUNC_INFO << resolution.width() << "x" << resolution.height();
00320                 listeResolutions.push_back(resolution);
00321             }
00322         }
00323         delete mediaRecorder;
00324         delete camera;
00325     }
00326     return listeResolutions;
00327 }
```

7.7 Référence du fichier camera.h

Fichier qui contient la déclaration de la classe Camera.

```
#include <QDebug>
#include <QObject>
#include <QThread>
#include <opencv.hpp>
#include <video/video.hpp>
#include <QPixmap>
#include <QImage>
#include <QTime>
#include <QCameraInfo>
#include "rov.h"
```

Classes

— class Camera
Class permettant de mettre en place une communication avec la camera.

Macros

— #define CAMERA_DEFAUT 0
Défini le numéro de la caméra par défaut.
— #define HAUTEUR_DEFAUT 1200
Défini la hauteur en pixel de la caméra par défaut.
— #define LARGEUR_DEFAUT 1600
Défini la largeur en pixel de la caméra par défaut.
— #define SEUIL_DEFAUT 0.5
Défini le seuil du contraste, luminosite et saturation de la caméra par défaut.

7.7.1 Description détaillée

Fichier qui contient la déclaration de la classe Camera.

Définition dans le fichier camera.h.

7.7.2 Documentation des macros

7.7.2.1 CAMERA_DEFAUT

```
#define CAMERA_DEFAUT 0
```

Défini le numéro de la caméra par défaut.

Définition à la ligne 29 du fichier camera.h.

Référencé par Rov : :demarrerCampagne().

7.7.2.2 HAUTEUR_DEFAUT

```
#define HAUTEUR_DEFAUT 1200
```

Défini la hauteur en pixel de la caméra par défaut.

Définition à la ligne 41 du fichier camera.h.

Référencé par Camera : :getResolution(), et Camera : :recupererListeResolutionsCamera().

7.7.2.3 LARGEUR_DEFAUT

```
#define LARGEUR_DEFAUT 1600
```

Défini la largeur en pixel de la caméra par défaut.

Définition à la ligne 35 du fichier camera.h.

Référencé par Camera : :getResolution(), et Camera : :recupererListeResolutionsCamera().

7.7.2.4 SEUIL_DEFAUT

```
#define SEUIL_DEFAUT 0.5
```

Défini le seuil du contraste, luminosite et saturation de la caméra par défaut.

Définition à la ligne 47 du fichier camera.h.

7.8 camera.h

```
00001
00007 #ifndef CAMERA_H
00008 #define CAMERA_H
00009
00010 #include <QDebug>
00011 #include <QObject>
00012 #include <QThread>
00013 #include <opencv.hpp>
00014 #include <video/video.hpp>
00015 #include <QPixmap>
00016 #include <QImage>
00017 #include <QTime>
00018 #include <QCameraInfo>
00019
00020 #include "rov.h"
00021
00022 //#define SANS_DETECTION
00023
00029 #define CAMERA_DEFAUT       0
00030
00035 #define LARGEUR_DEFAUT      1600 //2048
00036
00041 #define HAUTEUR_DEFAUT      1200 //1024
00042
00047 #define SEUIL_DEFAUT        0.5
00048
00049 using namespace cv;
00050
00051 class Rov;
00052
00058 class Camera : public QThread
00059 {
00060     Q_OBJECT
00061 private:
00062     Rov *rov;
00063     QString nomCamera;
00064     int numero;
00065     int largeur;
00066     int hauteur;
00067     double luminosite;
00068     double contraste;
00069     double saturation;
00070     bool changementProprietes;
00071     QList<QSize> listeResolutionsCamera;
00072     int choixResolution;
00073
00080     void acquerirImageVideo(cv::VideoCapture &camera, cv::Mat &frame);
00081
00082 public:
00090     Camera(Rov *rov, int numero, int choixResolution=-1);
00098     Camera(Rov *rov, QString nomCamera, int choixResolution=-1);
00099
00104     ~Camera();
00105
00110     void run();
00111
00117     QString getNom() const;
00123     void setProprietes(cv::VideoCapture &camera);
00129     QSize getResolution();
00135     int getChoixResolution();
00140     void recupererListeResolutionsCamera();
00146     void recupererListeResolutionsCamera(int numero);
00152     void recupererListeResolutionsCamera(QString nomCamera);
00158     void recupererListeResolutionsCamera(QCameraInfo &cameraInfo);
00164     QList<QSize> getListeResolutionsCamera();
00170     static int getNbCameras();
00177     static int extraireNumero(QString nomCamera);
00184     static QString creerNomCamera(int numero);
00191     static QList<QSize> lireListeResolutionsCamera(QCameraInfo &cameraInfo);
00192
00193 signals:
00199     void nouvelleImage(QPixmap image);
00204     void finVideo();
00205
00206 public slots:
00213     void setResolution(int largeur, int hauteur);
00219     void setResolution(QSize resolution);
00225     void setResolution(int choix);
00231     void setLuminosite(int luminosite);
00237     void setContraste(int contraste);
00243     void setSaturation(int saturation);
00244 };
00245
00246 #endif // CAMERA_H
```

7.9 Référence du fichier campagne.cpp

Fichier qui contient la définition de la classe Campagne.

```
#include "campagne.h"
```

7.9.1 Description détaillée

Fichier qui contient la définition de la classe Campagne.

Définition dans le fichier campagne.cpp.

7.10 campagne.cpp

```
00001
00007 #include "campagne.h"
00008
00009 Campagne::Campagne(QString nomCampagne, QString lieu, QString nomTechnicien, QString
      prenomTechnicien, QDateTime date, QObject *parent, int duree) : QObject (parent), nomCampagne(
      nomCampagne), nomTechnicien(nomTechnicien), prenomTechnicien(prenomTechnicien), lieu(lieu), date(date), duree(
      duree), estArchive(false), albumPhoto(), mesures(), nombrePhotos(0)
00010 {
00011     qDebug() << Q_FUNC_INFO;
00012 }
00013
00014 Campagne::~Campagne()
00015 {
00016     qDebug() << Q_FUNC_INFO;
00017 }
00018
00019 QString Campagne::getNomCampagne() const
00020 {
00021     return nomCampagne;
00022 }
00023
00024 QString Campagne::getNomTechnicien() const
00025 {
00026     return nomTechnicien;
00027 }
00028
00029 QString Campagne::getPrenomTechnicien() const
00030 {
00031     return prenomTechnicien;
00032 }
00033
00034 QString Campagne::getLieu() const
00035 {
00036     return lieu;
00037 }
00038
00039 QDateTime Campagne::getDate() const
00040 {
00041     return date;
00042 }
00043
00044 int Campagne::getDuree() const
00045 {
00046     return duree;
00047 }
00048
00049 void Campagne::setDuree(int duree)
00050 {
00051     this->duree = this->duree + duree;
00052 }
00053
00054
00055 QString Campagne::getCheminSauvegarde() const
00056 {
00057     return cheminSauvegardePhotos;
00058 }
00059
00060 void Campagne::setCheminSauvegarde(QString chemin)
00061 {
00062     cheminSauvegardePhotos = chemin;
00063 }
00064
```

```
00065 void Campagne::setNombrePhotos(int nombre)
00066 {
00067     nombrePhotos = nombre;
00068 }
00069
00070 QVector<Photo> Campagne::getAlbumPhoto() const
00071 {
00072     return albumPhoto;
00073 }
00074
00075 QVector<Mesure> Campagne::getMesures() const
00076 {
00077     return mesures;
00078 }
00079
00080 void Campagne::ajouterPhoto(Photo &photo)
00081 {
00082     albumPhoto.push_back(photo);
00083 }
00084
00085 void Campagne::modifierArchivePhoto(int numeroPhoto)
00086 {
00087     albumPhoto[numeroPhoto].aGarder = !(albumPhoto[numeroPhoto].aGarder);
00088 }
00089
00090 void Campagne::ajouterMesure(Mesure &mesure)
00091 {
00092     mesures.push_back(mesure);
00093 }
00094
00095 void Campagne::supprimerMesures()
00096 {
00097     mesures.clear();
00098 }
00099
00100 void Campagne::supprimerPhotos()
00101 {
00102     albumPhoto.clear();
00103 }
00104
00105 int Campagne::incrementeNombrePhoto()
00106 {
00107     nombrePhotos++;
00108     return nombrePhotos;
00109 }
```

7.11 Référence du fichier campagne.h

Fichier qui contient la déclaration de la classe Campagne.

```
#include <QObject>
#include <QTime>
#include <QVector>
#include "ihmalbumphoto.h"
```

Classes

- class Campagne
  Class contenant les informations de la campagne en cours.
- struct Mesure
  structure permettant de définir les propriété d'une mesure prise à une heure précise

7.11.1 Description détaillée

Fichier qui contient la déclaration de la classe Campagne.

Définition dans le fichier campagne.h.

7.12 campagne.h

```
00001
00007 #ifndef CAMPAGNE_H
00008 #define CAMPAGNE_H
00009
00010 #include <QObject>
00011 #include <QTime>
00012 #include <QVector>
00013
00014 #include "ihmalbumphoto.h"
00015
00021 struct Mesure
00022 {
00023     QDateTime dateheure;
00024     QString humidite;
00025     QString temperature;
00026     QString radiation;
00027 };
00028
00034 class Campagne : public QObject
00035 {
00036     Q_OBJECT
00037 private:
00038     QString nomCampagne;
00039     QString nomTechnicien;
00040     QString prenomTechnicien;
00041     QString lieu;
00042     QDateTime date;
00043     QString cheminSauvegardePhotos;
00044     int duree;
00045     bool estArchive;
00046     QVector<Photo> albumPhoto;
00047     QVector<Mesure> mesures;
00048     int nombrePhotos;
00049
00050 public:
00062     Campagne(QString nomCampagne, QString lieu, QString nomTechnicien, QString prenomTechnicien,
      QDateTime date, QObject *parent = nullptr, int duree = 0);
00067     ~Campagne();
00073     QString getNomCampagne() const;
00079     QString getNomTechnicien() const;
00085     QString getPrenomTechnicien() const;
00091     QString getLieu() const;
00097     QDateTime getDate() const;
00103     int getDuree() const;
00109     void setDuree(int duree);
00115     QString getCheminSauvegarde() const;
00121     void setCheminSauvegarde(QString chemin);
00127     void setNombrePhotos(int nombre);
00133     QVector<Photo> getAlbumPhoto() const;
00139     QVector<Mesure> getMesures() const;
00145     void ajouterPhoto(Photo &photo);
00151     void modifierArchivePhoto(int numeroPhoto);
00157     void ajouterMesure(Mesure &mesure);
00162     void supprimerMesures();
00167     void supprimerPhotos();
00173     int incrementeNombrePhoto();
00174
00175 signals:
00176
00177 public slots:
00178 };
00179
00180 #endif // CAMPAGNE_H
```

7.13 Référence du fichier capteurs.cpp

Fichier qui contient la définition de la classe Capteurs.

```
#include "capteurs.h"
```

7.13.1 Description détaillée

Fichier qui contient la définition de la classe Capteurs.

Définition dans le fichier capteurs.cpp.

7.14 capteurs.cpp

```
00001
00007 #include "capteurs.h"
00008
00009 Capteurs::Capteurs(QObject *parent) : QObject(parent), telemetrie("--,--"),
       angle(""), temperature("--,--"), humidite("--,--"), radiation("--,--")
00010 {
00011     qDebug() << Q_FUNC_INFO;
00012 }
00013
00014 Capteurs::~Capteurs()
00015 {
00016     qDebug() << Q_FUNC_INFO;
00017 }
00018
00019 void Capteurs::setTelemetrie(QString telemetrie)
00020 {
00021     this->telemetrie = telemetrie;
00022 }
00023
00024 void Capteurs::setTemperature(QString temperature)
00025 {
00026     this->temperature = temperature;
00027 }
00028
00029 void Capteurs::setHumidite(QString humidite)
00030 {
00031     this->humidite = humidite;
00032 }
00033
00034 void Capteurs::setRadiation(QString radiation)
00035 {
00036     this->radiation = radiation;
00037 }
00038
00039 void Capteurs::setAngle(QString angle)
00040 {
00041     this->angle = angle;
00042 }
00043
00044 QString Capteurs::getAngle() const
00045 {
00046     return angle;
00047 }
00048
00049 QString Capteurs::getTelemetrie() const
00050 {
00051     return telemetrie;
00052 }
00053
00054 QString Capteurs::getTemperature() const
00055 {
00056     return temperature;
00057 }
00058
00059 QString Capteurs::getHumidite() const
00060 {
00061     return humidite;
00062 }
00063
00064 QString Capteurs::getRadiation() const
00065 {
00066     return radiation;
00067 }
00068
```

7.15 Référence du fichier capteurs.h

Fichier qui contient la déclaration de la classe Capteurs.

```
#include <QObject>
#include <QDebug>
```

Classes

— class Capteurs
Classe contenant les dernières informations issues des capteurs du rov.

7.15.1 Description détaillée

Fichier qui contient la déclaration de la classe Capteurs.

Définition dans le fichier capteurs.h.

7.16 capteurs.h

```
00001
00007 #ifndef CAPTEURS_H
00008 #define CAPTEURS_H
00009
00010 #include <QObject>
00011 #include <QDebug>
00012
00018 class Capteurs : public QObject
00019 {
00020     Q_OBJECT
00021 private:
00022     QString telemetrie;
00023     QString angle;
00024     QString temperature;
00025     QString humidite;
00026     QString radiation;
00027
00028 public:
00034     Capteurs(QObject *parent = nullptr);
00039     ~Capteurs();
00045     void setTelemetrie(QString telemetrie);
00051     void setTemperature(QString temperature);
00057     void setHumidite(QString humidite);
00063     void setRadiation(QString radiation);
00069     void setAngle(QString angle);
00075     QString getTelemetrie() const;
00081     QString getTemperature() const;
00087     QString getHumidite() const;
00093     QString getRadiation() const;
00099     QString getAngle() const;
00100
00101 signals:
00102
00103 public slots:
00104 };
00105
00106 #endif // CAPTEURS_H
```

7.17 Référence du fichier Changelog.md

7.18 Changelog.md

```
00001 \page page_changelog Changelog
00002
00003
00004 r192 | abonnet | 2020-04-02 17:04:45 +0200 (jeu. 02 avril 2020) | 1 ligne
00005
00006 Réajustement documentation
00007
00008 r191 | abonnet | 2020-04-02 16:56:03 +0200 (jeu. 02 avril 2020) | 1 ligne
00009
00010 Réajustement documentation
00011
00012 r190 | abonnet | 2020-04-02 16:44:39 +0200 (jeu. 02 avril 2020) | 1 ligne
00013
00014 Suppression éléments non interessant pour le client dans le tag 0.1
00015
00016 r189 | abonnet | 2020-04-02 16:41:05 +0200 (jeu. 02 avril 2020) | 1 ligne
00017
00018 Création du tag 0.1
00019
00020 r188 | stenaille | 2020-04-02 16:30:44 +0200 (jeu. 02 avril 2020) | 2 lignes
00021
00022 mise a jour documentation
00023
00024 r187 | stenaille | 2020-04-02 16:28:51 +0200 (jeu. 02 avril 2020) | 2 lignes
00025
00026 mise a jour documentation
00027
```

```
00028 r186 | stenaille | 2020-04-02 16:24:35 +0200 (jeu. 02 avril 2020) | 2 lignes
00029
00030 Correction documentation
00031
00032 r185 | stenaille | 2020-04-02 16:16:57 +0200 (jeu. 02 avril 2020) | 2 lignes
00033
00034 correction documentation
00035
00036 r184 | stenaille | 2020-04-02 16:15:42 +0200 (jeu. 02 avril 2020) | 2 lignes
00037
00038 Mise a jour documentation
00039
00040 r183 | abonnet | 2020-04-02 16:11:18 +0200 (jeu. 02 avril 2020) | 1 ligne
00041
00042 Mis a jour documentation
00043
00044 r182 | stenaille | 2020-04-02 16:10:15 +0200 (jeu. 02 avril 2020) | 2 lignes
00045
00046 mise a jour des return
00047
00048 r181 | abonnet | 2020-04-02 16:08:04 +0200 (jeu. 02 avril 2020) | 1 ligne
00049
00050 Reéinitialisation de la base de données
00051
00052 r180 | stenaille | 2020-04-02 16:03:01 +0200 (jeu. 02 avril 2020) | 2 lignes
00053
00054 Suppression fn du main.cpp
00055
00056 r179 | abonnet | 2020-04-02 16:01:11 +0200 (jeu. 02 avril 2020) | 1 ligne
00057
00058 Suppression BDD obsolete
00059
00060 r178 | stenaille | 2020-04-02 15:59:19 +0200 (jeu. 02 avril 2020) | 2 lignes
00061
00062 Suppression fn de manette.h
00063
00064 r177 | abonnet | 2020-04-02 15:59:02 +0200 (jeu. 02 avril 2020) | 1 ligne
00065
00066 Suppréssion fichier sql obsolete
00067
00068 r176 | stenaille | 2020-04-02 15:55:25 +0200 (jeu. 02 avril 2020) | 2 lignes
00069
00070 Rectification documentation
00071
00072 r175 | abonnet | 2020-04-02 15:52:46 +0200 (jeu. 02 avril 2020) | 1 ligne
00073
00074 Mis a jour documentation
00075
00076 r174 | stenaille | 2020-04-02 15:47:03 +0200 (jeu. 02 avril 2020) | 2 lignes
00077
00078 retrait des fn dans rov.h
00079
00080 r173 | abonnet | 2020-04-02 15:45:09 +0200 (jeu. 02 avril 2020) | 1 ligne
00081
00082 Mis a jour documentation
00083
00084 r172 | stenaille | 2020-04-01 15:57:03 +0200 (mer. 01 avril 2020) | 2 lignes
00085
00086 Création slot enregisterMesureBDD() et signal enregistrerMesures()
00087
00088 r171 | abonnet | 2020-03-31 18:59:11 +0200 (mar. 31 mars 2020) | 1 ligne
00089
00090 Modification bug chargement des photos -> conteneur informationsPhotos de la méthode chargerCampagne()
       doit être vidé pour chaque campagne
00091
00092 r170 | tvaira | 2020-03-31 18:51:35 +0200 (mar. 31 mars 2020) | 2 lignes
00093
00094 Déplacement du define SANS_DETECTION dans camera.h
00095
00096 r169 | abonnet | 2020-03-31 18:22:23 +0200 (mar. 31 mars 2020) | 1 ligne
00097
00098 Modification structure Photo -> cette dernière ne comprend plus les données issues des capteurs
00099
00100 r168 | abonnet | 2020-03-31 18:19:11 +0200 (mar. 31 mars 2020) | 1 ligne
00101
00102 Ajout define SANS_DETECTION
00103
00104 r167 | abonnet | 2020-03-31 16:32:35 +0200 (mar. 31 mars 2020) | 1 ligne
00105
00106 Modification de l'archivage des photos -> ces dernières sont enregistré dans la BDD directement à la
       prise
00107
00108 r166 | tvaira | 2020-03-30 18:24:09 +0200 (lun. 30 mars 2020) | 2 lignes
00109
00110 Révision de code (voir mail)
00111
00112 r165 | stenaille | 2020-03-29 20:15:03 +0200 (dim. 29 mars 2020) | 2 lignes
00113
00114 Ajout suppression image en local et suppression dossier en local
00115
00116 r164 | abonnet | 2020-03-29 18:55:07 +0200 (dim. 29 mars 2020) | 1 ligne
```

```
00117
00118 Implementation méthode permettant de modifie l'etat des bouton de l'ihmReglageVideo
00119
00120 r163 | tvaira | 2020-03-29 18:53:18 +0200 (dim. 29 mars 2020) | 2 lignes
00121
00122 Démarrer/Arrêter Vidéo
00123
00124 r162 | abonnet | 2020-03-29 18:18:38 +0200 (dim. 29 mars 2020) | 1 ligne
00125
00126 Modification IHMReglageVideo -> choix de la camera disponible
00127
00128 r161 | abonnet | 2020-03-29 15:16:10 +0200 (dim. 29 mars 2020) | 1 ligne
00129
00130 Création d'un dossier du nom de la campagne à la création de cette dernière + enregistrement des
       photos prise dans ce dossier
00131
00132 r160 | abonnet | 2020-03-29 14:30:09 +0200 (dim. 29 mars 2020) | 1 ligne
00133
00134 mis en ordre du code
00135
00136 r159 | stenaille | 2020-03-28 18:42:48 +0100 (sam. 28 mars 2020) | 2 lignes
00137
00138 Ajout suppression des photos non voulu lors de l'archivage
00139
00140 r158 | abonnet | 2020-03-28 18:24:45 +0100 (sam. 28 mars 2020) | 1 ligne
00141
00142 Les photos s'enregistresous format PNG dans le dossier choisi
00143
00144 r157 | abonnet | 2020-03-28 18:01:12 +0100 (sam. 28 mars 2020) | 1 ligne
00145
00146 Modification méthode chargerCampagne() -> prend en compte le nombre de photos presente dans la base de
       données
00147
00148 r156 | abonnet | 2020-03-28 17:01:35 +0100 (sam. 28 mars 2020) | 1 ligne
00149
00150 Modification bug validerCampagne
00151
00152 r155 | abonnet | 2020-03-28 16:27:08 +0100 (sam. 28 mars 2020) | 1 ligne
00153
00154 Modification structure BDD ajout champs cheminSauvegarde à la table campagne
00155
00156 r154 | abonnet | 2020-03-28 16:02:22 +0100 (sam. 28 mars 2020) | 1 ligne
00157
00158 Implementation méthode updateCampagneBDD
00159
00160 r153 | stenaille | 2020-03-28 13:58:57 +0100 (sam. 28 mars 2020) | 2 lignes
00161
00162 Correction bug suppression photo
00163
00164 r152 | stenaille | 2020-03-27 18:39:27 +0100 (ven. 27 mars 2020) | 2 lignes
00165
00166 Modification methode supprimerCampagneBDD
00167
00168 r151 | stenaille | 2020-03-27 16:52:34 +0100 (ven. 27 mars 2020) | 2 lignes
00169
00170 Modification méthode archiverCampagne et supprimerCampagneBDD
00171
00172 r150 | stenaille | 2020-03-27 16:39:43 +0100 (ven. 27 mars 2020) | 2 lignes
00173
00174 Ajout méthode Archiver et suppressionBDD
00175
00176 r149 | abonnet | 2020-03-27 12:57:34 +0100 (ven. 27 mars 2020) | 1 ligne
00177
00178 Ajout liste des techniciens connus
00179
00180 r148 | abonnet | 2020-03-26 18:01:35 +0100 (jeu. 26 mars 2020) | 1 ligne
00181
00182 Modification méthode enregistrerCampagneBDD ->mise a jour base de données campagnes.sqlite
00183
00184 r147 | abonnet | 2020-03-26 17:40:23 +0100 (jeu. 26 mars 2020) | 1 ligne
00185
00186 Modification méthode enregistrerCampagneBDD ->mise a jour base de données campagnes.sqlite
00187
00188 r146 | abonnet | 2020-03-26 16:33:25 +0100 (jeu. 26 mars 2020) | 1 ligne
00189
00190 Modification methode chargerCampagnes() -> mise a jour base de données campagnes.sqlite
00191
00192 r145 | stenaille | 2020-03-26 16:21:16 +0100 (jeu. 26 mars 2020) | 1 ligne
00193
00194 Ajout diagrammeDeClassePersonnel
00195
00196 r144 | stenaille | 2020-03-26 15:23:46 +0100 (jeu. 26 mars 2020) | 1 ligne
00197
00198 Ajout diagramme de classe : diagrammeClasseDeplacerLeRobot, diagrammeClassePiloterLeBras,
       diagrammeClasseRecevoirMesures, diagrammeClasseVisualiserMesures
00199
00200 r143 | stenaille | 2020-03-26 14:07:03 +0100 (jeu. 26 mars 2020) | 2 lignes
00201
00202 Modification diagramme cas d'utilisation
00203
00204 r142 | tvaira | 2020-03-26 13:34:44 +0100 (jeu. 26 mars 2020) | 1 ligne
```

```
00205
00206 Modelisation BD
00207
00208 r141 | tvaira | 2020-03-25 17:25:20 +0100 (mer. 25 mars 2020) | 1 ligne
00209
00210 Agencement cas utilisation tenaille
00211
00212 r140 | stenaille | 2020-03-25 17:16:37 +0100 (mer. 25 mars 2020) | 1 ligne
00213
00214 Ajout diagramme recevoirMesure, visualiserMesures, casUtilisationPersonnel
00215
00216 r139 | abonnet | 2020-03-25 14:43:19 +0100 (mer. 25 mars 2020) | 1 ligne
00217
00218 ajout diagramme de cas d'utilisation
00219
00220 r138 | abonnet | 2020-03-25 13:54:22 +0100 (mer. 25 mars 2020) | 1 ligne
00221
00222 Modification class IHMCreationCampagne -> héritage QDialog
00223
00224 r137 | abonnet | 2020-03-25 12:27:45 +0100 (mer. 25 mars 2020) | 1 ligne
00225
00226 Rectification diagramme cas d'utilisation
00227
00228 r136 | tvaira | 2020-03-24 16:14:45 +0100 (mar. 24 mars 2020) | 1 ligne
00229
00230 Maj diagramme de séquence demarrerUneCampagne
00231
00232 r135 | tvaira | 2020-03-24 16:13:59 +0100 (mar. 24 mars 2020) | 2 lignes
00233
00234 Modification classe IHMAccueil en mode modale (type QDialog)
00235
00236 r134 | abonnet | 2020-03-24 15:05:30 +0100 (mar. 24 mars 2020) | 1 ligne
00237
00238 Ajout diagramme sequence et deploiement
00239
00240 r133 | abonnet | 2020-03-24 15:03:14 +0100 (mar. 24 mars 2020) | 1 ligne
00241
00242 modification diagramme deploiement
00243
00244 r132 | abonnet | 2020-03-24 13:26:51 +0100 (mar. 24 mars 2020) | 1 ligne
00245
00246 Création diagramme de deploiement
00247
00248 r131 | abonnet | 2020-03-24 12:47:45 +0100 (mar. 24 mars 2020) | 1 ligne
00249
00250 Création diagramme sequence piloterLaCamera et demarreruneCampagne
00251
00252 r130 | tvaira | 2020-03-21 16:54:23 +0100 (sam. 21 mars 2020) | 1 ligne
00253
00254 Validation BD SQLite v1.1
00255
00256 r129 | abonnet | 2020-03-21 14:39:48 +0100 (sam. 21 mars 2020) | 1 ligne
00257
00258 Modification BD ajout campagne-v1.1.sql
00259
00260 r128 | stenaille | 2020-03-20 17:46:19 +0100 (ven. 20 mars 2020) | 1 ligne
00261
00262 ajout .png diagramme piloterLeBras
00263
00264 r127 | stenaille | 2020-03-20 17:45:11 +0100 (ven. 20 mars 2020) | 1 ligne
00265
00266 Ajout diagramme de sequence piloter le bras
00267
00268 r126 | abonnet | 2020-03-20 15:41:11 +0100 (ven. 20 mars 2020) | 1 ligne
00269
00270 Modification BD
00271
00272 r125 | abonnet | 2020-03-20 15:25:48 +0100 (ven. 20 mars 2020) | 1 ligne
00273
00274 Modification BD + ajout lieu class Campagne
00275
00276 r124 | abonnet | 2020-03-20 15:18:55 +0100 (ven. 20 mars 2020) | 1 ligne
00277
00278 Modification BD + ajout lieu class Campagne
00279
00280 r123 | stenaille | 2020-03-20 11:13:08 +0100 (ven. 20 mars 2020) | 1 ligne
00281
00282 Ajout element diagramme de sequence
00283
00284 r122 | tvaira | 2020-03-20 07:34:40 +0100 (ven. 20 mars 2020) | 1 ligne
00285
00286 Validation du fichier SQL
00287
00288 r121 | stenaille | 2020-03-19 19:36:12 +0100 (jeu. 19 mars 2020) | 2 lignes
00289
00290 Ajout du de la fonction archiverCampagne
00291
00292 r120 | tvaira | 2020-03-19 18:40:29 +0100 (jeu. 19 mars 2020) | 1 ligne
00293
00294 Validation sd deplacerLeRobot
00295
```

```
00296 r119 | stenaille | 2020-03-19 18:26:04 +0100 (jeu. 19 mars 2020) | 2 lignes
00297
00298 Mise a jour structure Mesure
00299
00300 r118 | stenaille | 2020-03-19 18:23:25 +0100 (jeu. 19 mars 2020) | 2 lignes
00301
00302 Ajout structure Mesures
00303
00304 r117 | abonnet | 2020-03-19 18:20:23 +0100 (jeu. 19 mars 2020) | 1 ligne
00305
00306 Modification BDD
00307
00308 r116 | abonnet | 2020-03-19 18:20:05 +0100 (jeu. 19 mars 2020) | 1 ligne
00309
00310 Modification BDD
00311
00312 r115 | abonnet | 2020-03-19 18:19:37 +0100 (jeu. 19 mars 2020) | 1 ligne
00313
00314 Modification BDD
00315
00316 r114 | stenaille | 2020-03-19 18:19:09 +0100 (jeu. 19 mars 2020) | 1 ligne
00317
00318 Ajout campagnesArchives
00319
00320 r113 | abonnet | 2020-03-19 18:09:17 +0100 (jeu. 19 mars 2020) | 1 ligne
00321
00322 Ajout fichier campagnesEnCours-v1.sql
00323
00324 r112 | stenaille | 2020-03-19 18:00:25 +0100 (jeu. 19 mars 2020) | 1 ligne
00325
00326 Ajout de bouml-rovnet-tenaille
00327
00328 r111 | stenaille | 2020-03-19 17:59:02 +0100 (jeu. 19 mars 2020) | 1 ligne
00329
00330 Suppression boulm-rovnet-tenaille
00331
00332 r110 | stenaille | 2020-03-19 17:51:00 +0100 (jeu. 19 mars 2020) | 1 ligne
00333
00334 Ajout des parties manquantes
00335
00336 r109 | stenaille | 2020-03-19 17:45:19 +0100 (jeu. 19 mars 2020) | 1 ligne
00337
00338 r108 | abonnet | 2020-03-19 17:35:08 +0100 (jeu. 19 mars 2020) | 1 ligne
00339
00340 Modification BDD
00341
00342 r107 | abonnet | 2020-03-19 17:33:49 +0100 (jeu. 19 mars 2020) | 1 ligne
00343
00344 Modification BDD
00345
00346 r106 | abonnet | 2020-03-19 17:20:30 +0100 (jeu. 19 mars 2020) | 1 ligne
00347
00348 Modification BDD
00349
00350 r105 | stenaille | 2020-03-19 17:18:46 +0100 (jeu. 19 mars 2020) | 1 ligne
00351
00352 REcommit bouml
00353
00354 r104 | abonnet | 2020-03-19 17:14:01 +0100 (jeu. 19 mars 2020) | 1 ligne
00355
00356 modification BDD
00357
00358 r103 | abonnet | 2020-03-19 17:07:10 +0100 (jeu. 19 mars 2020) | 1 ligne
00359
00360 modification BDD
00361
00362 r102 | abonnet | 2020-03-19 17:04:58 +0100 (jeu. 19 mars 2020) | 1 ligne
00363
00364 Modification BDD
00365
00366 r101 | abonnet | 2020-03-19 17:04:21 +0100 (jeu. 19 mars 2020) | 1 ligne
00367
00368 Modification BDD
00369
00370 r100 | tvaira | 2020-03-19 17:03:48 +0100 (jeu. 19 mars 2020) | 1 ligne
00371
00372 Suppression de 2.lock
00373
00374 r99 | abonnet | 2020-03-19 16:54:55 +0100 (jeu. 19 mars 2020) | 1 ligne
00375
00376 Implémentation méthodes permettant d'ajouter une nouvelle campagne dans la BDD
00377
00378 r98 | abonnet | 2020-03-19 16:51:14 +0100 (jeu. 19 mars 2020) | 1 ligne
00379
00380 modification BDD
00381
00382 r97 | abonnet | 2020-03-19 16:38:14 +0100 (jeu. 19 mars 2020) | 1 ligne
00383
00384 Deplacement bdd
00385
00386 r96 | abonnet | 2020-03-19 16:38:02 +0100 (jeu. 19 mars 2020) | 1 ligne
```

```
00387
00388 Deplacement bdd
00389
00390 r95 | stenaille | 2020-03-19 16:29:48 +0100 (jeu. 19 mars 2020) | 1 ligne
00391
00392 modification diagramme séquence déplacer le robot format png
00393
00394 r94 | stenaille | 2020-03-19 16:26:18 +0100 (jeu. 19 mars 2020) | 1 ligne
00395
00396 diagrame séquence déplacer le robot format png
00397
00398 r93 | stenaille | 2020-03-19 16:23:28 +0100 (jeu. 19 mars 2020) | 1 ligne
00399
00400 diagrame séquence déplacer le robot
00401
00402 r92 | stenaille | 2020-03-19 16:21:17 +0100 (jeu. 19 mars 2020) | 2 lignes
00403
00404 reglage bug configuration camera
00405
00406 r91 | tvaira | 2020-03-19 16:19:05 +0100 (jeu. 19 mars 2020) | 2 lignes
00407
00408 Ajout du déploiement des fichiers SQLite de base
00409
00410 r90 | abonnet | 2020-03-19 15:54:19 +0100 (jeu. 19 mars 2020) | 1 ligne
00411
00412 Implémentation méthodes permettant de charger les campagnes présente dans la BDD
00413
00414 r89 | tvaira | 2020-03-19 08:29:10 +0100 (jeu. 19 mars 2020) | 4 lignes
00415
00416 Ajout des méthodes executer() et recuperer() dans la classe
00417 BaseDeDonnees
00418 Suppression des warnings
00419
00420 r88 | abonnet | 2020-03-18 18:01:11 +0100 (mer. 18 mars 2020) | 1 ligne
00421
00422 Implémentation classe BaseDeDonnees
00423
00424 r87 | stenaille | 2020-03-18 17:59:40 +0100 (mer. 18 mars 2020) | 1 ligne
00425
00426 Ajout boulm-rovnet-tenaille
00427
00428 r86 | abonnet | 2020-03-18 17:49:36 +0100 (mer. 18 mars 2020) | 1 ligne
00429
00430 Implémentation classe BaseDeDonnees
00431
00432 r85 | stenaille | 2020-03-18 17:32:44 +0100 (mer. 18 mars 2020) | 2 lignes
00433
00434 Ajout campagnesArchives
00435
00436 r84 | abonnet | 2020-03-18 17:28:05 +0100 (mer. 18 mars 2020) | 1 ligne
00437
00438 Changement nom classe BaseDeDonnee -> BaseDeDonnees
00439
00440 r83 | abonnet | 2020-03-18 17:18:20 +0100 (mer. 18 mars 2020) | 1 ligne
00441
00442 Modification campagneEnCours -> campagnesEnCours
00443
00444 r82 | abonnet | 2020-03-18 16:46:13 +0100 (mer. 18 mars 2020) | 1 ligne
00445
00446 Ajout fichier SQLite campagneEnCours
00447
00448 r81 | abonnet | 2020-03-18 15:52:18 +0100 (mer. 18 mars 2020) | 1 ligne
00449
00450 Realisation sd_prendreUnePhoto
00451
00452 r80 | stenaille | 2020-03-18 15:47:56 +0100 (mer. 18 mars 2020) | 2 lignes
00453
00454 Ajout de la classe BaseDeDonnee
00455
00456 r79 | stenaille | 2020-03-18 15:04:58 +0100 (mer. 18 mars 2020) | 2 lignes
00457
00458 Correction warning
00459
00460 r78 | abonnet | 2020-03-18 13:39:32 +0100 (mer. 18 mars 2020) | 1 ligne
00461
00462 Ajout diagrammeSequenceSysteme
00463
00464 r77 | abonnet | 2020-03-18 13:13:41 +0100 (mer. 18 mars 2020) | 1 ligne
00465
00466 Modification classe albumPhoto -> IHMAlbumPhoto
00467
00468 r76 | abonnet | 2020-03-17 17:26:48 +0100 (mar. 17 mars 2020) | 1 ligne
00469
00470 Implémentation méthode permettant de garder l'etat d'une photo lors de la reprise d'une campagne
00471
00472 r75 | abonnet | 2020-03-17 16:14:47 +0100 (mar. 17 mars 2020) | 1 ligne
00473
00474 Implémentation méthode permettant de garder l'etat d'une photo lors de la reprise d'une campagne
00475
00476 r74 | abonnet | 2020-03-17 14:21:25 +0100 (mar. 17 mars 2020) | 1 ligne
00477
```

```
00478 Implémentation méthode permettant de changer le status d'une photo dans l'album photo
00479
00480 r73 | tvaira | 2020-03-16 18:45:45 +0100 (lun. 16 mars 2020) | 2 lignes
00481
00482 Exemple QSignalMapper
00483
00484 r72 | abonnet | 2020-03-16 17:09:31 +0100 (lun. 16 mars 2020) | 1 ligne
00485
00486 Implémentation méthodes permettant d'avoir les informations d'une photo dans l'album photo
00487
00488 r71 | tvaira | 2020-03-16 16:32:12 +0100 (lun. 16 mars 2020) | 2 lignes
00489
00490 Correction erreur erase() campagneEnCours
00491
00492 r70 | abonnet | 2020-03-16 15:53:30 +0100 (lun. 16 mars 2020) | 1 ligne
00493
00494 Implémentation méthodes permettant de supprimer la campagne selectionne
00495
00496 r69 | abonnet | 2020-03-16 14:56:53 +0100 (lun. 16 mars 2020) | 1 ligne
00497
00498 Implementation méthodes permettant de reprendre une campagne arreté avec la bonne durée de campagne
00499
00500 r68 | abonnet | 2020-03-16 14:00:20 +0100 (lun. 16 mars 2020) | 1 ligne
00501
00502 Ajout d'une structure photo + modification de la classe albumPhoto en IHMAlbumPhoto
00503
00504 r67 | abonnet | 2020-03-15 17:22:51 +0100 (dim. 15 mars 2020) | 1 ligne
00505
00506 Implementation méthodes permettant de creer une nouvelle campagne
00507
00508 r66 | abonnet | 2020-03-14 13:50:57 +0100 (sam. 14 mars 2020) | 1 ligne
00509
00510 Modification sd:visualiser environnement
00511
00512 r65 | abonnet | 2020-03-14 13:50:44 +0100 (sam. 14 mars 2020) | 1 ligne
00513
00514 Modification sd:visualiser environnement
00515
00516 r64 | tvaira | 2020-03-14 07:52:53 +0100 (sam. 14 mars 2020) | 1 ligne
00517
00518 Correction sd visualierEnvironnement
00519
00520 r63 | tvaira | 2020-03-14 07:40:33 +0100 (sam. 14 mars 2020) | 1 ligne
00521
00522 Ne pas commiter le dossier lock
00523
00524 r62 | stenaille | 2020-03-13 20:09:44 +0100 (ven. 13 mars 2020) | 2 lignes
00525
00526 debug chemin image
00527
00528 r61 | abonnet | 2020-03-13 19:35:46 +0100 (ven. 13 mars 2020) | 1 ligne
00529
00530 Ajout dossier bouml au projet rovnet
00531
00532 r60 | abonnet | 2020-03-13 18:10:23 +0100 (ven. 13 mars 2020) | 1 ligne
00533
00534 Implementation classe IHMAccueil
00535
00536 r59 | abonnet | 2020-03-12 15:30:35 +0100 (jeu. 12 mars 2020) | 3 lignes
00537
00538 Implémentation méthodes permettant d'envoyer les trames correspondantes
00539 au pilotage de la caméra
00540
00541 r58 | stenaille | 2020-03-12 12:56:52 +0100 (jeu. 12 mars 2020) | 2 lignes
00542
00543 Ajout Humidité
00544
00545 r57 | stenaille | 2020-03-12 12:25:16 +0100 (jeu. 12 mars 2020) | 2 lignes
00546
00547 Documentation
00548
00549 r56 | stenaille | 2020-03-12 11:53:54 +0100 (jeu. 12 mars 2020) | 2 lignes
00550
00551 correction bug trame
00552
00553 r55 | abonnet | 2020-03-12 11:25:48 +0100 (jeu. 12 mars 2020) | 2 lignes
00554
00555 Modification classe manette -> heritage QGamepad
00556
00557 r54 | stenaille | 2020-03-12 10:59:35 +0100 (jeu. 12 mars 2020) | 2 lignes
00558
00559 Ajout structure bouton et modification contruction trame ordre et pince
00560
00561 r53 | abonnet | 2020-03-12 10:53:36 +0100 (jeu. 12 mars 2020) | 2 lignes
00562
00563 Mise a jour connect manette
00564
00565 r52 | abonnet | 2020-03-12 10:26:15 +0100 (jeu. 12 mars 2020) | 2 lignes
00566
00567 Documentation
00568
```

```
00569 r51 | stenaille | 2020-03-11 13:57:06 +0100 (mer. 11 mars 2020) | 2 lignes
00570
00571 Mise à jour code
00572
00573 r50 | abonnet | 2020-03-11 13:37:31 +0100 (mer. 11 mars 2020) | 1 ligne
00574
00575 Ajustation de la résolution de la caméra
00576
00577 r49 | stenaille | 2020-03-11 13:36:13 +0100 (mer. 11 mars 2020) | 2 lignes
00578
00579 Ajout des trames pince
00580
00581 r48 | stenaille | 2020-03-11 11:00:21 +0100 (mer. 11 mars 2020) | 2 lignes
00582
00583 optimisation du code sur la partie test des etat de la manette
00584
00585 r47 | stenaille | 2020-03-10 22:20:38 +0100 (mar. 10 mars 2020) | 2 lignes
00586
00587 Modification du code (maniere de tester l'etat de la manette)
00588
00589 r46 | stenaille | 2020-03-10 21:40:47 +0100 (mar. 10 mars 2020) | 2 lignes
00590
00591 Organisation du code
00592
00593 r45 | stenaille | 2020-03-10 17:05:27 +0100 (mar. 10 mars 2020) | 2 lignes
00594
00595 Organisation code
00596
00597 r44 | tvaira | 2020-03-09 12:02:19 +0100 (lun. 09 mars 2020) | 1 ligne
00598
00599 Revisions de code
00600
00601 r43 | abonnet | 2020-03-08 16:04:13 +0100 (dim. 08 mars 2020) | 1 ligne
00602
00603 Modification de la relation rov-ihm en relation bidirectionnelle
00604
00605 r42 | stenaille | 2020-03-07 18:08:42 +0100 (sam. 07 mars 2020) | 2 lignes
00606
00607 correction code non compilable
00608
00609 r41 | abonnet | 2020-03-06 17:56:17 +0100 (ven. 06 mars 2020) | 1 ligne
00610
00611 Mise a jour documentation
00612
00613 r40 | stenaille | 2020-03-06 17:08:45 +0100 (ven. 06 mars 2020) | 2 lignes
00614
00615 Ajout des différentes structures EtatManetteDeplacement
00616
00617 r39 | stenaille | 2020-03-06 16:51:30 +0100 (ven. 06 mars 2020) | 2 lignes
00618
00619 Modification du code pour correction de bug lors du pilotage du bras
00620
00621 r38 | abonnet | 2020-03-06 15:12:38 +0100 (ven. 06 mars 2020) | 1 ligne
00622
00623 Création classe ReglageVideo et AlbumPhoto
00624
00625 r37 | tvaira | 2020-03-06 06:52:47 +0100 (ven. 06 mars 2020) | 2 lignes
00626
00627 Test 18.04 + define NO_GAMEPAD pour une utilisation sans manette
00628
00629 r36 | stenaille | 2020-03-05 17:18:36 +0100 (jeu. 05 mars 2020) | 2 lignes
00630
00631 modification de la methode creationTramePilotage
00632
00633 r35 | abonnet | 2020-03-05 16:16:42 +0100 (jeu. 05 mars 2020) | 1 ligne
00634
00635 Implementation des methodes permettant de capturer plusieurs photos en un campagne et de pouvoir les
      visualiser clairement
00636
00637 r34 | abonnet | 2020-03-05 14:14:59 +0100 (jeu. 05 mars 2020) | 1 ligne
00638
00639 Implementation methodes permettant de capturer une image et de l'afficher dans l'album photo
00640
00641 r33 | stenaille | 2020-03-05 13:49:38 +0100 (jeu. 05 mars 2020) | 2 lignes
00642
00643 Ajout de la création des trames de pilotage
00644
00645 r32 | stenaille | 2020-03-04 17:05:47 +0100 (mer. 04 mars 2020) | 2 lignes
00646
00647 Création des connexions et des slots pour les bouton A, B, X, Y
00648
00649 r31 | abonnet | 2020-03-04 16:46:45 +0100 (mer. 04 mars 2020) | 1 ligne
00650
00651 Implementation methode permettant d'avoir les reglage de la vidéo dans une nouvelle fenetre
00652
00653 r30 | stenaille | 2020-03-04 16:05:29 +0100 (mer. 04 mars 2020) | 2 lignes
00654
00655 rectification du code
00656
00657 r29 | stenaille | 2020-03-04 15:38:44 +0100 (mer. 04 mars 2020) | 2 lignes
00658
```

```
00659 oublie
00660
00661 r28 | stenaille | 2020-03-04 15:24:49 +0100 (mer. 04 mars 2020) | 3 lignes
00662
00663 Ajout de l'affichage des valeurs températures et radiations de la trame
00664 capteur
00665
00666 r27 | tvaira | 2020-03-04 14:01:11 +0100 (mer. 04 mars 2020) | 2 lignes
00667
00668 Support pour OpenCV 2 (Ubuntu 16.04) ou 3 (Ubuntu 18.04)
00669
00670 r26 | abonnet | 2020-03-04 11:14:21 +0100 (mer. 04 mars 2020) | 2 lignes
00671
00672 Suppression label données telemetrie
00673
00674 r25 | abonnet | 2020-03-04 10:01:04 +0100 (mer. 04 mars 2020) | 1 ligne
00675
00676 Incrustation telemetrie flux video
00677
00678 r24 | abonnet | 2020-02-21 15:01:22 +0100 (ven. 21 févr. 2020) | 1 ligne
00679
00680 Implémentation méthodes permettant une incrustation de l'heure et des données télémétriques sur le
       flux vidéo
00681
00682 r23 | stenaille | 2020-02-20 20:21:15 +0100 (jeu. 20 févr. 2020) | 3 lignes
00683
00684 Mise à jour du code (retrait fonction inutile et ajout du code d'envoye
00685 des trames déplacement)
00686
00687 r22 | stenaille | 2020-02-19 18:19:29 +0100 (mer. 19 févr. 2020) | 2 lignes
00688
00689 Modification bug
00690
00691 r21 | stenaille | 2020-02-19 18:00:22 +0100 (mer. 19 févr. 2020) | 2 lignes
00692
00693 Modification de la connexion creerTrameDeplacement
00694
00695 r20 | abonnet | 2020-02-19 16:48:47 +0100 (mer. 19 févr. 2020) | 1 ligne
00696
00697 Ajout methode getCamera
00698
00699 r19 | stenaille | 2020-02-19 16:03:10 +0100 (mer. 19 févr. 2020) | 2 lignes
00700
00701 création de la connexion creerTrameDeplacement
00702
00703 r18 | abonnet | 2020-02-18 13:11:01 +0100 (mar. 18 févr. 2020) | 1 ligne
00704
00705 Implémentation méthodes permettant de modifier les parametres du flux video
00706
00707 r17 | stenaille | 2020-02-17 13:53:38 +0100 (lun. 17 févr. 2020) | 2 lignes
00708
00709 suppression du todo des structures
00710
00711 r16 | stenaille | 2020-02-17 13:52:40 +0100 (lun. 17 févr. 2020) | 3 lignes
00712
00713 Créations des structures de la manette et ajout des connexion entre la
00714 mannette et l'application Rov
00715
00716 r15 | tvaira | 2020-02-15 14:24:56 +0100 (sam. 15 févr. 2020) | 2 lignes
00717
00718 Insertion d'un TODO pour une structure Manette
00719
00720 r14 | stenaille | 2020-02-14 17:12:57 +0100 (ven. 14 févr. 2020) | 2 lignes
00721
00722 Todo connexion entre creationTrameDeplacement et creerTrameDeplacement
00723
00724 r13 | stenaille | 2020-02-14 17:03:42 +0100 (ven. 14 févr. 2020) | 5 lignes
00725
00726 création du signal creationTrameDeplacement(char deplacementAxeX, int
00727 puissance, char deplacementAxeY) dans la classe manette et la méthode
00728 slot creerTrameDeplacement(char deplacementAxeX, int puissance, char
00729 deplacementAxeY) dans la classe rov
00730
00731 r12 | stenaille | 2020-02-14 16:48:16 +0100 (ven. 14 févr. 2020) | 2 lignes
00732
00733 Ajout documentation classe manette
00734
00735 r11 | stenaille | 2020-02-14 16:33:36 +0100 (ven. 14 févr. 2020) | 2 lignes
00736
00737 Amélioration du code de la fonction creerTrameDeplacement()
00738
00739 r10 | abonnet | 2020-02-14 15:58:20 +0100 (ven. 14 févr. 2020) | 3 lignes
00740
00741 Implémentation des méthodes permettant de décoder la trame de télémtrie
00742 et afficher les données correspondantes
00743
00744 r9 | stenaille | 2020-02-14 15:50:55 +0100 (ven. 14 févr. 2020) | 2 lignes
00745
00746 Documentation
00747
00748 r8 | stenaille | 2020-02-14 14:30:28 +0100 (ven. 14 févr. 2020) | 2 lignes
```

```
00749
00750 ajout class manette
00751
00752 r7 | stenaille | 2020-02-14 14:27:25 +0100 (ven. 14 févr. 2020) | 2 lignes
00753
00754 Ajout de la methode detecterManette()
00755
00756 r6 | abonnet | 2020-02-14 13:50:04 +0100 (ven. 14 févr. 2020) | 2 lignes
00757
00758 Implémentation des méthodes permmettant d'afficher le flux vidéo sur l'IHM
00759
00760 r5 | abonnet | 2020-02-13 16:18:17 +0100 (jeu. 13 févr. 2020) | 1 ligne
00761
00762 r4 | abonnet | 2020-02-13 16:17:44 +0100 (jeu. 13 févr. 2020) | 2 lignes
00763
00764 Ajout des attribut de la caméra
00765
00766 r3 | abonnet | 2020-02-13 13:33:24 +0100 (jeu. 13 févr. 2020) | 2 lignes
00767
00768 Ajout des classes camera et rov
00769
00770 r2 | abonnet | 2020-02-12 14:28:56 +0100 (mer. 12 févr. 2020) | 2 lignes
00771
00772 Création du projet Qt
00773
00774 r1 | www-data | 2020-02-01 15:03:29 +0100 (sam. 01 févr. 2020) | 1 ligne
00775
00776 Creating initial repository structure
00777
```

7.19 Référence du fichier communicationrov.cpp

Fichier qui contient la définition de la classe CommunicationRov.

```
#include "communicationrov.h"
```

7.19.1 Description détaillée

Fichier qui contient la définition de la classe CommunicationRov.

Définition dans le fichier communicationrov.cpp.

7.20 communicationrov.cpp

```
00001
00007 #include "communicationrov.h"
00008
00009 CommunicationRov::CommunicationRov(QObject *parent) :
      QObject(parent)
00010 {
00011     qDebug() << Q_FUNC_INFO;
00012     port = new QSerialPort(this);
00013
00014     connect(port, SIGNAL(errorOccurred(QSerialPort::SerialPortError)), this, SLOT(
      gererErreur(QSerialPort::SerialPortError)));
00015 }
00016
00017 CommunicationRov::~CommunicationRov()
00018 {
00019     if(port->isOpen())
00020         port->close();
00021     qDebug() << Q_FUNC_INFO;
00022 }
00023
00024 bool CommunicationRov::ouvrirPort()
00025 {
00026     if(port->open(QIODevice::ReadWrite))
00027     {
00028         qDebug() << Q_FUNC_INFO << "Port ouvert" << port->isOpen();
00029         if(port->isOpen())
00030         {
00031             emit etatPortModifie(true, port->portName());
00032             connect(port, SIGNAL(readyRead()), this, SLOT(recevoir()));
```

```
00033             return true;
00034         }
00035         else
00036             return false;
00037     }
00038     #ifdef MODE_CONNECTE
00039     else
00040     {
00041         if(!port->isOpen())
00042         {
00043             QMessageBox::critical(nullptr, "Communication rov", QString::fromUtf8("Erreur ouverture du port
      série !"));
00044             return false;
00045         }
00046     }
00047     #endif
00048 }
00049
00050 void CommunicationRov::fermerPort()
00051 {
00052     port->close();
00053     disconnect(port, SIGNAL(readyRead()), this, SLOT(recevoir()));
00054     qDebug() << Q_FUNC_INFO << "Port fermé" << !port->isOpen();
00055     emit etatPortModifie(false, port->portName());
00056 }
00057
00058 void CommunicationRov::setConfiguration(
      Configuration maConfiguration)
00059 {
00060     port->setPortName(maConfiguration.port);
00061     port->setBaudRate(maConfiguration.debit);
00062     port->setDataBits((QSerialPort::DataBits)maConfiguration.bitsDonnees);
00063     port->setStopBits((QSerialPort::StopBits)maConfiguration.bitStop);
00064     port->setParity(QSerialPort::NoParity);
00065     port->setFlowControl(QSerialPort::NoFlowControl);
00066 }
00067
00068 int CommunicationRov::emettreTrame(QString trame)
00069 {
00070     int nombresOctets = -1;
00071
00072     if (port == NULL || !port->isOpen())
00073     {
00074        return -1;
00075     }
00076
00077     nombresOctets = port->write(trame.toLatin1());
00078     return nombresOctets;
00079 }
00080
00081 void CommunicationRov::recevoir()
00082 {
00083     while(port->bytesAvailable())
00084     {
00085         donnees += port->readAll();
00086         //qDebug() << Q_FUNC_INFO << "bytesAvailable" << port->bytesAvailable() << "donnees" << donnees;
00087     }
00088
00089     if(donnees.startsWith("$") && donnees.endsWith("\r\n"))
00090     {
00091         trameRecue = QString(donnees.data());
00092         //qDebug() << Q_FUNC_INFO << "trameRecue" << trameRecue;
00093         emit nouvelleTrame(trameRecue);
00094         donnees.clear();
00095     }
00096 }
00097
00098 QStringList CommunicationRov::detecterPortsDisponibles()
00099 {
00100     QStringList listePortsDetectes;
00101
00102     foreach(const QSerialPortInfo &info, QSerialPortInfo::availablePorts())
00103     {
00104         listePortsDetectes.push_back(info.portName());
00105     }
00106
00107     return listePortsDetectes;
00108 }
00109
00110 bool CommunicationRov::getEtatPort()
00111 {
00112     return port->isOpen();
00113 }
00114
00115 void CommunicationRov::gererErreur(QSerialPort::SerialPortError erreur)
00116 {
00117     switch(erreur)
00118     {
00119         case QSerialPort::ResourceError : fermerPort();
00120             break;
00121     }
```

```
00122 }
00123
```

7.21 Référence du fichier communicationrov.h

Fichier qui contient la déclaration de la classe CommunicationRov.

```
#include <QObject>
#include <QSerialPort>
#include <QSerialPortInfo>
#include <QDebug>
#include <QMessageBox>
```

Classes

- class CommunicationRov
  Class permettant de mettre en place une communication avec le rov.
- struct Configuration
  structure permettant de configurer une communication

Macros

- #define CONFIGURATION_MANUELLE
- #define MODE_CONNECTE

7.21.1 Description détaillée

Fichier qui contient la déclaration de la classe CommunicationRov.

Définition dans le fichier communicationrov.h.

7.21.2 Documentation des macros

7.21.2.1 CONFIGURATION_MANUELLE

```
#define CONFIGURATION_MANUELLE
```

Définition à la ligne 17 du fichier communicationrov.h.

7.21.2.2 MODE_CONNECTE

```
#define MODE_CONNECTE
```

Définition à la ligne 16 du fichier communicationrov.h.

7.22 communicationrov.h

```
00001
00007 #ifndef COMMUNICATIONROV_H
00008 #define COMMUNICATIONROV_H
00009
00010 #include <QObject>
00011 #include <QSerialPort>
00012 #include <QSerialPortInfo>
00013 #include <QDebug>
00014 #include <QMessageBox>
00015
00016 #define MODE_CONNECTE
00017 #define CONFIGURATION_MANUELLE
00018
00024 struct Configuration
00025 {
00026     QString port;
00027     int debit;
00028     int bitsDonnees;
00029     int bitStop;
00030 };
00031
00032
00038 class CommunicationRov : public QObject
00039 {
00040     Q_OBJECT
00041 private:
00042     QSerialPort *port;
00043     QByteArray donnees;
00044     QString trameRecue;
00045
00046 public:
00052     CommunicationRov(QObject *parent = nullptr);
00057     ~CommunicationRov();
00062     bool ouvrirPort();
00067     void fermerPort();
00073     void setConfiguration(Configuration maConfiguration);
00080     int emettreTrame(QString trame);
00086     static QStringList detecterPortsDisponibles();
00092     bool getEtatPort();
00093
00094 signals:
00100     void nouvelleTrame(QString trame);
00107     void etatPortModifie(bool etat, QString information);
00108
00109 public slots:
00114     void recevoir();
00115     void gererErreur(QSerialPort::SerialPortError);
00116 };
00117
00118 #endif // COMMUNICATIONROV_H
```

7.23 Référence du fichier ihmaccueil.cpp

Fichier qui contient la définition de la classe IHMAccueil.

```
#include "ihmaccueil.h"
#include "ihmrov.h"
#include "ihmcreationcampagne.h"
#include "campagne.h"
#include "basededonnees.h"
#include "ihmgraphiques.h"
#include <QMessageBox>
```

7.23.1 Description détaillée

Fichier qui contient la définition de la classe IHMAccueil.

Définition dans le fichier ihmaccueil.cpp.

7.24 ihmaccueil.cpp

```
00001
00007 #include "ihmaccueil.h"
00008 #include "ihmrov.h"
00009 #include "ihmcreationcampagne.h"
00010 #include "campagne.h"
00011 #include "basededonnees.h"
00012 #include "ihmgraphiques.h"
00013 #include <QMessageBox>
00014
00015 IHMAccueil::IHMAccueil(QWidget *parent) : QWidget(parent),
      campagnesEnCours()
00016 {
00017     qDebug() << Q_FUNC_INFO;
00018     baseDeDonnees = BaseDeDonnees::getInstance();
00019
00020     initialisationWidgets();
00021     initialisationDesignWidgets();
00022     initialiserEvenements();
00023     initialiserLayouts();
00024     configurerWidgets();
00025
00026     chargerCampagnes();
00027     rechargerListeCampagnes();
00028
00029     ihmRov = new IHMRov(this);
00030     ihmRov->hide();
00031 }
00032
00033 IHMAccueil::~IHMAccueil()
00034 {
00035     BaseDeDonnees::detruireInstance();
00036     qDebug() << Q_FUNC_INFO;
00037 }
00038
00039 void IHMAccueil::initialisationWidgets()
00040 {
00041     archives = new QLabel("Archives :", this);
00042     rechercheCampagneArchive = new QLineEdit(this);
00043     boutonImagesArchives = new QPushButton("Photos", this);
00044     boutonStatistiquesArchives = new QPushButton("Graphiques", this);
00045     boutonCreationCampagne = new QPushButton("Créer campagne", this);
00046     boutonDemarrerCampagne = new QPushButton("Démarrer", this);
00047     boutonArchiverCampagne = new QPushButton("Archiver", this);
00048     boutonSupprimerCampagne = new QPushButton("Supprimer", this);
00049     listeCampagne = new QComboBox(this);
00050     logoAccueil = new QLabel(this);
00051 }
00052
00053 void IHMAccueil::initialisationDesignWidgets()
00054 {
00055     QFont policeBouton("", 15, 75, false);
00056     QFont policeTexte("", 13, 75, false);
00057
00058     archives->setFixedSize(80,50);
00059     archives->setFont(policeTexte);
00060
00061     rechercheCampagneArchive->setFixedSize(200,40);
00062     rechercheCampagneArchive->setFont(policeTexte);
00063     rechercheCampagneArchive->setStyleSheet("QLineEdit {border-image:
      url(design/QLine_200x40.png)}" "QLineEdit:hover {border-image: url(design/QLine_200x40_survole.png)}");
00064
00065     boutonImagesArchives->setFixedSize(157,50);
00066     boutonImagesArchives->setFont(policeBouton);
00067     boutonImagesArchives->setStyleSheet("QPushButton {border-image:
      url(design/bouton_157x50.png)}" "QPushButton:hover {border-image: url(design/bouton_157x50_survole.png)}");
00068
00069     boutonStatistiquesArchives->setFixedSize(157,50);
00070     boutonStatistiquesArchives->setFont(policeBouton);
00071     boutonStatistiquesArchives->setStyleSheet("QPushButton {border-image:
      url(design/bouton_157x50.png)}" "QPushButton:hover {border-image: url(design/bouton_157x50_survole.png)}");
00072
00073     boutonCreationCampagne->setFixedSize(302,50);
00074     boutonCreationCampagne->setFont(policeBouton);
00075     boutonCreationCampagne->setStyleSheet("QPushButton {border-image:
      url(design/bouton_302x50.png)}" "QPushButton:hover {border-image: url(design/bouton_302x50_survole.png)}");
00076
00077     boutonDemarrerCampagne->setFixedSize(302,50);
00078     boutonDemarrerCampagne->setFont(policeBouton);
00079     boutonDemarrerCampagne->setStyleSheet("QPushButton {border-image:
      url(design/bouton_302x50.png)}" "QPushButton:hover {border-image: url(design/bouton_302x50_survole.png)}");
00080
00081     boutonArchiverCampagne->setFixedSize(199,50);
00082     boutonArchiverCampagne->setFont(policeBouton);
00083     boutonArchiverCampagne->setStyleSheet("QPushButton {border-image:
      url(design/bouton_199x50.png)}" "QPushButton:hover {border-image: url(design/bouton_199x50_survole.png)}");
00084
00085     boutonSupprimerCampagne->setFixedSize(199,50);
00086     boutonSupprimerCampagne->setFont(policeBouton);
```

```
00087     boutonSupprimerCampagne->setStyleSheet("QPushButton {border-image:
      url(design/bouton_199x50.png)}" "QPushButton:hover {border-image: url(design/bouton_199x50_survole.png)}");
00088
00089     listeCampagne->setFixedSize(199,50);
00090     listeCampagne->setFont(policeTexte);
00091     listeCampagne->setStyleSheet("QComboBox {border-image: url(design/combobox_199x50.png)}" "
      QComboBox:hover {border-image: url(design/combobox_199x50_survole.png)}" "QComboBox::drop-down
       {border-image: url(rien.png)}" "QComboBox {padding: 0 0 0 15px}");
00092
00093
00094 }
00095
00096 void IHMAccueil::initialiserLayouts()
00097 {
00098     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00099     QHBoxLayout *layoutSuperieur = new QHBoxLayout;
00100     QVBoxLayout *layoutCentral = new QVBoxLayout;
00101     QHBoxLayout *layoutConfigurationCampagne = new QHBoxLayout;
00102     QHBoxLayout *layoutCampagne = new QHBoxLayout;
00103
00104     layoutPrincipal->addLayout(layoutSuperieur);
00105     layoutSuperieur->addWidget(archives);
00106     layoutSuperieur->addWidget(rechercheCampagneArchive);
00107     layoutSuperieur->addWidget(boutonImagesArchives);
00108     layoutSuperieur->addWidget(boutonStatistiquesArchives);
00109
00110     layoutPrincipal->addLayout(layoutCentral);
00111     layoutCentral->setAlignment(Qt::AlignCenter);
00112     layoutCentral->addWidget(logoAccueil);
00113
00114     layoutPrincipal->addLayout(layoutConfigurationCampagne);
00115     layoutConfigurationCampagne->addWidget(listeCampagne);
00116     layoutConfigurationCampagne->addWidget(boutonArchiverCampagne);
00117     layoutConfigurationCampagne->addWidget(boutonSupprimerCampagne);
00118
00119     layoutPrincipal->addLayout(layoutCampagne);
00120     layoutCampagne->addWidget(boutonCreationCampagne);
00121     layoutCampagne->addWidget(boutonDemarrerCampagne);
00122
00123     setLayout(layoutPrincipal);
00124     setWindowTitle(NOM_FENETRE_ACCUEIL);
00125     setStyleSheet("background:#C1BEBE;");
00126 }
00127
00128 void IHMAccueil::configurerWidgets()
00129 {
00130     qDebug() << Q_FUNC_INFO << qApp->applicationDirPath() + "/images/Robot.png";
00131     logoAccueil->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/Robot.png"));
00132     listeCampagne->setEditable(false);
00133     rechercheCampagneArchive->setPlaceholderText("Nom de la campagne");
00134     rechercheCampagneArchive->setTextMargins(10,0,0,0);
00135 }
00136
00137 void IHMAccueil::initialiserEvenements()
00138 {
00139     connect(rechercheCampagneArchive, SIGNAL(textChanged(QString)), this, SLOT(
      rechercherCampagne(QString)));
00140     connect(boutonDemarrerCampagne, SIGNAL(clicked()), this, SLOT(
      demarrerCampagne()));
00141     connect(boutonCreationCampagne, SIGNAL(clicked()), this, SLOT(
      creerCampagne()));
00142     connect(boutonSupprimerCampagne, SIGNAL(clicked()), this, SLOT(
      supprimerCampagne()));
00143     connect(boutonArchiverCampagne, SIGNAL(clicked()), this, SLOT(
      archiverCampagne()));
00144     connect(boutonImagesArchives, SIGNAL(clicked()), this, SLOT(
      ouvrirArchive()));
00145     connect(boutonStatistiquesArchives, SIGNAL(clicked()), this, SLOT(
      ouvrirGraphiques()));
00146 }
00147
00148 void IHMAccueil::rechargerListeCampagnes()
00149 {
00150     listeCampagne->clear();
00151
00152     for(QVector<Campagne*>::iterator it = campagnesEnCours.begin(); it !=
      campagnesEnCours.end(); it++)
00153     {
00154         listeCampagne->addItem((*it)->getNomCampagne());
00155     }
00156
00157     if(listeCampagne->currentText().isEmpty())
00158     {
00159         boutonDemarrerCampagne->setDisabled(true);
00160         boutonDemarrerCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_302x50_grise.png)}");
00161
00162         boutonArchiverCampagne->setDisabled(true);
00163         boutonArchiverCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_199x50_grise.png)}");
00164
```

```
00165         boutonSupprimerCampagne->setDisabled(true);
00166         boutonSupprimerCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_199x50_grise.png)}");
00167     }
00168     else
00169     {
00170         boutonDemarrerCampagne->setEnabled(true);
00171         boutonDemarrerCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_302x50.png)}" "QPushButton:hover {border-image: url(design/bouton_302x50_survole.png)}");
00172
00173         boutonArchiverCampagne->setEnabled(true);
00174         boutonArchiverCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_199x50.png)}" "QPushButton:hover {border-image: url(design/bouton_199x50_survole.png)}");
00175
00176         boutonSupprimerCampagne->setEnabled(true);
00177         boutonSupprimerCampagne->setStyleSheet("QPushButton {border-image:
       url(design/bouton_199x50.png)}" "QPushButton:hover {border-image: url(design/bouton_199x50_survole.png)}");
00178     }
00179 }
00180
00181 void IHMAccueil::recupererNbPhotos(QString &nombrePhotos, QString &
      requeteNombrePhotos)
00182 {
00183     baseDeDonnees->recuperer(requeteNombrePhotos, nombrePhotos);
00184 }
00185
00186 void IHMAccueil::recupererCampagneEnCours(bool &retourCampagne, QString
       &requeteInformationsCampagne, QVector<QStringList> &campagnesEnCours)
00187 {
00188     retourCampagne = baseDeDonnees->recuperer(requeteInformationsCampagne,
      campagnesEnCours);
00189 }
00190
00191 void IHMAccueil::recupererPhotos(bool &retourPhoto, QString &
      requeteInformationsPhotos, QVector<QStringList> &informationsPhotos)
00192 {
00193     retourPhoto = baseDeDonnees->recuperer(requeteInformationsPhotos,
      informationsPhotos);
00194 }
00195
00196 void IHMAccueil::construireListe(QVector<QString> liste)
00197 {
00198     QCompleter *completeur = new QCompleter(liste.toList(),this);
00199     rechercheCampagneArchive->setCompleter(completeur);
00200 }
00201
00202 void IHMAccueil::chargerCampagnes()
00203 {
00204     baseDeDonnees->ouvrir("campagnes.sqlite");
00205
00206     bool retourCampagne, retourPhoto;
00207     QVector<QStringList> campagnesEnCours;
00208     QString nombrePhotos;
00209     QVector<QStringList> informationsPhotos;
00210
00211     QString requeteInformationsCampagne = "SELECT campagne.idCampagne, campagne.nom, campagne.lieu,
       technicien.nom, technicien.prenom, campagne.date, campagne.duree, campagne.cheminSauvegarde FROM campagne INNER
       JOIN technicien ON campagne.idTechnicien = technicien.idTechnicien WHERE campagne.enCours = '1' ORDER BY
       campagne.date DESC";
00212
00213     #ifdef DEBUG_BASEDEDONNEES
00214     qDebug() << Q_FUNC_INFO << QString::fromUtf8("requête : ") << requeteInformationsCampagne;
00215     #endif
00216
00217     recupererCampagneEnCours(retourCampagne, requeteInformationsCampagne,
      campagnesEnCours);
00218     if(retourCampagne)
00219     {
00220         for(int i=0; i < campagnesEnCours.size(); i++)
00221         {
00222             QStringList informationsCampagne = campagnesEnCours.at(i);
00223             QString requeteNombrePhotos = "SELECT COUNT(IdPhoto) FROM photo INNER JOIN campagne ON
       photo.IdCampagne = campagne.IdCampagne WHERE campagne.IdCampagne = '" + informationsCampagne.at(0) + "' ";
00224             recupererNbPhotos(nombrePhotos, requeteNombrePhotos);
00225
00226             #ifdef DEBUG_BASEDEDONNEES
00227             qDebug() << Q_FUNC_INFO << QString::fromUtf8("%0 %1 %2 %3 %4 %5 %6 %7").arg(
      informationsCampagne.at(0)).arg(informationsCampagne.at(1)).arg(informationsCampagne.at(2)).arg(informationsCampagne.at(3)).arg
      (informationsCampagne.at(4)).arg(informationsCampagne.at(5)).arg(informationsCampagne.at(6)).arg(
      informationsCampagne.at(7));
00228             #endif
00229             Campagne *campagne = new Campagne(informationsCampagne.at(1),
      informationsCampagne.at(2), informationsCampagne.at(3), informationsCampagne.at(4), QDateTime::fromString(
      informationsCampagne.at(5)), this, informationsCampagne.at(6).toInt());
00230
00231             campagne->setCheminSauvegarde(informationsCampagne.at(7));
00232             campagne->setNombrePhotos(nombrePhotos.toInt());
00233
00234             QString requeteInformationsPhotos ="SELECT cheminImage, aGarder, dateHeure FROM photo INNER
       JOIN campagne ON photo.IdCampagne = campagne.IdCampagne WHERE campagne.enCours = '1' AND photo.IdCampagne = '"
      + informationsCampagne.at(0) + "'";
```

```
00235             recupererPhotos(retourPhoto, requeteInformationsPhotos, informationsPhotos);
00236             if(retourPhoto)
00237             {
00238                 for(int i=0; i < informationsPhotos.size(); i++)
00239                 {
00240                     QStringList informationPhoto = informationsPhotos.at(i);
00241                     Photo photo;
00242                     photo.cheminSauvegarde = informationPhoto.at(0);
00243                     photo.aGarder = informationPhoto.at(1).toInt();
00244                     photo.image = QPixmap(informationPhoto.at(0));
00245                     photo.dateheure = QDateTime::fromString(informationPhoto.at(2));
00246 
00247                     campagne->ajouterPhoto(photo);
00248                 }
00249             }
00250             ajouterCampagne(campagne);
00251             informationsPhotos.clear();
00252         }
00253      }
00254      else
00255      {
00256         #ifdef DEBUG_BASEDEDONNEES
00257         qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00258         #endif
00259      }
00260 }
00261 
00262 void IHMAccueil::supprimerCampagneListe()
00263 {
00264     int nbCampagnes = campagnesEnCours.size();
00265     int n = 0;
00266     for(QVector<Campagne*>::iterator it = campagnesEnCours.begin(); n < nbCampagnes; it++)
00267     {
00268         if((*it)->getNomCampagne() == listeCampagne->currentText())
00269         {
00270             delete (*it);
00271             campagnesEnCours.erase(it);
00272         }
00273         n++;
00274     }
00275     rechargerListeCampagnes();
00276 }
00277 
00278 QString IHMAccueil::recupererIdCampagne()
00279 {
00280     bool retourRequeteIdCampagne;
00281     QString idCampagne;
00282     QString requeteIdCampagne = "SELECT idCampagne FROM campagne WHERE campagne.nom = '" +
      listeCampagne->currentText() + "'";
00283 
00284     retourRequeteIdCampagne = baseDeDonnees->recuperer(requeteIdCampagne, idCampagne)
      ;
00285 
00286     if(!retourRequeteIdCampagne)
00287     {
00288         #ifdef DEBUG_BASEDEDONNEES
00289             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00290         #endif
00291         return "0";
00292     }
00293     else
00294         return idCampagne;
00295 }
00296 
00297 void IHMAccueil::supprimerPhotoLocal(QString requete)
00298 {
00299     QVector<QString> photoASupprimer;
00300     baseDeDonnees->recuperer(requete, photoASupprimer);
00301     for(QVector<QString>::Iterator it = photoASupprimer.begin(); it != photoASupprimer.end(); ++it)
00302     {
00303        QFile::remove((*it));
00304     }
00305 }
00306 
00307 void IHMAccueil::supprimerDossierPhotoLocal()
00308 {
00309     QDir qDir;
00310 
00311     for(QVector<Campagne*>::Iterator it = campagnesEnCours.begin(); it !=
      campagnesEnCours.end(); ++it)
00312     {
00313         if((*it)->getNomCampagne() == listeCampagne->currentText())
00314         {
00315             if(qDir.exists((*it)->getCheminSauvegarde() + "/" + (*it)->getNomCampagne()))
00316             {
00317                 if(!qDir.rmdir((*it)->getCheminSauvegarde() + "/" + (*it)->getNomCampagne()))
00318                 {
00319                     qDebug() << Q_FUNC_INFO << "Erreur : impossible de supprimer le dossier" << (*it)->
      getCheminSauvegarde() + "/" + (*it)->getNomCampagne();
00320                     QMessageBox::critical(this, "Erreur", "Erreur : impossible de supprimer le dossier " +
      (*it)->getCheminSauvegarde() + "/" + (*it)->getNomCampagne() + " !");
```

```
00321                 }
00322             }
00323         }
00324     }
00325 }
00326
00327 void IHMAccueil::ajouterCampagne(Campagne *campagne, bool verification)
00328 {
00329     QDir dossierCampagne(campagne->getCheminSauvegarde());
00330     if(dossierCampagne.exists())
00331     {
00332         if(!dossierCampagne.mkdir(campagne->getNomCampagne()))
00333         {
00334             qDebug() << Q_FUNC_INFO << "Erreur : impossible de créer le dossier" << campagne->
      getCheminSauvegarde() << "campagne" << campagne->
      getNomCampagne();
00335             if(verification)
00336             {
00337                 QMessageBox::critical(this, "Erreur", "Erreur : impossible de créer le dossier " + campagne
      ->getCheminSauvegarde() + " !");
00338                 return;
00339             }
00340         }
00341     }
00342     campagnesEnCours.push_back(campagne);
00343     rechargerListeCampagnes();
00344 }
00345
00346 void IHMAccueil::enregistrerCampagneBDD(
      Campagne *campagne)
00347 {
00348     #ifdef DEBUG_BASEDEDONNEES
00349     qDebug() << Q_FUNC_INFO << campagne->getNomCampagne();
00350     #endif
00351
00352     baseDeDonnees->ouvrir("campagnes.sqlite");
00353
00354     QString idTechnicien;
00355     QString requeteId = "SELECT technicien.IdTechnicien FROM technicien WHERE technicien.nom = '" +
      campagne->getNomTechnicien() + "' AND technicien.prenom = '" + campagne->
      getPrenomTechnicien() + "'";
00356     bool retour = baseDeDonnees->recuperer(requeteId, idTechnicien);
00357     if(!retour)
00358     {
00359         QString requeteInformation = "INSERT INTO technicien (nom, prenom) VALUES ('" + campagne->
      getNomTechnicien() + "','" + campagne->getPrenomTechnicien() + "')";
00360         baseDeDonnees->executer(requeteInformation);
00361         baseDeDonnees->recuperer(requeteId, idTechnicien);
00362     }
00363
00364     QString requeteInsertionCampagne = "INSERT INTO campagne (idTechnicien, nom, lieu, date, duree,
       enCours, cheminSauvegarde) VALUES ('" + idTechnicien + "', '" + campagne->getNomCampagne() + "', '" +
       campagne->getLieu() + "', '" + campagne->getDate().toString() + "', '" + QString::number(
      campagne->getDuree()) + "', '1', '" + campagne->getCheminSauvegarde() + "')";
00365     baseDeDonnees->executer(requeteInsertionCampagne);
00366 }
00367
00368 void IHMAccueil::modifierCampagneBDD(Campagne *campagne)
00369 {
00370     QString idCampagne, requeteUpdateDuree, requeteIdCampagne, requeteInsertionMesures,
      requeteModifierPhotos;
00371
00372     requeteUpdateDuree = "UPDATE campagne SET duree = '"+ QString::number(campagne->
      getDuree()) + "' WHERE campagne.nom = '" + campagne->getNomCampagne() + "'";
00373     baseDeDonnees->executer(requeteUpdateDuree);
00374
00375     requeteIdCampagne = "SELECT campagne.idCampagne FROM campagne WHERE campagne.nom = '" + campagne->
      getNomCampagne() + "'";
00376     baseDeDonnees->recuperer(requeteIdCampagne, idCampagne);
00377
00378     for(int i =0; i < campagne->getAlbumPhoto().size(); ++i)
00379     {
00380         requeteModifierPhotos = "UPDATE photo set aGarder = '" + QString::number(campagne->
      getAlbumPhoto()[i].aGarder) + "' WHERE cheminImage = '" + campagne->
      getAlbumPhoto()[i].cheminSauvegarde + "'";
00381         baseDeDonnees->executer(requeteModifierPhotos);
00382     }
00383     campagne->supprimerMesures();
00384 }
00385
00386 void IHMAccueil::ajouterPhotoBDD(Photo &photo,
      Campagne *campagne)
00387 {
00388     QString idCampagne;
00389     QString requeteIdCampagne = "SELECT IdCampagne FROM campagne WHERE campagne.nom = '" + campagne->
      getNomCampagne() + "'";
00390     baseDeDonnees->recuperer(requeteIdCampagne, idCampagne);
00391
00392     QString requeteInsertion = "INSERT INTO photo (idCampagne, cheminImage, aGarder, dateHeure) VALUES ('"
      + idCampagne + "', '" + photo.cheminSauvegarde + "', '1', '" + photo.
      dateheure.toString() + "')";
```

```
00393     baseDeDonnees->executer(requeteInsertion);
00394 }
00395
00396 void IHMAccueil::demarrerCampagne()
00397 {
00398     for(QVector<Campagne*>::iterator it = campagnesEnCours.begin(); it !=
      campagnesEnCours.end(); it++)
00399     {
00400         if((*it)->getNomCampagne() == listeCampagne->currentText())
00401         {
00402             qDebug() << Q_FUNC_INFO << listeCampagne->currentText();
00403             ihmRov->setCampagne(*it);
00404             ihmRov->showMinimized();
00405             ihmRov->gererCampagne();
00406             this->setVisible(false);
00407             break;
00408         }
00409     }
00410 }
00411
00412 void IHMAccueil::creerCampagne()
00413 {
00414     QVector<QStringList> listeTechniciens;
00415     QString requete = "SELECT technicien.nom, technicien.prenom FROM technicien";
00416
00417     baseDeDonnees->recuperer(requete, listeTechniciens);
00418
00419     IHMCreationCampagne *ihmCreationCampagne = new
      IHMCreationCampagne(this, listeTechniciens);
00420     ihmCreationCampagne->setFixedSize(416,278);
00421     ihmCreationCampagne->exec();
00422 }
00423
00424 void IHMAccueil::archiverCampagne()
00425 {
00426     baseDeDonnees->ouvrir("campagnes.sqlite");
00427     bool retourRequeteArchiver;
00428     bool retourRequeteSuppressionPhoto;
00429     QString requeteArchiver, requeteSuppressionPhoto;
00430
00431     supprimerPhotoLocal("SELECT photo.cheminImage FROM photo WHERE photo.IdCampagne = '"
       + recupererIdCampagne() + "' AND photo.aGarder = '0'");
00432
00433     requeteArchiver = "UPDATE campagne SET enCours = '0' WHERE campagne.nom = '" +
      listeCampagne->currentText() + "'";
00434     requeteSuppressionPhoto = "DELETE FROM photo WHERE photo.IdCampagne = '" +
      recupererIdCampagne() + "' AND photo.aGarder = '0'";
00435
00436     retourRequeteArchiver = baseDeDonnees->executer(requeteArchiver);
00437     retourRequeteSuppressionPhoto = baseDeDonnees->executer(requeteSuppressionPhoto);
00438
00439     if(!retourRequeteArchiver && !retourRequeteSuppressionPhoto)
00440     {
00441         #ifdef DEBUG_BASEDEDONNEES
00442             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00443         #endif
00444     }
00445     else
00446         supprimerCampagneListe();
00447 }
00448
00449 void IHMAccueil::supprimerCampagne()
00450 {
00451     baseDeDonnees->ouvrir("campagnes.sqlite");
00452     bool retourRequeteSuppressionMesures;
00453     bool retourRequeteSuppressionPhotos;
00454     bool retourRequeteSuppressionCampagne;
00455
00456     supprimerPhotoLocal("SELECT photo.cheminImage FROM photo WHERE photo.IdCampagne = '"
       + recupererIdCampagne() + "'");
00457
00458     supprimerDossierPhotoLocal();
00459
00460     QString requeteSuppressionMesures = "DELETE FROM mesure WHERE mesure.IdCampagne = '" +
      recupererIdCampagne() + "'";
00461     QString requeteSuppressionPhotos = "DELETE FROM photo WHERE photo.IdCampagne = '" +
      recupererIdCampagne() + "'";
00462     QString requeteSuppressionCampagne = "DELETE FROM campagne WHERE campagne.IdCampagne = '" +
      recupererIdCampagne() + "'";
00463
00464     retourRequeteSuppressionMesures = baseDeDonnees->executer(
      requeteSuppressionMesures);
00465     retourRequeteSuppressionPhotos = baseDeDonnees->executer(requeteSuppressionPhotos)
      ;
00466     retourRequeteSuppressionCampagne = baseDeDonnees->executer(
      requeteSuppressionCampagne);
00467
00468     if(!retourRequeteSuppressionMesures && !retourRequeteSuppressionPhotos && !
      retourRequeteSuppressionCampagne)
00469     {
00470         #ifdef DEBUG_BASEDEDONNEES
```

```
00471             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00472         #endif
00473     }
00474     else
00475         supprimerCampagneListe();
00476 }
00477
00478 void IHMAccueil::enregistrerMesureBDD(QString temperature, QString humidite,
      QString radiation)
00479 {
00480     baseDeDonnees->ouvrir("campagnes.sqlite");
00481     bool retourRequeteEnregistrementMesure;
00482
00483     QString requeteEnregistrementMesure = "INSERT INTO mesure (idCampagne, heure, temperature, radiation,
       humidite) VALUES (" + recupererIdCampagne() + ",'" + QDateTime::currentDateTime().toString
      () + "'," + temperature + "," + radiation + "," + humidite + ")";
00484
00485     retourRequeteEnregistrementMesure = baseDeDonnees->executer(
      requeteEnregistrementMesure);
00486
00487     if(!retourRequeteEnregistrementMesure)
00488     {
00489         #ifdef DEBUG_BASEDEDONNEES
00490             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00491         #endif
00492     }
00493 }
00494
00495 void IHMAccueil::rechercherCampagne(QString texte)
00496 {
00497     #ifdef DEBUG_BASEDEDONNEES
00498         qDebug() << Q_FUNC_INFO;
00499     #endif
00500
00501     baseDeDonnees->ouvrir("campagnes.sqlite");
00502     QVector<QString> listeCampagnesRecherchees;
00503
00504     bool retourRequeteRechercheCampagne;
00505
00506     QString requeteRechercheCampagne = "SELECT campagne.nom FROM campagne WHERE campagne.nom LIKE '" +
      texte + "%' AND campagne.enCours = '0'";
00507
00508     retourRequeteRechercheCampagne = baseDeDonnees->recuperer(
      requeteRechercheCampagne, listeCampagnesRecherchees);
00509
00510     if(!retourRequeteRechercheCampagne)
00511     {
00512         #ifdef DEBUG_BASEDEDONNEES
00513             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur !");
00514         #endif
00515     }
00516     else
00517     {
00518         construireListe(listeCampagnesRecherchees);
00519     }
00520 }
00521
00522 void IHMAccueil::ouvrirArchive()
00523 {
00524     QStringList cheminSauvegarde;
00525     QString requete = "SELECT campagne.cheminSauvegarde, campagne.nom FROM campagne WHERE campagne.nom = '"
       + rechercheCampagneArchive->text() + "'";
00526
00527     baseDeDonnees->ouvrir("campagnes.sqlite");
00528     bool retour = baseDeDonnees->recuperer(requete, cheminSauvegarde);
00529     if(retour)
00530     {
00531         if(!(QDesktopServices::openUrl(QUrl(cheminSauvegarde.at(0) + "/" + cheminSauvegarde.at(1),
      QUrl::TolerantMode))))
00532             QMessageBox::critical(this, "Erreur", "Erreur : Chemin introuvable ");
00533     }
00534     else
00535     {
00536         QMessageBox::critical(this, "Erreur", "Erreur : Nom de campagne introuvable ");
00537     }
00538 }
00539
00540 void IHMAccueil::ouvrirGraphiques()
00541 {
00542     QVector<QStringList> mesures;
00543     QString requete = "SELECT mesure.heure, mesure.radiation, mesure.temperature, mesure.humidite FROM
       mesure INNER JOIN campagne ON campagne.IdCampagne = mesure.IdCampagne WHERE campagne.nom = '" +
      rechercheCampagneArchive->text() + "'";
00544
00545     baseDeDonnees->ouvrir("campagnes.sqlite");
00546     bool retour = baseDeDonnees->recuperer(requete, mesures);
00547     if(retour)
00548     {
00549         if(!mesures.isEmpty())
00550         {
00551             IHMGraphiques *ihmGraphique = new IHMGraphiques(mesures);
```

```
00552             ihmGraphique->show();
00553         }
00554         else
00555         {
00556             QMessageBox::critical(this, "Erreur", "Erreur : Aucune données trouvées !");
00557         }
00558     }
00559 }
```

7.25 Référence du fichier ihmaccueil.h

Fichier qui contient la déclaration de la classe IHMAccueil.

```
#include <QVector>
#include <QtWidgets>
#include <QString>
```

Classes

— class IHMAccueil
Class permettant de créer une nouvelle campagne, reprendre une campagne mise en pause, archiver une campagne, supprimer une campagne, accéder à la base de données et configurer le matériel.

Macros

— #define NOM_FENETRE_ACCUEIL "Projet Rovnet 2020"
— #define RATIO30 0.3
— #define RATIO40 0.4
— #define RATIO50 0.5

7.25.1 Description détaillée

Fichier qui contient la déclaration de la classe IHMAccueil.

Définition dans le fichier ihmaccueil.h.

7.25.2 Documentation des macros

7.25.2.1 NOM_FENETRE_ACCUEIL

```
#define NOM_FENETRE_ACCUEIL "Projet Rovnet 2020"
```

Définition à la ligne 14 du fichier ihmaccueil.h.

Référencé par IHMAccueil : :initialiserLayouts().

7.25.2.2 RATIO30

```
#define RATIO30 0.3
```

Définition à la ligne 15 du fichier ihmaccueil.h.

7.25.2.3 RATIO40

```
#define RATIO40 0.4
```

Définition à la ligne 16 du fichier ihmaccueil.h.

7.25.2.4 RATIO50

```
#define RATIO50 0.5
```

Définition à la ligne 17 du fichier ihmaccueil.h.

7.26 ihmaccueil.h

```
00001
00007 #ifndef IHMACCUEIL_H
00008 #define IHMACCUEIL_H
00009
00010 #include <QVector>
00011 #include <QtWidgets>
00012 #include <QString>
00013
00014 #define NOM_FENETRE_ACCUEIL "Projet Rovnet 2020"
00015 #define RATIO30 0.3
00016 #define RATIO40 0.4
00017 #define RATIO50 0.5
00018
00019 class Campagne;
00020 class BaseDeDonnees;
00021 class IHMRov;
00022 struct Photo;
00023
00029 class IHMAccueil : public QWidget
00030 {
00031     Q_OBJECT
00032
00033 private:
00034     QPushButton *boutonImagesArchives;
00035     QPushButton *boutonStatistiquesArchives;
00036     QPushButton *boutonCreationCampagne;
00037     QPushButton *boutonDemarrerCampagne;
00038     QPushButton *boutonArchiverCampagne;
00039     QPushButton *boutonSupprimerCampagne;
00040     QComboBox *listeCampagne;
00041     QLineEdit *rechercheCampagneArchive;
00042     QLabel *logoAccueil;
00043     QLabel *archives;
00044     QVector<Campagne*> campagnesEnCours;
00045     BaseDeDonnees *baseDeDonnees;
00046     IHMRov *ihmRov;
00047
00052     void initialisationWidgets();
00057     void initialisationDesignWidgets();
00062     void initialiserLayouts();
00067     void configurerWidgets();
00072     void initialiserEvenements();
00077     void rechargerListeCampagnes();
00082     void chargerCampagnes();
00087     void supprimerCampagneListe();
00093     QString recupererIdCampagne();
00099     void supprimerPhotoLocal(QString requete);
00104     void supprimerDossierPhotoLocal();
00105
00112     void recupererNbPhotos(QString &nombrePhotos, QString &requeteNombrePhotos);
```

```
00120     void recupererCampagneEnCours(bool &retourCampagne, QString &
      requeteInformationsCampagne, QVector<QStringList> &campagnesEnCours);
00128     void recupererPhotos(bool &retourPhoto, QString &requeteInformationsPhotos,
      QVector<QStringList> &informationsPhotos);
00134     void construireListe(QVector<QString> liste);
00135
00136 public:
00142     explicit IHMAccueil(QWidget *parent = nullptr);
00147     ~IHMAccueil();
00154     void ajouterCampagne(Campagne *campagne, bool verification=false);
00160     void enregistrerCampagneBDD(Campagne *campagne);
00166     void modifierCampagneBDD(Campagne *campagne);
00173     void ajouterPhotoBDD(Photo &photo, Campagne *campagne);
00174
00175
00176 signals:
00177
00178 public slots:
00183     void demarrerCampagne();
00188     void creerCampagne();
00193     void archiverCampagne();
00198     void supprimerCampagne();
00206     void enregisterMesureBDD(QString temperature, QString humidite, QString radiation);
00212     void rechercherCampagne(QString texte);
00217     void ouvrirArchive();
00222     void ouvrirGraphiques();
00223 };
00224
00225 #endif // IHMACCUEIL_H
```

7.27 Référence du fichier ihmalbumphoto.cpp

Fichier qui contient la définition de la classe IHMAlbumPhoto.

```
#include "ihmalbumphoto.h"
#include "ihmrov.h"
#include "campagne.h"
```

7.27.1 Description détaillée

Fichier qui contient la définition de la classe IHMAlbumPhoto.

Définition dans le fichier ihmalbumphoto.cpp.

7.28 ihmalbumphoto.cpp

```
00001
00007 #include "ihmalbumphoto.h"
00008 #include "ihmrov.h"
00009 #include "campagne.h"
00010
00011 IHMAlbumPhoto::IHMAlbumPhoto(IHMRov *ihmRov,
      QWidget *parent) : QWidget(parent), imhRov(ihmRov)
00012 {
00013     qDebug() << Q_FUNC_INFO;
00014     photos = new QWidget();
00015     layoutPhotos = new QHBoxLayout;
00016     layoutAlbumPhoto = new QVBoxLayout;
00017     scrollArea = new QScrollArea();
00018
00019     layoutPhotos->setAlignment(Qt::AlignCenter);
00020     layoutAlbumPhoto->setAlignment(Qt::AlignCenter);
00021
00022     photos->setLayout(layoutAlbumPhoto);
00023     scrollArea->setWidgetResizable(true);
00024     scrollArea->setFrameStyle(QFrame::Panel);
00025     scrollArea->setWidget(photos);
00026     layoutPhotos->addWidget(scrollArea);
00027
00028     setLayout(layoutPhotos);
00029
00030     int width = qApp->desktop()->availableGeometry().width();
```

```
00031     int height = qApp->desktop()->availableGeometry().height();
00032     resize(width, height);
00033     setStyleSheet("background:#202020;color:white;");
00034 }
00035
00036 IHMAlbumPhoto::~IHMAlbumPhoto()
00037 {
00038     qDebug() << Q_FUNC_INFO;
00039 }
00040
00041 void IHMAlbumPhoto::ouvrirAlbumPhotos(QVector<Photo>
      albumPhoto)
00042 {
00043     if(albumPhoto.isEmpty())
00044     {
00045         QMessageBox::critical(this, "Erreur", "Listes photos vide !");
00046         return;
00047     }
00048
00049     QFont police("", 15, 50, false);
00050
00051     this->albumPhoto = albumPhoto;
00052     signalMapper = new QSignalMapper(this);
00053     int numeroPhoto = 0;
00054     for(QVector<Photo>::iterator it = albumPhoto.begin(); it != albumPhoto.end(); ++it, numeroPhoto++)
00055     {
00056         QVBoxLayout *layoutPhoto = new QVBoxLayout;
00057         QHBoxLayout *layoutInformationsPhotos = new QHBoxLayout;
00058         QHBoxLayout *layoutPhotoAGArder = new QHBoxLayout;
00059
00060         QLabel *photo = new QLabel(this);
00061         photo->setPixmap((*it).image);
00062
00063         QLabel *dateHeure = new QLabel((*it).dateheure.toString(), this);
00064         QLabel *chemin = new QLabel((*it).cheminSauvegarde,this);
00065         QCheckBox *photoGarde = new QCheckBox(this);
00066         QWidget *information = new QWidget(this);
00067         QLabel *photoAGarder = new QLabel("Photo à garder :", this);
00068
00069         layoutPhotoAGArder->setAlignment(Qt::AlignRight);
00070         information->setFixedWidth((*it).image.width());
00071
00072         if(albumPhoto[numeroPhoto].aGarder)
00073             photoGarde->setChecked(true);
00074         else
00075             photoGarde->setChecked(false);
00076
00077         connect(photoGarde, SIGNAL(clicked()), signalMapper, SLOT(map()));
00078         signalMapper->setMapping(photoGarde, numeroPhoto);
00079
00080         layoutAlbumPhoto->addLayout(layoutPhoto);
00081         layoutPhoto->addWidget(information);
00082         information->setLayout(layoutInformationsPhotos);
00083         layoutPhoto->addWidget(photo);
00084         layoutInformationsPhotos->addWidget(dateHeure);
00085         layoutInformationsPhotos->addWidget(chemin);
00086         layoutInformationsPhotos->addLayout(layoutPhotoAGArder);
00087         layoutPhotoAGArder->addWidget(photoAGarder);
00088         layoutPhotoAGArder->addWidget(photoGarde);
00089
00090         dateHeure->setFont(police);
00091         chemin->setFont(police);
00092         photoAGarder->setFont(police);
00093         information->setStyleSheet("background-color: white");
00094         dateHeure->setStyleSheet("color: black");
00095         chemin->setStyleSheet("color: black");
00096         photoAGarder->setStyleSheet("color: black");
00097         photoGarde->setStyleSheet("color: black");
00098     }
00099     connect(signalMapper, SIGNAL(mapped(int)), this, SLOT(
      selectionnerPhoto(int)));
00100
00101     this->show();
00102 }
00103
00104 void IHMAlbumPhoto::selectionnerPhoto(int numeroPhoto)
00105 {
00106     if(numeroPhoto < albumPhoto.size())
00107     {
00108         imhRov->getCampagne()->modifierArchivePhoto(numeroPhoto);
00109         qDebug() << Q_FUNC_INFO << "numeroPhoto" << numeroPhoto << "A garder" <<
      imhRov->getCampagne()->getAlbumPhoto()[numeroPhoto].aGarder;
00110     }
00111 }
```

7.29 Référence du fichier ihmalbumphoto.h

Fichier qui contient la déclaration de la classe IHMAlbumPhoto.

```
#include <QtWidgets>
#include <QVector>
```

Classes

— class IHMAlbumPhoto
Class permettant de visualiser les photos en cours de campagne.
— struct Photo
structure contenant les informations d'une photo de campagne

Macros

— #define NOM_FENETRE_ALBUMPHOTO "Projet Rovnet 2020"

7.29.1 Description détaillée

Fichier qui contient la déclaration de la classe IHMAlbumPhoto.

Définition dans le fichier ihmalbumphoto.h.

7.29.2 Documentation des macros

7.29.2.1 NOM_FENETRE_ALBUMPHOTO

```
#define NOM_FENETRE_ALBUMPHOTO "Projet Rovnet 2020"
```

Définition à la ligne 13 du fichier ihmalbumphoto.h.

7.30 ihmalbumphoto.h

```
00001
00007 #ifndef ALBUMPHOTO_H
00008 #define ALBUMPHOTO_H
00009
00010 #include <QtWidgets>
00011 #include <QVector>
00012
00013 #define NOM_FENETRE_ALBUMPHOTO "Projet Rovnet 2020"
00014
00020 struct Photo
00021 {
00022     QPixmap image;
00023     QDateTime dateheure;
00024     bool aGarder;
00025     QString cheminSauvegarde;
00026 };
00027
00028 class IHMRov;
00029
00035 class IHMAlbumPhoto : public QWidget
00036 {
00037     Q_OBJECT
00038 private:
00039     QWidget *photos;
00040     QHBoxLayout *layoutPhotos;
00041     QVBoxLayout *layoutAlbumPhoto;
00042     QScrollArea *scrollArea;
00043     QSignalMapper *signalMapper;
00044     QVector<Photo> albumPhoto;
00045     IHMRov *imhRov;
00046
00047 public:
00054     IHMAlbumPhoto(IHMRov *ihmRov, QWidget *parent = nullptr);
00059     ~IHMAlbumPhoto();
00064     void ouvrirAlbumPhotos(QVector<Photo> albumPhoto);
00065
00066 signals:
00067
00068 public slots:
00074     void selectionnerPhoto(int numero);
00075 };
00076
00077 #endif // ALBUMPHOTO_H
```

7.31 Référence du fichier ihmconfiguration.cpp

Fichier qui contient la définition de la classe IHMConfiguration.

```
#include "ihmconfiguration.h"
```

7.31.1 Description détaillée

Fichier qui contient la définition de la classe IHMConfiguration.

Définition dans le fichier ihmconfiguration.cpp.

7.32 ihmconfiguration.cpp

```
00001
00007 #include "ihmconfiguration.h"
00008
00009 IHMConfiguration::IHMConfiguration(Rov *rov,
      QWidget *parent) : QWidget(parent), rov(rov)
00010 {
00011     qDebug() << Q_FUNC_INFO;
00012
00013     initialiserWidgets();
00014     configurerWidgets();
00015     initialiserLayouts();
00016     initialiserEvenements();
00017
00018     modifierConfiguration();
00019     rov->getCommunicationRov()->ouvrirPort();
00020     modifieEtatBoutons();
00021 }
00022
00023 IHMConfiguration::~IHMConfiguration()
00024 {
00025     qDebug() << Q_FUNC_INFO;
00026 }
00027
00028 void IHMConfiguration::initialiserWidgets()
00029 {
00030     listePortsDisponibles = new QComboBox(this);
00031     listeDebit = new QComboBox(this);
00032     listeBitsDonnees = new QComboBox(this);
00033     listeBitsStop = new QComboBox(this);
00034     boutonGererPort = new QPushButton("Fermer", this);
00035 }
00036
00037 void IHMConfiguration::configurerWidgets()
00038 {
00039     listePortsDisponibles->addItems(
      CommunicationRov::detecterPortsDisponibles());
00040     if(listePortsDisponibles->currentText() == "")
00041         listePortsDisponibles->addItem("Aucun port détécté");
00042     listeDebit->addItem("9600");
00043     listeDebit->addItem("115200");
00044
00045     listeBitsDonnees->addItem("7");
00046     listeBitsDonnees->addItem("8");
00047     listeBitsDonnees->setCurrentIndex(1);
00048
00049     listeBitsStop->addItem("1");
00050     listeBitsStop->addItem("2");
00051 }
00052
00053 void IHMConfiguration::initialiserLayouts()
00054 {
00055     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00056     QHBoxLayout *layoutInformation = new QHBoxLayout;
00057     QFormLayout *layoutConfiguration = new QFormLayout;
00058     QVBoxLayout *layoutCommande = new QVBoxLayout;
00059
00060     layoutCommande->setAlignment(Qt::AlignTop);
00061
00062     layoutPrincipal->addLayout(layoutInformation);
00063     layoutInformation->addLayout(layoutConfiguration);
00064     layoutInformation->addLayout(layoutCommande);
00065     layoutConfiguration->addRow("Port:", listePortsDisponibles);
```

```
00066     layoutConfiguration->addRow("Débit:", listeDebit);
00067     layoutConfiguration->addRow("Bits de données:", listeBitsDonnees);
00068     layoutConfiguration->addRow("Bits de stop:", listeBitsStop);
00069     layoutCommande->addWidget(boutonGererPort);
00070
00071     setLayout(layoutPrincipal);
00072     setStyleSheet("background:#202020;color:white;");
00073 }
00074
00075 void IHMConfiguration::initialiserEvenements()
00076 {
00077     connect(listePortsDisponibles, SIGNAL(currentIndexChanged(int)), this, SLOT(
      modifierConfiguration()));
00078     connect(listeDebit, SIGNAL(currentIndexChanged(int)), this, SLOT(
      modifierConfiguration()));
00079     connect(listeBitsDonnees, SIGNAL(currentIndexChanged(int)), this, SLOT(
      modifierConfiguration()));
00080     connect(listeBitsStop, SIGNAL(currentIndexChanged(int)), this, SLOT(
      modifierConfiguration()));
00081     connect(boutonGererPort, SIGNAL(clicked()), this, SLOT(
      modifierEtatPort()));
00082 }
00083
00084 void IHMConfiguration::modifieEtatBoutons()
00085 {
00086     if(rov->getCommunicationRov()->getEtatPort())
00087     {
00088         listePortsDisponibles->setDisabled(true);
00089         listeDebit->setDisabled(true);
00090         listeBitsDonnees->setDisabled(true);
00091         listeBitsStop->setDisabled(true);
00092     }
00093     else
00094     {
00095         boutonGererPort->setText("Ouvrir");
00096         listePortsDisponibles->setEnabled(true);
00097         listeDebit->setEnabled(true);
00098         listeBitsDonnees->setEnabled(true);
00099         listeBitsStop->setEnabled(true);
00100     }
00101 }
00102
00103 void IHMConfiguration::modifierConfiguration()
00104 {
00105     Configuration configuration;
00106
00107     configuration.port = listePortsDisponibles->currentText();
00108     configuration.debit = listeDebit->currentText().toInt();
00109     configuration.bitsDonnees = listeBitsDonnees->currentText().toInt();
00110     configuration.bitStop = listeBitsStop->currentText().toInt();
00111
00112     rov->modifierConfiguration(configuration);
00113 }
00114
00115 void IHMConfiguration::modifierEtatPort()
00116 {
00117     if(boutonGererPort->text() == "Ouvrir")
00118     {
00119         if(rov->getCommunicationRov()->ouvrirPort())
00120         {
00121             boutonGererPort->setText("Fermer");
00122             listePortsDisponibles->setDisabled(true);
00123             listeDebit->setDisabled(true);
00124             listeBitsDonnees->setDisabled(true);
00125             listeBitsStop->setDisabled(true);
00126         }
00127     }
00128     else
00129     {
00130         rov->getCommunicationRov()->fermerPort();
00131         boutonGererPort->setText("Ouvrir");
00132         listePortsDisponibles->setEnabled(true);
00133         listeDebit->setEnabled(true);
00134         listeBitsDonnees->setEnabled(true);
00135         listeBitsStop->setEnabled(true);
00136     }
00137 }
00138
00139 void IHMConfiguration::actualisePortsDisponibles()
00140 {
00141     listePortsDisponibles->clear();
00142     listePortsDisponibles->addItems(
      CommunicationRov::detecterPortsDisponibles());
00143     if(listePortsDisponibles->currentText() == "")
00144         listePortsDisponibles->addItem("Aucun port détécté");
00145 }
```

7.33 Référence du fichier ihmconfiguration.h

Fichier qui contient la déclaration de la classe IHMConfiguration.

```
#include <QWidget>
#include "rov.h"
```

Classes

— class IHMConfiguration
Class permettant de configurer la communication avec le rov.

7.33.1 Description détaillée

Fichier qui contient la déclaration de la classe IHMConfiguration.

Définition dans le fichier ihmconfiguration.h.

7.34 ihmconfiguration.h

```
00001
00007 #ifndef IHMCONFIGURATION_H
00008 #define IHMCONFIGURATION_H
00009
00010 #include <QWidget>
00011 #include "rov.h"
00012
00013 class Rov;
00014
00020 class IHMConfiguration : public QWidget
00021 {
00022     Q_OBJECT
00023 private:
00024     Rov *rov;
00025     QComboBox *listePortsDisponibles;
00026     QComboBox *listeDebit;
00027     QComboBox *listeBitsDonnees;
00028     QComboBox *listeBitsStop;
00029     QPushButton *boutonGererPort;
00030
00035     void initialiserWidgets();
00040     void configurerWidgets();
00045     void initialiserLayouts();
00050     void initialiserEvenements();
00051
00052 public:
00059     IHMConfiguration(Rov *rov, QWidget *parent = nullptr);
00064     ~IHMConfiguration();
00065
00066 public slots:
00071     void modifierConfiguration();
00076     void modifierEtatPort();
00081     void modifieEtatBoutons();
00086     void actualisePortsDisponibles();
00087
00088 signals:
00089
00090
00091 };
00092
00093 #endif // IHMCONFIGURATION_H
```

7.35 Référence du fichier ihmcreationcampagne.cpp

Fichier qui contient la définition de la classe IHMCreationCampagne.

```
#include "ihmcreationcampagne.h"
#include "campagne.h"
#include "ihmaccueil.h"
```

7.35.1 Description détaillée

Fichier qui contient la définition de la classe IHMCreationCampagne.

Définition dans le fichier ihmcreationcampagne.cpp.

7.36 ihmcreationcampagne.cpp

```
00001
00007 #include "ihmcreationcampagne.h"
00008 #include "campagne.h"
00009 #include "ihmaccueil.h"
00010
00011 IHMCreationCampagne::IHMCreationCampagne(
      IHMAccueil *ihmAccueil, QVector<QStringList> &listeTechniciens) :
      QDialog(ihmAccueil, Qt::Dialog), ihmAccueil(ihmAccueil), listeTechniciens(listeTechniciens)
00012 {
00013     qDebug() << Q_FUNC_INFO;
00014
00015     initialisationWidgets();
00016     initialisationDesignWidgets();
00017     initialiserEvenements();
00018     initialiserLayouts();
00019     configurerWidgets();
00020     chargerListeTechniciens();
00021 }
00022
00023 IHMCreationCampagne::~IHMCreationCampagne()
00024 {
00025     qDebug() << Q_FUNC_INFO;
00026 }
00027
00028 void IHMCreationCampagne::initialisationWidgets()
00029 {
00030     nomCampagne = new QLineEdit(this);
00031     nomTechnicien = new QLineEdit(this);
00032     prenomTechnicien = new QLineEdit(this);
00033     lieu = new QLineEdit(this);
00034     cheminSauvegarde = new QLineEdit(this);
00035     boutonValider = new QPushButton("Valider", this);
00036     boutonAnnuler = new QPushButton("Annuler", this);
00037     boutonChoixChemin = new QPushButton("...", this);
00038     techniciens = new QComboBox(this);
00039 }
00040
00041 void IHMCreationCampagne::initialisationDesignWidgets()
00042 {
00043     QFont policeBouton("", 13, 75, false);
00044     QFont policeText("", 13, 0, false);
00045
00046     nomCampagne->setFixedSize(194,30);
00047     nomCampagne->setFont(policeText);
00048     nomCampagne->setStyleSheet("QLineEdit {border-image: url(design/QLine_194x30.png)}" "
      QLineEdit:hover {border-image: url(design/QLine_194x30_survole.png)}");
00049
00050     nomTechnicien->setFixedSize(194,30);
00051     nomTechnicien->setFont(policeText);
00052     nomTechnicien->setStyleSheet("QLineEdit {border-image: url(design/QLine_194x30.png)}" "
      QLineEdit:hover {border-image: url(design/QLine_194x30_survole.png)}" "QLineEdit:disable {border-image:
       url(design/QLine_194x30_grise.png)}");
00053
00054     prenomTechnicien->setFixedSize(194,30);
00055     prenomTechnicien->setFont(policeText);
00056     prenomTechnicien->setStyleSheet("QLineEdit {border-image: url(design/QLine_194x30.png)}
      " "QLineEdit:hover {border-image: url(design/QLine_194x30_survole.png)}");
00057
00058     lieu->setFixedSize(194,30);
00059     lieu->setFont(policeText);
00060     lieu->setStyleSheet("QLineEdit {border-image: url(design/QLine_194x30.png)}" "QLineEdit:hover
       {border-image: url(design/QLine_194x30_survole.png)}");
00061
00062     cheminSauvegarde->setFixedSize(108,30);
00063     cheminSauvegarde->setFont(policeText);
00064     cheminSauvegarde->setStyleSheet("QLineEdit {border-image: url(design/QLine_108x30.png)}
      " "QLineEdit:hover {border-image: url(design/QLine_108x30_survole.png)}" );
00065
00066     boutonValider->setFixedSize(194,40);
00067     boutonValider->setFont(policeBouton);
00068     boutonValider->setStyleSheet("QPushButton {border-image: url(design/bouton_194x40.png)}" "
      QPushButton:hover {border-image: url(design/bouton_194x40_survole.png)}");
00069
00070     boutonAnnuler->setFixedSize(194,40);
00071     boutonAnnuler->setFont(policeBouton);
00072     boutonAnnuler->setStyleSheet("QPushButton {border-image: url(design/bouton_194x40.png)}" "
```

```
    QPushButton:hover {border-image: url(design/bouton_194x40_survole.png)}");
00073
00074     boutonChoixChemin->setFixedSize(80,30);
00075     boutonChoixChemin->setFont(policeText);
00076     boutonChoixChemin->setStyleSheet("QPushButton {border-image:
     url(design/bouton_80x30.png)}" "QPushButton:hover {border-image: url(design/bouton_80x30_survole.png)}");
00077
00078     techniciens->setMinimumSize(194,30);
00079     techniciens->setFont(policeText);
00080     techniciens->setStyleSheet("QComboBox {border-image: url(design/combobox_194x30.png)}" "
    QComboBox:hover {border-image: url(design/combobox_194x30_survole.png)}" "QComboBox::drop-down {border-image:
     url(rien.png)}" "QComboBox {padding: 0 0 0 10px}");
00081
00082     qDebug() << "nomCampagne" << nomCampagne->size();
00083
00084 }
00085
00086 void IHMCreationCampagne::initialiserEvenements()
00087 {
00088     connect(boutonValider, SIGNAL(clicked()), this, SLOT(
    validerCampagne()));
00089     connect(boutonAnnuler, SIGNAL(clicked()), this, SLOT(close()));
00090     connect(boutonChoixChemin, SIGNAL(clicked()), this, SLOT(
    choixCheminSauvegarde()));
00091     connect(techniciens, SIGNAL(currentIndexChanged(int)), this, SLOT(
    modifierEtatBouton(int)));
00092 }
00093
00094 void IHMCreationCampagne::initialiserLayouts()
00095 {
00096     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00097     QFormLayout *layoutFormulaireCampagne = new QFormLayout;
00098     QHBoxLayout *layoutValidation = new QHBoxLayout;
00099     QHBoxLayout *layoutChoixChemin = new QHBoxLayout;
00100
00101     layoutPrincipal->addLayout(layoutFormulaireCampagne);
00102     layoutPrincipal->addLayout(layoutValidation);
00103     layoutChoixChemin->addWidget(cheminSauvegarde);
00104     layoutChoixChemin->addWidget(boutonChoixChemin);
00105
00106     layoutFormulaireCampagne->addRow("Nom campagne : ", nomCampagne);
00107     layoutFormulaireCampagne->addRow("Techniciens : ", techniciens);
00108     layoutFormulaireCampagne->addRow("Nom technicien : ", nomTechnicien);
00109     layoutFormulaireCampagne->addRow("Prenom technicien : ", prenomTechnicien);
00110     layoutFormulaireCampagne->addRow("Lieu campagne : ", lieu);
00111     layoutFormulaireCampagne->addRow("Chemin sauvegarde photos : ", layoutChoixChemin);
00112
00113     layoutValidation->addWidget(boutonValider);
00114     layoutValidation->addWidget(boutonAnnuler);
00115
00116     layoutValidation->setAlignment(Qt::AlignBottom);
00117
00118     setLayout(layoutPrincipal);
00119 }
00120
00121 void IHMCreationCampagne::configurerWidgets()
00122 {
00123     nomCampagne->setTextMargins(10,0,0,0);
00124     nomTechnicien->setTextMargins(10,0,0,0);
00125     prenomTechnicien->setTextMargins(10,0,0,0);
00126     lieu->setTextMargins(10,0,0,0);
00127     cheminSauvegarde->setTextMargins(10,0,0,0);
00128 }
00129
00130 void IHMCreationCampagne::chargerListeTechniciens()
00131 {
00132     techniciens->addItem("Ajouter technicien");
00133
00134     for(QVector<QStringList>::iterator it = listeTechniciens.begin(); it !=
    listeTechniciens.end(); ++it)
00135     {
00136         techniciens->addItem((*it).at(0) + " " + (*it).at(1));
00137     }
00138 }
00139
00140 void IHMCreationCampagne::validerCampagne()
00141 {
00142     if(nomCampagne->text().isEmpty() || nomTechnicien->text().isEmpty() ||
    prenomTechnicien->text().isEmpty() || lieu->text().isEmpty() ||
    cheminSauvegarde->text().isEmpty())
00143     {
00144         QMessageBox::critical(nullptr, "Création campagne", "Informations incomplètes !");
00145         return;
00146     }
00147
00148     Campagne *campagne = new Campagne(nomCampagne->text(),
    lieu->text(), nomTechnicien->text(), prenomTechnicien->text(),
    QDateTime::currentDateTime(), this);
00149     campagne->setCheminSauvegarde(cheminSauvegarde->text());
00150     ihmAccueil->ajouterCampagne(campagne, true);
00151     ihmAccueil->enregistrerCampagneBDD(campagne);
```

```
00152     close();
00153 }
00154
00155 void IHMCreationCampagne::modifierEtatBouton(int index)
00156 {
00157     if(index > 0)
00158     {
00159         nomTechnicien->setDisabled(true);
00160         nomTechnicien->setText(listeTechniciens[index - 1].at(0));
00161         nomTechnicien->setStyleSheet("QLineEdit {border-image:
       url(design/QLine_194x30_grise.png)}");
00162         prenomTechnicien->setDisabled(true);
00163         prenomTechnicien->setText(listeTechniciens[index - 1].at(1));
00164         prenomTechnicien->setStyleSheet("QLineEdit {border-image:
       url(design/QLine_194x30_grise.png)}");
00165     }
00166     else
00167     {
00168         nomTechnicien->clear();
00169         prenomTechnicien->clear();
00170         nomTechnicien->setEnabled(true);
00171         nomTechnicien->setStyleSheet("QLineEdit {border-image: url(design/QLine_194x30.png)}")
      ;
00172         prenomTechnicien->setEnabled(true);
00173         prenomTechnicien->setStyleSheet("QLineEdit {border-image:
       url(design/QLine_194x30.png)}");
00174     }
00175 }
00176
00177 void IHMCreationCampagne::choixCheminSauvegarde()
00178 {
00179     cheminSauvegarde->setText(QFileDialog::getExistingDirectory(this, "Choix du chemin de
       sauvegarde des photos"));
00180 }
```

7.37 Référence du fichier ihmcreationcampagne.h

Fichier qui contient la déclaration de la classe IHMCreationCampagne.

```
#include <QtWidgets>
```

Classes

— class IHMCreationCampagne
Class permettant de créer une nouvelle campagne.

Macros

— #define NOM_FENETRE_CREATIONCAMPAGNE "Projet Rovnet 2020"

7.37.1 Description détaillée

Fichier qui contient la déclaration de la classe IHMCreationCampagne.

Définition dans le fichier ihmcreationcampagne.h.

7.37.2 Documentation des macros

7.37.2.1 NOM_FENETRE_CREATIONCAMPAGNE

```
#define NOM_FENETRE_CREATIONCAMPAGNE "Projet Rovnet 2020"
```

Définition à la ligne 12 du fichier ihmcreationcampagne.h.

7.38 ihmcreationcampagne.h

```
00001
00007 #ifndef IHMCREATIONCAMPAGNE_H
00008 #define IHMCREATIONCAMPAGNE_H
00009
00010 #include <QtWidgets>
00011
00012 #define NOM_FENETRE_CREATIONCAMPAGNE "Projet Rovnet 2020"
00013
00014 class IHMAccueil;
00015 class BaseDeDonnees;
00016
00022 class IHMCreationCampagne : public QDialog
00023 {
00024     Q_OBJECT
00025 private:
00026     IHMAccueil *ihmAccueil;
00027     QLineEdit *nomCampagne;
00028     QLineEdit *nomTechnicien;
00029     QLineEdit *prenomTechnicien;
00030     QLineEdit *lieu;
00031     QLineEdit *cheminSauvegarde;
00032     QPushButton *boutonValider;
00033     QPushButton *boutonAnnuler;
00034     QPushButton *boutonChoixChemin;
00035     QComboBox *techniciens;
00036     QVector<QStringList> listeTechniciens;
00037
00042     void initialisationWidgets();
00047     void initialisationDesignWidgets();
00052     void initialiserEvenements();
00057     void initialiserLayouts();
00062     void configurerWidgets();
00067     void chargerListeTechniciens();
00068
00069 public:
00076     explicit IHMCreationCampagne(IHMAccueil *ihmAccueil, QVector<QStringList>
      &listeTechniciens);
00081     ~IHMCreationCampagne();
00082
00083 signals:
00084
00085 public slots:
00090     void validerCampagne();
00096     void modifierEtatBouton(int index);
00101     void choixCheminSauvegarde();
00102 };
00103
00104 #endif // IHMCREATIONCAMPAGNE_H
```

7.39 Référence du fichier ihmgraphiques.cpp

Fichier qui contient la définition de la classe IHMGraphiques.

```
#include "ihmgraphiques.h"
```

7.39.1 Description détaillée

Fichier qui contient la définition de la classe IHMGraphiques.

Définition dans le fichier ihmgraphiques.cpp.

7.40 ihmgraphiques.cpp

```
00001
00007 #include "ihmgraphiques.h"
00008
00009 IHMGraphiques::IHMGraphiques(QVector<QStringList> mesures,
      QWidget *parent) : QWidget(parent), mesures(mesures), valeurMaxRadiation(0.0),
      valeurMaxTemperature(0.0), valeurMaxHumidite(0.0)
00010 {
00011     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00012     QHBoxLayout *layoutGrapheRadiation = new QHBoxLayout;
00013     QHBoxLayout *layoutGrapheHumidite = new QHBoxLayout;
00014     QHBoxLayout *layoutGrapheTemperature = new QHBoxLayout;
00015
00016     initialisationGraphiqueRadiation();
00017     initialisationGraphiqueHumidite();
00018     initialisationGraphiqueTemperature();
00019
00020     resize(640, 480);
00021     setStyleSheet("background:#202020");
00022
00023     layoutPrincipal->addLayout(layoutGrapheRadiation);
00024     layoutPrincipal->addLayout(layoutGrapheHumidite);
00025     layoutPrincipal->addLayout(layoutGrapheTemperature);
00026     layoutGrapheRadiation->addWidget(graphiqueRadiation);
00027     layoutGrapheHumidite->addWidget(graphiqueHumidite);
00028     layoutGrapheTemperature->addWidget(graphiqueTemperature);
00029     setLayout(layoutPrincipal);
00030     showMaximized();
00031 }
00032
00033 void IHMGraphiques::initialisationGraphiqueRadiation()
00034 {
00035     courbeRadiation = new QLineSeries();
00036     doseLimiteRadiation = new QLineSeries();
00037
00038     for(QVector<QStringList>::iterator it = mesures.begin(); it !=
      mesures.end(); it++)
00039     {
00040         courbeRadiation->append(QDateTime::fromString((*it).at(0)).toMSecsSinceEpoch(),
      (*it).at(1).toFloat());
00041         if((*it).at(1).toFloat() > valeurMaxRadiation)
00042             valeurMaxRadiation = (*it).at(1).toFloat();
00043     }
00044
00045     doseLimiteRadiation->append(QDateTime::fromString(mesures[0].at(0)).
      toMSecsSinceEpoch(), 0.1);
00046     doseLimiteRadiation->append(QDateTime::fromString(mesures[
      mesures.size() - 1].at(0)).toMSecsSinceEpoch(), 0.1);
00047
00048     courbeRadiation->setName(QString::fromUtf8("<font color=\"#FFFFFF\">Radiation</font>"));
00049     doseLimiteRadiation->setName(QString::fromUtf8("<font color=\"#FFFFFF\">Dose
       limite</font>"));
00050     QPen pen;
00051     pen.setColor(QColor(Qt::darkGreen));
00052     pen.setWidth(2);
00053     courbeRadiation->setPen(pen);
00054     pen.setColor(QColor(Qt::darkRed));
00055     pen.setStyle(Qt::DashLine);
00056     doseLimiteRadiation->setPen(pen);
00057
00058     grapheRadiation = new QChart();
00059     grapheRadiation->setBackgroundVisible(false);
00060     grapheRadiation->addSeries(courbeRadiation);
00061     grapheRadiation->addSeries(doseLimiteRadiation);
00062     grapheRadiation->setBackgroundBrush(QColor(0xFFFFFF));
00063
00064     QDateTimeAxis *axeXRadiation = new QDateTimeAxis;
00065     axeXRadiation->setTickCount(10);
00066     axeXRadiation->setFormat("hh:mm:ss");
00067     axeXRadiation->setTitleText("Heure");
00068     axeXRadiation->setLabelsColor(0xFFFFFF);
00069     axeXRadiation->setTitleBrush(QColor(0xFFFFFF));
00070     axeXRadiation->setMin(QDateTime::fromString(mesures[0].at(0)));
00071     axeXRadiation->setMax(QDateTime::fromString(mesures[mesures.size()-1].at(0)));
00072     grapheRadiation->addAxis(axeXRadiation, Qt::AlignBottom);
00073     courbeRadiation->attachAxis(axeXRadiation);
00074     doseLimiteRadiation->attachAxis(axeXRadiation);
00075
00076     QValueAxis *axeYRadiation = new QValueAxis;
00077     axeYRadiation->setRange(0, int(valeurMaxRadiation) + 0.2);
00078     axeYRadiation->setLabelFormat("%.2f");
00079     axeYRadiation->setTitleText(QString::fromUtf8("Radiation en µsV/h"));
00080     axeYRadiation->setLabelsColor(0xFFFFFF);
00081     axeYRadiation->setTitleBrush(QColor(0xFFFFFF));
00082     grapheRadiation->addAxis(axeYRadiation, Qt::AlignLeft);
00083
00084     doseLimiteRadiation->setPointsVisible(true);
00085     doseLimiteRadiation->setPointLabelsFormat("@yPoint"
      ESPACE_LISIBILITE);
```

```
00086     doseLimiteRadiation->setPointLabelsVisible(true);
00087     doseLimiteRadiation->attachAxis(axeYRadiation);
00088 
00089     courbeRadiation->setPointsVisible(true);
00090     courbeRadiation->setPointLabelsFormat("@yPoint"
      ESPACE_LISIBILITE);
00091     courbeRadiation->setPointLabelsVisible(true);
00092     courbeRadiation->attachAxis(axeYRadiation);
00093 
00094     courbeRadiation->setPointLabelsColor(0xFFFFFF);
00095     doseLimiteRadiation->setPointLabelsColor(0xFFFFFF);
00096 
00097     graphiqueRadiation = new QChartView(grapheRadiation);
00098     graphiqueRadiation->setRenderHint(QPainter::Antialiasing);
00099 }
00100 
00101 void IHMGraphiques::initialisationGraphiqueHumidite()
00102 {
00103     courbeHumidite = new QLineSeries();
00104 
00105     for(QVector<QStringList>::iterator it = mesures.begin(); it !=
      mesures.end(); it++)
00106     {
00107         courbeHumidite->append(QDateTime::fromString((*it).at(0)).toMSecsSinceEpoch(), (*it).
      at(3).toFloat());
00108         if((*it).at(1).toFloat() > valeurMaxHumidite)
00109             valeurMaxHumidite = (*it).at(3).toFloat();
00110     }
00111 
00112     courbeHumidite->setName(QString::fromUtf8("<font color=\"#FFFFFF\">Humidité</font>"));
00113     QPen pen;
00114     pen.setColor(QColor(Qt::darkGreen));
00115     pen.setWidth(2);
00116     courbeHumidite->setPen(pen);
00117 
00118     grapheHumidite = new QChart();
00119     grapheHumidite->setBackgroundVisible(false);
00120     grapheHumidite->addSeries(courbeHumidite);
00121     grapheHumidite->setBackgroundBrush(QColor(0xFFFFFF));
00122 
00123     QDateTimeAxis *axeXHumidite = new QDateTimeAxis;
00124     axeXHumidite->setTickCount(10);
00125     axeXHumidite->setFormat("hh:mm:ss");
00126     axeXHumidite->setTitleText("Heure");
00127     axeXHumidite->setLabelsColor(0xFFFFFF);
00128     axeXHumidite->setTitleBrush(QColor(0xFFFFFF));
00129     axeXHumidite->setMin(QDateTime::fromString(mesures[0].at(0)));
00130     axeXHumidite->setMax(QDateTime::fromString(mesures[mesures.size()-1].at(0)));
00131     grapheHumidite->addAxis(axeXHumidite, Qt::AlignBottom);
00132     courbeHumidite->attachAxis(axeXHumidite);
00133 
00134     QValueAxis *axeYHumidite = new QValueAxis;
00135     axeYHumidite->setRange(0, 100);
00136     axeYHumidite->setLabelFormat("%d");
00137     axeYHumidite->setTitleText(QString::fromUtf8("Humidité en %"));
00138     axeYHumidite->setLabelsColor(0xFFFFFF);
00139     axeYHumidite->setTitleBrush(QColor(0xFFFFFF));
00140     grapheHumidite->addAxis(axeYHumidite, Qt::AlignLeft);
00141     courbeHumidite->setPointsVisible(true);
00142     courbeHumidite->setPointLabelsFormat("@yPoint"
      ESPACE_LISIBILITE);
00143     courbeHumidite->setPointLabelsVisible(true);
00144     courbeHumidite->attachAxis(axeYHumidite);
00145 
00146     courbeHumidite->setPointLabelsColor(0xFFFFFF);
00147 
00148     graphiqueHumidite = new QChartView(grapheHumidite);
00149     graphiqueHumidite->setRenderHint(QPainter::Antialiasing);
00150 }
00151 
00152 void IHMGraphiques::initialisationGraphiqueTemperature()
00153 {
00154     courbeTemperature = new QLineSeries();
00155 
00156     for(QVector<QStringList>::iterator it = mesures.begin(); it !=
      mesures.end(); it++)
00157     {
00158         courbeTemperature->append(QDateTime::fromString((*it).at(0)).toMSecsSinceEpoch(),
      (*it).at(2).toFloat());
00159         if((*it).at(2).toFloat() > valeurMaxTemperature)
00160             valeurMaxTemperature = (*it).at(2).toFloat();
00161     }
00162 
00163     courbeTemperature->setName(QString::fromUtf8("<font color=\"#FFFFFF\"
      >Température</font>"));
00164     QPen pen;
00165     pen.setColor(QColor(Qt::darkGreen));
00166     pen.setWidth(2);
00167     courbeTemperature->setPen(pen);
00168 
00169     grapheTemperature = new QChart();
```

```
00170     grapheTemperature->setBackgroundVisible(false);
00171     grapheTemperature->addSeries(courbeTemperature);
00172     grapheTemperature->setBackgroundBrush(QColor(0xFFFFFF));
00173
00174     QDateTimeAxis *axeXTemperature = new QDateTimeAxis;
00175     axeXTemperature->setTickCount(10);
00176     axeXTemperature->setFormat("hh:mm:ss");
00177     axeXTemperature->setTitleText("Heure");
00178     axeXTemperature->setLabelsColor(0xFFFFFF);
00179     axeXTemperature->setTitleBrush(QColor(0xFFFFFF));
00180     axeXTemperature->setMin(QDateTime::fromString(mesures[0].at(0)));
00181     axeXTemperature->setMax(QDateTime::fromString(mesures[mesures.size()-1].at(0)));
00182     grapheTemperature->addAxis(axeXTemperature, Qt::AlignBottom);
00183     courbeTemperature->attachAxis(axeXTemperature);
00184
00185     QValueAxis *axeYTemperature = new QValueAxis;
00186     axeYTemperature->setRange(0, int(valeurMaxTemperature) + 5);
00187     axeYTemperature->setLabelFormat("%.1f");
00188     axeYTemperature->setTitleText(QString::fromUtf8("Temperature en °C"));
00189     axeYTemperature->setLabelsColor(0xFFFFFF);
00190     axeYTemperature->setTitleBrush(QColor(0xFFFFFF));
00191     grapheTemperature->addAxis(axeYTemperature, Qt::AlignLeft);
00192     courbeTemperature->setPointsVisible(true);
00193     courbeTemperature->setPointLabelsFormat("@yPoint"
    ESPACE_LISIBILITE);
00194     courbeTemperature->setPointLabelsVisible(true);
00195     courbeTemperature->attachAxis(axeYTemperature);
00196
00197     courbeTemperature->setPointLabelsColor(0xFFFFFF);
00198
00199     graphiqueTemperature = new QChartView(grapheTemperature);
00200     graphiqueTemperature->setRenderHint(QPainter::Antialiasing);
00201 }
```

7.41 Référence du fichier ihmgraphiques.h

Fichier qui contient la déclaration de la classe IHMGraphiques.

```
#include <QtWidgets>
#include <QtCharts>
```

Classes

— class IHMGraphiques
Class permettant de visualiser les graphiques des campagnes archivés.

Macros

— #define ESPACE_LISIBILITE " "

7.41.1 Description détaillée

Fichier qui contient la déclaration de la classe IHMGraphiques.

Définition dans le fichier ihmgraphiques.h.

7.41.2 Documentation des macros

7.41.2.1 ESPACE_LISIBILITE

```
#define ESPACE_LISIBILITE " "
```

Définition à la ligne 13 du fichier ihmgraphiques.h.

Référencé par IHMGraphiques : :initialisationGraphiqueHumidite(), IHMGraphiques : :initialisationGraphiqueRadiation(), et IHM↩ Graphiques : :initialisationGraphiqueTemperature().

7.42 ihmgraphiques.h

```
00001
00007 #ifndef IHMGRAPHIQUES_H
00008 #define IHMGRAPHIQUES_H
00009
00010 #include <QtWidgets>
00011 #include <QtCharts>
00012
00013 #define ESPACE_LISIBILITE "              "
00014
00020 class IHMGraphiques : public QWidget
00021 {
00022     Q_OBJECT
00023 private:
00024     QVector<QStringList> mesures;
00025     QChartView *graphiqueRadiation;
00026     QChart *grapheRadiation;
00027     QLineSeries *courbeRadiation;
00028     QLineSeries *doseLimiteRadiation;
00029
00030     QChartView *graphiqueHumidite;
00031     QChart *grapheHumidite;
00032     QLineSeries *courbeHumidite;
00033
00034     QChartView *graphiqueTemperature;
00035     QChart *grapheTemperature;
00036     QLineSeries *courbeTemperature;
00037
00038     float valeurMaxRadiation;
00039     float valeurMaxTemperature;
00040     float valeurMaxHumidite;
00041
00042     void initialisationGraphiqueRadiation();
00043     void initialisationGraphiqueHumidite();
00044     void initialisationGraphiqueTemperature();
00045
00046 public:
00053     IHMGraphiques(QVector<QStringList> mesures, QWidget *parent = nullptr);
00054
00055 signals:
00056
00057 };
00058
00059 #endif // IHMGRAPHIQUES_H
```

7.43 Référence du fichier ihmreglagevideo.cpp

Fichier qui contient la définition de la classe ReglageVideo.

```
#include "ihmreglagevideo.h"
```

7.43.1 Description détaillée

Fichier qui contient la définition de la classe ReglageVideo.

Définition dans le fichier ihmreglagevideo.cpp.

7.44 ihmreglagevideo.cpp

```
00001
00007 #include "ihmreglagevideo.h"
00008
00009 IHMReglageVideo::IHMReglageVideo(Rov *rov,
      QWidget *parent) : QWidget(parent), rov(rov)
00010 {
00011     qDebug() << Q_FUNC_INFO;
00012     initialiserWidgets();
00013     configurerWidgets();
00014     initialiserLayouts();
00015     initialiserEvenements();
00016 }
00017
00018 IHMReglageVideo::~IHMReglageVideo()
00019 {
00020     qDebug() << Q_FUNC_INFO;
00021 }
00022
00023 void IHMReglageVideo::initialiserWidgets()
00024 {
00025     sliderLuminositeVideo = new QSlider(Qt::Horizontal, this);
00026     sliderContrasteVideo = new QSlider(Qt::Horizontal, this);
00027     sliderSaturationVideo = new QSlider(Qt::Horizontal, this);
00028
00029     luminosite = new QLabel("Luminosité", this);
00030     contraste = new QLabel("Contraste", this);
00031     saturation = new QLabel("Saturation", this);
00032
00033     luminositeVideo = new QSpinBox(this);
00034     contrasteVideo = new QSpinBox(this);
00035     saturationVideo = new QSpinBox(this);
00036
00037     camera = new QLabel("Camera(s): ", this);
00038     resolution = new QLabel("Résolution: ", this);
00039
00040     listeCameras = new QComboBox(this);
00041     listeResolutions = new QComboBox(this);
00042
00043     boutonVideo = new QPushButton("Arrêter", this);
00044 }
00045
00046 void IHMReglageVideo::configurerWidgets()
00047 {
00048     sliderLuminositeVideo->setValue(50);
00049     sliderContrasteVideo->setValue(50);
00050     sliderSaturationVideo->setValue(50);
00051
00052     luminositeVideo->setValue(50);
00053     contrasteVideo->setValue(50);
00054     saturationVideo->setValue(50);
00055
00056     listeCameras->setEnabled(false);
00057 }
00058
00059 void IHMReglageVideo::initialiserLayouts()
00060 {
00061     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00062     QHBoxLayout *layoutReglageVideo = new QHBoxLayout;
00063     QVBoxLayout *layoutConfigurationLuminosite = new QVBoxLayout;
00064     QVBoxLayout *layoutConfigurationContraste = new QVBoxLayout;
00065     QVBoxLayout *layoutConfigurationSaturation = new QVBoxLayout;
00066     QHBoxLayout *layoutCamera = new QHBoxLayout;
00067     QHBoxLayout *layoutBoutonCamera = new QHBoxLayout;
00068
00069     layoutReglageVideo->setAlignment(Qt::AlignLeft);
00070     layoutConfigurationContraste->setAlignment(Qt::AlignTop);
00071     layoutConfigurationLuminosite->setAlignment(Qt::AlignTop);
00072     layoutConfigurationSaturation->setAlignment(Qt::AlignTop);
00073     layoutBoutonCamera->setAlignment(Qt::AlignLeft);
00074
00075     layoutPrincipal->addLayout(layoutReglageVideo);
00076     layoutPrincipal->addLayout(layoutCamera);
00077     layoutPrincipal->addLayout(layoutBoutonCamera);
00078
00079     layoutReglageVideo->addLayout(layoutConfigurationLuminosite);
00080     layoutReglageVideo->addLayout(layoutConfigurationContraste);
00081     layoutReglageVideo->addLayout(layoutConfigurationSaturation);
00082
00083     layoutConfigurationLuminosite->addWidget(luminosite);
00084     layoutConfigurationLuminosite->addWidget(sliderLuminositeVideo);
00085     layoutConfigurationLuminosite->addWidget(luminositeVideo);
00086     layoutConfigurationContraste->addWidget(contraste);
00087     layoutConfigurationContraste->addWidget(sliderContrasteVideo);
00088     layoutConfigurationContraste->addWidget(contrasteVideo);
00089     layoutConfigurationSaturation->addWidget(saturation);
00090     layoutConfigurationSaturation->addWidget(sliderSaturationVideo);
00091     layoutConfigurationSaturation->addWidget(saturationVideo);
00092     layoutCamera->addWidget(camera);
```

```
00093     layoutCamera->addWidget(listeCameras);
00094     layoutCamera->addWidget(resolution);
00095     layoutCamera->addWidget(listeResolutions);
00096     layoutBoutonCamera->addWidget(boutonVideo);
00097
00098     setLayout(layoutPrincipal);
00099     setWindowTitle(NOM_FENETRE_REGLAGEVIDEO);
00100     setStyleSheet("background:#202020;color:white;");
00101 }
00102
00103 void IHMReglageVideo::initialiserEvenements()
00104 {
00105     connect(sliderLuminositeVideo, SIGNAL(valueChanged(int)),
      luminositeVideo, SLOT(setValue(int)));
00106     connect(sliderContrasteVideo, SIGNAL(valueChanged(int)),
      contrasteVideo, SLOT(setValue(int)));
00107     connect(sliderSaturationVideo, SIGNAL(valueChanged(int)),
      saturationVideo, SLOT(setValue(int)));
00108     connect(luminositeVideo, SIGNAL(valueChanged(int)),
      sliderLuminositeVideo, SLOT(setValue(int)));
00109     connect(contrasteVideo, SIGNAL(valueChanged(int)),
      sliderContrasteVideo, SLOT(setValue(int)));
00110     connect(saturationVideo, SIGNAL(valueChanged(int)),
      sliderSaturationVideo, SLOT(setValue(int)));
00111     connect(boutonVideo, SIGNAL(clicked()), this, SLOT(gererVideo()));
00112 }
00113
00114 void IHMReglageVideo::gererVideo()
00115 {
00116     if(boutonVideo->text() == "Arrêter")
00117     {
00118         rov->arreterVideo();
00119         boutonVideo->setText("Démarrer");
00120     }
00121     else
00122     {
00123         activerCamera();
00124         boutonVideo->setText("Arrêter");
00125     }
00126 }
00127
00128 void IHMReglageVideo::initialiserEvenementsCamera()
00129 {
00130     if(rov->getCamera() != nullptr)
00131     {
00132         if(rov->getCamera()->isRunning())
00133             return;
00134         chargerListeCameraDisponible();
00135         connect(luminositeVideo, SIGNAL(valueChanged(int)), rov->
      getCamera(), SLOT(setLuminosite(int)));
00136         connect(contrasteVideo, SIGNAL(valueChanged(int)), rov->
      getCamera(), SLOT(setContraste(int)));
00137         connect(saturationVideo, SIGNAL(valueChanged(int)), rov->
      getCamera(), SLOT(setSaturation(int)));
00138         connect(listeCameras, SIGNAL(currentIndexChanged(int)), this, SLOT(
      chargerListeResolutionDisponible(int)));
00139         connect(listeResolutions, SIGNAL(currentIndexChanged(int)),
      rov->getCamera(), SLOT(setResolution(int)));
00140     }
00141 }
00142
00143 void IHMReglageVideo::chargerListeCameraDisponible()
00144 {
00145     int nbCameras = QCameraInfo::availableCameras().count();
00146     qDebug() << Q_FUNC_INFO << "Caméra(s) disponible(s)" << QCameraInfo::availableCameras().count();
00147     if (nbCameras > 0)
00148     {
00149        #ifndef SANS_DETECTION
00150        QList<QCameraInfo> cameras = QCameraInfo::availableCameras();
00151        listeCameras->clear();
00152        int choix = -1, i = 0;
00153        foreach (const QCameraInfo &cameraInfo, cameras)
00154        {
00155            listeCameras->addItem(cameraInfo.deviceName());
00156            qDebug() << Q_FUNC_INFO << "Device" << cameraInfo.deviceName();
00157            qDebug() << Q_FUNC_INFO << "Description" << cameraInfo.description();
00158            if(rov->getCamera() != nullptr)
00159            {
00160                if(cameraInfo.deviceName() == rov->getCamera()->
      getNom())
00161                {
00162                    choix = i;
00163                }
00164            }
00165            i++;
00166        }
00167        #else
00168        int choix = 0;
00169        qDebug() << Q_FUNC_INFO << "Device" << rov->getCamera()->
      getNom();
00170        listeCameras->addItem(rov->getCamera()->getNom());
```

```
00171         #endif
00172
00173         if(choix != -1)
00174         {
00175             if(rov->getCamera() != nullptr)
00176                 chargerListeResolutionDisponible(
      rov->getCamera()->getNom());
00177             listeCameras->setCurrentIndex(choix);
00178         }
00179     }
00180
00181 }
00182
00183 void IHMReglageVideo::chargerListeResolutionDisponible(int
       index)
00184 {
00185     if(index < 0)
00186         return;
00187     chargerListeResolutionDisponible(
      listeCameras->currentText());
00188 }
00189
00190 void IHMReglageVideo::chargerListeResolutionDisponible(
      QString nom)
00191 {
00192     #ifndef SANS_DETECTION
00193     QCameraInfo cameraInfo(nom.toLatin1());
00194     QList<QSize> liste = Camera::lireListeResolutionsCamera(cameraInfo);
00195     listeResolutions->clear();
00196     foreach (const QSize &resolution, liste)
00197     {
00198         qDebug() << Q_FUNC_INFO << resolution.width() << "x" << resolution.height();
00199         listeResolutions->addItem(QString::number(resolution.width()) + QString("x") +
      QString::number(resolution.height()));
00200     }
00201     #else
00202     listeResolutions->clear();
00203     QSize resolutionDefaut = rov->getCamera()->getResolution();
00204     qDebug() << Q_FUNC_INFO << resolutionDefaut.width() << "x" << resolutionDefaut.height();
00205     listeResolutions->addItem(QString::number(resolutionDefaut.width()) + QString("x") +
      QString::number(resolutionDefaut.height()));
00206     #endif
00207     if(rov->getCamera() != nullptr)
00208     {
00209         qDebug() << Q_FUNC_INFO << "choixResolution" << rov->getCamera()->
      getChoixResolution();
00210         listeResolutions->setCurrentIndex(rov->getCamera()->
      getChoixResolution());
00211     }
00212 }
00213
00214 void IHMReglageVideo::activerCamera()
00215 {
00216     qDebug() << Q_FUNC_INFO << listeCameras->currentText() <<
      listeResolutions->currentText() << listeResolutions->currentIndex();
00217     rov->demarrerVideo(listeCameras->currentText(),
      listeResolutions->currentIndex());
00218     listeCameras->setEnabled(false);
00219 }
00220
00221 void IHMReglageVideo::modifierEtatBoutons()
00222 {
00223     if(rov->getCamera() != nullptr)
00224     {
00225         disconnect(luminositeVideo, SIGNAL(valueChanged(int)),
      rov->getCamera(), SLOT(setLuminosite(int)));
00226         disconnect(contrasteVideo, SIGNAL(valueChanged(int)), rov->
      getCamera(), SLOT(setContraste(int)));
00227         disconnect(saturationVideo, SIGNAL(valueChanged(int)),
      rov->getCamera(), SLOT(setSaturation(int)));
00228         disconnect(listeCameras, SIGNAL(currentIndexChanged(int)), this, SLOT(
      chargerListeResolutionDisponible(int)));
00229         disconnect(listeResolutions, SIGNAL(currentIndexChanged(int)),
      rov->getCamera(), SLOT(setResolution(int)));
00230     }
00231     listeCameras->setEnabled(true);
00232 }
```

7.45 Référence du fichier ihmreglagevideo.h

Fichier qui contient la déclaration de la classe IHMReglageVideo.

```
#include <QtWidgets>
#include <QCameraInfo>
#include "rov.h"
```

Classes

— class IHMReglageVideo
Classe permettant de regler l'affichage du flux video.

Macros

— #define NOM_FENETRE_REGLAGEVIDEO "Réglages vidéo"

7.45.1 Description détaillée

Fichier qui contient la déclaration de la classe IHMReglageVideo.

Définition dans le fichier ihmreglagevideo.h.

7.45.2 Documentation des macros

7.45.2.1 NOM_FENETRE_REGLAGEVIDEO

```
#define NOM_FENETRE_REGLAGEVIDEO "Réglages vidéo"
```

Définition à la ligne 14 du fichier ihmreglagevideo.h.

Référencé par IHMReglageVideo : :initialiserLayouts().

7.46 ihmreglagevideo.h

```
00001
00007 #ifndef REGLAGEVIDEO_H
00008 #define REGLAGEVIDEO_H
00009
00010 #include <QtWidgets>
00011 #include <QCameraInfo>
00012 #include "rov.h"
00013
00014 #define NOM_FENETRE_REGLAGEVIDEO "Réglages vidéo"
00015
00016 class Rov;
00017
00023 class IHMReglageVideo : public QWidget
00024 {
00025     Q_OBJECT
00026 private:
00027     Rov *rov;
00028     QSlider *sliderLuminositeVideo;
00029     QSlider *sliderContrasteVideo;
00030     QSlider *sliderSaturationVideo;
00031     QLabel *luminosite;
00032     QLabel *contraste;
00033     QLabel *saturation;
00034     QSpinBox *luminositeVideo;
00035     QSpinBox *contrasteVideo;
00036     QSpinBox *saturationVideo;
00037     QLabel *camera;
00038     QComboBox *listeCameras;
00039     QLabel *resolution;
00040     QComboBox *listeResolutions;
00041     QPushButton *boutonVideo;
00042     QPushButton *boutonArreterVideo;
00043
00048     void initialiserWidgets();
00053     void configurerWidgets();
00058     void initialiserLayouts();
00063     void initialiserEvenements();
```

```
00068     void chargerListeCameraDisponible();
00069
00070 public:
00077     IHMReglageVideo(Rov *rov, QWidget *parent = nullptr);
00082     ~IHMReglageVideo();
00087     void initialiserEvenementsCamera();
00088
00089 signals:
00090
00091 public slots:
00096     void activerCamera();
00101     void modifierEtatBoutons();
00107     void chargerListeResolutionDisponible(int index);
00113     void chargerListeResolutionDisponible(QString nom);
00118     void gererVideo();
00119 };
00120
00121 #endif // REGLAGEVIDEO_H
```

7.47 Référence du fichier ihmrov.cpp

Fichier qui contient la définition de la classe IHMRov.

```
#include "ihmrov.h"
#include "rov.h"
#include "ihmreglagevideo.h"
#include "ihmalbumphoto.h"
#include "ihmaccueil.h"
#include "campagne.h"
#include "ihmconfiguration.h"
#include <QtMath>
```

7.47.1 Description détaillée

Fichier qui contient la définition de la classe IHMRov.

Définition dans le fichier ihmrov.cpp.

7.48 ihmrov.cpp

```
00001
00007 #include "ihmrov.h"
00008 #include "rov.h"
00009 #include "ihmreglagevideo.h"
00010 #include "ihmalbumphoto.h"
00011 #include "ihmaccueil.h"
00012 #include "campagne.h"
00013 #include "ihmconfiguration.h"
00014 #include <QtMath>
00015
00016 IHMRov::IHMRov(IHMAccueil *ihmAccueil, QWidget *parent) :
      QWidget(parent), campagneEnCours(nullptr), ihmAccueil(ihmAccueil), etatRadar(true)
00017 {
00018     qDebug() << Q_FUNC_INFO << this << "width" << qApp->desktop()->screen()->width() << "height" << qApp->
      desktop()->screen()->height();
00019     rov = new Rov(this);
00020     reglageVideo = nullptr;
00021     configuration = nullptr;
00022
00023     initialiserWidgets();
00024     configurerWidgets();
00025     initialiserLayouts();
00026     initialiserEvenements();
00027
00028     if(!rov->getManettes().isEmpty())
00029         modifieEtatManette(true);
00030 }
00031
00032 IHMRov::~IHMRov()
00033 {
```

```
00034     qDebug() << Q_FUNC_INFO;
00035 }
00036
00037 void IHMRov::initialiserWidgets()
00038 {
00039     fluxVideo = new QLabel("Aucune image détéctée",this);
00040     photosEnCours = new QPushButton("Album Photo", this);
00041     boutonReglageVideo = new QPushButton("Réglages Vidéo", this);
00042     boutonCampagne = new QPushButton(QString::fromUtf8("Démarrer"), this);
00043     boutonConfiguration = new QPushButton("Communication", this);
00044     hautDePage = new QLabel(this);
00045     basDePage = new QLabel(this);
00046     logoEtatPortSerie = new QLabel(this);
00047     logoEtatCamera = new QLabel(this);
00048     logoEtatManette = new QLabel(this);
00049     etatPortSerie = new QLabel(this);
00050     etatCamera = new QLabel(this);
00051     etatManette = new QLabel(this);
00052     portSerie = new QLabel("Port série :", this);
00053     camera = new QLabel("Caméra :", this);
00054     manette = new QLabel("Manette :", this);
00055     indicateurTemperature = new QwtThermo(this);
00056     indicateurRadiation = new QwtThermo(this);
00057     zoneEtatMateriel = new QGroupBox(this);
00058     zoneInformationSeuils = new QGroupBox(this);
00059     temperature = new QLabel("Température\n", this);
00060     radiation = new QLabel("Radiation\n", this);
00061
00062     QFont police1("", 15, 75, false);
00063     QFont police2("Cursive", 12, 40, true);
00064
00065     portSerie->setFont(police1);
00066     camera->setFont(police1);
00067     manette->setFont(police1);
00068     etatPortSerie->setFont(police2);
00069     etatCamera->setFont(police2);
00070     etatManette->setFont(police2);
00071     temperature->setFont(police2);
00072     radiation->setFont(police2);
00073
00074     fluxVideo->setFixedSize(230,50);
00075     fluxVideo->setFont(police1);
00076
00077     photosEnCours->setFixedSize(230,50);
00078     photosEnCours->setFont(police1);
00079     photosEnCours->setStyleSheet("QPushButton {border-image: url(design/bouton_230x50.png)}" "
      QPushButton:hover {border-image: url(design/bouton_230x50_survole.png)}");
00080     //photosEnCours->setStyleSheet("QPushButton {border-image: url(design/bouton_230x50_survole.png)}"
       "QPushButton:hover {border-image: url(design/bouton_230x50.png)}");
00081
00082     boutonReglageVideo->setFixedSize(230,50);
00083     boutonReglageVideo->setFont(police1);
00084     boutonReglageVideo->setStyleSheet("QPushButton {border-image:
       url(design/bouton_230x50.png)}" "QPushButton:hover {border-image: url(design/bouton_230x50_survole.png)}");
00085     //boutonReglageVideo->setStyleSheet("QPushButton {border-image: url(design/bouton_230x50_survole.png)}"
       "QPushButton:hover {border-image: url(design/bouton_230x50.png)}");
00086
00087     boutonCampagne->setFixedSize(230,50);
00088     boutonCampagne->setFont(police1);
00089     boutonCampagne->setStyleSheet("QPushButton {border-image: url(design/bouton_230x50.png)}"
       "QPushButton:hover {border-image: url(design/bouton_230x50_survole.png)}");
00090     //boutonCampagne->setStyleSheet("QPushButton {border-image: url(design/bouton_230x50_survole.png)}"
       "QPushButton:hover {border-image: url(design/bouton_230x50.png)}");
00091
00092     boutonConfiguration->setFixedSize(230,50);
00093     boutonConfiguration->setFont(police1);
00094     boutonConfiguration->setStyleSheet("QPushButton {border-image:
       url(design/bouton_230x50.png)}" "QPushButton:hover {border-image: url(design/bouton_230x50_survole.png)}");
00095     //boutonConfiguration->setStyleSheet("QPushButton {border-image:
       url(design/bouton_230x50_survole.png)}" "QPushButton:hover {border-image: url(design/bouton_230x50.png)}");
00096
00097     hautDePage->setMinimumHeight(1);
00098     //hautDePage->setStyleSheet("QLabel {border-image: url(design/fond_noir.png)}");
00099     basDePage->setMinimumHeight(1);
00100     //basDePage->setStyleSheet("QLabel {border-image: url(design/fond_noir.png)}");
00101
00102     #ifdef PAS_DE_MANETTE
00103     testCapturePhoto = new QPushButton("Capturer", this);
00104     testCapturePhoto->setFixedSize(230,50);
00105     //testCapturePhoto->setFont(police);
00106     testCapturePhoto->setStyleSheet("QPushButton {border-image:
       url(design/bouton_230x50.png)}" "QPushButton:hover {border-image: url(design/bouton_230x50_survole.png)}");
00107     QAction *actionCapturerPhoto = new QAction(this);
00108     actionCapturerPhoto->setShortcut(QKeySequence(Qt::Key_C));
00109     addAction(actionCapturerPhoto);
00110     connect(actionCapturerPhoto, SIGNAL(triggered()), this, SLOT(capturerImage()));
00111     #endif
00112 }
00113
00114 void IHMRov::configurerWidgets()
00115 {
```

```
00116     int width = int(qApp->desktop()->screen()->width() * RATIO);
00117     int height = int(qApp->desktop()->screen()->height() * RATIO);
00118     fluxVideo->setFixedSize(width, height);
00119     fluxVideo->setScaledContents(true);
00120     fluxVideo->setAlignment(Qt::AlignHCenter | Qt::AlignVCenter);
00121     fluxVideo->setPixmap(QPixmap(qApp->applicationDirPath() +  "/images/signal-interrompu.jpg"));
00122
00123     logoEtatPortSerie->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/inactif.png
      ").scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00124     etatPortSerie->setText("Fermé");
00125
00126     logoEtatCamera->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/inactif.png").
      scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00127     etatCamera->setText("Eteinte");
00128
00129     logoEtatManette->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/inactif.png").
      scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00130     etatManette->setText("Déconnectée");
00131
00132     indicateurTemperature->setScale(QwtInterval(
      SEUIL_TEMPERATURE_NEGATIVE_ACCEPTABLE,
      SEUIL_TEMPERATURE_POSITIVE_ACCEPTABLE));
00133     indicateurRadiation->setScale(QwtInterval(0,
      SEUIL_RADIATION_ACCEPTABLE));
00134
00135     indicateurTemperature->setOrigin(0.);
00136     indicateurTemperature->setOriginMode(QwtThermo::OriginMode::OriginCustom);
00137
00138     zoneEtatMateriel->setStyleSheet("QGroupBox {background: white} QGroupBox{border: 2px
       solid black}");
00139     zoneInformationSeuils->setStyleSheet("QGroupBox {background: white}
       QGroupBox{border: 2px solid black}");
00140     portSerie->setStyleSheet("QLabel {background: white}");
00141     camera->setStyleSheet("QLabel {background: white}");
00142     manette->setStyleSheet("QLabel {background: white}");
00143     etatPortSerie->setStyleSheet("QLabel {background: white}");
00144     etatCamera->setStyleSheet("QLabel {background: white}");
00145     etatManette->setStyleSheet("QLabel {background: white}");
00146     logoEtatPortSerie->setStyleSheet("QLabel {background: white}");
00147     logoEtatCamera->setStyleSheet("QLabel {background: white}");
00148     logoEtatManette->setStyleSheet("QLabel {background: white}");
00149     indicateurRadiation->setStyleSheet("QwtThermo {background: white}");
00150     indicateurTemperature->setStyleSheet("QwtThermo {background: white}");
00151     temperature->setStyleSheet("QLabel {background: white}");
00152     radiation->setStyleSheet("QLabel {background: white}");
00153 }
00154
00155 void IHMRov::initialiserLayouts()
00156 {
00157     QVBoxLayout *layoutPrincipal = new QVBoxLayout;
00158     QHBoxLayout *layoutInformationRov = new QHBoxLayout;
00159     QVBoxLayout *layoutCamera = new QVBoxLayout;
00160     QVBoxLayout *layoutOptionVideo = new QVBoxLayout;
00161     QVBoxLayout *layoutGestionCampagne = new QVBoxLayout;
00162     QVBoxLayout *layoutReglageVideo = new QVBoxLayout;
00163     QVBoxLayout *layoutInformationMateriel = new QVBoxLayout;
00164     QHBoxLayout *layoutEtatPortSerie = new QHBoxLayout;
00165     QHBoxLayout *layoutEtatCamera = new QHBoxLayout;
00166     QHBoxLayout *layoutEtatManette = new QHBoxLayout;
00167     QHBoxLayout *layoutInformationSeuils = new QHBoxLayout;
00168     QVBoxLayout *layoutSeuilTemperature = new QVBoxLayout;
00169     QVBoxLayout *layoutSeuilRadiation = new QVBoxLayout;
00170
00171     layoutOptionVideo->setAlignment(Qt::AlignTop);
00172     layoutGestionCampagne->setAlignment(Qt::AlignBottom);
00173     layoutInformationMateriel->setAlignment(Qt::AlignTop);
00174     layoutCamera->addWidget(fluxVideo);
00175
00176     layoutOptionVideo->addWidget(boutonReglageVideo);
00177     layoutOptionVideo->addWidget(boutonConfiguration);
00178     layoutOptionVideo->addWidget(photosEnCours);
00179     #ifdef PAS_DE_MANETTE
00180     layoutOptionVideo->addWidget(testCapturePhoto);
00181     #endif
00182
00183     layoutInformationMateriel->addWidget(portSerie);
00184     layoutInformationMateriel->addLayout(layoutEtatPortSerie);
00185     layoutInformationMateriel->addWidget(camera);
00186     layoutInformationMateriel->addLayout(layoutEtatCamera);
00187     layoutInformationMateriel->addWidget(manette);
00188     layoutInformationMateriel->addLayout(layoutEtatManette);
00189
00190     layoutEtatPortSerie->setAlignment(Qt::AlignLeft);
00191     layoutEtatCamera->setAlignment(Qt::AlignLeft);
00192     layoutEtatManette->setAlignment(Qt::AlignLeft);
00193
00194     layoutEtatPortSerie->addWidget(logoEtatPortSerie);
00195     layoutEtatPortSerie->addWidget(etatPortSerie);
00196
00197     layoutEtatCamera->addWidget(logoEtatCamera);
00198     layoutEtatCamera->addWidget(etatCamera);
```

```
00199
00200     layoutEtatManette->addWidget(logoEtatManette);
00201     layoutEtatManette->addWidget(etatManette);
00202
00203     layoutSeuilTemperature->addWidget(temperature);
00204     layoutSeuilTemperature->addWidget(indicateurTemperature);
00205
00206     layoutSeuilRadiation->addWidget(radiation);
00207     layoutSeuilRadiation->addWidget(indicateurRadiation);
00208
00209     layoutInformationSeuils->addLayout(layoutSeuilTemperature);
00210     layoutInformationSeuils->addLayout(layoutSeuilRadiation);
00211
00212     layoutPrincipal->addWidget(hautDePage);
00213     layoutPrincipal->addLayout(layoutInformationRov);
00214     layoutInformationRov->addLayout(layoutCamera);
00215     layoutInformationRov->addStretch();
00216     layoutInformationRov->addLayout(layoutReglageVideo);
00217     layoutReglageVideo->addLayout(layoutOptionVideo);
00218
00219     //layoutReglageVideo->addLayout(layoutInformationMateriel);
00220     zoneEtatMateriel->setLayout(layoutInformationMateriel);
00221     layoutReglageVideo->addWidget(zoneEtatMateriel);
00222     //layoutReglageVideo->addLayout(layoutInformationSeuils);
00223     zoneInformationSeuils->setLayout(layoutInformationSeuils);
00224     layoutReglageVideo->addWidget(zoneInformationSeuils);
00225
00226     layoutReglageVideo->addLayout(layoutGestionCampagne);
00227     layoutGestionCampagne->addWidget(boutonCampagne);
00228     layoutPrincipal->addWidget(basDePage);
00229
00230     setLayout(layoutPrincipal);
00231     resize(width(), fluxVideo->maximumHeight());
00232     //setStyleSheet("background:#101010;");
00233     setStyleSheet("background:#C1BEBE;");
00234
00235     setWindowFlags(windowFlags() & ~Qt::WindowCloseButtonHint);
00236     showMinimized();
00237 }
00238
00239 void IHMRov::initialiserEvenements()
00240 {
00241     connect(boutonReglageVideo, SIGNAL(clicked()), this, SLOT(
      reglerVideo()));
00242     connect(boutonConfiguration, SIGNAL(clicked()), this, SLOT(
      reglerConfiguration()));
00243     connect(photosEnCours, SIGNAL(clicked()), this, SLOT(
      chargerPhotos()));
00244     #ifdef PAS_DE_MANETTE
00245     connect(testCapturePhoto, SIGNAL(clicked(bool)), this, SLOT(
      capturerImage(bool)));
00246     #endif
00247     connect(boutonCampagne, SIGNAL(clicked()), this, SLOT(
      gererCampagne()));
00248     connect(rov, SIGNAL(enregistrerMesures(QString, QString, QString)),
      ihmAccueil, SLOT(enregisterMesureBDD(QString, QString, QString)));
00249 }
00250
00251 void IHMRov::actualiserInformations(QPixmap &image)
00252 {
00253     QPainter p(&image);
00254     QPen pen;
00255     QPen penCadre;
00256     QFont fontHaut("Open Sans");
00257     QFont fontBas("Open Sans");
00258     int marge = 0;
00259     pen.setWidth(qApp->desktop()->screen()->width() * 0.005);
00260
00261     //QFontDatabase fontDatabase;
00262     //qDebug() << fontDatabase.families();
00263
00264     // règle la taille de police
00265     fontHaut.setPixelSize(image.height()*0.045); // 4.5 % de la hauteur de l'image en pixel
00266     fontBas.setPixelSize(fontHaut.pixelSize()*0.75); // 75% de la hauteur normale
00267
00268     /*
00269     // pour le dessin du bandeau
00270     penCadre.setWidth(1);
00271     penCadre.setBrush(Qt::lightGray);
00272     penCadre.setCapStyle(Qt::RoundCap);
00273     penCadre.setJoinStyle(Qt::RoundJoin);
00274     p.setPen(penCadre);
00275     // dessine un bandeau en haut (hauteur 5% de la hauteur de l'image)
00276     p.drawLine( 0, (image.height()*0.05)+1, image.width(), (image.height()*0.05)+1 );
00277     // dessine un bandeau en bas (hauteur 5% de la hauteur de l'image)
00278     p.drawLine( 0, (image.height()*0.95)-1, image.width(), (image.height()*0.95)-1 );
00279     */
00280     QRect bandeauHaut( 0, 0, image.width(), image.height()*0.05 );
00281     QRect bandeauBas( 0, image.height()*0.95, image.width(), image.height()*0.05 );
00282     penCadre.setBrush(QBrush(QColor(255, 255, 255, 255)));
00283     p.setPen(penCadre);
```

```
00284     p.fillRect(bandeauHaut, QBrush(QColor(128, 128, 128, 64)));
00285     p.fillRect(bandeauBas, QBrush(QColor(128, 128, 128, 64)));
00286
00287     // découpe le bandeau en trois cadres (largeurs : 25 % 50 % 25% de la largeur de l'image)
00288     QRect bandeauHautGauche( 0, 0, image.width()*0.25, image.height()*0.05 );
00289     QRect bandeauHautCentre( image.width()*0.25, 0, image.width()*0.5, image.height()*0.05 );
00290     QRect bandeauHautDroite( image.width()*0.75, 0, image.width()*0.25, image.height()*0.05 );
00291
00292     // découpe le bandeau en trois cadres (largeurs : 25 % 25 % 25% 25% de la largeur de l'image)
00293     QRect bandeauBasGauche( 0, image.height()*0.95, image.width()*0.25, image.height()*0.05 );
00294     QRect bandeauBasCentreGauche( image.width()*0.25, image.height()*0.95, image.width()*0.25, image.height
      ()*0.05 );
00295     QRect bandeauBasCentreDroite( image.width()*0.50, image.height()*0.95, image.width()*0.25, image.height
      ()*0.05 );
00296     QRect bandeauRadar( image.width()*0.75, image.height() - image.width()*0.25, image.width()*0.25, image.
      width()*0.25);
00297     QRect bandeauBasDroite( image.width()*0.75, image.height()*0.95, image.width()*0.25, image.height()*0.0
      5 );
00298
00299     /*
00300     // pour le dessin des cadres
00301     p.setPen(pen);
00302     // dessine les trois cadres du haut
00303     p.drawRect( bandeauHautGauche ); // haut gauche
00304     p.fillRect(bandeauHautGauche, QBrush(QColor(128, 128, 128, 64)));
00305     p.drawRect( bandeauHautCentre ); // haut centre
00306     p.fillRect(bandeauHautCentre, QBrush(QColor(128, 128, 128, 64)));
00307     p.drawRect( bandeauHautDroite ); // haut droite
00308     p.fillRect(bandeauHautDroite, QBrush(QColor(128, 128, 128, 64)));
00309     // dessine les quatres cadres du bas
00310     p.drawRect( bandeauBasGauche ); // bas gauche
00311     p.fillRect(bandeauBasGauche, QBrush(QColor(128, 128, 128, 64)));
00312     p.drawRect( bandeauBasCentreGauche ); // bas centre gauche
00313     p.fillRect(bandeauBasCentreGauche, QBrush(QColor(128, 128, 128, 64)));
00314     p.drawRect( bandeauBasCentreDroite ); // bas centre droite
00315     p.fillRect(bandeauBasCentreDroite, QBrush(QColor(128, 128, 128, 64)));
00316     p.drawRect( bandeauBasDroite ); // bas droite
00317     p.fillRect(bandeauBasDroite, QBrush(QColor(128, 128, 128, 64)));
00318     */
00319
00320     // pour le dessin des textes et images
00321     pen.setBrush(Qt::darkRed);
00322     p.setPen(pen);
00323     p.setFont(fontHaut);
00324
00325     marge = image.width()*0.0025;
00326     QRect cadreLogoHorloge( bandeauHautGauche.x(), bandeauHautGauche.y(), bandeauHautGauche.width()*0.1,
      bandeauHautGauche.height() ); // 10% du bandeau
00327     cadreLogoHorloge.adjust(marge, marge, -marge, -marge);
00328     QImage logoHeure(qApp->applicationDirPath() + "/images/logo_heure.png");
00329     p.drawImage(cadreLogoHorloge, logoHeure);
00330     p.drawText(bandeauHautGauche, Qt::AlignHCenter|Qt::AlignVCenter, QTime::currentTime().toString());
00331
00332     QRect cadreLogoDuree( bandeauHautDroite.x(), bandeauHautDroite.y(), bandeauHautDroite.width()*0.1,
      bandeauHautGauche.height() ); // 10% du bandeau
00333     cadreLogoDuree.adjust(marge, marge, -marge, -marge);
00334     QImage logoDuree(qApp->applicationDirPath() + "/images/logo_duree.png");
00335     p.drawImage(cadreLogoDuree, logoDuree);
00336     p.drawText(bandeauHautDroite, Qt::AlignHCenter|Qt::AlignVCenter, rov->
      getTempsCampagne());
00337
00338     p.setFont(fontBas);
00339     QRect cadreLogoTemperature( bandeauBasGauche.x(), bandeauBasGauche.y(), bandeauBasGauche.width()*0.1,
      bandeauBasGauche.height() ); // 10% du bandeau
00340     cadreLogoTemperature.adjust(marge, marge, -marge, -marge);
00341     QImage logoTemperature(qApp->applicationDirPath() + "/images/logo_temperature.png");
00342     p.drawImage(cadreLogoTemperature, logoTemperature);
00343     p.drawText(bandeauBasGauche, Qt::AlignHCenter|Qt::AlignVCenter, rov->
      getCapteurs()->getTemperature() + " °C");
00344
00345     QRect cadreLogoHumidite( bandeauBasCentreGauche.x(), bandeauBasCentreGauche.y(), bandeauBasCentreGauche
      .width()*0.1, bandeauBasCentreGauche.height() ); // 10% du bandeau
00346     cadreLogoHumidite.adjust(marge, marge, -marge, -marge);
00347     QImage logoHumditie(qApp->applicationDirPath() + "/images/logo_humidite.png");
00348     p.drawImage(cadreLogoHumidite, logoHumditie);
00349     p.drawText(bandeauBasCentreGauche, Qt::AlignHCenter|Qt::AlignVCenter, rov->
      getCapteurs()->getHumidite() + "%");
00350
00351     QRect cadreLogoRadiation( bandeauBasCentreDroite.x(), bandeauBasCentreDroite.y(),
      bandeauBasCentreDroite.width()*0.1, bandeauBasCentreDroite.height() ); // 10% du bandeau
00352     cadreLogoRadiation.adjust(marge, marge, -marge, -marge);
00353     QImage logoRadiation(qApp->applicationDirPath() + "/images/logo_radiation.png");
00354     p.drawImage(cadreLogoRadiation, logoRadiation);
00355     p.drawText(bandeauBasCentreDroite, Qt::AlignHCenter|Qt::AlignVCenter, rov->
      getCapteurs()->getRadiation() + " uS/h");
00356
00357     if(getEtatRadar())
00358     {
00359         pen.setBrush(Qt::green);
00360         p.setPen(pen);
00361         //p.drawPoint(image.width()*0.875, image.height() - image.width()*0.125); // placement du point 0
```

```
 sur le radar
00362         p.drawImage(bandeauRadar, QImage(qApp->applicationDirPath() + "/images/RADAR.png"));
00363
00364         if(rov->getCapteurs()->getTelemetrie().toInt() <=
      DISTANCE_MAX_RADAR && rov->getCapteurs()->
      getTelemetrie().toInt() >= 0 && rov->getCapteurs()->
      getAngle().toInt() >= 0 && rov->getCapteurs()->getAngle().toInt() <=
      ANGLE_MAX_RADAR && rov->getCapteurs()->getAngle() != "" &&
      rov->getCapteurs()->getTelemetrie() != "")
00365         {
00366             if(rov->getCapteurs()->getAngle().toInt() ==
      ANGLE_MAX_RADAR || rov->getCapteurs()->getAngle().toInt() ==
      ANGLE_MIN_RADAR)
00367             {
00368                 pointsRadar.clear();
00369             }
00370             pointsRadar.push_back(QPoint(calculCoordonneesX(image),
      calculCoordonneesY(image)));
00371             for (QVector<QPoint>::iterator it = pointsRadar.begin(); it !=
      pointsRadar.end(); ++it)
00372             {
00373                p.drawPoint((*it).x(),(*it).y());
00374             }
00375         }
00376     }
00377     else
00378     {
00379         QRect cadreLogoObstacle( bandeauBasDroite.x(), bandeauBasDroite.y(), bandeauBasDroite.width()*0.1,
      bandeauBasDroite.height() ); // 10% du bandeau
00380         cadreLogoObstacle.adjust(marge, marge, -marge, -marge);
00381         QImage logoObstacle(qApp->applicationDirPath() + "/images/logo_telemetrie.png");
00382         p.drawImage(cadreLogoObstacle, logoObstacle);
00383         p.drawText(bandeauBasDroite, Qt::AlignHCenter|Qt::AlignVCenter, "Obstacle : " +
      rov->getCapteurs()->getTelemetrie() + " cm");
00384
00385     }
00386     p.end();
00387 }
00388
00389 void IHMRov::actualiserInformationsSeuils()
00390 {
00391     if(rov->getCapteurs()->getRadiation().toDouble() >
      SEUIL_RADIATION_ACCEPTABLE)
00392     {
00393         indicateurRadiation->setFillBrush(QBrush(QColor(Qt::GlobalColor::darkRed)));
00394     }
00395     else
00396     {
00397         indicateurRadiation->setFillBrush(QBrush(QColor(Qt::GlobalColor::darkGreen)));
00398     }
00399
00400     if(rov->getCapteurs()->getTemperature().toDouble() >
      SEUIL_TEMPERATURE_POSITIVE_ACCEPTABLE ||
      rov->getCapteurs()->getTemperature().toDouble() <
      SEUIL_TEMPERATURE_NEGATIVE_ACCEPTABLE)
00401     {
00402         indicateurTemperature->setFillBrush(QBrush(QColor(Qt::GlobalColor::darkRed)));
00403     }
00404     else
00405     {
00406         indicateurTemperature->setFillBrush(QBrush(QColor(Qt::GlobalColor::darkGreen))
      );
00407     }
00408
00409     indicateurRadiation->setValue(rov->getCapteurs()->
      getRadiation().toDouble());
00410     indicateurTemperature->setValue(rov->getCapteurs()->
      getTemperature().toDouble());
00411 }
00412
00413 double IHMRov::calculCoordonneesX(QPixmap &image)
00414 {
00415     if(rov->getCapteurs()->getAngle().toDouble() > 90)
00416         return image.width()*(0.875 - 0.125*qSin(qDegreesToRadians(90.0 - (rov->
      getCapteurs()->getAngle().toDouble() - 2 * (rov->
      getCapteurs()->getAngle().toDouble() - 90))))* rov->
      getCapteurs()->getTelemetrie().toDouble()/
      DISTANCE_MAX_RADAR);
00417     else
00418         return image.width()*(0.875 + 0.125*qSin(qDegreesToRadians(90.0 - rov->
      getCapteurs()->getAngle().toDouble()))* rov->getCapteurs()->
      getTelemetrie().toDouble()/DISTANCE_MAX_RADAR);
00419 }
00420
00421 double IHMRov::calculCoordonneesY(QPixmap &image)
00422 {
00423     return (image.height() - image.width()*0.125) - image.width()*0.125*qSin(qDegreesToRadians(
      rov->getCapteurs()->getAngle().toDouble()))*rov->
      getCapteurs()->getTelemetrie().toDouble()/
      DISTANCE_MAX_RADAR;
00424 }
```

```
00425
00426 void IHMRov::setCampagne(Campagne *campagne)
00427 {
00428     campagneEnCours = campagne;
00429     setWindowTitle(NOM_FENETRE_ROV  " " + campagne->
      getNomCampagne() + " " + campagne->getDate().toString());
00430 }
00431
00432 Campagne* IHMRov::getCampagne()
00433 {
00434     return campagneEnCours;
00435 }
00436
00437 void IHMRov::setEtatRadar(bool etatRadar)
00438 {
00439     this->etatRadar = etatRadar;
00440 }
00441
00442 bool IHMRov::getEtatRadar()
00443 {
00444     return etatRadar;
00445 }
00446
00447 void IHMRov::afficherImage(QPixmap image)
00448 {
00449     derniereImageVideo = image;
00450     actualiserInformations(image);
00451     fluxVideo->setPixmap(image);
00452 }
00453
00454 void IHMRov::reglerVideo()
00455 {
00456     if(reglageVideo != nullptr)
00457     {
00458         reglageVideo->show();
00459         reglageVideo->raise();
00460     }
00461 }
00462
00463 void IHMRov::reglerConfiguration()
00464 {
00465     if(configuration != nullptr)
00466     {
00467         configuration->actualisePortsDisponibles();
00468         configuration->show();
00469         configuration->raise();
00470     }
00471 }
00472
00473 void IHMRov::capturerImage(bool etat)
00474 {
00475     #ifndef PAS_DE_MANETTE
00476     if(etat)
00477     #endif
00478     //Q_UNUSED(etat)
00479     {
00480         Photo photo;
00481
00482         photo.image = dernieReImageVideo;
00483         photo.dateheure = QDateTime::currentDateTime();
00484         photo.aGarder = true;
00485         photo.cheminSauvegarde = campagneEnCours->
      getCheminSauvegarde() + "/" + campagneEnCours->
      getNomCampagne() + "/" + "Capture_" + QString::number(
      campagneEnCours->incrementeNombrePhoto());
00486
00487         campagneEnCours->ajouterPhoto(photo);
00488         qDebug() << Q_FUNC_INFO << "Photo capturée";
00489
00490         photo.image.save(photo.cheminSauvegarde, "PNG");
00491
00492         ihmAccueil->ajouterPhotoBDD(photo,
      campagneEnCours);
00493     }
00494 }
00495
00496 void IHMRov::gererCampagne()
00497 {
00498     qDebug() << Q_FUNC_INFO << boutonCampagne->text();
00499     if(boutonCampagne->text() == QString::fromUtf8("Démarrer"))
00500     {
00501         if(reglageVideo == nullptr)
00502             reglageVideo = new IHMReglageVideo(rov);
00503         if(configuration == nullptr)
00504             configuration = new IHMConfiguration(
      rov);
00505         if(rov->demarrerCampagne())
00506         {
00507             boutonCampagne->setText(QString::fromUtf8("Arrêter"));
00508         }
00509     }
```

```
00510     else if(boutonCampagne->text() == QString::fromUtf8("Arrêter"))
00511     {
00512         fermer();
00513         //this->close();
00514     }
00515 }
00516
00517 void IHMRov:: chargerPhotos()
00518 {
00519     IHMAlbumPhoto *ihmAlbumPhoto = new IHMAlbumPhoto(this);
00520     ihmAlbumPhoto->ouvrirAlbumPhotos(campagneEnCours->
      getAlbumPhoto());
00521 }
00522
00523 void IHMRov::initialiserEvenementsCamera()
00524 {
00525     reglageVideo->initialiserEvenementsCamera();
00526 }
00527
00528 void IHMRov::arreterVideo()
00529 {
00530     reglageVideo->modifierEtatBoutons();
00531     fluxVideo->setPixmap(QPixmap(qApp->applicationDirPath() +  "/images/signal-interrompu.jpg"));
00532 }
00533
00534 void IHMRov::closeEvent(QCloseEvent *event)
00535 {
00536     qDebug() << Q_FUNC_INFO << rov->getCampagneEncours();
00537     if(rov->getCampagneEncours())
00538     {
00539         fermer();
00540         event->accept(); // -> close
00541     }
00542     else
00543     {
00544         event->ignore();
00545     }
00546 }
00547
00548 void IHMRov::fermer()
00549 {
00550     rov->arreterCampagne();
00551     delete reglageVideo;
00552     delete configuration;
00553     reglageVideo = nullptr;
00554     configuration = nullptr;
00555     boutonCampagne->setText(QString::fromUtf8("Démarrer"));
00556     ihmAccueil->modifierCampagneBDD(campagneEnCours);
00557     this->setVisible(false);
00558     ihmAccueil->setVisible(true);
00559 }
00560
00561 void IHMRov::modifieEtatPortSerie(bool etat, QString information)
00562 {
00563     if(etat)
00564     {
00565         logoEtatPortSerie->setPixmap(QPixmap(qApp->applicationDirPath() + "
      /images/actif.png").scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00566         etatPortSerie->setText(information);
00567     }
00568     else
00569     {
00570         configuration->modifieEtatBoutons();
00571         logoEtatPortSerie->setPixmap(QPixmap(qApp->applicationDirPath() + "
      /images/inactif.png").scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00572         etatPortSerie->setText("Fermé");
00573     }
00574 }
00575
00576 void IHMRov::modifieEtatCamera(bool etat, QString information)
00577 {
00578     if(etat)
00579     {
00580         logoEtatCamera->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/actif.png").
      scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00581         etatCamera->setText(information);
00582     }
00583     else
00584     {
00585         logoEtatCamera->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/inactif.png")
      .scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00586         etatCamera->setText("Eteinte");
00587     }
00588 }
00589
00590 void IHMRov::modifieEtatManette(bool etat)
00591 {
00592     if(etat)
00593     {
00594         logoEtatManette->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/actif.png")
      .scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
```

```
00595         etatManette->setText("Connectée");
00596     }
00597     else
00598     {
00599         logoEtatManette->setPixmap(QPixmap(qApp->applicationDirPath() + "/images/inactif.png
      ").scaled(qApp->desktop()->screen()->width()*0.01, qApp->desktop()->screen()->width()*0.01));
00600         etatManette->setText("Déconnectée");
00601     }
00602 }
```

7.49 Référence du fichier ihmrov.h

Fichier qui contient la déclaration de la classe IHMRov.

```
#include <QtWidgets>
#include <QMessageBox>
#include <qwt/qwt_thermo.h>
#include <qwt/qwt_color_map.h>
#include <QColor>
```

Classes

— class IHMRov
IHM permettant d'obtenir le flux vidéo en direct placé sur le robot et d'obtenir les informations relatifs à ses capteurs.

Macros

— #define ANGLE_MAX_RADAR 180
Défini l'angle max supporté pas le radar.
— #define ANGLE_MIN_RADAR 0
Défini l'angle min supporté pas le radar.
— #define DISTANCE_MAX_RADAR 150
Défini la distance max supporté pas le radar.
— #define NOM_FENETRE_ROV "Projet Rovnet 2020"
Défini le nom du projet.
— #define PAS_DE_MANETTE
— #define RATIO 0.75
Défini le ratio pour la taille de l'affichage du flux vidéo.
— #define SEUIL_RADIATION_ACCEPTABLE 0.1
Défini le seuil à partir duquel la radiation n'est plus acceptable en microsievert.
— #define SEUIL_TEMPERATURE_NEGATIVE_ACCEPTABLE -40
— #define SEUIL_TEMPERATURE_POSITIVE_ACCEPTABLE 40

7.49.1 Description détaillée

Fichier qui contient la déclaration de la classe IHMRov.

Définition dans le fichier ihmrov.h.

7.49.2 Documentation des macros

7.49.2.1 ANGLE_MAX_RADAR

```
#define ANGLE_MAX_RADAR 180
```

Défini l'angle max supporté pas le radar.

Définition à la ligne 42 du fichier ihmrov.h.

Référencé par IHMRov : :actualiserInformations().

7.49.2.2 ANGLE_MIN_RADAR

```
#define ANGLE_MIN_RADAR 0
```

Défini l'angle min supporté pas le radar.

Définition à la ligne 48 du fichier ihmrov.h.

Référencé par IHMRov : :actualiserInformations().

7.49.2.3 DISTANCE_MAX_RADAR

```
#define DISTANCE_MAX_RADAR 150
```

Défini la distance max supporté pas le radar.

Définition à la ligne 36 du fichier ihmrov.h.

Référencé par IHMRov : :actualiserInformations(), IHMRov : :calculCoordonneesX(), et IHMRov : :calculCoordonneesY().

7.49.2.4 NOM_FENETRE_ROV

```
#define NOM_FENETRE_ROV "Projet Rovnet 2020"
```

Défini le nom du projet.

Définition à la ligne 23 du fichier ihmrov.h.

Référencé par IHMRov : :setCampagne().

7.49.2.5 PAS_DE_MANETTE

```
#define PAS_DE_MANETTE
```

Définition à la ligne 16 du fichier ihmrov.h.

7.49.2.6 RATIO

```
#define RATIO 0.75
```

Défini le ratio pour la taille de l'affichage du flux vidéo.

Définition à la ligne 30 du fichier ihmrov.h.

Référencé par IHMRov : :configurerWidgets().

7.49.2.7 SEUIL_RADIATION_ACCEPTABLE

```
#define SEUIL_RADIATION_ACCEPTABLE 0.1
```

Défini le seuil à partir duquel la radiation n'est plus acceptable en microsievert.

Définition à la ligne 66 du fichier ihmrov.h.

Référencé par IHMRov : :actualiserInformationsSeuils(), et IHMRov : :configurerWidgets().

7.49.2.8 SEUIL_TEMPERATURE_NEGATIVE_ACCEPTABLE

```
#define SEUIL_TEMPERATURE_NEGATIVE_ACCEPTABLE -40
```

Définition à la ligne 60 du fichier ihmrov.h.

Référencé par IHMRov : :actualiserInformationsSeuils(), et IHMRov : :configurerWidgets().

7.49.2.9 SEUIL_TEMPERATURE_POSITIVE_ACCEPTABLE

```
#define SEUIL_TEMPERATURE_POSITIVE_ACCEPTABLE 40
```

Définition à la ligne 54 du fichier ihmrov.h.

Référencé par IHMRov : :actualiserInformationsSeuils(), et IHMRov : :configurerWidgets().

7.50 ihmrov.h

```
00001
00007 #ifndef IHMROV_H
00008 #define IHMROV_H
00009
00010 #include <QtWidgets>
00011 #include <QMessageBox>
00012 #include <qwt/qwt_thermo.h>
00013 #include <qwt/qwt_color_map.h>
00014 #include <QColor>
00015
00016 #define PAS_DE_MANETTE
00017
00023 #define NOM_FENETRE_ROV "Projet Rovnet 2020"
00024
00030 #define RATIO           0.75
00031
00036 #define DISTANCE_MAX_RADAR      150
00037
00042 #define ANGLE_MAX_RADAR      180
00043
00048 #define ANGLE_MIN_RADAR      0
00049
00054 #define SEUIL_TEMPERATURE_POSITIVE_ACCEPTABLE 40
00055
00060 #define SEUIL_TEMPERATURE_NEGATIVE_ACCEPTABLE -40
00061
00066 #define SEUIL_RADIATION_ACCEPTABLE 0.1
00067
00068
00069 class IHMAlbumPhoto;
00070 class IHMReglageVideo;
00071 class Rov;
00072 class IHMAccueil;
00073 class Campagne;
00074 class IHMConfiguration;
00075
00081 class IHMRov : public QWidget
00082 {
00083     Q_OBJECT
00084 private:
00085     Campagne *campagneEnCours;
00086     IHMAccueil *ihmAccueil;
00087     Rov *rov;
00088     IHMReglageVideo *reglageVideo;
00089     IHMConfiguration *configuration;
00090     QLabel *fluxVideo;
00091     QPushButton *photosEnCours;
00092     QPushButton *boutonReglageVideo;
00093     QPixmap derniereImageVideo;
00094     #ifdef PAS_DE_MANETTE
00095     QPushButton *testCapturePhoto;
00096     #endif
00097     QPushButton *boutonCampagne;
00098     QPushButton *boutonConfiguration;
00099     QLabel *hautDePage;
00100     QLabel *basDePage;
00101     QLabel *logoEtatPortSerie;
00102     QLabel *logoEtatCamera;
00103     QLabel *logoEtatManette;
00104     QLabel *etatPortSerie;
00105     QLabel *etatCamera;
00106     QLabel *etatManette;
00107     QLabel *portSerie;
00108     QLabel *camera;
00109     QLabel *manette;
00110     bool etatRadar;
00111     QwtThermo *indicateurTemperature;
00112     QwtThermo *indicateurRadiation;
00113     QLabel *temperature;
00114     QLabel *radiation;
00115     QGroupBox *zoneEtatMateriel;
00116     QGroupBox *zoneInformationSeuils;
00117     QVector<QPoint> pointsRadar;
00118
00123     void initialiserWidgets();
00128     void configurerWidgets();
00133     void initialiserLayouts();
00138     void initialiserEvenements();
00144     void actualiserInformations(QPixmap &image);
00149     void fermer();
00156     double calculCoordonneesX(QPixmap &image);
00163     double calculCoordonneesY(QPixmap &image);
00164
00165 protected:
00171     void closeEvent(QCloseEvent *event);
00172
00173 public:
00180     IHMRov(IHMAccueil *ihmAccueil, QWidget *parent = nullptr);
```

```
00185     ~IHMRov();
00191     void setCampagne(Campagne *campagne);
00197     Campagne* getCampagne();
00203     void setEtatRadar(bool etatRadar);
00208     bool getEtatRadar();
00213     void actualiserInformationsSeuils();
00214
00215 public slots:
00221     void afficherImage(QPixmap image);
00226     void reglerVideo();
00231     void reglerConfiguration();
00237     void capturerImage(bool etat=false);
00242     void gererCampagne();
00247     void chargerPhotos();
00252     void initialiserEvenementsCamera();
00257     void arreterVideo();
00264     void modifieEtatPortSerie(bool etat, QString information);
00270     void modifieEtatCamera(bool etat, QString information);
00276     void modifieEtatManette(bool etat);
00277
00278 };
00279
00280 #endif // IHMROV_H
```

7.51 Référence du fichier main.cpp

Fichier main.

```
#include "ihmaccueil.h"
#include <QApplication>
```

Fonctions

— int main (int argc, char ∗argv[ ])
Programme principal qui instancie, affiche l'IHM du Rov et exécute la boucle d'évènements.

7.51.1 Description détaillée

Fichier main.

Définition dans le fichier main.cpp.

7.51.2 Documentation des fonctions

7.51.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Programme principal qui instancie, affiche l'IHM du Rov et exécute la boucle d'évènements.

Paramètres

argc	
argv	

Renvoie

retourne un entier correspond à l'état de la fermeture de l'application

Définition à la ligne 17 du fichier main.cpp.

```
00018 {
00019     QApplication a(argc, argv);
00020     IHMAccueil w;
00021     w.setFixedSize(633, 489);
00022     w.show();
00023
00024     return a.exec();
00025 }
```

7.52 main.cpp

```
00001
00007 #include "ihmaccueil.h"
00008 #include <QApplication>
00009
00017 int main(int argc, char *argv[])
00018 {
00019     QApplication a(argc, argv);
00020     IHMAccueil w;
00021     w.setFixedSize(633, 489);
00022     w.show();
00023
00024     return a.exec();
00025 }
00026
00027 // Pour la documentation Doxygen
00028
```

7.53 Référence du fichier manette.cpp

Fichier qui contient la définition de la classe Manette.

```
#include "manette.h"
```

7.53.1 Description détaillée

Fichier qui contient la définition de la classe Manette.

Définition dans le fichier manette.cpp.

7.54 manette.cpp

```
00001
00007 #include "manette.h"
00008
00009 Manette::Manette(int deviceID) : QGamepad(deviceID), modeDeplacement(true),
      modePilotageBras(false), puissanceGachetteDroite(0), puissanceGachetteGauche(0), joystickAxeXCamera("0"),
      joystickAxeYCamera("0")
00010 {
00011     qDebug() << Q_FUNC_INFO << this;
00012     initialiserEtats();
00013     initialiserEtatBouton();
00014     initialiserTypesPilotage();
00015     initialiserTypesDeplacement();
00016 }
00017
00018 Manette::~Manette()
00019 {
00020     qDebug() << Q_FUNC_INFO << this;
00021 }
```

```
00022
00023 void Manette::initialiserEtats()
00024 {
00025     maManettePilotage.flecheADroite = false;
00026     maManettePilotage.flecheAGauche = false;
00027     maManettePilotage.flecheEnArriere = false;
00028     maManettePilotage.flecheEnAvant = false;
00029     maManettePilotage.boutonHautDroit = false;
00030     maManettePilotage.boutonHautGauche = false;
00031
00032     maManetteDeplacement.joystickGaucheADroite = false;
00033     maManetteDeplacement.joystickGaucheAGauche = false;
00034     maManetteDeplacement.joystickGaucheEnArriere = false;
00035     maManetteDeplacement.joystickGaucheEnAvant = false;
00036     maManetteDeplacement.gachetteBasDroit = false;
00037     maManetteDeplacement.gachetteBasGauche = false;
00038 }
00039
00040 void Manette::initialiserTypesPilotage()
00041 {
00042     avance.flecheADroite = false;
00043     avance.flecheAGauche = false;
00044     avance.flecheEnArriere = false;
00045     avance.flecheEnAvant = true;
00046     avance.boutonHautDroit = false;
00047     avance.boutonHautGauche = false;
00048
00049     recule.flecheADroite = false;
00050     recule.flecheAGauche = false;
00051     recule.flecheEnArriere = true;
00052     recule.flecheEnAvant = false;
00053     recule.boutonHautDroit = false;
00054     recule.boutonHautGauche = false;
00055
00056     gauche.flecheADroite = false;
00057     gauche.flecheAGauche = true;
00058     gauche.flecheEnArriere = false;
00059     gauche.flecheEnAvant = false;
00060     gauche.boutonHautDroit = false;
00061     gauche.boutonHautGauche = false;
00062
00063     droite.flecheADroite = true;
00064     droite.flecheAGauche = false;
00065     droite.flecheEnArriere = false;
00066     droite.flecheEnAvant = false;
00067     droite.boutonHautDroit = false;
00068     droite.boutonHautGauche = false;
00069
00070     monte.flecheADroite = false;
00071     monte.flecheAGauche = false;
00072     monte.flecheEnArriere = false;
00073     monte.flecheEnAvant = false;
00074     monte.boutonHautDroit = true;
00075     monte.boutonHautGauche = false;
00076
00077     descend.flecheADroite = false;
00078     descend.flecheAGauche = false;
00079     descend.flecheEnArriere = false;
00080     descend.flecheEnAvant = false;
00081     descend.boutonHautDroit = false;
00082     descend.boutonHautGauche = true;
00083 }
00084
00085 void Manette::initialiserTypesDeplacement()
00086 {
00087     initialisationStructuresEnAvant();
00088     initialisationStructuresEnArriere();
00089     initialisationStructureRotationAGauche();
00090     initialisationStructuresRotationAGaucheDoucement();
00091     initialisationStructureRotationADroite();
00092     initialisationStructuresRotationADroiteDoucement();
00093     initialisationStructureViragеAvantAGauche();
00094     initialisationStructuresViragеAvantAGaucheDoucement(
      );
00095     initialisationStructureViragеAvantADroite();
00096     initialisationStructuresViragеAvantADroiteDoucement(
      );
00097     initialisationStructureViragеArriereAGauche();
00098     initialisationStructuresViragеArriereAGaucheDoucement
      ();
00099     initialisationStructureViragеArriereADroite();
00100     initialisationStructuresViragеArriereADroiteDoucement
      ();
00101 }
00102
00103 void Manette::initialisationStructuresEnAvant()
00104 {
00105     enAvantCas1.joystickGaucheEnAvant = false;
00106     enAvantCas1.joystickGaucheEnArriere = false;
00107     enAvantCas1.joystickGaucheAGauche = false;
00108     enAvantCas1.joystickGaucheADroite = false;
```

```
00109     enAvantCas1.gachetteBasGauche = false;
00110     enAvantCas1.gachetteBasDroit = true;
00111 
00112     enAvantCas2.joystickGaucheEnAvant = true;
00113     enAvantCas2.joystickGaucheEnArriere = false;
00114     enAvantCas2.joystickGaucheAGauche = false;
00115     enAvantCas2.joystickGaucheADroite = false;
00116     enAvantCas2.gachetteBasGauche = false;
00117     enAvantCas2.gachetteBasDroit = true;
00118 
00119     enAvantCas3.joystickGaucheEnAvant = false;
00120     enAvantCas3.joystickGaucheEnArriere = true;
00121     enAvantCas3.joystickGaucheAGauche = false;
00122     enAvantCas3.joystickGaucheADroite = false;
00123     enAvantCas3.gachetteBasGauche = false;
00124     enAvantCas3.gachetteBasDroit = true;
00125 }
00126 
00127 void Manette::initialisationStructuresEnArriere()
00128 {
00129     enArriereCas1.joystickGaucheEnAvant = false;
00130     enArriereCas1.joystickGaucheEnArriere = false;
00131     enArriereCas1.joystickGaucheAGauche = false;
00132     enArriereCas1.joystickGaucheADroite = false;
00133     enArriereCas1.gachetteBasGauche = true;
00134     enArriereCas1.gachetteBasDroit = false;
00135 
00136     enArriereCas2.joystickGaucheEnAvant = true;
00137     enArriereCas2.joystickGaucheEnArriere = false;
00138     enArriereCas2.joystickGaucheAGauche = false;
00139     enArriereCas2.joystickGaucheADroite = false;
00140     enArriereCas2.gachetteBasGauche = true;
00141     enArriereCas2.gachetteBasDroit = false;
00142 
00143     enArriereCas3.joystickGaucheEnAvant = false;
00144     enArriereCas3.joystickGaucheEnArriere = true;
00145     enArriereCas3.joystickGaucheAGauche = false;
00146     enArriereCas3.joystickGaucheADroite = false;
00147     enArriereCas3.gachetteBasGauche = true;
00148     enArriereCas3.gachetteBasDroit = false;
00149 }
00150 
00151 void Manette::initialisationStructureRotationAGauche()
00152 {
00153     rotationAGauche.joystickGaucheEnAvant = false;
00154     rotationAGauche.joystickGaucheEnArriere = false;
00155     rotationAGauche.joystickGaucheAGauche = true;
00156     rotationAGauche.joystickGaucheADroite = false;
00157     rotationAGauche.gachetteBasGauche = false;
00158     rotationAGauche.gachetteBasDroit = false;
00159 }
00160 
00161 void Manette::initialisationStructuresRotationAGaucheDoucement
      ()
00162 {
00163     rotationAGaucheDoucementCas1.
      joystickGaucheEnAvant = false;
00164     rotationAGaucheDoucementCas1.
      joystickGaucheEnArriere = true;
00165     rotationAGaucheDoucementCas1.
      joystickGaucheAGauche = true;
00166     rotationAGaucheDoucementCas1.
      joystickGaucheADroite = false;
00167     rotationAGaucheDoucementCas1.gachetteBasGauche = false;
00168     rotationAGaucheDoucementCas1.gachetteBasDroit = false;
00169 
00170     rotationAGaucheDoucementCas2.
      joystickGaucheEnAvant = true;
00171     rotationAGaucheDoucementCas2.
      joystickGaucheEnArriere = false;
00172     rotationAGaucheDoucementCas2.
      joystickGaucheAGauche = true;
00173     rotationAGaucheDoucementCas2.
      joystickGaucheADroite = false;
00174     rotationAGaucheDoucementCas2.gachetteBasGauche = false;
00175     rotationAGaucheDoucementCas2.gachetteBasDroit = false;
00176 }
00177 
00178 void Manette::initialisationStructureRotationADroite()
00179 {
00180     rotationADroite.joystickGaucheEnAvant = false;
00181     rotationADroite.joystickGaucheEnArriere = false;
00182     rotationADroite.joystickGaucheAGauche = false;
00183     rotationADroite.joystickGaucheADroite = true;
00184     rotationADroite.gachetteBasGauche = false;
00185     rotationADroite.gachetteBasDroit = false;
00186 }
00187 
00188 void Manette::initialisationStructuresRotationADroiteDoucement
      ()
00189 {
```

```
00190     rotationADroiteDoucementCas1.
      joystickGaucheEnAvant = false;
00191     rotationADroiteDoucementCas1.
      joystickGaucheEnArriere = true;
00192     rotationADroiteDoucementCas1.
      joystickGaucheAGauche = false;
00193     rotationADroiteDoucementCas1.
      joystickGaucheADroite = true;
00194     rotationADroiteDoucementCas1.gachetteBasGauche = false;
00195     rotationADroiteDoucementCas1.gachetteBasDroit = false;
00196
00197     rotationADroiteDoucementCas2.
      joystickGaucheEnAvant = true;
00198     rotationADroiteDoucementCas2.
      joystickGaucheEnArriere = false;
00199     rotationADroiteDoucementCas2.
      joystickGaucheAGauche = false;
00200     rotationADroiteDoucementCas2.
      joystickGaucheADroite = true;
00201     rotationADroiteDoucementCas2.gachetteBasGauche = false;
00202     rotationADroiteDoucementCas2.gachetteBasDroit = false;
00203 }
00204
00205 void Manette::initialisationStructureVirageAvantAGauche()
00206 {
00207     virageAvantAGauche.joystickGaucheEnAvant = false;
00208     virageAvantAGauche.joystickGaucheEnArriere = false;
00209     virageAvantAGauche.joystickGaucheAGauche = true;
00210     virageAvantAGauche.joystickGaucheADroite = false;
00211     virageAvantAGauche.gachetteBasGauche = false;
00212     virageAvantAGauche.gachetteBasDroit = true;
00213 }
00214
00215 void Manette::initialisationStructuresVirageAvantAGaucheDoucement
      ()
00216 {
00217     virageAvantAGaucheDoucementCas1.
      joystickGaucheEnAvant = true;
00218     virageAvantAGaucheDoucementCas1.
      joystickGaucheEnArriere = false;
00219     virageAvantAGaucheDoucementCas1.
      joystickGaucheAGauche = true;
00220     virageAvantAGaucheDoucementCas1.
      joystickGaucheADroite = false;
00221     virageAvantAGaucheDoucementCas1.
      gachetteBasGauche = false;
00222     virageAvantAGaucheDoucementCas1.
      gachetteBasDroit = true;
00223
00224     virageAvantAGaucheDoucementCas2.
      joystickGaucheEnAvant = false;
00225     virageAvantAGaucheDoucementCas2.
      joystickGaucheEnArriere = true;
00226     virageAvantAGaucheDoucementCas2.
      joystickGaucheAGauche = true;
00227     virageAvantAGaucheDoucementCas2.
      joystickGaucheADroite = false;
00228     virageAvantAGaucheDoucementCas2.
      gachetteBasGauche = false;
00229     virageAvantAGaucheDoucementCas2.
      gachetteBasDroit = true;
00230 }
00231
00232 void Manette::initialisationStructureVirageAvantADroite()
00233 {
00234     virageAvantADroite.joystickGaucheEnAvant = false;
00235     virageAvantADroite.joystickGaucheEnArriere = false;
00236     virageAvantADroite.joystickGaucheAGauche = false;
00237     virageAvantADroite.joystickGaucheADroite = true;
00238     virageAvantADroite.gachetteBasGauche = false;
00239     virageAvantADroite.gachetteBasDroit = true;
00240 }
00241
00242 void Manette::initialisationStructuresVirageAvantADroiteDoucement
      ()
00243 {
00244     virageAvantADroiteDoucementCas1.
      joystickGaucheEnAvant = true;
00245     virageAvantADroiteDoucementCas1.
      joystickGaucheEnArriere = false;
00246     virageAvantADroiteDoucementCas1.
      joystickGaucheAGauche = false;
00247     virageAvantADroiteDoucementCas1.
      joystickGaucheADroite = true;
00248     virageAvantADroiteDoucementCas1.
      gachetteBasGauche = false;
00249     virageAvantADroiteDoucementCas1.
      gachetteBasDroit = true;
00250
00251     virageAvantADroiteDoucementCas2.
      joystickGaucheEnAvant = false;
```

```
00252     virageAvantADroiteDoucementCas2.
      joystickGaucheEnArriere = true;
00253     virageAvantADroiteDoucementCas2.
      joystickGaucheAGauche = false;
00254     virageAvantADroiteDoucementCas2.
      joystickGaucheADroite = true;
00255     virageAvantADroiteDoucementCas2.
      gachetteBasGauche = false;
00256     virageAvantADroiteDoucementCas2.
      gachetteBasDroit = true;
00257 }
00258
00259 void Manette::initialisationStructureVirageArriereAGauche
      ()
00260 {
00261     virageArriereAGauche.joystickGaucheEnAvant = false;
00262     virageArriereAGauche.joystickGaucheEnArriere = false;
00263     virageArriereAGauche.joystickGaucheAGauche = true;
00264     virageArriereAGauche.joystickGaucheADroite = false;
00265     virageArriereAGauche.gachetteBasGauche = true;
00266     virageArriereAGauche.gachetteBasDroit = false;
00267 }
00268
00269 void Manette::initialisationStructuresVirageArriereAGaucheDoucement
      ()
00270 {
00271     virageArriereAGaucheDoucementCas1.
      joystickGaucheEnAvant = true;
00272     virageArriereAGaucheDoucementCas1.
      joystickGaucheEnArriere = false;
00273     virageArriereAGaucheDoucementCas1.
      joystickGaucheAGauche = true;
00274     virageArriereAGaucheDoucementCas1.
      joystickGaucheADroite = false;
00275     virageArriereAGaucheDoucementCas1.
      gachetteBasGauche = true;
00276     virageArriereAGaucheDoucementCas1.
      gachetteBasDroit = false;
00277
00278     virageArriereAGaucheDoucementCas2.
      joystickGaucheEnAvant = false;
00279     virageArriereAGaucheDoucementCas2.
      joystickGaucheEnArriere = true;
00280     virageArriereAGaucheDoucementCas2.
      joystickGaucheAGauche = true;
00281     virageArriereAGaucheDoucementCas2.
      joystickGaucheADroite = false;
00282     virageArriereAGaucheDoucementCas2.
      gachetteBasGauche = true;
00283     virageArriereAGaucheDoucementCas2.
      gachetteBasDroit = false;
00284 }
00285
00286 void Manette::initialisationStructureVirageArriereADroite
      ()
00287 {
00288     virageArriereADroite.joystickGaucheEnAvant = false;
00289     virageArriereADroite.joystickGaucheEnArriere = false;
00290     virageArriereADroite.joystickGaucheAGauche = false;
00291     virageArriereADroite.joystickGaucheADroite = true;
00292     virageArriereADroite.gachetteBasGauche = true;
00293     virageArriereADroite.gachetteBasDroit = false;
00294 }
00295
00296 void Manette::initialisationStructuresVirageArriereADroiteDoucement
      ()
00297 {
00298     virageArriereADroiteDoucementCas1.
      joystickGaucheEnAvant = true;
00299     virageArriereADroiteDoucementCas1.
      joystickGaucheEnArriere = false;
00300     virageArriereADroiteDoucementCas1.
      joystickGaucheAGauche = false;
00301     virageArriereADroiteDoucementCas1.
      joystickGaucheADroite = true;
00302     virageArriereADroiteDoucementCas1.
      gachetteBasGauche = true;
00303     virageArriereADroiteDoucementCas1.
      gachetteBasDroit = false;
00304
00305     virageArriereADroiteDoucementCas2.
      joystickGaucheEnAvant = false;
00306     virageArriereADroiteDoucementCas2.
      joystickGaucheEnArriere = true;
00307     virageArriereADroiteDoucementCas2.
      joystickGaucheAGauche = false;
00308     virageArriereADroiteDoucementCas2.
      joystickGaucheADroite = true;
00309     virageArriereADroiteDoucementCas2.
      gachetteBasGauche = true;
00310     virageArriereADroiteDoucementCas2.
```

```
    gachetteBasDroit = false;
00311 }
00312
00313 void Manette::initialiserEtatBouton()
00314 {
00315     maManetteBouton.boutonA = false;
00316     maManetteBouton.boutonB = false;
00317     maManetteBouton.boutonX = false;
00318     maManetteBouton.boutonY = false;
00319
00320     ouvrirPince.boutonA = false;
00321     ouvrirPince.boutonB = false;
00322     ouvrirPince.boutonX = true;
00323     ouvrirPince.boutonY = false;
00324
00325     fermerPince.boutonA = false;
00326     fermerPince.boutonB = true;
00327     fermerPince.boutonX = false;
00328     fermerPince.boutonY = false;
00329
00330     poserObjetDansBac.boutonA = true;
00331     poserObjetDansBac.boutonB = false;
00332     poserObjetDansBac.boutonX = false;
00333     poserObjetDansBac.boutonY = false;
00334
00335     revenirEtatInitial.boutonA = false;
00336     revenirEtatInitial.boutonB = false;
00337     revenirEtatInitial.boutonX = false;
00338     revenirEtatInitial.boutonY = true;
00339 }
00340
00341 void Manette::determinerTrameDeplacementRobot()
00342 {
00343     if(maManetteDeplacement == enAvantCas1 ||
    maManetteDeplacement == enAvantCas2 ||
    maManetteDeplacement == enAvantCas3)
00344         emit creationTrameDeplacement('A',
    puissanceGachetteDroite, '0');
00345     else if(maManetteDeplacement == enArriereCas1 ||
    maManetteDeplacement == enArriereCas2 ||
    maManetteDeplacement == enArriereCas3)
00346         emit creationTrameDeplacement('R',
    puissanceGachetteGauche, '0');
00347     else if(maManetteDeplacement == rotationAGauche)
00348         emit creationTrameDeplacement('0', 100, 'G');
00349     else if(maManetteDeplacement ==
    rotationAGaucheDoucementCas1 || maManetteDeplacement ==
    rotationAGaucheDoucementCas2)
00350         emit creationTrameDeplacement('0', int(100 *
    REDUCTION_VITESSE), 'G');
00351     else if(maManetteDeplacement == rotationADroite)
00352         emit creationTrameDeplacement('0', 100, 'D');
00353     else if(maManetteDeplacement ==
    rotationADroiteDoucementCas1 || maManetteDeplacement ==
    rotationADroiteDoucementCas2)
00354         emit creationTrameDeplacement('0', int(100 *
    REDUCTION_VITESSE), 'D');
00355     else if(maManetteDeplacement == virageAvantAGauche)
00356         emit creationTrameDeplacement('A',
    puissanceGachetteDroite, 'G');
00357     else if(maManetteDeplacement ==
    virageAvantAGaucheDoucementCas1 ||
    maManetteDeplacement == virageAvantAGaucheDoucementCas2)
00358         emit creationTrameDeplacement('A', int(
    puissanceGachetteDroite * REDUCTION_VITESSE), 'G');
00359     else if(maManetteDeplacement == virageAvantADroite)
00360         emit creationTrameDeplacement('A',
    puissanceGachetteDroite, 'D');
00361     else if(maManetteDeplacement ==
    virageAvantADroiteDoucementCas1 ||
    maManetteDeplacement == virageAvantADroiteDoucementCas2)
00362         emit creationTrameDeplacement('A', int(
    puissanceGachetteDroite * REDUCTION_VITESSE), 'D');
00363     else if(maManetteDeplacement == virageArriereAGauche)
00364         emit creationTrameDeplacement('R',
    puissanceGachetteGauche, 'G');
00365     else if(maManetteDeplacement ==
    virageArriereAGaucheDoucementCas1 ||
    maManetteDeplacement == virageArriereAGaucheDoucementCas2
    )
00366         emit creationTrameDeplacement('R', int(
    puissanceGachetteGauche * REDUCTION_VITESSE), 'G');
00367     else if(maManetteDeplacement == virageArriereADroite)
00368         emit creationTrameDeplacement('R',
    puissanceGachetteGauche, 'D');
00369     else if(maManetteDeplacement ==
    virageArriereADroiteDoucementCas1 ||
    maManetteDeplacement == virageArriereADroiteDoucementCas2
    )
00370         emit creationTrameDeplacement('R', int(
    puissanceGachetteGauche * REDUCTION_VITESSE), 'D');
```

```
00371     else
00372         emit creationTrameDeplacement('0', 0, '0');
00373 }
00374
00375 void Manette::determinerTramePilotageBras()
00376 {
00377     if(maManettePilotage == avance)
00378         emit creationTramePilotage(AVANCER);
00379     else if(maManettePilotage == recule)
00380         emit creationTramePilotage(RECULER);
00381     else if(maManettePilotage == gauche)
00382         emit creationTramePilotage(GAUCHE);
00383     else if(maManettePilotage == droite)
00384         emit creationTramePilotage(DROITE);
00385     else if(maManettePilotage == monte)
00386         emit creationTramePilotage(MONTER);
00387     else if(maManettePilotage == descend)
00388         emit creationTramePilotage(DESCENDRE);
00389     else
00390         emit creationTramePilotage(IMMOBILE);
00391 }
00392
00393 void Manette::determinerTrameBouton()
00394 {
00395     if(maManetteBouton == ouvrirPince)
00396         emit creationTramePince(OUVERTURE_PINCE);
00397     else if(maManetteBouton == fermerPince)
00398         emit creationTramePince(FERMETURE_PINCE);
00399     else if(maManetteBouton == revenirEtatInitial)
00400         emit creationTrameOrdre(RETOUR_ETAT_INITIAL);
00401     else if(maManetteBouton == poserObjetDansBac)
00402         emit creationTrameOrdre(POSER_OBJET_DANS_BAC);
00403     else
00404         emit creationTramePince(IMMOBILE);
00405 }
00406
00407 void Manette::changerMode()
00408 {
00409     if(modeDeplacement)
00410     {
00411         modePilotageBras = true;
00412         modeDeplacement = false;
00413     }
00414     else
00415     {
00416         modePilotageBras = false;
00417         modeDeplacement = true;
00418     }
00419 }
00420
00421 void Manette::changerAxeXJoystickGauche(double valeur)
00422 {
00423     if(modeDeplacement)
00424     {
00425         if(valeur <= -TAUX_VALIDITE_JOYSTICK)
00426             maManetteDeplacement.joystickGaucheAGauche = true;
00427         else if(valeur >= TAUX_VALIDITE_JOYSTICK)
00428             maManetteDeplacement.joystickGaucheADroite = true;
00429         else if(valeur > -TAUX_VALIDITE_JOYSTICK && valeur <
    TAUX_VALIDITE_JOYSTICK)
00430         {
00431             maManetteDeplacement.joystickGaucheADroite = false;
00432             maManetteDeplacement.joystickGaucheAGauche = false;
00433         }
00434         determinerTrameDeplacementRobot();
00435     }
00436 }
00437
00438 void Manette::changerAxeYJoystickGauche(double valeur)
00439 {
00440     if(modeDeplacement)
00441     {
00442         if(valeur >= TAUX_VALIDITE_JOYSTICK)
00443             maManetteDeplacement.joystickGaucheEnAvant = true;
00444         else if(valeur <= -TAUX_VALIDITE_JOYSTICK)
00445             maManetteDeplacement.joystickGaucheEnArriere = true;
00446         else if(valeur > -TAUX_VALIDITE_JOYSTICK && valeur <
    TAUX_VALIDITE_JOYSTICK)
00447         {
00448             maManetteDeplacement.joystickGaucheEnAvant = false;
00449             maManetteDeplacement.joystickGaucheEnArriere = false
    ;
00450         }
00451         determinerTrameDeplacementRobot();
00452     }
00453 }
00454
00455 void Manette::changerAxeXJoystickDroit(double valeur)
00456 {
00457     if(valeur >= TAUX_VALIDITE_JOYSTICK)
00458         joystickAxeXCamera = "D";
```

```
00459     else if(valeur <= -TAUX_VALIDITE_JOYSTICK)
00460         joystickAxeXCamera = "G";
00461     else
00462         joystickAxeXCamera = "0";
00463
00464     emit nouvelleTrameCamera(joystickAxeXCamera,
      joystickAxeYCamera);
00465 }
00466
00467 void Manette::changerAxeYJoystickDroit(double valeur)
00468 {
00469     if(valeur >= TAUX_VALIDITE_JOYSTICK)
00470         joystickAxeYCamera = "B";
00471     else if(valeur <= -TAUX_VALIDITE_JOYSTICK)
00472         joystickAxeYCamera = "H";
00473     else
00474         joystickAxeYCamera = "0";
00475
00476     emit nouvelleTrameCamera(joystickAxeXCamera,
      joystickAxeYCamera);
00477 }
00478
00479 void Manette::changerBoutonHautGauche(bool etat)
00480 {
00481     maManettePilotage.boutonHautGauche = etat;
00482     if(modePilotageBras)
00483         determinerTramePilotageBras();
00484 }
00485
00486 void Manette::changerBoutonHautDroit(bool etat)
00487 {
00488     maManettePilotage.boutonHautDroit = etat;
00489     if(modePilotageBras)
00490         determinerTramePilotageBras();
00491 }
00492
00493 void Manette::changerGachetteBasGauche(double valeur)
00494 {
00495     puissanceGachetteGauche = int(valeur*100);
00496     if (valeur > 0)
00497         maManetteDeplacement.gachetteBasGauche = true;
00498     else
00499         maManetteDeplacement.gachetteBasGauche = false;
00500     if(modeDeplacement)
00501         determinerTrameDeplacementRobot();
00502 }
00503
00504 void Manette::changerGachetteBasDroit(double valeur)
00505 {
00506     puissanceGachetteDroite = int(valeur*100);
00507     if (valeur > 0)
00508         maManetteDeplacement.gachetteBasDroit = true;
00509     else
00510         maManetteDeplacement.gachetteBasDroit = false;
00511     if(modeDeplacement)
00512         determinerTrameDeplacementRobot();
00513 }
00514
00515 void Manette::changerFlecheEnAvant(bool etat)
00516 {
00517     maManettePilotage.flecheEnAvant = etat;
00518     if(modeDeplacement)
00519         determinerTrameDeplacementRobot();
00520     else
00521         determinerTramePilotageBras();
00522 }
00523
00524 void Manette::changerFlecheEnArriere(bool etat)
00525 {
00526     maManettePilotage.flecheEnArriere = etat;
00527     if(modeDeplacement)
00528         determinerTrameDeplacementRobot();
00529     else
00530         determinerTramePilotageBras();
00531 }
00532
00533 void Manette::changerFlecheAGauche(bool etat)
00534 {
00535     maManettePilotage.flecheAGauche = etat;
00536     if(modeDeplacement)
00537         determinerTrameDeplacementRobot();
00538     else
00539         determinerTramePilotageBras();
00540 }
00541
00542 void Manette::changerFlecheADroite(bool etat)
00543 {
00544     maManettePilotage.flecheADroite = etat;
00545     if(modeDeplacement)
00546         determinerTrameDeplacementRobot();
00547     else
```

```
00548         determinerTramePilotageBras();
00549 }
00550
00551 void Manette::changerBoutonA(bool etat)
00552 {
00553     maManetteBouton.boutonA = etat;
00554     if(modePilotageBras)
00555         determinerTrameBouton();
00556 }
00557
00558 void Manette::changerBoutonB(bool etat)
00559 {
00560     maManetteBouton.boutonB = etat;
00561     if(modePilotageBras)
00562         determinerTrameBouton();
00563 }
00564
00565 void Manette::changerBoutonX(bool etat)
00566 {
00567     maManetteBouton.boutonX = etat;
00568     if(modePilotageBras)
00569         determinerTrameBouton();
00570 }
00571
00572 void Manette::changerBoutonY(bool etat)
00573 {
00574     maManetteBouton.boutonY = etat;
00575     if(modePilotageBras)
00576         determinerTrameBouton();
00577 }
00578
00579 void Manette::changerBoutonSelect(bool etat)
00580 {
00581     if(etat)
00582     {
00583         changerMode();
00584     }
00585     qDebug() << Q_FUNC_INFO << "Button Select" << etat ;
00586 }
00587
00588 void Manette::fermerApplication(bool etat)
00589 {
00590     Q_UNUSED(etat);
00591     QCoreApplication::quit();
00592 }
00593
00594 bool EtatManetteBouton::operator==(const
      EtatManetteBouton &maStructure) const
00595 {
00596     if(this->boutonA != maStructure.boutonA)
00597         return false;
00598     else if(this->boutonB != maStructure.boutonB)
00599         return false;
00600     else if(this->boutonX != maStructure.boutonX)
00601         return false;
00602     else if(this->boutonY != maStructure.boutonY)
00603         return false;
00604     else
00605         return true;
00606 }
00607
00608 bool EtatManetteBouton::operator!=(const
      EtatManetteBouton &maStructure) const
00609 {
00610     return !(*this == maStructure);
00611 }
00612
00613 bool EtatManettePilotage::operator==(const
      EtatManettePilotage &maStructure) const
00614 {
00615     if(this->flecheEnAvant != maStructure.flecheEnAvant)
00616         return false;
00617     else if(this->flecheEnArriere != maStructure.flecheEnArriere)
00618         return false;
00619     else if(this->flecheAGauche != maStructure.flecheAGauche)
00620         return false;
00621     else if(this->flecheADroite != maStructure.flecheADroite)
00622         return false;
00623     else if(this->boutonHautGauche != maStructure.boutonHautGauche)
00624         return false;
00625     else if(this->boutonHautDroit != maStructure.boutonHautDroit)
00626         return false;
00627     else
00628         return true;
00629 }
00630
00631 bool EtatManettePilotage::operator!=(const
      EtatManettePilotage &maStructure) const
00632 {
00633     return !(*this == maStructure);
00634 }
```

```
00635
00636 bool EtatManetteDeplacement::operator==(const
      EtatManetteDeplacement &maStructure) const
00637 {
00638     if(this->gachetteBasDroit != maStructure.gachetteBasDroit)
00639         return false;
00640     else if(this->gachetteBasGauche != maStructure.gachetteBasGauche)
00641         return false;
00642     else if(this->joystickGaucheADroite != maStructure.joystickGaucheADroite)
00643         return false;
00644     else if(this->joystickGaucheAGauche != maStructure.joystickGaucheAGauche)
00645         return false;
00646     else if(this->joystickGaucheEnArriere != maStructure.joystickGaucheEnArriere)
00647         return false;
00648     else if(this->joystickGaucheEnAvant != maStructure.joystickGaucheEnAvant)
00649         return false;
00650     else
00651         return true;
00652 }
00653
00654 bool EtatManetteDeplacement::operator!=(const
      EtatManetteDeplacement &maStructure) const
00655 {
00656     return !(*this == maStructure);
00657 }
00658
00659 QString Manette::getNom()
00660 {
00661     return name();
00662 }
```

7.55 Référence du fichier manette.h

Fichier qui contient la déclaration de la classe Manette.

```
#include <QObject>
#include <QtCore/QCoreApplication>
#include <QtGamepad/QGamepad>
#include <QDebug>
```

Classes

- struct EtatManetteBouton
  Structure qui définit l'état de la manette en mode pilotage de la pince.
- struct EtatManetteDeplacement
  Structure qui définit l'état de la manette en mode déplacement du robot.
- struct EtatManettePilotage
  Structure qui définit l'état de la manette en mode pilotage de la pince.
- class Manette
  Classe permettant une communication entre le rov et la manette.

Macros

- #define AVANCER "AVANCER"
  défini la valeur du paramètre de la trame lors du mouvement AVANCER
- #define CAMERA_BAS 1
- #define CAMERA_DROITE 1
  défini la valeur du double retourné par le signal lorsque le joystick de la manette est poussé a fond sur la droite
- #define CAMERA_GAUCHE -1
  défini la valeur du double retourné par le signal lorsque le joystick de la manette est poussé a fond sur la gauche
- #define CAMERA_HAUT -1
  défini la valeur du double retourné par le signal lorsque le joystick de la manette est poussé a fond sur le haut
- #define DESCENDRE "DESCENDRE"
  défini la valeur du paramètre de la trame lors du mouvement DESCENDRE
- #define DROITE "DROITE"
  défini la valeur du paramètre de la trame lors du mouvement DROITE
- #define FERMETURE_PINCE "F"
  défini la valeur du paramètre de la trame lors de la fermeture de la pince
- #define GAUCHE "GAUCHE"

défini la valeur du paramètre de la trame lors du mouvement GAUCHE

— #define IMMOBILE "0"
défini la valeur du paramètre de la trame lors du mouvement IMMOBILE

— #define MONTER "MONTER"
défini la valeur du paramètre de la trame lors du mouvement MONTER

— #define OUVERTURE_PINCE "O"
défini la valeur du paramètre de la trame lors de l'ouverture de la pince

— #define POSER_OBJET_DANS_BAC "BAC"
défini la valeur du paramètre de la trame lorsque l'on pose l'objet dans le bac

— #define RECULER "RECULER"
défini la valeur du paramètre de la trame lors du mouvement RECULER

— #define REDUCTION_VITESSE 0.75
défini le taux de reduction de la puissance lors d'un mouvement

— #define RETOUR_ETAT_INITIAL "INIT"
défini la valeur du paramètre de la trame lors d'un retour à l'etat initial

— #define TAUX_VALIDITE_JOYSTICK 0.50
défini le taux d'acceptation de la valeur du joystick

7.55.1 Description détaillée

Fichier qui contient la déclaration de la classe Manette.

Définition dans le fichier manette.h.

7.55.2 Documentation des macros

7.55.2.1 AVANCER

```
#define AVANCER "AVANCER"
```

défini la valeur du paramètre de la trame lors du mouvement AVANCER

Définition à la ligne 38 du fichier manette.h.

Référencé par Manette : :determinerTramePilotageBras().

7.55.2.2 CAMERA_BAS

```
#define CAMERA_BAS 1
```

Définition à la ligne 116 du fichier manette.h.

7.55.2.3 CAMERA_DROITE

```
#define CAMERA_DROITE 1
```

défini la valeur du double retourné par le signal lorsque le joystick de la manette est poussé a fond sur la droite

Définition à la ligne 98 du fichier manette.h.

7.55.2.4 CAMERA_GAUCHE

```
#define CAMERA_GAUCHE -1
```

défini la valeur du double retourné par le signal lorsque le joystick de la manette est poussé a fond sur la gauche

Définition à la ligne 104 du fichier manette.h.

7.55.2.5 CAMERA_HAUT

```
#define CAMERA_HAUT -1
```

défini la valeur du double retourné par le signal lorsque le joystick de la manette est poussé a fond sur le haut

défini la valeur du double retourné par le signal lorsque le joystick de la manette est poussé a fond sur le bass

7.55.2.6 DESCENDRE

```
#define DESCENDRE "DESCENDRE"
```

défini la valeur du paramètre de la trame lors du mouvement DESCENDRE

Définition à la ligne 50 du fichier manette.h.

Référencé par Manette : :determinerTramePilotageBras().

7.55.2.7 DROITE

```
#define DROITE "DROITE"
```

défini la valeur du paramètre de la trame lors du mouvement DROITE

Définition à la ligne 68 du fichier manette.h.

Référencé par Manette : :determinerTramePilotageBras().

7.55.2.8 FERMETURE_PINCE

```
#define FERMETURE_PINCE "F"
```

défini la valeur du paramètre de la trame lors de la fermeture de la pince

Définition à la ligne 86 du fichier manette.h.

Référencé par Manette : :determinerTrameBouton().

7.55.2.9 GAUCHE

```
#define GAUCHE "GAUCHE"
```

défini la valeur du paramètre de la trame lors du mouvement GAUCHE

Définition à la ligne 62 du fichier manette.h.

Référencé par Manette : :determinerTramePilotageBras().

7.55.2.10 IMMOBILE

```
#define IMMOBILE "0"
```

défini la valeur du paramètre de la trame lors du mouvement IMMOBILE

Définition à la ligne 32 du fichier manette.h.

Référencé par Manette : :determinerTrameBouton(), et Manette : :determinerTramePilotageBras().

7.55.2.11 MONTER

```
#define MONTER "MONTER"
```

défini la valeur du paramètre de la trame lors du mouvement MONTER

Définition à la ligne 56 du fichier manette.h.

Référencé par Manette : :determinerTramePilotageBras().

7.55.2.12 OUVERTURE_PINCE

```
#define OUVERTURE_PINCE "O"
```

défini la valeur du paramètre de la trame lors de l'ouverture de la pince

Définition à la ligne 80 du fichier manette.h.

Référencé par Manette : :determinerTrameBouton().

7.55.2.13 POSER_OBJET_DANS_BAC

```
#define POSER_OBJET_DANS_BAC "BAC"
```

défini la valeur du paramètre de la trame lorsque l'on pose l'objet dans le bac

Définition à la ligne 92 du fichier manette.h.

Référencé par Manette : :determinerTrameBouton().

7.55.2.14 RECULER

```
#define RECULER "RECULER"
```

défini la valeur du paramètre de la trame lors du mouvement RECULER

Définition à la ligne 44 du fichier manette.h.

Référencé par Manette : :determinerTramePilotageBras().

7.55.2.15 REDUCTION_VITESSE

```
#define REDUCTION_VITESSE 0.75
```

défini le taux de reduction de la puissance lors d'un mouvement

Définition à la ligne 26 du fichier manette.h.

Référencé par Manette : :determinerTrameDeplacementRobot().

7.55.2.16 RETOUR_ETAT_INITIAL

```
#define RETOUR_ETAT_INITIAL "INIT"
```

défini la valeur du paramètre de la trame lors d'un retour à l'etat initial

Définition à la ligne 74 du fichier manette.h.

Référencé par Manette : :determinerTrameBouton().

7.55.2.17 TAUX_VALIDITE_JOYSTICK

```
#define TAUX_VALIDITE_JOYSTICK 0.50
```

défini le taux d'acceptation de la valeur du joystick

Définition à la ligne 20 du fichier manette.h.

Référencé par Manette : :changerAxeXJoystickDroit(), Manette : :changerAxeXJoystickGauche(), Manette : :changerAxeYJoystick↩ Droit(), et Manette : :changerAxeYJoystickGauche().

7.56 manette.h

```
00001
00007 #ifndef MANETTE_H
00008 #define MANETTE_H
00009
00010 #include <QObject>
00011 #include <QtCore/QCoreApplication>
00012 #include <QtGamepad/QGamepad>
00013 #include <QDebug>
00014
00020 #define TAUX_VALIDITE_JOYSTICK 0.50
00021
00026 #define REDUCTION_VITESSE 0.75
00027
00032 #define IMMOBILE "0"
00033
00038 #define AVANCER "AVANCER"
00039
00044 #define RECULER "RECULER"
00045
00050 #define DESCENDRE "DESCENDRE"
00051
00056 #define MONTER "MONTER"
00057
00062 #define GAUCHE "GAUCHE"
00063
00068 #define DROITE "DROITE"
00069
00074 #define RETOUR_ETAT_INITIAL "INIT"
00075
00080 #define OUVERTURE_PINCE "O"
00081
00086 #define FERMETURE_PINCE "F"
00087
00092 #define POSER_OBJET_DANS_BAC "BAC"
00093
00098 #define CAMERA_DROITE  1
00099
00104 #define CAMERA_GAUCHE -1
00105
00110 #define CAMERA_HAUT   -1
00111
00116 #define CAMERA_BAS     1
00117
00122 struct EtatManetteBouton
00123 {
00124     bool boutonA;
00125     bool boutonB;
00126     bool boutonX;
00127     bool boutonY;
00128
00135     bool operator==(const EtatManetteBouton &maStructure) const;
00142     bool operator!=(const EtatManetteBouton &maStructure) const;
00143 };
00144
00150 struct EtatManettePilotage
00151 {
00152     bool flecheEnAvant;
00153     bool flecheEnArriere;
00154     bool flecheAGauche;
00155     bool flecheADroite;
00156     bool boutonHautGauche;
00157     bool boutonHautDroit;
00158
00165     bool operator==(const EtatManettePilotage &maStructure) const;
00172     bool operator!=(const EtatManettePilotage &maStructure) const;
00173 };
00178 struct EtatManetteDeplacement
00179 {
00180     bool joystickGaucheEnAvant;
00181     bool joystickGaucheEnArriere;
00182     bool joystickGaucheAGauche;
00183     bool joystickGaucheADroite;
00184     bool gachetteBasGauche;
00185     bool gachetteBasDroit;
00186
00192     bool operator==(const EtatManetteDeplacement &maStructure) const;
00199     bool operator!=(const EtatManetteDeplacement &maStructure) const;
00200 };
00201
00202 class Rov;
00203
00204
00210 class Manette : public QGamepad
00211 {
00212     Q_OBJECT
00213
00214 private:
00215     EtatManetteBouton maManetteBouton;
```

```
00216     EtatManetteBouton ouvrirPince;
00217     EtatManetteBouton fermerPince;
00218     EtatManetteBouton poserObjetDansBac;
00219     EtatManetteBouton revenirEtatInitial;
00220     EtatManetteDeplacement maManetteDeplacement;
00221     EtatManetteDeplacement enAvantCas1;
00222     EtatManetteDeplacement enAvantCas2;
00223     EtatManetteDeplacement enAvantCas3;
00224     EtatManetteDeplacement enArriereCas1;
00225     EtatManetteDeplacement enArriereCas2;
00226     EtatManetteDeplacement enArriereCas3;
00227     EtatManetteDeplacement rotationAGauche;
00228     EtatManetteDeplacement rotationAGaucheDoucementCas1;

00229     EtatManetteDeplacement rotationAGaucheDoucementCas2;

00230     EtatManetteDeplacement rotationADroite;
00231     EtatManetteDeplacement rotationADroiteDoucementCas1;

00232     EtatManetteDeplacement rotationADroiteDoucementCas2;

00233     EtatManetteDeplacement virageAvantAGauche;
00234     EtatManetteDeplacement virageAvantAGaucheDoucementCas1
      ;
00235     EtatManetteDeplacement virageAvantAGaucheDoucementCas2
      ;
00236     EtatManetteDeplacement virageAvantADroite;
00237     EtatManetteDeplacement virageAvantADroiteDoucementCas1
      ;
00238     EtatManetteDeplacement virageAvantADroiteDoucementCas2
      ;
00239     EtatManetteDeplacement virageArriereAGauche;
00240     EtatManetteDeplacement
      virageArriereAGaucheDoucementCas1;
00241     EtatManetteDeplacement
      virageArriereAGaucheDoucementCas2;
00242     EtatManetteDeplacement virageArriereADroite;
00243     EtatManetteDeplacement
      virageArriereADroiteDoucementCas1;
00244     EtatManetteDeplacement
      virageArriereADroiteDoucementCas2;
00245
00246     EtatManettePilotage maManettePilotage;
00247     EtatManettePilotage avance;
00248     EtatManettePilotage recule;
00249     EtatManettePilotage gauche;
00250     EtatManettePilotage droite;
00251     EtatManettePilotage monte;
00252     EtatManettePilotage descend;
00253     bool modeDeplacement;
00254     bool modePilotageBras;
00255     int puissanceGachetteDroite;
00256     int puissanceGachetteGauche;
00257     QString joystickAxeXCamera;
00258     QString joystickAxeYCamera;
00259
00264     void initialiserEtats();
00269     void initialiserTypesPilotage();
00274     void initialiserTypesDeplacement();
00279     void initialisationStructuresEnAvant();
00284     void initialisationStructuresEnArriere();
00289     void initialisationStructureRotationAGauche();
00294     void initialisationStructuresRotationAGaucheDoucement();
00299     void initialisationStructureRotationADroite();
00304     void initialisationStructuresRotationADroiteDoucement();
00309     void initialisationStructureVirageAvantAGauche();
00314     void initialisationStructuresVirageAvantAGaucheDoucement();
00319     void initialisationStructureVirageAvantADroite();
00324     void initialisationStructuresVirageAvantADroiteDoucement();
00329     void initialisationStructureVirageArriereAGauche();
00334     void initialisationStructuresVirageArriereAGaucheDoucement();
00339     void initialisationStructureVirageArriereADroite();
00344     void initialisationStructuresVirageArriereADroiteDoucement();
00349     void initialiserEtatBouton();
00350
00351 public:
00357     explicit Manette(int deviceId);
00362     ~Manette();
00363
00368     void determinerTrameDeplacementRobot();
00373     void determinerTramePilotageBras();
00378     void determinerTrameBouton();
00383     void changerMode();
00389     QString getNom();
00390
00391 signals:
00399     void creationTrameDeplacement(char deplacementAxeX, int puissance, char deplacementAxeY);
00405     void creationTramePilotage(QString direction);
00410     void prendrePhoto();
00416     void creationTrameOrdre(QString ordre);
00421     void creationTramePince(QString mouvementPince);
```

```
00428     void nouvelleTrameCamera(QString axeY, QString axeX);
00429
00430 public slots:
00436     void changerAxeXJoystickGauche(double valeur);
00442     void changerAxeYJoystickGauche(double valeur);
00448     void changerAxeXJoystickDroit(double valeur);
00454     void changerAxeYJoystickDroit(double valeur);
00459     void changerBoutonHautGauche(bool etat);
00464     void changerBoutonHautDroit(bool etat);
00470     void changerGachetteBasGauche(double valeur);
00476     void changerGachetteBasDroit(double valeur);
00482     void changerFlecheEnAvant(bool etat);
00488     void changerFlecheEnArriere(bool etat);
00494     void changerFlecheAGauche(bool etat);
00500     void changerFlecheADroite(bool etat);
00505     void changerBoutonA(bool);
00510     void changerBoutonB(bool);
00515     void changerBoutonX(bool);
00520     void changerBoutonY(bool);
00526     void changerBoutonSelect(bool etat);
00532     void fermerApplication(bool etat);
00533 };
00534
00535 #endif // MANETTE_H
```

7.57 Référence du fichier README.md

7.58 README.md

```
00001 \mainpage Le projet
00002
00003 \tableofcontents
00004
00005 \section section_tdm Table des matières
00006 - \ref page_README
00007 - \ref page_changelog
00008 - \ref page_about
00009 - \ref page_licence
00010
00011 \section section_infos Informations
00012
00013 \author Servan Tenaille <<servan.tenaille@gmail.com>>
00014 \author Anthony Bonnet <<bonnet.anthony0@gmail.com>>
00015 \date 2020
00016 \version 0.2
00017 \see https://svn.riouxsvn.com/rovnet/
00018
00019
00020 \page page_README README
00021
00022 [TOC]
00023
00024 # Projet {#projet}
00025
00026 ## Présentation {#presentation}
00027
00028 Les objectifs du projet ROV'NET sont de se déplacer dans un milieu contaminé afin de faire des prises
       de vues :
00029
00030 * Le déplacement se fera à partir d'un châssis en liaison filaire à 4 roues motorisées
       indépendamment.
00031 * Le ROV sera équipé :
00032     * d'une caméra d'aide au déplacements et/ou de capteurs d'obstacles
00033     * d'un capteur de température et de radioactivité
00034     * d'un dispositif de prise de vue motorisé
00035     * d'un bras de robotique avec pince de préhension
00036
00037 ## Base de données SQLite {#bdd}
00038
00039 ![](./sql/campagnes-v1.1.png)
00040
00041 ~~~ {.sql}
00042 PRAGMA foreign_keys = ON;
00043
00044 --
00045 -- Structure de la table `campagne`
00046 --
00047
00048 CREATE TABLE IF NOT EXISTS `campagne` (
00049     `IdCampagne`    INTEGER PRIMARY KEY AUTOINCREMENT,
00050     `IdTechnicien`  INTEGER NOT NULL,
00051     `nom`   TEXT NOT NULL UNIQUE,
00052     `lieu`  TEXT NOT NULL,
00053     `cheminSauvegarde`  TEXT NOT NULL,
```

```
00054    `date`  DATETIME NOT NULL,
00055    `duree` INTEGER NOT NULL,
00056    `enCours` NUMERIC NOT NULL,
00057    UNIQUE(`IdCampagne`, `IdTechnicien`),
00058    FOREIGN KEY(IdTechnicien) REFERENCES technicien(IdTechnicien)
00059 );
00060
00061 --
00062 -- Structure de la table `mesure`
00063 --
00064
00065 CREATE TABLE IF NOT EXISTS `mesure` (
00066    `IdMesure`  INTEGER PRIMARY KEY AUTOINCREMENT,
00067    `IdCampagne`    INTEGER NOT NULL,
00068    `heure` DATETIME NOT NULL,
00069    `temperature`   REAL NOT NULL,
00070    `radiation` REAL NOT NULL,
00071    `humidite`  REAL NOT NULL,
00072    UNIQUE(`IdMesure`, `IdCampagne`),
00073    FOREIGN KEY(IdCampagne) REFERENCES campagne(IdCampagne)
00074 );
00075
00076 --
00077 -- Structure de la table `photo`
00078 --
00079
00080 CREATE TABLE IF NOT EXISTS `photo` (
00081    `IdPhoto`   INTEGER PRIMARY KEY AUTOINCREMENT,
00082    `IdCampagne`    INTEGER NOT NULL,
00083    `cheminImage`   TEXT NOT NULL,
00084    `aGarder`   NUMERIC NOT NULL,
00085    UNIQUE(`IdPhoto`, `IdCampagne`),
00086    FOREIGN KEY(IdCampagne) REFERENCES campagne(IdCampagne)
00087 );
00088
00089 --
00090 -- Structure de la table `technicien`
00091 --
00092
00093 CREATE TABLE IF NOT EXISTS `technicien` (
00094    `IdTechnicien`  INTEGER PRIMARY KEY AUTOINCREMENT,
00095    `nom`   TEXT NOT NULL,
00096    `prenom`    TEXT NOT NULL
00097 );
00098 ~~~
00099
00100 ## Recette {#recette}
00101
00102 - Servan Tenaille
00103
00104     * Prendre en charge une manette par le logiciel
00105     * Recevoir et Visualiser les mesures des capteurs de température et d'irradiation
00106     * Déplacer le robot
00107     * Piloter le bras articulé
00108     * Envoyer les ordres de déplacement au robot et au bras
00109     * Archiver les mesures
00110
00111 - Anthony Bonnet
00112
00113     * Démarrer une campagne
00114     * Visualiser l'environnement (le flux vidéo de la caméra et les données de télémétrie)
00115     * Recevoir les données de télémétrie
00116     * Prendre une photo
00117     * Configurer le contrôle de la caméra
00118     * Archiver les photos
00119
00120 ## Informations {#informations}
00121
00122 \author Servan Tenaille <<servan.tenaille@gmail.com>>
00123 \author Anthony Bonnet <<bonnet.anthony0@gmail.com>>
00124 \date 2020
00125 \version 0.2
00126 \see https://svn.riouxsvn.com/rovnet/
00127
00128
00129 \page page_about A propos
00130
00131 \author Servan Tenaille <<servan.tenaille@gmail.com>>
00132 \author Anthony Bonnet <<bonnet.anthony0@gmail.com>>
00133 \date 2020
00134 \version 0.2
00135 \see https://svn.riouxsvn.com/rovnet/
00136
00137
00138 \page page_licence Licence GPL
00139
00140 This program is free software; you can redistribute it and/or modify
00141 it under the terms of the GNU General Public License as published by
00142 the Free Software Foundation; either version 2 of the License, or
00143 (at your option) any later version.
00144
```

```
00145 This program is distributed in the hope that it will be useful,
00146 but WITHOUT ANY WARRANTY; without even the implied warranty of
00147 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00148 GNU General Public License for more details.
00149
00150 You should have received a copy of the GNU General Public License
00151 along with this program; if not, write to the Free Software
00152 Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

7.59 Référence du fichier rov.cpp

Fichier qui contient la définition de la classe Rov.

```
#include "rov.h"
#include "campagne.h"
#include <iomanip>
```

7.59.1 Description détaillée

Fichier qui contient la définition de la classe Rov.

Définition dans le fichier rov.cpp.

7.60 rov.cpp

```
00001
00007 #include "rov.h"
00008 #include "campagne.h"
00009 #include <iomanip>
00010
00011 Rov::Rov(IHMRov *ihmRov, QObject *parent) : QObject(parent), ihmRov(ihmRov),
      camera(nullptr), trameDeplacement("$DEP;0;0;0\r\n"), tramePrecedenteDeplacement("$DEP;0;0;0\r\n"),
      tramePilotage("$BRAS;0\r\n"), tramePrecedentePilotage("$BRAS;0\r\n"), tramePince("$PINCE;0\r\n"), tramePrecedentePince(
      "$PINCE;0\r\n"), tramePrecedenteCamera(TRAME_CAMERA_IMMOBILE),
00012 campagneEnCours(false)
00013 {
00014     qDebug() << Q_FUNC_INFO;
00015     communicationRov = new CommunicationRov(this);
00016     capteurs = new Capteurs(this);
00017
00018     initialiserEvenements();
00019     detecterManette();
00020 }
00021
00022 Rov::~Rov()
00023 {
00024     supprimerManette();
00025     arreterCampagne();
00026     qDebug() << Q_FUNC_INFO;
00027 }
00028
00029 void Rov::detecterManette()
00030 {
00031     listeManettes = QGamepadManager::instance()->connectedGamepads();
00032
00033     if (listeManettes.isEmpty())
00034     {
00035         qDebug() << Q_FUNC_INFO << "Aucune manette détectée !";
00036     }
00037     else
00038     {
00039         qDebug() << Q_FUNC_INFO << "Nombre de manettes" << listeManettes.size();
00040     }
00041
00042     for (auto m : listeManettes)
00043     {
00044         qDebug() << Q_FUNC_INFO << "-> Manette" << m;
00045         Manette *manette = new Manette(m);
00046         manettes.push_back(manette);
00047         initialiserEvenementManette(manette);
00048     }
00049 }
```

```
00050
00051 void Rov::initialiserEvenementManette(Manette *manette)
00052 {
00053     connect(manette, SIGNAL(creationTrameDeplacement(char, int, char)), this, SLOT(
      creerTrameDeplacement(char, int, char)));
00054     connect(manette, SIGNAL(creationTramePilotage(QString)), this, SLOT(
      creerTramePilotage(QString)));
00055     connect(manette, SIGNAL(creationTrameOrdre(QString)), this, SLOT(
      creerTrameOrdre(QString)));
00056     connect(manette, SIGNAL(creationTramePince(QString)), this, SLOT(
      creerTramePince(QString)));
00057     connect(manette, SIGNAL(nouvelleTrameCamera(QString,QString)), this, SLOT(
      creerTrameCamera(QString,QString)));
00058
00059     connect(manette, SIGNAL(axisLeftXChanged(double)), manette, SLOT(changerAxeXJoystickGauche(double)));
00060     connect(manette, SIGNAL(axisLeftYChanged(double)), manette, SLOT(changerAxeYJoystickGauche(double)));
00061     connect(manette, SIGNAL(axisRightXChanged(double)), manette, SLOT(changerAxeXJoystickDroit(double)));
00062     connect(manette, SIGNAL(axisRightYChanged(double)), manette, SLOT(changerAxeYJoystickDroit(double)));
00063     connect(manette, SIGNAL(buttonL1Changed(bool)), manette, SLOT(changerBoutonHautGauche(bool)));
00064     connect(manette, SIGNAL(buttonR1Changed(bool)), manette, SLOT(changerBoutonHautDroit(bool)));
00065     connect(manette, SIGNAL(buttonL2Changed(double)), manette, SLOT(changerGachetteBasGauche(double)));
00066     connect(manette, SIGNAL(buttonR2Changed(double)), manette, SLOT(changerGachetteBasDroit(double)));
00067     connect(manette, SIGNAL(buttonUpChanged(bool)), manette, SLOT(changerFlecheEnAvant(bool)));
00068     connect(manette, SIGNAL(buttonDownChanged(bool)), manette, SLOT(changerFlecheEnArriere(bool)));
00069     connect(manette, SIGNAL(buttonLeftChanged(bool)), manette, SLOT(changerFlecheAGauche(bool)));
00070     connect(manette, SIGNAL(buttonRightChanged(bool)), manette, SLOT(changerFlecheADroite(bool)));
00071     connect(manette, SIGNAL(buttonAChanged(bool)), manette, SLOT(changerBoutonA(bool)));
00072     connect(manette, SIGNAL(buttonBChanged(bool)), manette, SLOT(changerBoutonB(bool)));
00073     connect(manette, SIGNAL(buttonXChanged(bool)), manette, SLOT(changerBoutonX(bool)));
00074     connect(manette, SIGNAL(buttonYChanged(bool)), manette, SLOT(changerBoutonY(bool)));
00075     connect(manette, SIGNAL(buttonSelectChanged(bool)), manette, SLOT(changerBoutonSelect(bool)));
00076     connect(manette, SIGNAL(buttonGuideChanged(bool)), manette, SLOT(fermerApplication(bool)));
00077     connect(manette, SIGNAL(buttonR3Changed(bool)), ihmRov, SLOT(capturerImage(bool)));
00078
00079     connect(manette, SIGNAL(connectedChanged(bool)), ihmRov, SLOT(modifieEtatManette(bool)));
00080 }
00081
00082 void Rov::decoderTrameTelemetrie(QString trameTelemetrie)
00083 {
00084     capteurs->setTelemetrie(trameTelemetrie.section(";",1,1));
00085     capteurs->setAngle(trameTelemetrie.section(";",2,2));
00086 }
00087
00088 void Rov::decoderTrameCapteur(QString trameCapteur)
00089 {
00090     QString temperature, humidite, radiation;
00091
00092     temperature = QString::number(trameCapteur.section(";",1,1).toDouble()/10);
00093     humidite = trameCapteur.section(";",2,2);
00094     radiation = trameCapteur.section(";",3,3);
00095
00096     capteurs->setTemperature(temperature);
00097     capteurs->setHumidite(humidite);
00098     capteurs->setRadiation(radiation);
00099
00100     Mesure mesure;
00101     mesure.dateheure = QDateTime::currentDateTime();
00102     mesure.temperature = temperature;
00103     mesure.humidite = humidite;
00104     mesure.radiation = radiation;
00105
00106     QDateTime date;
00107
00108     ihmRov->getCampagne()->ajouterMesure(mesure);
00109     ihmRov->actualiserInformationsSeuils();
00110
00111     emit enregistrerMesures(temperature, humidite, radiation);
00112
00113 }
00114
00115 void Rov::initialiserEvenements()
00116 {
00117     connect(communicationRov, SIGNAL(nouvelleTrame(QString)), this, SLOT(
      decoderTrame(QString)));
00118     connect(communicationRov, SIGNAL(etatPortModifie(bool, QString)),
      ihmRov, SLOT(modifieEtatPortSerie(bool, QString)));
00119 }
00120
00121 void Rov::supprimerManette()
00122 {
00123     for(int i=0;i<manettes.size();i++)
00124         delete manettes[i];
00125 }
00126
00127 bool Rov::demarrerCampagne()
00128 {
00129     qDebug() << Q_FUNC_INFO;
00130     if(Camera::getNbCameras() > 0)
00131     {
00132         dureeCampagne.start();
00133         QString nom = Camera::creerNomCamera(CAMERA_DEFAUT);
```

```
00134         if(demarrerVideo(nom))
00135             campagneEnCours = true;
00136     }
00137     return campagneEnCours;
00138 }
00139
00140 void Rov::arreterCampagne()
00141 {
00142     qDebug() << Q_FUNC_INFO;
00143     ihmRov->getCampagne()->setDuree(dureeCampagne.elapsed());
00144     arreterVideo();
00145     campagneEnCours = false;
00146 }
00147
00148 Camera* Rov::getCamera()
00149 {
00150     return camera;
00151 }
00152
00153 Capteurs* Rov::getCapteurs()
00154 {
00155     return capteurs;
00156 }
00157
00158 QString Rov::getTempsCampagne()
00159 {
00160     QTime duree(0,0);
00161     QTime dureeMission = duree.addMSecs(dureeCampagne.elapsed() +
      ihmRov->getCampagne()->getDuree());
00162     return dureeMission.toString("hh:mm:ss");
00163 }
00164
00165 bool Rov::demarrerVideo(QString nomCamera, int choixResolution)
00166 {
00167     qDebug() << Q_FUNC_INFO << "nomCamera" << nomCamera << "choixResolution" << choixResolution;
00168     if(camera == nullptr)
00169     {
00170         camera = new Camera(this, nomCamera, choixResolution);
00171         connect(camera, SIGNAL(nouvelleImage(QPixmap)), ihmRov, SLOT(afficherImage(QPixmap)));
00172         ihmRov->initialiserEvenementsCamera();
00173         camera->start();
00174         ihmRov->modifieEtatCamera(true, nomCamera);
00175         return true;
00176     }
00177     return false;
00178 }
00179
00180 bool Rov::getCampagneEncours() const
00181 {
00182     return campagneEnCours;
00183 }
00184
00185 void Rov::decoderTrame(QString trame)
00186 {
00187     QString trameTelemetrie, trameCapteur;
00188
00189     QStringList trames = trame.split("\r\n");
00190     qDebug() << Q_FUNC_INFO << "trame" << trames.size();
00191     for(int i = 0; i < trames.size(); i++)
00192     {
00193         if(!trames[i].isEmpty())
00194         {
00195             if(trames[i].startsWith(DEBUT_TRAME_TELEMETRIE))
00196             {
00197                 trameTelemetrie = trames[i];
00198                 decoderTrameTelemetrie(trameTelemetrie);
00199             }
00200             else if(trames[i].startsWith(DEBUT_TRAME_CAPTEUR))
00201             {
00202                 trameCapteur = trames[i];
00203                 decoderTrameCapteur(trameCapteur);
00204             }
00205         }
00206         else
00207         {
00208             qDebug() << Q_FUNC_INFO << "Trame inconnue !";
00209         }
00210     }
00211 }
00212
00213 void Rov::creerTrameDeplacement(char deplacementAxeX, int puissance, char
      deplacementAxeY)
00214 {
00215     trameDeplacement = DEBUT_TRAME_DEPLACEMENT ";" + QString(
      deplacementAxeX) + ";" + QString::number(puissance) + ";" + QString(deplacementAxeY) + "\r\n";
00216
00217     if(tramePrecedenteDeplacement != trameDeplacement)
00218     {
00219         if(TRAME_DEPLACEMENT_IMMOBILE ==
      trameDeplacement)
00220             ihmRov->setEtatRadar(true);
```

```
00221         else
00222             ihmRov->setEtatRadar(false);
00223         qDebug() << Q_FUNC_INFO << "trameDeplacement :" << trameDeplacement;
00224         tramePrecedenteDeplacement = trameDeplacement;
00225         communicationRov->emettreTrame(trameDeplacement);
00226     }
00227 }
00228
00229 void Rov::creerTramePilotage(QString deplacement)
00230 {
00231     tramePilotage = DEBUT_TRAME_PILOTAGE ";" + deplacement + "\r\n";
00232     if(tramePrecedentePilotage != tramePilotage &&
      tramePince == TRAME_PINCE_IMMOBILE)
00233     {
00234         qDebug() << Q_FUNC_INFO << "tramePilotage :" << tramePilotage;
00235         tramePrecedentePilotage = tramePilotage;
00236         communicationRov->emettreTrame(tramePilotage);
00237     }
00238 }
00239
00240 void Rov::creerTrameOrdre(QString ordre)
00241 {
00242     trameOrdre = DEBUT_TRAME_ORDRE ";" + ordre + "\r\n";
00243     if(tramePrecedentePilotage == TRAME_PILOTAGE_IMMOBILE &&
      tramePince == TRAME_PINCE_IMMOBILE)
00244     {
00245         qDebug() << Q_FUNC_INFO << "trameOrdre :" << trameOrdre;
00246         communicationRov->emettreTrame(trameOrdre);
00247     }
00248 }
00249
00250 void Rov::creerTramePince(QString mouvementPince)
00251 {
00252     tramePince = DEBUT_TRAME_PINCE ";" + mouvementPince + "\r\n";
00253     if(tramePrecedentePilotage == TRAME_PILOTAGE_IMMOBILE &&
      tramePrecedentePince != tramePince)
00254     {
00255         qDebug() << Q_FUNC_INFO << "tramePince :" << tramePince;
00256         tramePrecedentePince = tramePince;
00257         communicationRov->emettreTrame(tramePince);
00258     }
00259 }
00260
00261 void Rov::creerTrameCamera(QString axeX, QString axeY)
00262 {
00263     QString trameCamera = DEBUT_TRAME_CAMERA ";" + axeX + ";" + axeY + "\r\n";
00264
00265     if(tramePrecedenteCamera != trameCamera)
00266     {
00267         tramePrecedenteCamera = trameCamera;
00268         communicationRov->emettreTrame(trameCamera);
00269         qDebug() << Q_FUNC_INFO << "trame camera" << trameCamera;
00270     }
00271 }
00272
00273 void Rov::arreterVideo()
00274 {
00275     if(camera != nullptr)
00276     {
00277         qDebug() << Q_FUNC_INFO << "isRunning" << camera->isRunning();
00278         ihmRov->arreterVideo();
00279         disconnect(camera, SIGNAL(nouvelleImage(QPixmap)), ihmRov, SLOT(afficherImage(QPixmap))
      );
00280         camera->requestInterruption();
00281         camera->wait();
00282         delete camera;
00283         camera = nullptr;
00284         ihmRov->modifieEtatCamera(false, "");
00285     }
00286 }
00287
00288 void Rov::modifierConfiguration(Configuration &configuration)
00289 {
00290     communicationRov->setConfiguration(configuration);
00291 }
00292
00293 CommunicationRov* Rov::getCommunicationRov()
00294 {
00295     return communicationRov;
00296 }
00297
00298 QVector<Manette*> Rov::getManettes()
00299 {
00300     return manettes;
00301 }
```

7.61 Référence du fichier rov.h

Fichier qui contient la déclaration de la classe rov.

```
#include <QObject>
#include "camera.h"
#include "communicationrov.h"
#include "manette.h"
#include "capteurs.h"
#include "ihmrov.h"
```

Classes

— class Rov
Classe controlant tout les traitements en provenance et en direction de la communication avec le rov.

Macros

— #define DEBUT_TRAME_CAMERA "$CAM"
Constante contenant le début de la trame caméra selon le protocole.
— #define DEBUT_TRAME_CAPTEUR "$ENVI"
Constante contenant le début de la trame capteur selon le protocole.
— #define DEBUT_TRAME_DEPLACEMENT "$DEP"
Constante contenant le début de la trame déplacement selon le protocole.
— #define DEBUT_TRAME_ORDRE "$ORDRE"
Constante contenant le début de la trame ordre selon le protocole.
— #define DEBUT_TRAME_PILOTAGE "$BRAS"
Constante contenant le début de la trame pilotage selon le protocole.
— #define DEBUT_TRAME_PINCE "$PINCE"
Constante contenant le début de la trame pince selon le protocole.
— #define DEBUT_TRAME_TELEMETRIE "$TEL"
Constante contenant le début de la trame telemetrie selon le protocole.
— #define TRAME_CAMERA_IMMOBILE "$CAM ;0 ;0\r\n"
Constante contenant la trame pour immobiliser la caméra.
— #define TRAME_DEPLACEMENT_IMMOBILE "$DEP ;0 ;0 ;0\r\n"
— #define TRAME_PILOTAGE_IMMOBILE "$BRAS ;0\r\n"
Constante contenant la trame pilotage à zéro selon le protocole.
— #define TRAME_PINCE_IMMOBILE "$PINCE ;0\r\n"
Constante contenant la trame pince à zéro selon le protocole.

7.61.1 Description détaillée

Fichier qui contient la déclaration de la classe rov.

Définition dans le fichier rov.h.

7.61.2 Documentation des macros

7.61.2.1 DEBUT_TRAME_CAMERA

```
#define DEBUT_TRAME_CAMERA "$CAM"
```

Constante contenant le début de la trame caméra selon le protocole.

Définition à la ligne 71 du fichier rov.h.

Référencé par Rov : :creerTrameCamera().

7.61.2.2 DEBUT_TRAME_CAPTEUR

```
#define DEBUT_TRAME_CAPTEUR "$ENVI"
```

Constante contenant le début de la trame capteur selon le protocole.

Définition à la ligne 28 du fichier rov.h.

Référencé par Rov : :decoderTrame().

7.61.2.3 DEBUT_TRAME_DEPLACEMENT

```
#define DEBUT_TRAME_DEPLACEMENT "$DEP"
```

Constante contenant le début de la trame déplacement selon le protocole.

Définition à la ligne 34 du fichier rov.h.

Référencé par Rov : :creerTrameDeplacement().

7.61.2.4 DEBUT_TRAME_ORDRE

```
#define DEBUT_TRAME_ORDRE "$ORDRE"
```

Constante contenant le début de la trame ordre selon le protocole.

Définition à la ligne 46 du fichier rov.h.

Référencé par Rov : :creerTrameOrdre().

7.61.2.5 DEBUT_TRAME_PILOTAGE

```
#define DEBUT_TRAME_PILOTAGE "$BRAS"
```

Constante contenant le début de la trame pilotage selon le protocole.

Définition à la ligne 40 du fichier rov.h.

Référencé par Rov : :creerTramePilotage().

7.61.2.6 DEBUT_TRAME_PINCE

```
#define DEBUT_TRAME_PINCE "$PINCE"
```

Constante contenant le début de la trame pince selon le protocole.

Définition à la ligne 52 du fichier rov.h.

Référencé par Rov : :creerTramePince().

7.61.2.7 DEBUT_TRAME_TELEMETRIE

```
#define DEBUT_TRAME_TELEMETRIE "$TEL"
```

Constante contenant le début de la trame telemetrie selon le protocole.

Définition à la ligne 22 du fichier rov.h.

Référencé par Rov : :decoderTrame().

7.61.2.8 TRAME_CAMERA_IMMOBILE

```
#define TRAME_CAMERA_IMMOBILE "$CAM;0;0\r\n"
```

Constante contenant la trame pour immobiliser la caméra.

Définition à la ligne 77 du fichier rov.h.

7.61.2.9 TRAME_DEPLACEMENT_IMMOBILE

```
#define TRAME_DEPLACEMENT_IMMOBILE "$DEP;0;0;0\r\n"
```

Définition à la ligne 53 du fichier rov.h.

Référencé par Rov : :creerTrameDeplacement().

7.61.2.10 TRAME_PILOTAGE_IMMOBILE

```
#define TRAME_PILOTAGE_IMMOBILE "$BRAS;0\r\n"
```

Constante contenant la trame pilotage à zéro selon le protocole.

Définition à la ligne 59 du fichier rov.h.

Référencé par Rov : :creerTrameOrdre(), et Rov : :creerTramePince().

7.61.2.11 TRAME_PINCE_IMMOBILE

```
#define TRAME_PINCE_IMMOBILE "$PINCE;0\r\n"
```

Constante contenant la trame pince à zéro selon le protocole.

Définition à la ligne 65 du fichier rov.h.

Référencé par Rov : :creerTrameOrdre(), et Rov : :creerTramePilotage().

7.62 rov.h

```
00001
00007 #ifndef ROV_H
00008 #define ROV_H
00009
00010 #include <QObject>
00011 #include "camera.h"
00012 #include "communicationrov.h"
00013 #include "manette.h"
00014 #include "capteurs.h"
00015 #include "ihmrov.h"
00016
00022 #define DEBUT_TRAME_TELEMETRIE "$TEL"
00023
00028 #define DEBUT_TRAME_CAPTEUR    "$ENVI"
00029
00034 #define DEBUT_TRAME_DEPLACEMENT "$DEP"
00035
00040 #define DEBUT_TRAME_PILOTAGE "$BRAS"
00041
00046 #define DEBUT_TRAME_ORDRE "$ORDRE"
00047
00052 #define DEBUT_TRAME_PINCE "$PINCE"
00053 #define TRAME_DEPLACEMENT_IMMOBILE "$DEP;0;0;0\r\n"
00054
00059 #define TRAME_PILOTAGE_IMMOBILE "$BRAS;0\r\n"
00060
00065 #define TRAME_PINCE_IMMOBILE "$PINCE;0\r\n"
00066
00071 #define DEBUT_TRAME_CAMERA "$CAM"
00072
00077 #define TRAME_CAMERA_IMMOBILE "$CAM;0;0\r\n"
00078
00079 class Camera;
00080 class CommunicationRov;
00081 class Manette;
00082 class Capteurs;
00083 class IHMRov;
00084 class QCameraInfo;
00085
00091 class Rov : public QObject
00092 {
00093     Q_OBJECT
00094 private:
00095     IHMRov *ihmRov;
00096     Capteurs *capteurs;
00097     Camera *camera;
00098     CommunicationRov *communicationRov;
00099     QList<int> listeManettes;
00100     QVector<Manette*> manettes;
00101     QString trameDeplacement;
00102     QString tramePrecedenteDeplacement;
00103     QString tramePilotage;
00104     QString tramePrecedentePilotage;
00105     QString trameOrdre;
00106     QString tramePince;
00107     QString tramePrecedentePince;
00108     QString tramePrecedenteCamera;
00109     QString tramePinceImmobile;
00110     QTime dureeCampagne;
00111     bool campagneEnCours;
00112
00117     void detecterManette();
00123     void initialiserEvenementManette(Manette *manette);
00129     void decoderTrameTelemetrie(QString trameTelemetrie);
00135     void decoderTrameCapteur(QString trameCapteur);
00140     void initialiserEvenements();
00145     void supprimerManette();
00146
00147 public:
00154     Rov(IHMRov *ihmRov, QObject *parent = nullptr);
00159     ~Rov();
00160
00165     bool demarrerCampagne();
00170     void arreterCampagne();
00176     Camera* getCamera();
00182     Capteurs* getCapteurs();
00188     QString getTempsCampagne();
00195     bool demarrerVideo(QString nomCamera, int choixResolution=-1);
00201     bool getCampagneEncours() const;
00207     void modifierConfiguration(Configuration &configuration);
00213     CommunicationRov* getCommunicationRov();
00219     QVector<Manette*> getManettes();
00220
00221 signals:
00227     void donneesTelemetrieDecode(QString donnes);
00235     void enregistrerMesures(QString temperature, QString humidite, QString radiation);
00241     void styleRadar(bool etat);
00242
```

```
00243 public slots:
00249     void decoderTrame(QString trame);
00254     void creerTrameDeplacement(char deplacementAxeX, int puissance, char
      deplacementAxeY);
00260     void creerTramePilotage(QString deplacement);
00266     void creerTrameOrdre(QString ordre);
00272     void creerTramePince(QString mouvementPince);
00279     void creerTrameCamera(QString axeX, QString axeY);
00284     void arreterVideo();
00285 };
00286
00287 #endif // ROV_H
```