Projet ROV'NET

1.1

BTS SN-IR LaSalle Avignon 2019

REYNIER Jacques
BOFFREDO Nicolas

Table des matières

1 Le projet ROV'NET

Le système ROV'NET est capable d'obtenir et gérer des données d'entrée afin de définir un environnement TQC (Tel Que Construit) en se déplaçant dans un milieu contaminé.

Auteur

BOFFREDO Nicolas nboffredo@gmail.com
REYNIER Jacques reynier.jacques@gmail.com

1.1 Table des matières

2 Changelog

r137 | jreynier | 2019-06-05 16 :16 :38 +0200 (mer. 05 juin 2019) | 1 ligne

Modification des déplacements (provisoire).

r136 | nboffredo | 2019-06-04 17 :17 :17 +0200 (mar. 04 juin 2019) | 1 ligne

Correction trame déplacement caméra

r135 | nboffredo | 2019-06-04 17 :06 :42 +0200 (mar. 04 juin 2019) | 1 ligne

Modificatoin trame déplacement caméra

r134 | nboffredo | 2019-06-04 16 :56 :03 +0200 (mar. 04 juin 2019) | 1 ligne

Correction liste des caméras

r133 | jreynier | 2019-06-04 16 :26 :53 +0200 (mar. 04 juin 2019) | 1 ligne

Ajout de la fenêtre de Paramètres à la création d'une nouvelle campagne.

r132 | nboffredo | 2019-06-04 15 :07 :56 +0200 (mar. 04 juin 2019) | 1 ligne

Correction d'un crash + départ du chronomètre mis au départ d'une nouvelle campagne

r131 | nboffredo | 2019-06-04 13 :57 :00 +0200 (mar. 04 juin 2019) | 1 ligne

Correction crash au démarrage d'une nouvelle campagne

r130 | nboffredo | 2019-06-04 12 :14 :42 +0200 (mar. 04 juin 2019) | 1 ligne

Correction bug remplissage liste caméra IHM

r129 | jreynier | 2019-06-04 12 :03 :19 +0200 (mar. 04 juin 2019) | 1 ligne

Modification de la fenêtre Paramètres.

r128 | jreynier | 2019-06-04 09 :09 :33 +0200 (mar. 04 juin 2019) | 1 ligne

Ajout d'une fenetre d'erreur lorsque l'on choisi un port indisponible dans la fenetre de parametres + affichage du parametrage dés la création de la campagne.

r127 | nboffredo | 2019-06-04 08 :22 :59 +0200 (mar. 04 juin 2019) | 1 ligne

Correction bug démarrage caméra

r126 | nboffredo | 2019-06-04 08 :18 :16 +0200 (mar. 04 juin 2019) | 1 ligne

mise à jour de la base de donnée (correction des id)

r125 | nboffredo | 2019-06-04 08 :12 :06 +0200 (mar. 04 juin 2019) | 1 ligne

suppression d'un operateur

r124 | jreynier | 2019-05-29 17 :41 :42 +0200 (mer. 29 mai 2019) | 1 ligne

Modification envoi de trames

r123 | nboffredo | 2019-05-29 11 :52 :08 +0200 (mer. 29 mai 2019) | 1 ligne

Suppression d'un operteur dans la table operateur de la BDD

r122 | jreynier | 2019-05-29 11 :46 :17 +0200 (mer. 29 mai 2019) | 1 ligne

Modification de noms de signaux.

r121 | jreynier | 2019-05-28 17 :20 :20 +0200 (mar. 28 mai 2019) | 1 ligne

Ajout de l'actualisation des icones d'état dans l'IHM.

r120 | jreynier | 2019-05-28 13 :54 :21 +0200 (mar. 28 mai 2019) | 1 ligne

Mise en fonction de la modification de l'intervalle d'archivage dans la fenetre de parametres.

r119 | jreynier | 2019-05-28 12 :25 :35 +0200 (mar. 28 mai 2019) | 1 ligne

Mise en fonction du bouton Archiver mesures dans la fenetre de parametres.

r118 | tvaira | 2019-05-27 18 :30 :13 +0200 (lun. 27 mai 2019) | 2 lignes

Vérification Création nouvelle campagne

r117 | nboffredo | 2019-05-27 12 :37 :21 +0200 (lun. 27 mai 2019) | 1 ligne

Ajout de la récupération de l'idOperateur correspondant à l'opérateur sélèctionné dans la fenetre de création d'une nouvelle campagne

r116 | nboffredo | 2019-05-24 12 :29 :43 +0200 (ven. 24 mai 2019) | 1 ligne

Mise à jour de la création d'une nouvelle campagne

r115 | jreynier | 2019-05-24 08 :09 :50 +0200 (ven. 24 mai 2019) | 1 ligne

Ajout d'un accesseur pour idCampagne de la classe BaseDeDonnees

r114 | jreynier | 2019-05-23 16 :43 :34 +0200 (jeu. 23 mai 2019) | 1 ligne

Mise à jour de l'archivage des mesures

r113 | nboffredo | 2019-05-23 15 :54 :07 +0200 (jeu. 23 mai 2019) | 1 ligne

Ajout de la possibilité de démarrer une nouvelle campagne et mise à jour de la doc doxygen

r112 | nboffredo | 2019-05-23 10 :53 :43 +0200 (jeu. 23 mai 2019) | 1 ligne

Mise à jour de la BDD

r111 | jreynier | 2019-05-23 10 :07 :17 +0200 (jeu. 23 mai 2019) | 1 ligne

Ajout de l'archivage des mesures toutes les secondes (provisoire).

r110 | nboffredo | 2019-05-23 09 :57 :37 +0200 (jeu. 23 mai 2019) | 1 ligne

Mise à jour de la fenetre pour demarrer une nouvelle campagne

r109 | jreynier | 2019-05-23 08 :30 :03 +0200 (jeu. 23 mai 2019) | 1 ligne

Modification de noms de signaux de la classe Mesures

r108 | jreynier | 2019-05-21 10 :16 :27 +0200 (mar. 21 mai 2019) | 1 ligne

Modification de la méthode traiteTrame de la classe Mesures

r107 | nboffredo | 2019-05-21 08 :52 :16 +0200 (mar. 21 mai 2019) | 1 ligne

Mise à jour de la fenetre des archives : taille fixée

r106 | nboffredo | 2019-05-21 08 :45 :16 +0200 (mar. 21 mai 2019) | 1 ligne

Modification de l'accesseur getDateImage(const QModelIndex) de la classe Archive (changement d'un switch à un QStringList)

r105 | tvaira | 2019-05-20 11 :24 :11 +0200 (lun. 20 mai 2019) | 1 ligne

Ajout documentation Doxygen pour tag 0.2

r104 | tvaira | 2019-05-20 11 :21 :16 +0200 (lun. 20 mai 2019) | 1 ligne

Passage des fichiers Doxygen au format Markdown

r103 | tvaira | 2019-05-10 17 :28 :58 +0200 (ven. 10 mai 2019) | 1 ligne

création du tag 0.3

r102 | nboffredo | 2019-05-03 12 :46 :12 +0200 (ven. 03 mai 2019) | 1 ligne

Mise à jour de l'affichage des mesures en temps réel sur l'ihm

r101 | jreynier | 2019-05-03 11 :58 :05 +0200 (ven. 03 mai 2019) | 1 ligne

Modification classe Bras : modification de l'envoi des trames.

r100 | nboffredo | 2019-05-03 10 :40 :27 +0200 (ven. 03 mai 2019) | 1 ligne

Ajout de l'affichage des mesures des capteurs sur l'ihm

r99 | nboffredo | 2019-05-03 10 :32 :16 +0200 (ven. 03 mai 2019) | 1 ligne

Ajout de la partie distance dans l'affichage des mesures sur l'ihm

r98 | nboffredo | 2019-05-03 10 :14 :23 +0200 (ven. 03 mai 2019) | 1 ligne

Ajout de la barre de menu et d'une fentre d'aide

r97 | jreynier | 2019-05-03 09 :23 :12 +0200 (ven. 03 mai 2019) | 1 ligne

Réparation de la classe Mesures (suite au conflit).

r96 | nboffredo | 2019-05-02 17 :42 :27 +0200 (jeu. 02 mai 2019) | 1 ligne

Ajout d'une barre de menu à l'IHM pour démarrer une nouvelle campagne

r95 | jreynier | 2019-05-02 17 :07 :33 +0200 (jeu. 02 mai 2019) | 1 ligne

Modification des trames émises par la classe Bras pour tests avec le bras articulé

r94 | jreynier | 2019-05-02 12 :32 :12 +0200 (jeu. 02 mai 2019) | 1 ligne

Modification de la classe CommunicationRov et Rov, la liaisonserie entre le pc et le rov est fonctionnelle.

r93 | jreynier | 2019-04-15 11 :24 :27 +0200 (lun. 15 avril 2019) | 1 ligne

Modification des trames émises par la classe ControleCamera.

r92 | jreynier | 2019-04-15 11 :01 :30 +0200 (lun. 15 avril 2019) | 1 ligne

Modification des noms de méthodes de la classe ControleCamera + conversion des arguments temperature et distance de la classe Mesures en double (précédemment float).

r91 | jreynier | 2019-04-13 12 :14 :49 +0200 (sam. 13 avril 2019) | 1 ligne

Modification des noms des méthodes de la classe Bras.

r90 | jreynier | 2019-04-13 11 :43 :56 +0200 (sam. 13 avril 2019) | 1 ligne

Optimisation de la méthode traiteTrame de la classe Mesures.

r89 | jreynier | 2019-04-13 11 :29 :45 +0200 (sam. 13 avril 2019) | 1 ligne

Modification de la méthode traiteTrame de la classe Mesures pour prendre en compte la nouvelle nomenclature de trames + ajout d'un attribut distance, relevé du capteur de proximité.

r88 | jreynier | 2019-04-13 11 :04 :08 +0200 (sam. 13 avril 2019) | 1 ligne

Modification de la méthode envoieTrame de la classe CommunicationRov : résolution de l'avertissement, et suppression d'un debug incohérent.

r87 | jreynier | 2019-04-13 10 :50 :01 +0200 (sam. 13 avril 2019) | 1 ligne

Suppression des ID pour les trames à émettre (suite à la revue n°2).

r86 | tvaira | 2019-04-06 10 :20 :56 +0200 (sam. 06 avril 2019) | 1 ligne

Ajout de la documentation pour le tag 0.2

r85 | nboffredo | 2019-04-05 11 :37 :12 +0200 (ven. 05 avril 2019) | 1 ligne

Mise à jour du Changelog et du README

r84 | nboffredo | 2019-04-05 11 :36 :55 +0200 (ven. 05 avril 2019) | 1 ligne

Mise à jour du README

r83 | nboffredo | 2019-04-05 11 :07 :22 +0200 (ven. 05 avril 2019) | 1 ligne

Mise à jour des Changelog

r82 | jreynier | 2019-04-03 23 :25 :20 +0200 (mer. 03 avril 2019) | 1 ligne

Ajout de messages Debug en prévision des tests de la revue 2

r81 | nboffredo | 2019-04-03 08 :55 :30 +0200 (mer. 03 avril 2019) | 1 ligne

création du tag 0.2

r80 | nboffredo | 2019-04-03 08 :47 :35 +0200 (mer. 03 avril 2019) | 1 ligne

Mise à jour information doxygen

r79 | jreynier | 2019-04-02 20 :39 :59 +0200 (mar. 02 avril 2019) | 1 ligne

Mise à jour Doxygen

r78 | tvaira | 2019-04-01 18 :34 :48 +0200 (lun. 01 avril 2019) | 1 ligne

Controle Qualité Revue 2

r77 | jreynier | 2019-03-29 11 :52 :20 +0100 (ven. 29 mars 2019) | 1 ligne

Ajout de la connexion au fichier rovnet.sqlite par la classe BaseDeDonnees.

r76 | jreynier | 2019-03-29 11 :40 :21 +0100 (ven. 29 mars 2019) | 1 ligne

Correction des avertissements dans la classe BaseDeDonnees.

r75 | nboffredo | 2019-03-29 10 :43 :37 +0100 (ven. 29 mars 2019) | 1 ligne

Ajout du controle de la caméra

r74 | nboffredo | 2019-03-28 22 :10 :32 +0100 (jeu. 28 mars 2019) | 1 ligne

Miseà jour des commentaires doxygen

r73 | jreynier | 2019-03-28 22 :00 :11 +0100 (jeu. 28 mars 2019) | 1 ligne

Modification de la structure de la BDD.

r72 | nboffredo | 2019-03-28 21 :49 :55 +0100 (jeu. 28 mars 2019) | 1 ligne

Ajout d'un bouton pour fermer la fenetre des archives

r71 | nboffredo | 2019-03-28 20 :29 :15 +0100 (jeu. 28 mars 2019) | 1 ligne

Ajout de la possibilité de changer de caméra à travers une liste sur l'IHM

r70 | jreynier | 2019-03-28 18 :17 :02 +0100 (jeu. 28 mars 2019) | 1 ligne

Modification de la bdd rovnet.sqlite (création d'une table mesures comportant un attribut date, temperature, et irradiation).

r69 | jreynier | 2019-03-28 16 :33 :41 +0100 (jeu. 28 mars 2019) | 1 ligne

Changement du lieu de l'instanciation de la bdd (de IHMRov à Rov).

r68 | nboffredo | 2019-03-28 15 :19 :47 +0100 (jeu. 28 mars 2019) | 1 ligne

Ajout d'une liste de toute les caméra disponibles

r67 | jreynier | 2019-03-28 14 :34 :07 +0100 (jeu. 28 mars 2019) | 1 ligne

Modification de la méthode traiteTrame afin de prendre en compte la nouvelle nomenclature de trame.

r66 | jreynier | 2019-03-28 12 :05 :35 +0100 (jeu. 28 mars 2019) | 1 ligne

Modification de la classe Deplacement : 4 types de trames sont maintenant envoyées (AVN, REC, TDR, TGA) afin de respecter la nomenclature des trames mise en place.

r65 | jreynier | 2019-03-28 11 :35 :58 +0100 (jeu. 28 mars 2019) | 1 ligne

Mise en place d'une idTrame commune aux classes Deplacement et Bras (et dans le futur, de la classe Camera).

r64 | nboffredo | 2019-03-27 12 :28 :49 +0100 (mer. 27 mars 2019) | 1 ligne

Ajout d'un début de méthode pour l'affichage de la liste des caméra disponibles

r63 | jreynier | 2019-03-27 09 :04 :29 +0100 (mer. 27 mars 2019) | 1 ligne

Déplacement des méthodes traiteTrame et stockeDonnees de la classe Rov vers la classe Mesures (plus adapté).

r62 | jreynier | 2019-03-23 17 :20 :20 +0100 (sam. 23 mars 2019) | 1 ligne

Suppression de RovNet.pro.user (encore).

r61 | jreynier | 2019-03-23 17 :19 :29 +0100 (sam. 23 mars 2019) | 1 ligne

Création des méthodes traiteTrame et stockeDonnees de la classe Rov.

r60 | jreynier | 2019-03-23 15 :36 :39 +0100 (sam. 23 mars 2019) | 1 ligne

Ajout d'une méthode deposer, liée au bouton start de la manette. Permet de déposer dans le bac le contenu de la pince.

r59 | nboffredo | 2019-03-22 16 :46 :24 +0100 (ven. 22 mars 2019) | 1 ligne

Mise à jour de l'affichage des archives

r58 | nboffredo | 2019-03-22 15 :45 :41 +0100 (ven. 22 mars 2019) | 1 ligne

Mise à jour de l'affichage si aucune caméra n'est connecté + Mise à jour des dimensions des widgets dans l'interface

r57 | nboffredo | 2019-03-22 14 :55 :10 +0100 (ven. 22 mars 2019) | 1 ligne

Ajout d'un écran noir si aucune caméra n'est detecté

r56 | jreynier | 2019-03-22 14 :36 :14 +0100 (ven. 22 mars 2019) | 1 ligne

Modification des touches pour prendre en compte les modes de la manette.

r55 | jreynier | 2019-03-22 14 :07 :06 +0100 (ven. 22 mars 2019) | 1 ligne

Ajout d'un mode de manette (Bras ou Deplacement), fonctionnel.

r54 | nboffredo | 2019-03-22 13 :04 :26 +0100 (ven. 22 mars 2019) | 1 ligne

Déplacement de méthodes d'archivages dans la classe Archives

r53 | tvaira | 2019-03-22 06 :30 :31 +0100 (ven. 22 mars 2019) | 2 lignes

Ajout de la classe BaseDeDonnees

r52 | jreynier | 2019-03-21 16 :51 :36 +0100 (jeu. 21 mars 2019) | 1 ligne

Mise à jour Doxygene classes Bras et Deplacement

r51 | nboffredo | 2019-03-21 16 :36 :56 +0100 (jeu. 21 mars 2019) | 1 ligne

MaJ des commentaires doxygen

r50 | jreynier | 2019-03-21 16 :14 :57 +0100 (jeu. 21 mars 2019) | 1 ligne

Modification de noms de méthodes tournerEpaule et tournerPoignet dans la classe Bras.

r49 | jreynier | 2019-03-21 15 :59 :11 +0100 (jeu. 21 mars 2019) | 1 ligne

Modification méthodes de classe Bras

r48 | jreynier | 2019-03-21 14 :31 :27 +0100 (jeu. 21 mars 2019) | 1 ligne

Mise à jour des trames envoyées par la manette (classes Bras et Deplacement).

r47 | nboffredo | 2019-03-21 13 :55 :49 +0100 (jeu. 21 mars 2019) | 1 ligne

Correction du bug faisant crash l'IHM lors d'un prise de photo sans caméra detectée

r46 | nboffredo | 2019-03-21 12 :26 :01 +0100 (jeu. 21 mars 2019) | 1 ligne

Mise à jour de l'interface de visualisation des photos archivés, ajout des informations propres aux photo (date, heure, rad, temp)

r45 | nboffredo | 2019-03-21 10 :47 :32 +0100 (jeu. 21 mars 2019) | 1 ligne

Ajout de la visualisation des photos

r44 | tvaira | 2019-03-20 21 :14 :42 +0100 (mer. 20 mars 2019) | 2 lignes

Suppresion fichier inutile .pro.user !!!!!!!!!!!!!

r43 | jreynier | 2019-03-20 12 :34 :04 +0100 (mer. 20 mars 2019) | 1 ligne

Mise à jour du constructeur ControleRov, pour prendre en argument l'objet rov connu par l'IHM

r42 | nboffredo | 2019-03-20 12 :19 :50 +0100 (mer. 20 mars 2019) | 1 ligne

Déplacement de la gestion des archives dans la classe Archives

r41 | jreynier | 2019-03-20 12 :16 :25 +0100 (mer. 20 mars 2019) | 1 ligne

Création des liens entre IHMRov, Rov, et CommunicationRov

r40 | jreynier | 2019-03-20 11 :45 :16 +0100 (mer. 20 mars 2019) | 1 ligne

Mise à jour des commentaires Doxygen des classes : manette, mesures, controlerov, bras, deplacement, communicationrov (nouvelle nomenclature)

r39 | tvaira | 2019-03-18 18 :59 :08 +0100 (lun. 18 mars 2019) | 1 ligne

Suppression fichier inutile .pro.user

r38 | tvaira | 2019-03-18 18 :58 :03 +0100 (lun. 18 mars 2019) | 2 lignes

Mise à jour de la documentation du code

r37 | nboffredo | 2019-03-15 18 :23 :33 +0100 (ven. 15 mars 2019) | 2 lignes

Ajout d'une interface de gestion des archives.

r36 | jreynier | 2019-03-15 12 :58 :07 +0100 (ven. 15 mars 2019) | 1 ligne

Refonte de la classe Manette et ControleRov

r35 | jreynier | 2019-03-15 10 :54 :37 +0100 (ven. 15 mars 2019) | 1 ligne

Modification relations entre classes.

r34 | nboffredo | 2019-03-15 10 :33 :38 +0100 (ven. 15 mars 2019) | 1 ligne

Ajout d'un fichier de ressources pour le stockage des logos

r33 | tvaira | 2019-03-15 06 :10 :02 +0100 (ven. 15 mars 2019) | 1 ligne

Modificiation Doxyfile

r32 | jreynier | 2019-03-14 16 :36 :03 +0100 (jeu. 14 mars 2019) | 1 ligne

Mise à jour CommunicationRov

r31 | nboffredo | 2019-03-14 16 :26 :40 +0100 (jeu. 14 mars 2019) | 1 ligne

Ajout de l'intégralité des Widgets, de l'indicateur de disponibilité de la caméra et d'un chronomètre sur l'IHM

r30 | jreynier | 2019-03-14 15 :34 :01 +0100 (jeu. 14 mars 2019) | 1 ligne

Structuration de la classe Mesures + description Doxygen pour la classe Mesures et CommunicationRov

r29 | nboffredo | 2019-03-14 13 :48 :10 +0100 (jeu. 14 mars 2019) | 1 ligne

Mise à jour de l'IHM

r28 | jreynier | 2019-03-14 11 :57 :02 +0100 (jeu. 14 mars 2019) | 1 ligne

Structuration de la classe CommunicationRov

r27 | jreynier | 2019-03-14 11 :17 :03 +0100 (jeu. 14 mars 2019) | 1 ligne

Les SLOTS de déplacement et de mouvements du bras émettent maintenant des signaux contenant les trames correspondantes + ajout du module serialport au projer.

r26 | tvaira | 2019-03-14 06 :53 :47 +0100 (jeu. 14 mars 2019) | 1 ligne

Ajout du fichier Doxyfile

r25 | tvaira | 2019-03-14 06 :50 :00 +0100 (jeu. 14 mars 2019) | 1 ligne

Suppression du trunk de la documentation du code (seulement dans les version taguées)

r24 | nboffredo | 2019-03-13 17 :49 :42 +0100 (mer. 13 mars 2019) | 1 ligne

Ajout du flux video sur l'IHM

r23 | jreynier | 2019-03-13 17 :14 :39 +0100 (mer. 13 mars 2019) | 1 ligne

Création d'un prototype de trame pour les méthodes Avancer et Tourner

r22 | jreynier | 2019-03-13 16 :55 :30 +0100 (mer. 13 mars 2019) | 1 ligne

Mise à jour des slots dans ControleBras

r21 | jreynier | 2019-03-13 14 :29 :01 +0100 (mer. 13 mars 2019) | 1 ligne

Ajout des slots de controle de bras dans ControleBras

r20 | jreynier | 2019-03-13 13 :53 :46 +0100 (mer. 13 mars 2019) | 1 ligne

Modification ihmrov.cpp pour instancier dynamiquement une manette

r19 | jreynier | 2019-03-13 13 :44 :39 +0100 (mer. 13 mars 2019) | 1 ligne

Modification nom de classes : GestionnaireMesures devient Mesures et GestionnaireManette devient Manette.

r18 | nboffredo | 2019-03-13 13 :40 :16 +0100 (mer. 13 mars 2019) | 1 ligne

Modification nom des classes

r17 | nboffredo | 2019-03-13 12 :28 :42 +0100 (mer. 13 mars 2019) | 1 ligne

Mise à jour fichiers camera

r16 | nboffredo | 2019-03-13 12 :22 :02 +0100 (mer. 13 mars 2019) | 1 ligne

Mise à jour du gestionnaire de la caméra

r15 | jreynier | 2019-03-13 12 :19 :53 +0100 (mer. 13 mars 2019) | 1 ligne

Mise à jour des classes GestionnaireManette et ControleDeplacement. Nous pouvons maintenant récupérer des signaux de la manette et les traiter dans les classes correspondantes.

r14 | jreynier | 2019-03-13 11 :56 :02 +0100 (mer. 13 mars 2019) | 1 ligne

Ajout des squelettes des classes du projet RovNet

r13 | jreynier | 2019-03-13 11 :49 :59 +0100 (mer. 13 mars 2019) | 1 ligne

Création du projet Qt RovNet

r12 | nboffredo | 2019-03-13 11 :44 :23 +0100 (mer. 13 mars 2019) | 1 ligne

Mise à jour de l'organisation des fichiers

r11 | nboffredo | 2019-03-13 11 :34 :05 +0100 (mer. 13 mars 2019) | 1 ligne

Déplacement fichiers Doxygenes

r10 | nboffredo | 2019-03-13 10 :48 :50 +0100 (mer. 13 mars 2019) | 1 ligne

Mise à jour du Doxygène

r9 | nboffredo | 2019-03-13 10 :44 :34 +0100 (mer. 13 mars 2019) | 1 ligne

Ajout d'une méthode pour le retour vidéo

r8 | nboffredo | 2019-03-13 09 :42 :42 +0100 (mer. 13 mars 2019) | 1 ligne

Ajout des fichiers Doxygene

r7 | nboffredo | 2019-03-13 09 :39 :01 +0100 (mer. 13 mars 2019) | 1 ligne

Ajout du répertoire Camera

r6 | tvaira | 2019-03-09 07 :57 :28 +0100 (sam. 09 mars 2019) | 1 ligne

Initialisation Doxygen

r5 | nboffredo | 2019-03-07 15 :58 :55 +0100 (jeu. 07 mars 2019) | 1 ligne

Ajout de l'exemple d'une interface Qt fonctionnel pour la caméra

r4 | jreynier | 2019-03-07 10 :54 :07 +0100 (jeu. 07 mars 2019) | 1 ligne

TestManette : suppression des fichiers qml

r3 | jreynier | 2019-03-07 10 :44 :46 +0100 (jeu. 07 mars 2019) | 1 ligne

Création d'un dossier Tests dans le trunk

r2 | jreynier | 2019-02-07 13 :36 :51 +0100 (jeu. 07 févr. 2019) | 1 ligne

Création du README et du Changelog.

r1 | www-data | 2019-02-06 20 :08 :54 +0100 (mer. 06 févr. 2019) | 1 ligne

Creating initial repository structure

3 Installation

Fabrication

```
$ qmake
$ make
$ sudo make install
```

Paquets nécessaires

```
$ sudo apt-get install xboxdrv
$ sudo apt-get install xboxdrv-daemon
$ sudo apt-get install v4l-utils
$ sudo apt-get install libpulse-dev
```

4 README

Nom : ROV'NET

Le système ROV'NET est capable d'obtenir et gérer des données d'entrée afin de définir un environnement TQC (Tel Que Construit) en se déplaçant dans un milieu contaminé.

Numéro de version : 1.0

Auteurs

Auteur

BOFFREDO Nicolas nboffredo@gmail.com
REYNIER Jacques reynier.jacques@gmail.com

Dépôt SVN

https://svn.riouxsvn.com/rov-net

Recette IR

— Étudiant : BOFFREDO Nicolas
— Étudiant : REYNIER Jacques

Base de données SQLite

```
CREATE TABLE 'campagnes' ( 'nom' VARCHAR, 'description' VARCHAR, 'date' DATETIME, 'cheminArchives' VARCHAR,
       'idOperateur' INTEGER, 'idCampagne' INTEGER PRIMARY KEY AUTOINCREMENT, FOREIGN KEY(idOperateur) REFERENCES
       operateurs(idOperateur) )

CREATE TABLE 'mesures' ( 'date' DATETIME, 'temperature' DOUBLE, 'irradiation' DOUBLE, 'idCampagne' INTEGER,
       FOREIGN KEY(idCampagne) REFERENCES campagnes(idCampagne) )

CREATE TABLE 'operateurs' ( 'nom' VARCHAR, 'prenom' VARCHAR, 'idOperateur' INTEGER PRIMARY KEY
       AUTOINCREMENT )
```

5 A propos

Auteur

BOFFREDO Nicolas nboffredo@gmail.com
REYNIER Jacques reynier.jacques@gmail.com

Version

1.0

Date

2019

6 Licence GPL

7 Liste des choses à faire

Membre Bras : :leveCoude (bool appuye)

Changer l'envoie de trame. Envoyer 1 quand Triangle, -1 quand Croix. -> Diminution nombre code différents dans la trame, simplification du décodage des trames.

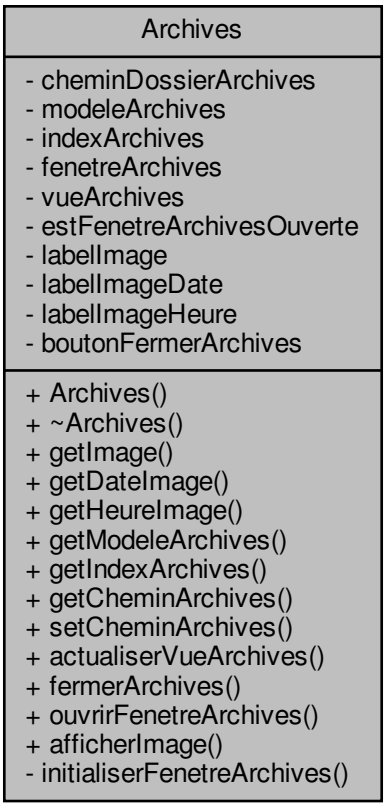
8 Documentation des classes

8.1 Référence de la classe Archives

Déclaration de la classe Archives.

```
#include <archives.h>
```

Graphe de collaboration de Archives :



Connecteurs publics

- void fermerArchives ()
  Ferme la fenetre des archives.
- void ouvrirFenetreArchives ()
  Ouvre une fenetre menant aux archives des captures d'écrans.
- void afficherImage (const QModelIndex &indexArchives)
  Affiche l'image séléctionnée et les informations correspondantes.

Fonctions membres publiques

- Archives (QObject *parent=nullptr)
- ~Archives ()
- QString getImage (const QModelIndex &indexArchives)
  Renvoie l'image des archives.
- QString getDateImage (const QModelIndex &indexArchives)
  Retourne la date de prise de l'image sélectionnée.
- QString getHeureImage (const QModelIndex &indexArchives)
  Retourne l'heure de prise de l'image sélectionnée.
- QFileSystemModel * getModeleArchives ()
  Accesseur renvoyant le modèle de données.
- QModelIndex getIndexArchives ()
  Accesseur renvoyant l'index du modèle de données.
- QString getCheminArchives ()
  Accesseur renvoyant le chemin du dossier de stockage des photos.
- void setCheminArchives (QString nouveauCheminArchives)
  Accesseur permettant de modifier le chemin vers les Archives.
- void actualiserVueArchives ()
  Actualise la vue des archives.

Fonctions membres privées

- void initialiserFenetreArchives ()
  Initialise la fenetre pour naviguer dans les archives.

Attributs privés

- QString cheminDossierArchives
  le chemin du dossier de stockage des photos
- QFileSystemModel * modeleArchives
  le modèle de données
- QModelIndex indexArchives
  l'index du modèle de données
- QDialog * fenetreArchives
  fenetre pour naviguer dans les archives
- QListView * vueArchives
  la vue de la liste des archives
- bool estFenetreArchivesOuverte
  booléen indiquant si la fenêtre est ouverte
- QLabel * labelImage
  label pour l'affichage de la photo sélectionnée
- QLabel * labelImageDate
  label pour l'affichage de la date de la photo sélectionnée
- QLabel * labelImageHeure
  label pour l'affichage de l'heure de la photo sélectionnée
- QPushButton * boutonFermerArchives
  bouton de fermeture de la fenêtre

8.1.1 Description détaillée

Auteur

Nicolas BOFFREDO

Version

1.1

Date

Mercredi 12 Juin 2019

8.1.2 Documentation des constructeurs et destructeur

8.1.2.1 Archives()

```
Archives::Archives (
            QObject * parent = nullptr )  [explicit]
```

Constructeur de la classe Archives

Paramètres

parent	QObject*

Références initialiserFenetreArchives().

```
00021                                   : QObject(parent), cheminDossierArchives(" "),
      modeleArchives(nullptr)
00022 {
00023     initialiserFenetreArchives();
00024 }
```

8.1.2.2 ~Archives()

```
Archives::~Archives ( )
```

Destructeur de la classe Archives

```
00031 {
00032 }
```

8.1.3 Documentation des fonctions membres

8.1.3.1 actualiserVueArchives()

```
void Archives::actualiserVueArchives ( )
```

Est appellée à chaque fois que la fenêtre de navigation dans les archives est ouverte.

Références getCheminArchives(), getIndexArchives(), getModeleArchives(), indexArchives, modeleArchives, et vueArchives.

Référencé par ouvrirFenetreArchives().

```
00186 {
00187     modeleArchives->setRootPath(getCheminArchives());
00188     indexArchives = modeleArchives->index(
      getCheminArchives());
00189
00190     vueArchives->setModel(this->getModeleArchives());
00191     vueArchives->setRootIndex(this->getIndexArchives());
00192 }
```

8.1.3.2 afficherImage

```
void Archives::afficherImage (
            const QModelIndex & indexArchives )  [slot]
```

Paramètres

indexArchives	QModelIndex

Références getDateImage(), getHeureImage(), getImage(), labelImage, labelImageDate, et labelImageHeure.

Référencé par initialiserFenetreArchives().

```
00210 {
00211     labelImage->setPixmap(QPixmap(this->getImage(indexArchives)));
00212     labelImage->setScaledContents(true);
00213     labelImageDate->setText("Date : " + this->getDateImage(
      indexArchives));
00214     labelImageHeure->setText("Heure : " + this->getHeureImage(
      indexArchives));
00215 }
```

8.1.3.3 fermerArchives

```
void Archives::fermerArchives ( )  [slot]
```

Références fenetreArchives.

Référencé par initialiserFenetreArchives().

```
00222 {
00223     fenetreArchives->close();
00224 }
```

8.1.3.4 getCheminArchives()

```
QString Archives::getCheminArchives ( )
```

Renvoie

Un *QString* indiquant le chemin du dossier de stockage des photos.

Références cheminDossierArchives.

Référencé par actualiserVueArchives(), Rov : :creerNouvelleCampagne(), getImage(), initialiserFenetreArchives(), et Camera↩ : :nommerCapture().

```
00040 {
00041     qDebug() << Q_FUNC_INFO;
00042     return cheminDossierArchives;
00043 }
```

8.1.3.5 getDateImage()

```
QString Archives::getDateImage (
            const QModelIndex & indexArchives )
```

Paramètres

indexArchives	const QModelIndex &, L'index de l'image des archives

Renvoie

Un *QString* indiquant la date de prise de l'image sélectionnée

Référencé par afficherImage().

```
00096 {
00097     QString dateImage = indexArchives.data().toString().left(10);
00098     QStringList mois;
00099
00100     dateImage.replace(2, 1, " ");
00101     dateImage.replace(5, 1, " ");
00102
00103     mois << QString::fromUtf8("Janvier") << QString::fromUtf8("Février") << QString::fromUtf8("Mars") <<
      QString::fromUtf8("Avril") << QString::fromUtf8("Mai") << QString::fromUtf8("Juin") << QString::fromUtf8("
      Juillet") << QString::fromUtf8("Août") << QString::fromUtf8("Septembre") << QString::fromUtf8("Octobre") <<
      QString::fromUtf8("Novembre") << QString::fromUtf8("Décembre");
00104     dateImage.replace(3, 2, mois.at(dateImage.left(5).right(3).toInt() - 1));
00105
00106     return dateImage;
00107 }
```

8.1.3.6 getHeureImage()

```
QString Archives::getHeureImage (
            const QModelIndex & indexArchives )
```

Paramètres

indexArchives	const QModelIndex &, L'index de l'images des archives

Renvoie

Un *QString* indiquant l'heure de prise de l'image sélectionnée

Référencé par afficherImage().

```
00116 {
00117     QString heureImage = indexArchives.data().toString().remove(19, 4).right(8);
00118     heureImage.replace(2, 1, "h ");
00119     heureImage.replace(6, 1, "m ");
00120     heureImage.append('s');
00121
00122     return heureImage;
00123 }
```

8.1.3.7 getImage()

```
QString Archives::getImage (
            const QModelIndex & indexArchives )
```

Paramètres

indexArchives	QModelIndex & index sur le modèle des fichiers contenus dans le QListView

Renvoie

un *QString* indiquant le nom de l'image

Références getCheminArchives().

Référencé par afficherImage().

```
00063 {
00064     QString cheminImage = getCheminArchives() + indexArchives.data().toString
      ();
00065     qDebug() << Q_FUNC_INFO << "cheminImage" << cheminImage;
00066     return cheminImage;
00067 }
```

8.1.3.8 getIndexArchives()

```
QModelIndex Archives::getIndexArchives ( )
```

Renvoie

Un *QModelIndex* indiquant l'index du modèle de données.

Références indexArchives.

Référencé par actualiserVueArchives(), et initialiserFenetreArchives().

```
00085 {
00086     return indexArchives;
00087 }
```

8.1.3.9 getModeleArchives()

```
QFileSystemModel * Archives::getModeleArchives ( )
```

Renvoie

Un *QFileSystemModel*∗ indiquant le modèle de données.

Références modeleArchives.

Référencé par actualiserVueArchives(), et initialiserFenetreArchives().

```
00075 {
00076     return modeleArchives;
00077 }
```

8.1.3.10 initialiserFenetreArchives()

```
void Archives::initialiserFenetreArchives ( )  [private]
```

La taille de la fenetre est fixer en fonction de la resolution des photos prises.

Références afficherImage(), boutonFermerArchives, fenetreArchives, fermerArchives(), getCheminArchives(), getIndexArchives(), getModeleArchives(), indexArchives, labelImage, labelImageDate, labelImageHeure, modeleArchives, et vueArchives.

Référencé par Archives().

```
00131 {
00132     modeleArchives = new QFileSystemModel;
00133     modeleArchives->setRootPath(getCheminArchives());
00134     indexArchives = modeleArchives->index(
      getCheminArchives());
00135
00136     fenetreArchives = new QDialog();
00137     vueArchives = new QListView;
00138     labelImage = new QLabel;
00139     labelImageDate = new QLabel;
00140     labelImageHeure = new QLabel;
00141     boutonFermerArchives = new QPushButton("&Fermer");
00142
00143     const int hauteurImage = 480;
00144     const int largeurImage = 640;
00145     const int hauteurInformations = (hauteurImage/10);
00146     const int largeurMAX = largeurImage + largeurImage/2;
00147     const int hauteurMAX = hauteurImage + hauteurInformations;
00148
00149     vueArchives->setModel(this->getModeleArchives());
00150     vueArchives->setRootIndex(this->getIndexArchives());
00151
00152     QHBoxLayout *hLayoutPrincipalArchives = new QHBoxLayout;
00153     QVBoxLayout *vLayoutSelections = new QVBoxLayout;
00154     QVBoxLayout *vLayoutImage = new QVBoxLayout;
00155     QHBoxLayout *hLayoutInformationsImage = new QHBoxLayout;
00156
00157     hLayoutPrincipalArchives->addLayout(vLayoutImage);
00158     hLayoutPrincipalArchives->addLayout(vLayoutSelections);
00159     vLayoutSelections->addWidget(vueArchives);
00160     vLayoutSelections->addWidget(boutonFermerArchives);
00161     vLayoutImage->addLayout(hLayoutInformationsImage);
00162     vLayoutImage->addWidget(labelImage);
00163     hLayoutInformationsImage->addWidget(labelImageDate);
00164     hLayoutInformationsImage->addWidget(labelImageHeure);
00165
00166     fenetreArchives->setWindowTitle("Archives Photo");
00167     fenetreArchives->setFixedSize(largeurMAX, hauteurMAX);
00168     fenetreArchives->setLayout(hLayoutPrincipalArchives);
00169     labelImage->setFixedHeight(hauteurImage);
00170     labelImage->setFixedWidth(largeurImage);
00171     labelImageDate->setFixedHeight(hauteurInformations);
00172     labelImageHeure->setFixedHeight(hauteurInformations);
00173
00174     connect(vueArchives, SIGNAL(doubleClicked(const QModelIndex)), this, SLOT(
      afficherImage(const QModelIndex)));
00175     connect(boutonFermerArchives, SIGNAL(clicked()), this, SLOT(
      fermerArchives()));
00176
00177     fenetreArchives->setWindowFlags(Qt::Dialog | Qt::WindowCloseButtonHint);
00178 }
```

8.1.3.11 ouvrirFenetreArchives

```
void Archives::ouvrirFenetreArchives ( )  [slot]
```

Références actualiserVueArchives(), et fenetreArchives.

```
00199 {
00200     actualiserVueArchives();
00201     fenetreArchives->exec();
00202 }
```

8.1.3.12 setCheminArchives()

```
void Archives::setCheminArchives (
            QString nouveauCheminArchives )
```

Paramètres

nouveauCheminArchives	QString le nouveau chemin des Archives.

Références cheminDossierArchives.

Référencé par Rov : :creerDossierArchives().

```
00051 {
00052     qDebug() << Q_FUNC_INFO << nouveauCheminArchives;
00053     cheminDossierArchives = nouveauCheminArchives;
00054 }
```

8.1.4 Documentation des données membres

8.1.4.1 boutonFermerArchives

```
QPushButton* Archives::boutonFermerArchives  [private]
```

Référencé par initialiserFenetreArchives().

8.1.4.2 cheminDossierArchives

```
QString Archives::cheminDossierArchives  [private]
```

Référencé par getCheminArchives(), et setCheminArchives().

8.1.4.3 estFenetreArchivesOuverte

```
bool Archives::estFenetreArchivesOuverte  [private]
```

8.1.4.4 fenetreArchives

```
QDialog* Archives::fenetreArchives  [private]
```

Référencé par fermerArchives(), initialiserFenetreArchives(), et ouvrirFenetreArchives().

8.1.4.5 indexArchives

```
QModelIndex Archives::indexArchives  [private]
```

Référencé par actualiserVueArchives(), getIndexArchives(), et initialiserFenetreArchives().

8.1.4.6 labelImage

```
QLabel* Archives::labelImage  [private]
```

Référencé par afficherImage(), et initialiserFenetreArchives().

8.1.4.7 labelImageDate

```
QLabel* Archives::labelImageDate  [private]
```

Référencé par afficherImage(), et initialiserFenetreArchives().

8.1.4.8 labelImageHeure

```
QLabel* Archives::labelImageHeure  [private]
```

Référencé par afficherImage(), et initialiserFenetreArchives().

8.1.4.9 modeleArchives

```
QFileSystemModel* Archives::modeleArchives  [private]
```

Référencé par actualiserVueArchives(), getModeleArchives(), et initialiserFenetreArchives().

8.1.4.10 vueArchives

```
QListView* Archives::vueArchives  [private]
```

Référencé par actualiserVueArchives(), et initialiserFenetreArchives().

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

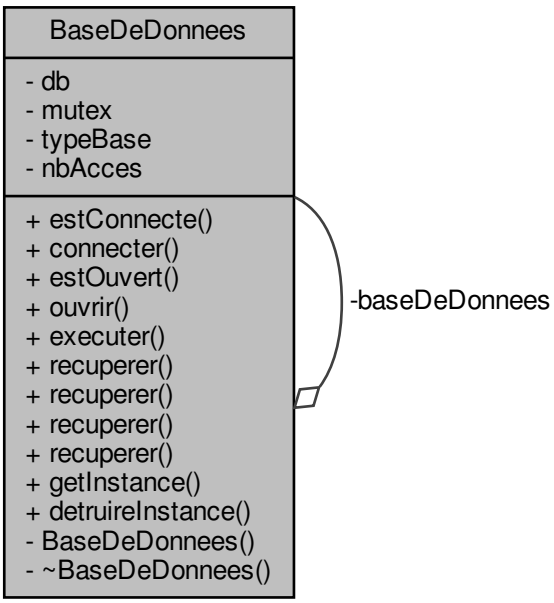
- archives.h
- archives.cpp

8.2 Référence de la classe BaseDeDonnees

Déclaration de la classe BaseDeDonnees.

`#include <basededonnees.h>`

Graphe de collaboration de BaseDeDonnees :



Fonctions membres publiques

- bool estConnecte ()
- bool connecter (QString nomBase, QString username=BDD_USERNAME, QString password=BDD_PASSWORD, QString serveur=BDD_HOSTNAME)
- bool estOuvert ()
- bool ouvrir (QString fichierBase)
- bool executer (QString requete)
- bool recuperer (QString requete, QString &donnees)
- bool recuperer (QString requete, QStringList &donnees)
- bool recuperer (QString requete, QVector< QString > &donnees)
- bool recuperer (QString requete, QVector< QStringList > &donnees)

Fonctions membres publiques statiques

- static BaseDeDonnees * getInstance (QString type="QMYSQL")
- static void detruireInstance ()

Fonctions membres privées

- BaseDeDonnees (QString type)
- ~BaseDeDonnees ()

Attributs privés

- QSqlDatabase db
- QMutex mutex

Attributs privés statiques

- static BaseDeDonnees * baseDeDonnees = nullptr
- static QString typeBase = "QMYSQL"
- static int nbAcces = 0

8.2.1 Description détaillée

Auteur

Thierry VAIRA

Version

1.1

8.2.2 Documentation des constructeurs et destructeur

8.2.2.1 BaseDeDonnees()

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

Références db, et typeBase.

Référencé par getInstance().

```
00023 {
00024     #ifdef DEBUG_BASEDEDONNEES
00025     qDebug() << Q_FUNC_INFO << type;
00026     #endif
00027     db = QSqlDatabase::addDatabase(type);
00028     typeBase = type;
00029 }
```

8.2.2.2 ~BaseDeDonnees()

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

```
00032 {
00033     #ifdef DEBUG_BASEDEDONNEES
00034     qDebug() << Q_FUNC_INFO;
00035     #endif
00036 }
```

8.2.3 Documentation des fonctions membres

8.2.3.1 connecter()

```
bool BaseDeDonnees::connecter (
            QString nomBase,
            QString username = BDD_USERNAME,
            QString password = BDD_PASSWORD,
            QString serveur = BDD_HOSTNAME )
```

Références db, mutex, et typeBase.

```
00076 {
00077     if(typeBase != "QMYSQL")
00078         return false;
00079     QMutexLocker verrou(&mutex);
00080     if(!db.isOpen())
00081     {
00082        db.setHostName(serveur);
00083        db.setUserName(username);
00084        db.setPassword(password);
00085        db.setDatabaseName(nomBase);
00086
00087        #ifdef DEBUG_BASEDEDONNEES
00088        qDebug() << Q_FUNC_INFO;
00089        qDebug() << "HostName : " << db.hostName();
00090        qDebug() << "UserName : " << db.userName();
00091        qDebug() << "DatabaseName : " << db.databaseName();
00092        #endif
00093        if(db.open())
00094        {
00095            #ifdef DEBUG_BASEDEDONNEES
00096            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Connexion réussie à %1").arg(
      db.hostName());
00097            #endif
00098            return true;
00099        }
00100        else
00101        {
00102            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : impossible de se connecter à la base de
       données !");
00103            QMessageBox::critical(nullptr, QString::fromUtf8("BaseDeDonnees"), QString::fromUtf8("Impossible
       de se connecter à la base de données !"));
00104            return false;
00105        }
00106     }
00107     else
00108         return true;
00109 }
```

8.2.3.2 detruireInstance()

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

Références baseDeDonnees, et nbAcces.

Référencé par IHMRov : :~IHMRov(), et Rov : :~Rov().

```
00051 {
00052     // instance ?
00053     if(baseDeDonnees != nullptr)
00054     {
00055         if(nbAcces > 0)
00056             nbAcces--;
00057         #ifdef DEBUG_BASEDEDONNEES
00058         qDebug() << Q_FUNC_INFO << "nbAcces restants" << nbAcces;
00059         #endif
00060         // dernier ?
00061         if(nbAcces == 0)
00062         {
00063             delete baseDeDonnees;
00064             baseDeDonnees = nullptr;
00065         }
00066     }
00067 }
```

8.2.3.3 estConnecte()

```
bool BaseDeDonnees::estConnecte ( )
```

Références db, et mutex.

```
00070 {
00071     QMutexLocker verrou(&mutex);
00072     return db.isOpen();
00073 }
```

8.2.3.4 estOuvert()

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

Références db, et mutex.

Référencé par Rov : :Rov().

```
00112 {
00113     QMutexLocker verrou(&mutex);
00114     return db.isOpen();
00115 }
```

8.2.3.5 executer()

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

Références db, et mutex.

Référencé par Rov : :creerNouvelleCampagne(), et Rov : :stockeMesuresBDD().

```
00150 {
00151     QMutexLocker verrou(&mutex);
00152     QSqlQuery r;
00153     bool retour;
00154
00155     if(db.isOpen())
00156     {
00157         if(requete.contains("UPDATE") || requete.contains("INSERT") || requete.contains("DELETE"))
00158         {
00159             retour = r.exec(requete);
00160             #ifdef DEBUG_BASEDEDONNEES
00161             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00162             #endif
00163             if(retour)
00164             {
00165                 return true;
00166             }
00167             else
00168             {
00169                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00170                 return false;
00171             }
00172         }
00173         else
00174         {
00175             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00176             return false;
00177         }
00178     }
00179     else
00180         return false;
00181
00182 }
```

8.2.3.6 getInstance()

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

Références BaseDeDonnees(), baseDeDonnees, et nbAcces.

Référencé par Rov : :Rov().

```
00039 {
00040     if(baseDeDonnees == nullptr)
00041         baseDeDonnees = new BaseDeDonnees(type);
00042     nbAcces++;
00043     #ifdef DEBUG_BASEDEDONNEES
00044     qDebug() << Q_FUNC_INFO << "nbAcces" << nbAcces;
00045     #endif
00046
00047     return baseDeDonnees;
00048 }
```

8.2.3.7 ouvrir()

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

Références db, mutex, et typeBase.

Référencé par Rov : :Rov().

```
00118 {
00119     if(typeBase != "QSQLITE")
00120         return false;
00121     QMutexLocker verrou(&mutex);
00122     if(!db.isOpen())
00123     {
00124        db.setDatabaseName(fichierBase);
00125
00126        #ifdef DEBUG_BASEDEDONNEES
00127        qDebug() << Q_FUNC_INFO << db.databaseName();
00128        #endif
00129        if(db.open())
00130        {
00131            #ifdef DEBUG_BASEDEDONNEES
00132            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Ouvertir réussie à %1").arg(
      db.databaseName());
00133            #endif
00134
00135            return true;
00136        }
00137        else
00138        {
00139            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : impossible d'ouvrir la base de données !"
      );
00140            QMessageBox::critical(nullptr, QString::fromUtf8("BaseDeDonnees"), QString::fromUtf8("Impossible
       d'ouvrir la base de données !"));
00141            return false;
00142        }
00143     }
00144     else
00145         return true;
00146 }
```

8.2.3.8 recuperer() [1/4]

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

Références db, et mutex.

Référencé par Rov : :creerNouvelleCampagne(), Rov : :recupererListeNomsOperateurs(), et Rov : :recupererListePrenoms↩ Operateurs().

```
00188 {
00189     QMutexLocker verrou(&mutex);
00190     QSqlQuery r;
00191     bool retour;
00192 
00193     if(db.isOpen())
00194     {
00195         if(requete.contains("SELECT"))
00196         {
00197             retour = r.exec(requete);
00198             #ifdef DEBUG_BASEDEDONNEES
00199             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00200             #endif
00201             if(retour)
00202             {
00203                 // on se positionne sur l'enregistrement
00204                 r.first();
00205 
00206                 // on vérifie l'état de l'enregistrement retourné
00207                 if(!r.isValid())
00208                 {
00209                     #ifdef DEBUG_BASEDEDONNEES
00210                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Résultat non valide !");
00211                     #endif
00212                     return false;
00213                 }
00214 
00215                 // on récupère sous forme de QString la valeur du champ
00216                 if(r.isNull(0))
00217                 {
00218                     #ifdef DEBUG_BASEDEDONNEES
00219                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Aucun résultat !");
00220                     #endif
00221                     return false;
00222                 }
00223                 donnees = r.value(0).toString();
00224                 #ifdef DEBUG_BASEDEDONNEES
00225                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00226                 #endif
00227                 return true;
00228             }
00229             else
00230             {
00231                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00232                 return false;
00233             }
00234         }
00235         else
00236         {
00237             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00238             return false;
00239         }
00240     }
00241     else
00242         return false;
00243 }
```

8.2.3.9 recuperer() [2/4]

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

Références db, et mutex.

```
00249 {
00250     QMutexLocker verrou(&mutex);
00251     QSqlQuery r;
00252     bool retour;
00253 
00254     if(db.isOpen())
00255     {
00256         if(requete.contains("SELECT"))
00257         {
00258             retour = r.exec(requete);
00259             #ifdef DEBUG_BASEDEDONNEES
00260             qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString, QStringList)> retour %1 pour
       la requete : %2").arg(QString::number(retour)).arg(requete);
00261             #endif
00262             if(retour)
00263             {
00264                 // on se positionne sur l'enregistrement
00265                 r.first();
00266 
00267                 // on vérifie l'état de l'enregistrement retourné
00268                 if(!r.isValid())
00269                 {
00270                     #ifdef DEBUG_BASEDEDONNEES
00271                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Résultat non valide !");
00272                     #endif
00273                     return false;
00274                 }
00275 
00276                 // on récupère sous forme de QString la valeur de tous les champs sélectionnés
00277                 // et on les stocke dans une liste de QString
00278                 for(int i=0;i<r.record().count();i++)
00279                     if(!r.isNull(i))
00280                         donnees << r.value(i).toString();
00281                 #ifdef DEBUG_BASEDEDONNEES
00282                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00283                 #endif
00284                 return true;
00285             }
00286             else
00287             {
00288                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00289                 return false;
00290             }
00291         }
00292         else
00293         {
00294             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00295             return false;
00296         }
00297     }
00298     else
00299         return false;
00300 }
```

8.2.3.10 recuperer() [3/4]

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

Références db, et mutex.

```
00306 {
00307     QMutexLocker verrou(&mutex);
00308     QSqlQuery r;
00309     bool retour;
00310     QString data;
00311 
00312     if(db.isOpen())
00313     {
00314         if(requete.contains("SELECT"))
00315         {
00316             retour = r.exec(requete);
00317             #ifdef DEBUG_BASEDEDONNEES
00318             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00319             #endif
00320             if(retour)
```

```
00321             {
00322                 // pour chaque enregistrement
00323                 while ( r.next() )
00324                 {
00325                     // on récupère sous forme de QString la valeur du champs sélectionné
00326                     data = r.value(0).toString();
00327 
00328                     #ifdef DEBUG_BASEDEDONNEES
00329                     //qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00330                     #endif
00331 
00332                     // on stocke l'enregistrement dans le QVector
00333                     donnees.push_back(data);
00334                 }
00335                 #ifdef DEBUG_BASEDEDONNEES
00336                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00337                 #endif
00338                 return true;
00339             }
00340             else
00341             {
00342                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00343                 return false;
00344             }
00345         }
00346         else
00347         {
00348             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00349             return false;
00350         }
00351     }
00352     else
00353         return false;
00354 }
```

8.2.3.11 recuperer() [4/4]

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

Références db, et mutex.

```
00360 {
00361     QMutexLocker verrou(&mutex);
00362     QSqlQuery r;
00363     bool retour;
00364     QStringList data;
00365 
00366     if(db.isOpen())
00367     {
00368         if(requete.contains("SELECT"))
00369         {
00370             retour = r.exec(requete);
00371             #ifdef DEBUG_BASEDEDONNEES
00372             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00373             #endif
00374             if(retour)
00375             {
00376                 // pour chaque enregistrement
00377                 while ( r.next() )
00378                 {
00379                     // on récupère sous forme de QString la valeur de tous les champs sélectionnés
00380                     // et on les stocke dans une liste de QString
00381                     for(int i=0;i<r.record().count();i++)
00382                         data << r.value(i).toString();
00383 
00384                     #ifdef DEBUG_BASEDEDONNEES
00385                     //qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00386                     /*for(int i=0;i<r.record().count();i++)
00387                         qDebug() << r.value(i).toString();*/
00388                     #endif
00389 
00390                     // on stocke l'enregistrement dans le QVector
00391                     donnees.push_back(data);
00392 
00393                     // on efface la liste de QString pour le prochain enregistrement
```

```
00394                     data.clear();
00395                 }
00396                 #ifdef DEBUG_BASEDEDONNEES
00397                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00398                 #endif
00399                 return true;
00400             }
00401             else
00402             {
00403                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00404                 return false;
00405             }
00406         }
00407         else
00408         {
00409             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00410             return false;
00411         }
00412     }
00413     else
00414         return false;
00415 }
```

8.2.4 Documentation des données membres

8.2.4.1 baseDeDonnees

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

Référencé par detruireInstance(), et getInstance().

8.2.4.2 db

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

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

8.2.4.3 mutex

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

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

8.2.4.4 nbAcces

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

Référencé par detruireInstance(), et getInstance().

8.2.4.5 typeBase

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

Référencé par BaseDeDonnees(), connecter(), et ouvrir().

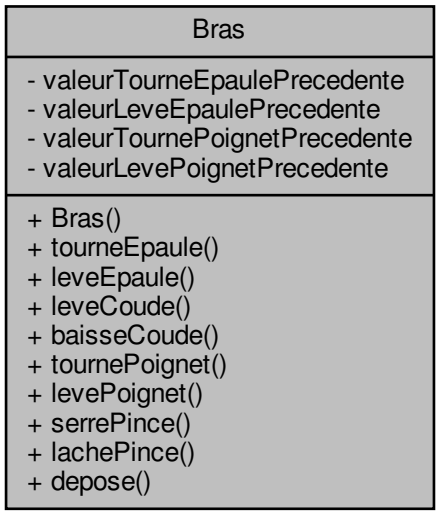
La documentation de cette classe a été générée à partir des fichiers suivants :
- basededonnees.h
- basededonnees.cpp

8.3 Référence de la classe Bras

Déclaration de la classe Bras. Réceptionne les signaux de la manette destiné aux mouvements du bras, et émet les trames correspondantes.

```
#include <bras.h>
```

Graphe de collaboration de Bras :



Connecteurs publics

- void tourneEpaule (double valeur)
  Contrôle la rotation de l'épaule (entre autre, la direction du bras), en émettant la trame correspondante. Correspond au joystick droite de la manette.
- void leveEpaule (double valeur)
  Contrôle l'angle de levé de l'épaule, en émettant la trame correspondante. Correspond au joystick droite de la manette.
- void leveCoude (bool appuye)
  Permet de lever le coude, en émettant la trame correspondante. Correspond au bouton triangle de la manette.
- void baisseCoude (bool appuye)
  Permet de plier le coude, en émettant la trame correspondante. Correspond au bouton X de la manette.
- void tournePoignet (double valeur)
  Permet de pivoter le poignet, en émettant la trame correspondante. Correspond au joystick gauche axe X de la manette.
- void levePoignet (double valeur)
  Permet de lever ou baisser le poignet, en émettant la trame correspondante. Correspond au joystick gauche axe Y de la manette.
- void serrePince (bool appuye)
  Permet de serrer la pince, en émettant la trame correspondante. Correspond au bouton R1 de la manette.
- void lachePince (bool appuye)
  Permet de relâcher la pince, en émettant la trame correspondante. Correspond au bouton L1 de la manette.
- void depose (bool appuye)
  Emet la trame : poser dans le bac le contenu de la pince.

Signaux

— void trameCree (QString trame)
Signal émis lorsqu'une trame a été créée et prête à être transmise.

Fonctions membres publiques

— Bras (QObject ∗parent=nullptr)
Constructeur de la classe Bras.

Attributs privés

— int valeurTourneEpaulePrecedente
Dernière valeur de la trame TourneEpaule émise.
— int valeurLeveEpaulePrecedente
Dernière valeur de la trame LeveEpaule émise.
— int valeurTournePoignetPrecedente
Dernière valeur de la trame TournePoignet émise.
— int valeurLevePoignetPrecedente
Dernière valeur de la trame LevePoignet émise.

8.3.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Juin 2019

8.3.2 Documentation des constructeurs et destructeur

8.3.2.1 Bras()

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

Paramètres

parent	QObject∗ objet Qt parent

```
00023                         : QObject(parent), valeurTourneEpaulePrecedente(0),
      valeurLeveEpaulePrecedente(0),
      valeurTournePoignetPrecedente(0),
      valeurLevePoignetPrecedente(0)
00024 {
00025
00026 }
```

8.3.3 Documentation des fonctions membres

8.3.3.1 baisseCoude

```
Bras::baisseCoude (
            bool appuye )  [slot]
```

Emet la trame : plier le coude.

Paramètres

appuye	bool Bouton appuyé, ou non.
appuye	bool Touche appuyée ou non.

Références trameCree().

Référencé par ControleRov : :changeConnexions().

```
00098 {
00099     QString trame = "$LCO" + QString::number(-appuye) + "\n";
00100
00101     emit trameCree(trame);
00102 }
```

8.3.3.2 depose

```
Bras::depose (
            bool appuye )  [slot]
```

Paramètres

appuye	bool Touche appuyée ou non.

Références trameCree().

Référencé par ControleRov : :changeConnexions().

```
00182 {
00183     QString trame = "$DEP" + QString::number(appuye) + "/";
00184
00185     emit trameCree(trame);
00186 }
```

8.3.3.3 lachePince

```
Bras::lachePince (
            bool appuye )  [slot]
```

Emet la trame : relâcher la pince.

Paramètres

appuye	bool Bouton appuyé, ou non.
appuye	bool Touche appuyée ou non.

Références trameCree().

Référencé par ControleRov : :changeConnexions().

```
00170 {
00171     QString trame = "$OPI" + QString::number(-appuye) + "\n";
00172
00173     emit trameCree(trame);
00174 }
```

8.3.3.4 leveCoude

```
Bras::leveCoude (
            bool appuye )  [slot]
```

Emet la trame : lever le coude.

Paramètres

appuye	bool Bouton appuyé, ou non.

A faire Changer l'envoie de trame. Envoyer 1 quand Triangle, -1 quand Croix. -> Diminution nombre code différents dans la trame, simplification du décodage des trames.

Paramètres

appuye	bool Touche appuyée ou non.

Références trameCree().

Référencé par ControleRov : :changeConnexions().

```
00086 {
00087     QString trame = "$LCO" + QString::number(appuye) + "\n";
00088
00089     emit trameCree(trame);
00090 }
```

8.3.3.5 leveEpaule

```
Bras::leveEpaule (
            double valeur )  [slot]
```

Emet la trame : lever ou baisser l'épaule du bras.

Paramètres

valeur	double Force de l'appui sur le joystick (entre -1 et 1).
valeur	double Force d'appui sur le joystick.

Références trameCree(), et valeurLeveEpaulePrecedente.

Référencé par ControleRov : :changeConnexions().

```
00060 {
00061     QString trame = "$LEP";
00062     int direction = 0;       // Par défaut, la direction est indiquée à 0
00063
00064     if (valeur <= -0.5)
00065         direction = 1;
00066
00067     else if (valeur >= 0.5)
00068         direction = -1;
00069
00070     if (valeurLeveEpaulePrecedente != direction)
00071     {
00072         valeurLeveEpaulePrecedente = direction;
00073         trame += QString::number(direction) + "\n";
00074        emit trameCree(trame);
00075     }
00076 }
```

8.3.3.6 levePoignet

```
Bras::levePoignet (
             double valeur )  [slot]
```

Paramètres

valeur	double Force d'appui sur le joystick.

Références trameCree(), et valeurLevePoignetPrecedente.

Référencé par ControleRov : :changeConnexions().

```
00134 {
00135     QString trame = "$LPO";
00136     int direction = 0;       // Par défaut, la direction est indiquée à 0
00137
00138     if (valeur <= -0.5)
00139         direction = 1;
00140
00141     else if (valeur >= 0.5)
00142         direction = -1;
00143
00144     if (valeurLevePoignetPrecedente != direction)
00145     {
00146         valeurLevePoignetPrecedente = direction;
00147         trame += QString::number(direction) + "\n";
00148         emit trameCree(trame);
00149     }
00150 }
```

8.3.3.7 serrePince

```
Bras::serrePince (
             bool appuye )  [slot]
```

Emet la trame : serrer la pince.

Paramètres

appuye	bool Bouton appuyé, ou non.
appuye	bool Touche appuyée ou non.

Références trameCree().

Référencé par ControleRov : :changeConnexions().

```
00158 {
00159     QString trame = "$FPI" + QString::number(appuye) + "\n";
00160
00161     emit trameCree(trame);
00162 }
```

8.3.3.8 tourneEpaule

```
Bras::tourneEpaule (
            double valeur )  [slot]
```

Emet la trame : tourner l'épaule du bras à droite/gauche.

Paramètres

valeur	double Force de l'appui sur le joystick (entre -1 et 1).
valeur	double Force d'appui sur le joystick.

Références trameCree(), et valeurTourneEpaulePrecedente.

Référencé par ControleRov : :changeConnexions().

```
00035 {
00036     QString trame = "$TEP";
00037     int direction = 0;      // Par défaut, la direction est indiquée à 0
00038
00039     if (valeur <= -0.5)
00040         direction = 1;
00041
00042     else if (valeur >= 0.5)
00043         direction = -1;
00044
00045     if (valeurTourneEpaulePrecedente != direction)
00046     {
00047         valeurTourneEpaulePrecedente = direction;
00048         trame += QString::number(direction) + "\n";
00049         if (direction != 0)
00050             emit trameCree(trame);
00051     }
00052 }
```

8.3.3.9 tournePoignet

```
Bras::tournePoignet (
            double valeur )  [slot]
```

Paramètres

valeur	double Force d'appui sur le joystick.

Références trameCree(), et valeurTournePoignetPrecedente.

Référencé par ControleRov : :changeConnexions().

```
00110 {
00111     QString trame = "$TPO";
00112     int direction = 0;      // Par défaut, la direction est indiquée à 0
00113
00114     if (valeur <= -0.5)
00115         direction = -1;
00116
00117     else if (valeur >= 0.5)
00118         direction = 1;
00119
00120     if (valeurTournePoignetPrecedente != direction)
00121     {
00122         valeurTournePoignetPrecedente = direction;
00123         trame += QString::number(direction) + "\n";
00124         emit trameCree(trame);
00125     }
00126 }
```

8.3.3.10 trameCree

```
void Bras::trameCree (
             QString trame )  [signal]
```

Référencé par baisseCoude(), ControleRov : :ControleRov(), depose(), lachePince(), leveCoude(), leveEpaule(), levePoignet(), serrePince(), tourneEpaule(), et tournePoignet().

8.3.4 Documentation des données membres

8.3.4.1 valeurLeveEpaulePrecedente

```
int Bras::valeurLeveEpaulePrecedente  [private]
```

Référencé par leveEpaule().

8.3.4.2 valeurLevePoignetPrecedente

```
int Bras::valeurLevePoignetPrecedente  [private]
```

Référencé par levePoignet().

8.3.4.3 valeurTourneEpaulePrecedente

```
int Bras::valeurTourneEpaulePrecedente  [private]
```

Référencé par tourneEpaule().

8.3.4.4 valeurTournePoignetPrecedente

```
int Bras::valeurTournePoignetPrecedente  [private]
```

Référencé par tournePoignet().

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

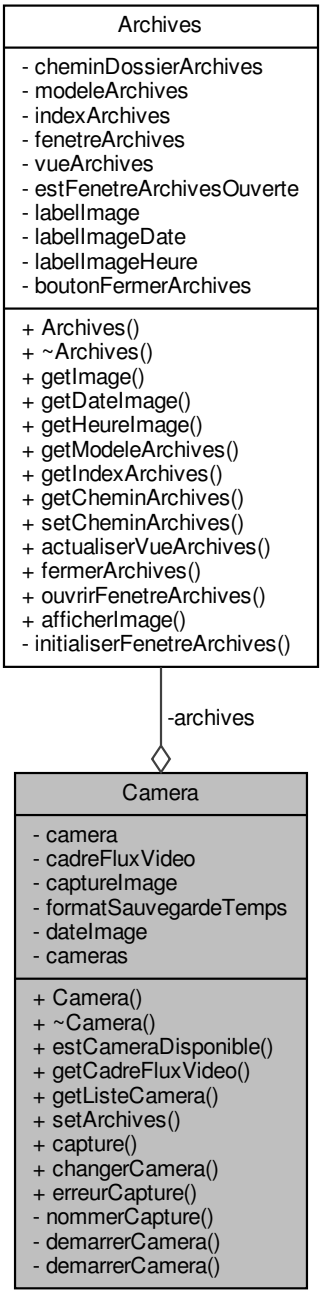
— bras.h
— bras.cpp

8.4 Référence de la classe Camera

Déclaration de la classe Camera.

```
#include <camera.h>
```

Graphe de collaboration de Camera :



Connecteurs publics

- void capture ()
  Capture l'image du flux video.
- void changerCamera (QString)
  Permet de démarrer le flux vidéo de la caméra choisir dans le QComboBox sur l'IHM.
- void erreurCapture (int id, QCameraImageCapture : :Error error, const QString &errorString)

Fonctions membres publiques

- Camera (QObject ∗parent=nullptr)

- ~Camera ()
- bool estCameraDisponible ()
  Retourne un booléen sur l'état de disponibilité de la caméra.
- QCameraViewfinder * getCadreFluxVideo ()
  retourne le flux video
- QList< QCameraInfo > getListeCamera ()
  retourne toute les caméras disponible
- void setArchives (Archives *archives)

Fonctions membres privées

- QString nommerCapture ()
  Renomme la photo capturée au format : "yyyy-MM-dd_HH :mm :ss".
- void demarrerCamera ()
- void demarrerCamera (QCameraInfo)
  Demarre le retour vidéo sur l'IHM.

Attributs privés

- QCamera * camera
  Permet la connexion avec la caméra.
- Archives * archives
  Permet la connexion avec les archives.
- QCameraViewfinder * cadreFluxVideo
  Permet l'affichage du flux vidéo.
- QCameraImageCapture * captureImage
  Permet la capture d'image.
- QString formatSauvegardeTemps
  Le format de sauvegarde du temps pour l'archivages.
- QString dateImage
  Stock la date de la prise de photo pour l'archivage.
- QList< QCameraInfo > cameras
  Stock la liste des caméras disponibles.

8.4.1 Description détaillée

Auteur

Nicolas BOFFREDO

Version

1.1

Date

Jeudi 13 Juin 2019

8.4.2 Documentation des constructeurs et destructeur

8.4.2.1 Camera()

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

Références cadreFluxVideo, camera, cameras, captureImage, demarrerCamera(), et estCameraDisponible().

```
00022                               : QObject(parent),
00023   camera(nullptr),
00024   archives(nullptr),
00025   cadreFluxVideo(nullptr),
00026   captureImage(nullptr),
00027   formatSauvegardeTemps("dd-MM-yyyy_HH:mm:ss")
00028 {
00029     cadreFluxVideo = new QCameraViewfinder;
00030     camera = new QCamera;
00031     captureImage = new QCameraImageCapture(camera);
00032     cameras = QCameraInfo::availableCameras();
00033
00034     if(estCameraDisponible())
00035         demarrerCamera(cameras[0]);
00036
00037     qDebug() << Q_FUNC_INFO << cameras;
00038 }
```

8.4.2.2 ~Camera()

```
Camera::~Camera ( )
```

Références cadreFluxVideo, et camera.

```
00041 {
00042     delete cadreFluxVideo;
00043     delete camera;
00044     qDebug() << Q_FUNC_INFO;
00045 }
```

8.4.3 Documentation des fonctions membres

8.4.3.1 capture

```
void Camera::capture ( )  [slot]
```

Références captureImage, et nommerCapture().

```
00116 {
00117     qDebug() << Q_FUNC_INFO;
00118     QString nomCapture = this->nommerCapture();
00119     qDebug() << Q_FUNC_INFO << "nomCapture" << nomCapture;
00120     captureImage->capture(nomCapture);
00121 }
```

8.4.3.2 changerCamera

```
void Camera::changerCamera (
            QString nomCamera )  [slot]
```

Paramètres

nomCamera	Un *QString*, le nom de la caméra

Références cameras, et demarrerCamera().

```
00142 {
00143     QString cameraTrouvee;
00144     QList<QCameraInfo> cameras = QCameraInfo::availableCameras();
00145     foreach (const QCameraInfo &cameraInfo, cameras)
00146     {
00147         cameraTrouvee = cameraInfo.description() + " (" + cameraInfo.deviceName()+ ")";
00148         if (cameraTrouvee == nomCamera)
00149         {
00150             this->demarrerCamera(cameraInfo);
00151         }
00152     }
00153 }
```

8.4.3.3 demarrerCamera() [1/2]

```
void Camera::demarrerCamera ( )  [private]
```

Référencé par Camera(), et changerCamera().

8.4.3.4 demarrerCamera() [2/2]

```
void Camera::demarrerCamera (
            QCameraInfo cameraSelectionnee )  [private]
```

Par défaut la méthode reçoit la première caméra trouvée

Paramètres

cameraSelectionnee	

Références cadreFluxVideo, camera, captureImage, erreurCapture(), et estCameraDisponible().

```
00092 {
00093     if(camera != nullptr)
00094         delete camera;
00095     if(captureImage != nullptr)
00096         delete captureImage;
00097
00098     if(estCameraDisponible())
00099     {
00100         qDebug() << Q_FUNC_INFO << "cameraSelectionnee" << cameraSelectionnee.deviceName();
00101         camera = new QCamera(cameraSelectionnee);
00102         camera->setViewfinder(cadreFluxVideo);
00103         camera->setCaptureMode(QCamera::CaptureStillImage);
00104         captureImage = new QCameraImageCapture(camera);
00105         captureImage->setCaptureDestination(QCameraImageCapture::CaptureToBuffer);
00106         connect(captureImage, SIGNAL(error(int,QCameraImageCapture::Error,QString)), this, SLOT
      (erreurCapture(int,QCameraImageCapture::Error,QString)));
00107
00108         camera->start();
00109     }
00110 }
```

8.4.3.5 erreurCapture

```
void Camera::erreurCapture (
            int id,
            QCameraImageCapture::Error error,
            const QString & errorString )  [slot]
```

Référencé par demarrerCamera().

```
00156 {
00157     Q_UNUSED(id)
00158     Q_UNUSED(error)
00159     qDebug() << Q_FUNC_INFO << errorString;
00160 }
```

8.4.3.6 estCameraDisponible()

```
bool Camera::estCameraDisponible ( )
```

Renvoie

Un *booléen*, vrai si la caméra est disponible, faux sinon.

Références cameras.

Référencé par IHMRov : :actualiseIconesEtat(), Camera(), et demarrerCamera().

```
00052 {
00053     if (cameras.count() > 0)
00054         return true;
00055
00056     else
00057         return false;
00058 }
```

8.4.3.7 getCadreFluxVideo()

```
QCameraViewfinder * Camera::getCadreFluxVideo ( )
```

Un assesseur permettant d'avoir le retour vidéo

Références cadreFluxVideo.

Référencé par IHMRov : :IHMRov().

```
00082 {
00083     return cadreFluxVideo;
00084 }
```

8.4.3.8 getListeCamera()

```
QList< QCameraInfo > Camera::getListeCamera ( )
```

Références cameras.

Référencé par IHMRov : :initialiserListeCamera().

```
00063 {
00064     QList<QCameraInfo> listeCamera;
00065     foreach (const QCameraInfo &cameraInfo, cameras)
00066     {
00067         listeCamera.append(cameraInfo);
00068     }
00069     return listeCamera;
00070 }
```

8.4.3.9 nommerCapture()

```
QString Camera::nommerCapture ( )  [private]
```

Indique le chemin vers un dossier de stockage des photos, à l'emplacement de l'éxécutable.

Références archives, dateImage, formatSauvegardeTemps, et Archives : :getCheminArchives().

Référencé par capture().

```
00128 {
00129     QString nom = QApplication::applicationDirPath() + "/defaut";
00130     QDateTime dateCapture = QDateTime::currentDateTime();
00131     dateImage = dateCapture.toString(formatSauvegardeTemps);
00132     nom = archives->getCheminArchives() + "/" +
      dateImage;
00133     qDebug() << Q_FUNC_INFO << nom;
00134     return nom;
00135 }
```

8.4.3.10 setArchives()

```
void Camera::setArchives (
            Archives * archives )
```

Références archives.

Référencé par IHMRov : :IHMRov().

```
00073 {
00074     this->archives = archives;
00075 }
```

8.4.4 Documentation des données membres

8.4.4.1 archives

```
Archives* Camera::archives  [private]
```

Référencé par nommerCapture(), et setArchives().

8.4.4.2 cadreFluxVideo

```
QCameraViewfinder* Camera::cadreFluxVideo  [private]
```

Référencé par Camera(), demarrerCamera(), getCadreFluxVideo(), et ~Camera().

8.4.4.3 camera

```
QCamera* Camera::camera  [private]
```

Référencé par Camera(), demarrerCamera(), et ~Camera().

8.4.4.4 cameras

```
QList<QCameraInfo> Camera::cameras  [private]
```

Référencé par Camera(), changerCamera(), estCameraDisponible(), et getListeCamera().

8.4.4.5 captureImage

```
QCameraImageCapture* Camera::captureImage  [private]
```

Référencé par Camera(), capture(), et demarrerCamera().

8.4.4.6 dateImage

```
QString Camera::dateImage  [private]
```

Référencé par nommerCapture().

8.4.4.7 formatSauvegardeTemps

```
QString Camera::formatSauvegardeTemps  [private]
```

Référencé par nommerCapture().

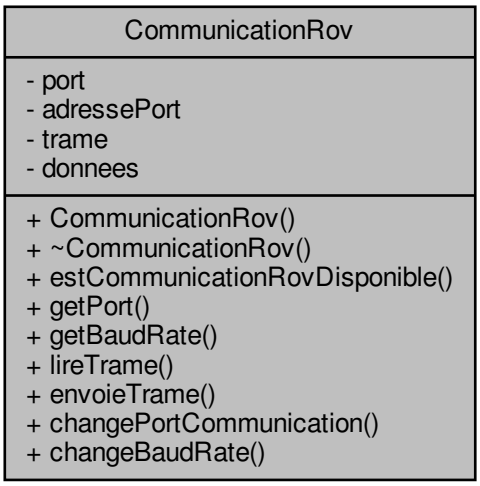
La documentation de cette classe a été générée à partir des fichiers suivants :
- camera.h
- camera.cpp

8.5 Référence de la classe CommunicationRov

Déclaration de la classe CommunicationRov. Gère la communication entre le Rov et le Rov.

`#include <communicationrov.h>`

Graphe de collaboration de CommunicationRov :



Connecteurs publics

- void lireTrame ()
  Récupère la trame envoyée par le Rov, et la renvoie sous un signal si cette dernière est complète.
- bool envoieTrame (QString trame)
  Envoie une trame au Rov.
- void changePortCommunication (QString nouveauPort)
  Modifie le port de communication du rov.
- void changeBaudRate (QString nouveauBaudRate)
  Modifie le baudrate utilisé.

Signaux

- void trameRecue (QString trame)
  Signal émis lorsque des nouvelles données ont été reçues.

Fonctions membres publiques

- CommunicationRov (QObject ∗parent=nullptr)
- ∼CommunicationRov ()
- bool estCommunicationRovDisponible ()
  Retourne l'é&tat d'ouverture du port de communication vers le Rov.
- const QString getPort ()
  Retourne le port de communication utilisé.
- const QString getBaudRate ()
  Retourne le baudrate utilisé.

Attributs privés

- QSerialPort * port
  Port série pour la communication le programme et le Rov.
- QString adressePort
  Adresse du port de communication.
- QString trame
  Trame reçue par le port.
- QByteArray donnees
  Dernière donnée reçue (ou en cours de réception)

8.5.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Juin 2019

8.5.2 Documentation des constructeurs et destructeur

8.5.2.1 CommunicationRov()

```
CommunicationRov::CommunicationRov (
            QObject * parent = nullptr )  [explicit]
```

Références adressePort, lireTrame(), et port.

```
00017                                                : QObject(parent), adressePort("/dev/ttyUSB0")
00018 {
00019     port = new QSerialPort(adressePort);
00020
00021     port->setBaudRate(QSerialPort::Baud38400);
00022
00023     port->open(QIODevice::ReadWrite);
00024     qDebug() << Q_FUNC_INFO << "Port ouvert :" << port->isOpen();
00025
00026     if(port->isOpen())
00027         connect(port, SIGNAL(readyRead()), this, SLOT(lireTrame()));
00028 }
```

8.5.2.2 ~CommunicationRov()

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

Références port.

```
00031 {
00032     port->close();
00033     delete port;
00034 }
```

8.5.3 Documentation des fonctions membres

8.5.3.1 changeBaudRate

```
void CommunicationRov::changeBaudRate (
            QString nouveauBaudRate )  [slot]
```

Change le débit (baudrate) utilisé pour la communication avec le rov.

Paramètres

nouveauBaudRate	QString le nouveau baudrate à utiliser.

Références port.

Référencé par IHMRov : :creerFenetreParametres().

```
00076 {
00077     port->setBaudRate(nouveauBaudRate.toInt());
00078     qDebug() << Q_FUNC_INFO << nouveauBaudRate.toInt();
00079 }
```

8.5.3.2 changePortCommunication

```
void CommunicationRov::changePortCommunication (
            QString nouveauPort )  [slot]
```

Change le port de communication utilisé pour la liaison avec le rov.

Paramètres

nouveauPort	QString le nouveau port auquel se connecter.

Références adressePort, et port.

Référencé par IHMRov : :creerFenetreParametres().

```
00062 {
00063     port->close();
00064     port->setPortName(nouveauPort);
00065     port->open(QIODevice::ReadWrite);
00066     adressePort = nouveauPort;
00067     qDebug() << "Port : " << port->portName() << "Ouvert : " << port->isOpen();
00068 }
```

8.5.3.3 envoieTrame

```
CommunicationRov::envoieTrame (
            QString trame )  [slot]
```

Envoie la trame passée en argument au Rov par liaison série.

Paramètres

trame	QString Trame à envoyer.

Renvoie

bool envoye, vrai si la trame a été envoyée, sinon faux

Paramètres

trame	QString le contenu de la trame envoyée au Rov

Références estCommunicationRovDisponible(), port, et trame.

Référencé par ControleRov : :ControleRov().

```
00118 {
00119     if(!estCommunicationRovDisponible())
00120         return false;
00121
00122     qDebug() << Q_FUNC_INFO << "trame" << trame;
00123     qint64 nbEnvoyes = port->write(trame.toLocal8Bit());
00124     //port->waitForReadyRead(1100);
00125
00126     if(nbEnvoyes > 0)
00127         return true;
00128     return false;
00129 }
```

8.5.3.4 estCommunicationRovDisponible()

```
CommunicationRov::estCommunicationRovDisponible ( )
```

Renvoie

bool disponible, vrai si le port vers le Rov est ouvert sinon faux
bool vrai si le port vers le Rov est ouvert sinon faux

Références port.

Référencé par IHMRov : :actualiseIconesEtat(), Rov : :creerNouvelleCampagne(), IHMRov : :enregistrerParametres(), et envoie↩Trame().

```
00087 {
00088     return port->isOpen();
00089 }
```

8.5.3.5 getBaudRate()

```
const QString CommunicationRov::getBaudRate ( )
```

Indique le Baudrate utilisé pour la communication avec le rov.

Renvoie

baudrate QString le baudrate utilisé actuellement pour la communication.

Références port.

Référencé par IHMRov : :creerFenetreParametres().

```
00052 {
00053     return QString::number(port->baudRate());
00054 }
```

8.5.3.6 getPort()

```
const QString CommunicationRov::getPort ( )
```

Indique le port utilisé pour la communication avec le rov.

Renvoie

adressePort QString le port de communication utilisé actuellement.

Références adressePort.

Référencé par IHMRov : :creerFenetreParametres().

```
00042 {
00043     return adressePort;
00044 }
```

8.5.3.7 lireTrame

```
CommunicationRov::lireTrame ( )  [slot]
```

Récupère la trame envoyée par le rov, et émet un signal.

Références donnees, port, et trameRecue().

Référencé par CommunicationRov().

```
00096 {
00097     donnees += port->readAll();
00098     qDebug() << donnees;
00099
00100     if(donnees.endsWith("\n"))
00101     {
00102         if(donnees.startsWith("$"))
00103         {
00104             qDebug() << Q_FUNC_INFO << donnees;
00105             emit trameRecue(donnees);
00106         }
00107
00108         donnees.clear();
00109     }
00110 }
```

8.5.3.8 trameRecue

```
CommunicationRov::trameRecue (
            QString trame )  [signal]
```

Paramètres

trame	QString le contenu de la trame reçue du Rov

Référencé par lireTrame().

8.5.4 Documentation des données membres

8.5.4.1 adressePort

```
QString CommunicationRov::adressePort  [private]
```

Référencé par changePortCommunication(), CommunicationRov(), et getPort().

8.5.4.2 donnees

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

Référencé par lireTrame().

8.5.4.3 port

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

Référencé par changeBaudRate(), changePortCommunication(), CommunicationRov(), envoieTrame(), estCommunicationRov↩Disponible(), getBaudRate(), lireTrame(), et ∼CommunicationRov().

8.5.4.4 trame

```
QString CommunicationRov::trame  [private]
```

Référencé par envoieTrame().

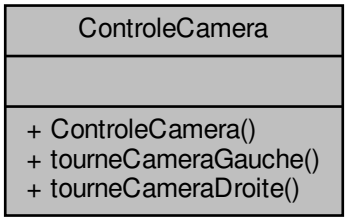
La documentation de cette classe a été générée à partir des fichiers suivants :
— communicationrov.h
— communicationrov.cpp

8.6 Référence de la classe ControleCamera

Déclaration de la classe ControleCamera.

```
#include <controlecamera.h>
```

Graphe de collaboration de ControleCamera :



Connecteurs publics

- void tourneCameraGauche (bool boutonAppuye)
  Permet de pivoter la caméra vers la gauche, en émettant la trame correspondante. Correspond à la croix directionnelle gauche de la manette.
- void tourneCameraDroite (bool boutonAppuye)
  Permet de pivoter la caméra vers la droite, en émettant la trame correspondante. Correspond à la croix directionnelle droite de la manette.

Signaux

- void trameCree (QString trame)
  Signal émis lorsqu'une trame a été crée et prête à être transmise.

Fonctions membres publiques

- ControleCamera (QObject ∗parent=nullptr)

8.6.1 Description détaillée

Auteur

Nicolas BOFFREDO

Version

1.1

Date

Jeudi 13 Juin 2019

8.6.2 Documentation des constructeurs et destructeur

8.6.2.1 ControleCamera()

```
ControleCamera::ControleCamera (
            QObject * parent = nullptr )  [explicit]
```

```
00017                                           : QObject(parent)
00018 {
00019
00020 }
```

8.6.3 Documentation des fonctions membres

8.6.3.1 tourneCameraDroite

```
ControleCamera::tourneCameraDroite (
            bool boutonAppuye )  [slot]
```

Paramètres

boutonAppuye	bool vrai si le bouton est appuyé, faux sinon.

Références trameCree().

Référencé par ControleRov : :ControleRov().

```
00039 {
00040     QString trame = "$TCA" + QString::number(boutonAppuye) + "\n";
00041     emit trameCree(trame);
00042 }
```

8.6.3.2 tourneCameraGauche

```
ControleCamera::tourneCameraGauche (
            bool boutonAppuye )  [slot]
```

Paramètres

boutonAppuye	bool vrai si le bouton est appuyé, faux sinon.

Références trameCree().

Référencé par ControleRov : :ControleRov().

```
00028 {
00029     QString trame = "$TCA" + QString::number(-boutonAppuye) + "\n";
00030     emit trameCree(trame);
00031 }
```

8.6.3.3 trameCree

```
void ControleCamera::trameCree (
            QString trame )  [signal]
```

Référencé par tourneCameraDroite(), et tourneCameraGauche().

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

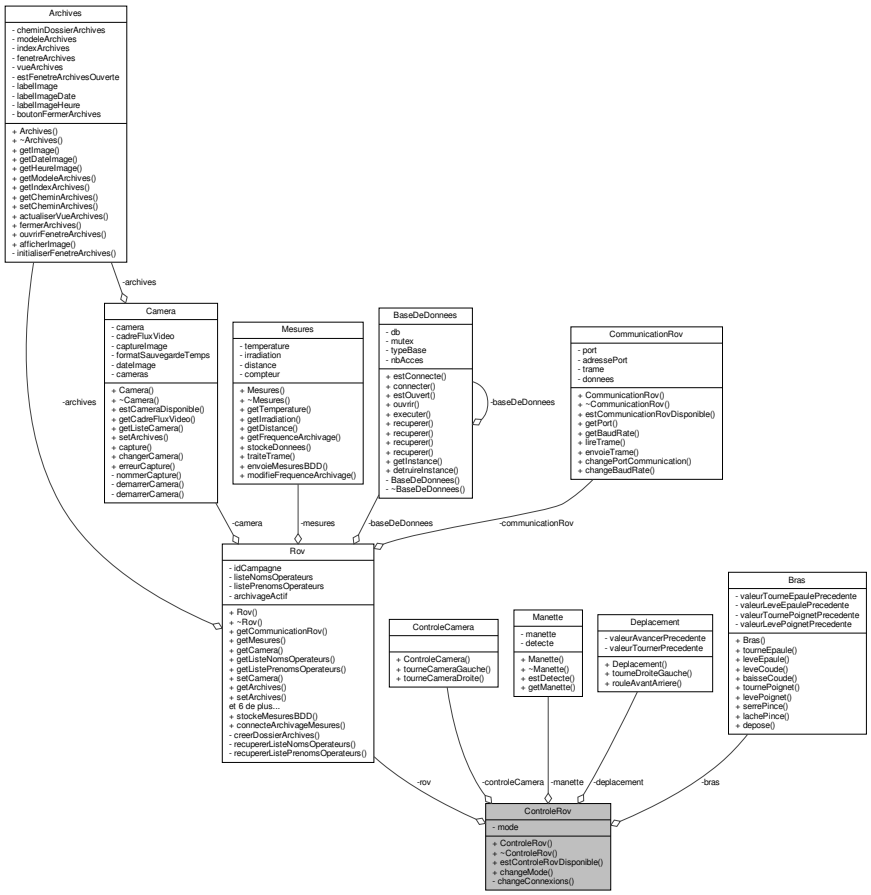
- controlecamera.h
- controlecamera.cpp

8.7 Référence de la classe ControleRov

Déclaration de la classe ControleRov. Permet le contrôle des éléments du rov, en reliant la manette aux méthodes de déplacement.

`#include <controlerov.h>`

Graphe de collaboration de ControleRov :



Connecteurs publics

— void changeMode (bool appuye)
change le mode de la manette (Deplacement / Bras).

Signaux

— void trameCree (QString trame)
Signal émis lorsqu'une trame a été crée et est destinée à l'envoi.

Fonctions membres publiques

— ControleRov (QObject ∗parent=nullptr, Rov ∗rov=nullptr)
— ∼ControleRov ()
— bool estControleRovDisponible () const
Renvoi l'état de connexion de la manette.

Fonctions membres privées

— void changeConnexions (int mode)
Permet de modifier les connexions entre la manette et les actions selon le mode de la manette.

Attributs privés

— Manette * manette
manette utilisée
— Deplacement * deplacement
objet contenant les méthodes utilisées pour le déplacement du rov
— Bras * bras
objet contenant les méthodes utilisées pour les mouvements du bras articulé
— ControleCamera * controleCamera
objet contenant les méthodes utilisées pour les mouvements de la caméra
— Rov * rov
objet Rov contenant l'accès à la communication
— unsigned int mode
Mode de la manette (0 pour déplacement, 1 pour bras)

8.7.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Juin 2019

8.7.2 Documentation des constructeurs et destructeur

8.7.2.1 ControleRov()

```
ControleRov::ControleRov (
            QObject * parent = nullptr,
            Rov * rov = nullptr )  [explicit]
```

Références bras, changeMode(), controleCamera, deplacement, CommunicationRov : :envoieTrame(), Manette : :estDetecte(), Rov↩ : :getCommunicationRov(), Manette : :getManette(), manette, Deplacement : :rouleAvantArriere(), ControleCamera : :tourneCamera↩ Droite(), ControleCamera : :tourneCameraGauche(), Deplacement : :tourneDroiteGauche(), Deplacement : :trameCree(), trameCree(), et Bras : :trameCree().

```
00021                                                 : QObject(parent), rov(rov),
      mode(MODE_DEPLACEMENT)
00022 {
00023     this->bras = new Bras(this);
00024     this->deplacement = new Deplacement(this);
00025     this->controleCamera = new ControleCamera(this);
00026     this->manette = new Manette(this);
00027
00028     if(manette->estDetecte())
00029     {
00030         connect(manette->getManette(), &QGamepad::axisLeftXChanged,
      deplacement, &Deplacement::tourneDroiteGauche);
00031         connect(manette->getManette(), &QGamepad::axisLeftYChanged,
      deplacement, &Deplacement::rouleAvantArriere);
00032         connect(manette->getManette(), &QGamepad::buttonLeftChanged,
      controleCamera, &ControleCamera::tourneCameraGauche);
00033         connect(manette->getManette(), &QGamepad::buttonRightChanged,
      controleCamera, &ControleCamera::tourneCameraDroite);
00034         connect(manette->getManette(), &QGamepad::buttonSelectChanged, this, &
      ControleRov::changeMode);
00035     }
00036
00037     // Connexion entre le signal émis d'une trame à envoyer au slot d'envoi de trame de la classe
       CommunicationRov.
00038     connect(bras, &Bras::trameCree, this->rov->
      getCommunicationRov(), &CommunicationRov::envoieTrame);
00039     // Connexion entre le signal émis d'une trame à envoyer au slot d'envoi de trame de la classe
       CommunicationRov.
00040     connect(deplacement, &Deplacement::trameCree, this->rov->
      getCommunicationRov(), &CommunicationRov::envoieTrame);
00041
00042     connect(this, &ControleRov::trameCree, this->rov->
      getCommunicationRov(), &CommunicationRov::envoieTrame);
00043 }
```

8.7.2.2 ~ControleRov()

```
ControleRov::~ControleRov ( )
```

```
00046 {
00047 }
```

8.7.3 Documentation des fonctions membres

8.7.3.1 changeConnexions()

```
ControleRov::changeConnexions (
            int mode )  [private]
```

Paramètres

mode	int
mode	int Nouveau mode.

Références Bras : :baisseCoude(), bras, deplacement, Bras : :depose(), Manette : :getManette(), Bras : :lachePince(), Bras : :leve↩ Coude(), Bras : :leveEpaule(), Bras : :levePoignet(), manette, MODE_BRAS, MODE_DEPLACEMENT, Deplacement : :rouleAvant↩ Arriere(), Bras : :serrePince(), Deplacement : :tourneDroiteGauche(), Bras : :tourneEpaule(), et Bras : :tournePoignet().

Référencé par changeMode().

```
00093 {
```

```
00094     if (mode == MODE_BRAS)
00095     {
00096         disconnect(manette->getManette(), &QGamepad::axisLeftXChanged,
      deplacement, &Deplacement::tourneDroiteGauche);
00097         disconnect(manette->getManette(), &QGamepad::axisLeftYChanged,
      deplacement, &Deplacement::rouleAvantArriere);
00098
00099         connect(manette->getManette(), &QGamepad::axisLeftXChanged,
      bras, &Bras::tournePoignet);
00100         connect(manette->getManette(), &QGamepad::axisLeftYChanged,
      bras, &Bras::levePoignet);
00101
00102         connect(manette->getManette(), &QGamepad::axisRightXChanged,
      bras, &Bras::tourneEpaule);
00103         connect(manette->getManette(), &QGamepad::axisRightYChanged,
      bras, &Bras::leveEpaule);
00104
00105         connect(manette->getManette(), &QGamepad::buttonUpChanged,
      bras, &Bras::leveCoude);
00106         connect(manette->getManette(), &QGamepad::buttonDownChanged,
      bras, &Bras::baisseCoude);
00107
00108         connect(manette->getManette(), &QGamepad::buttonR1Changed,
      bras, &Bras::serrePince);
00109         connect(manette->getManette(), &QGamepad::buttonL1Changed,
      bras, &Bras::lachePince);
00110
00111         connect(manette->getManette(), &QGamepad::buttonStartChanged,
      bras, &Bras::depose);
00112     }
00113     else if (mode == MODE_DEPLACEMENT)
00114     {
00115         disconnect(manette->getManette(), &QGamepad::axisLeftXChanged,
      bras, &Bras::tournePoignet);
00116         disconnect(manette->getManette(), &QGamepad::axisLeftYChanged,
      bras, &Bras::levePoignet);
00117
00118         disconnect(manette->getManette(), &QGamepad::axisRightXChanged,
      bras, &Bras::tourneEpaule);
00119         disconnect(manette->getManette(), &QGamepad::axisRightYChanged,
      bras, &Bras::leveEpaule);
00120
00121         disconnect(manette->getManette(), &QGamepad::buttonUpChanged,
      bras, &Bras::leveCoude);
00122         disconnect(manette->getManette(), &QGamepad::buttonDownChanged,
      bras, &Bras::baisseCoude);
00123
00124         disconnect(manette->getManette(), &QGamepad::buttonR1Changed,
      bras, &Bras::serrePince);
00125         disconnect(manette->getManette(), &QGamepad::buttonL1Changed,
      bras, &Bras::lachePince);
00126
00127         disconnect(manette->getManette(), &QGamepad::buttonStartChanged,
      bras, &Bras::depose);
00128
00129         connect(manette->getManette(), &QGamepad::axisLeftXChanged,
      deplacement, &Deplacement::tourneDroiteGauche);
00130         connect(manette->getManette(), &QGamepad::axisLeftYChanged,
      deplacement, &Deplacement::rouleAvantArriere);
00131     }
00132     else
00133     {
00134         qDebug() << Q_FUNC_INFO << "ERREUR ! mode inconnu";
00135     }
00136 }
```

8.7.3.2 changeMode

```
void ControleRov::changeMode (
             bool appuye )  [slot]
```

Permet de changer le mode de la manette, à la réception du signal d'appui de la touche Select.

Références changeConnexions(), mode, MODE_BRAS, MODE_DEPLACEMENT, et trameCree().

Référencé par ControleRov().

```
00064 {
00065     if (appuye == 1 && mode == MODE_DEPLACEMENT)
00066     {
00067         qDebug() << Q_FUNC_INFO << "Passage en mode Bras";
00068         mode = MODE_BRAS;
00069         emit trameCree("$CBR1\n");
00070         changeConnexions(MODE_BRAS);
00071     }
00072     else if (appuye == 1 && mode == MODE_BRAS)
00073     {
00074         qDebug() << Q_FUNC_INFO << "Passage en mode Deplacement";
00075         mode = MODE_DEPLACEMENT;
00076         emit trameCree("$CRO1\n");
00077         changeConnexions(MODE_DEPLACEMENT);
00078     }
00079     else if (mode != MODE_DEPLACEMENT && mode !=
      MODE_BRAS)
00080     {
00081         qDebug() << Q_FUNC_INFO << "Erreur de mode ! Retour au mode de déplacement";
00082         mode = MODE_DEPLACEMENT;
00083         changeConnexions(MODE_DEPLACEMENT);
00084     }
00085 }
```

8.7.3.3 estControleRovDisponible()

```
ControleRov::estControleRovDisponible ( ) const
```

Indique si le Rov est contrôlable, ou non.

Renvoie

bool Etat de la manette (connectée, ou non).

Références Manette : :estDetecte(), et manette.

Référencé par IHMRov : :actualiseIconesEtat().

```
00055 {
00056     return manette->estDetecte();
00057 }
```

8.7.3.4 trameCree

```
void ControleRov::trameCree (
            QString trame )  [signal]
```

Référencé par changeMode(), et ControleRov().

8.7.4 Documentation des données membres

8.7.4.1 bras

```
Bras* ControleRov::bras  [private]
```

Référencé par changeConnexions(), et ControleRov().

8.7.4.2 controleCamera

```
ControleCamera* ControleRov::controleCamera  [private]
```

Référencé par ControleRov().

8.7.4.3 deplacement

```
Deplacement* ControleRov::deplacement  [private]
```

Référencé par changeConnexions(), et ControleRov().

8.7.4.4 manette

```
Manette* ControleRov::manette  [private]
```

Référencé par changeConnexions(), ControleRov(), et estControleRovDisponible().

8.7.4.5 mode

```
unsigned int ControleRov::mode  [private]
```

Référencé par changeMode().

8.7.4.6 rov

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

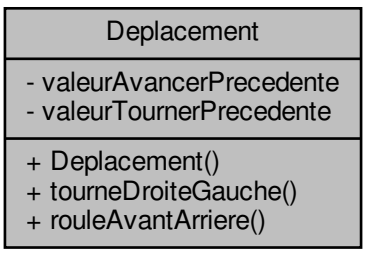
La documentation de cette classe a été générée à partir des fichiers suivants :
- controlerov.h
- controlerov.cpp

8.8 Référence de la classe Deplacement

Déclaration de la classe Deplacement. Réceptionne les signaux de la manette destiné aux déplacements du Rov, et émet les trames correspondantes.

```
#include <deplacement.h>
```

Graphe de collaboration de Deplacement :



Connecteurs publics

- void tourneDroiteGauche (double valeur)
  Slot activé lorsque le joystick gauche est poussé à droite ou gauche. Emet un signal contenant la trame correspondante (déplacement : tourner à droite/gauche).
- void rouleAvantArriere (double valeur)
  Slot activé lorsque le joystick gauche est poussé en avant ou arrière. Emet un signal contenant la trame correspondante (déplacement : avancer/reculer).

Signaux

- void trameCree (QString trame)
  Signal émis lorsqu'une trame est prête à être transmise.

Fonctions membres publiques

- Deplacement (QObject ∗parent=nullptr)

Attributs privés

- QString valeurAvancerPrecedente
  Dernière valeur de la trame Avancer émise.
- QString valeurTournerPrecedente
  Dernière valeur de la trame Tourner émise.

8.8.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Juin 2019

8.8.2 Documentation des constructeurs et destructeur

8.8.2.1 Deplacement()

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

```
00018                                       : QObject(parent),
      valeurAvancerPrecedente("0"), valeurTournerPrecedente("0")
00019 {
00020
00021 }
```

8.8.3 Documentation des fonctions membres

8.8.3.1 rouleAvantArriere

```
Deplacement::rouleAvantArriere (
            double valeur )  [slot]
```

Emet la trame : avancer/reculer.

Paramètres

valeur	double Force de l'appui sur le joystick (compris entre -1 et 1).

Références trameCree(), et valeurAvancerPrecedente.

Référencé par ControleRov : :changeConnexions(), et ControleRov : :ControleRov().

```
00055 {
00056     QString trame = "";
00057     valeur = round(valeur * 3);
00058
00059     if(valeur > 0)
00060         trame = "$RRO" + QString::number(-valeur) + "\n";
00061     else if(valeur < 0)
00062         trame = "$ARO" + QString::number(-valeur) + "\n";
00063     else
00064         trame = "$ARO0\n";
00065
00066     if (QString::number(valeur) != valeurAvancerPrecedente)
00067     {
00068         emit trameCree(trame);
00069         valeurAvancerPrecedente = QString::number(valeur);
00070     }
00071 }
```

8.8.3.2 tourneDroiteGauche

```
Deplacement::tourneDroiteGauche (
             double valeur )  [slot]
```

Emet la trame : tourner à droite/gauche.

Paramètres

valeur	double Force de l'appui sur le joystick (compris entre -1 et 1).

Références trameCree(), et valeurTournerPrecedente.

Référencé par ControleRov : :changeConnexions(), et ControleRov : :ControleRov().

```
00030 {
00031     QString trame = "";
00032
00033     valeur = round(valeur);
00034
00035     if(valeur > 0)
00036         trame = "$DRO" + QString::number(valeur) + "\n";
00037     else if(valeur < 0)
00038         trame = "$GRO" + QString::number(-valeur) + "\n";
00039     else
00040         trame = "$DRO0\n";
00041
00042     if (QString::number(valeur) != valeurTournerPrecedente)
00043     {
00044         emit trameCree(trame);
00045         valeurTournerPrecedente = QString::number(valeur);
00046     }
00047 }
```

8.8.3.3 trameCree

```
void Deplacement::trameCree (
             QString trame )  [signal]
```

Paramètres

trame	QString Trame à transmettre.

Référencé par ControleRov : :ControleRov(), rouleAvantArriere(), et tourneDroiteGauche().

8.8.4 Documentation des données membres

8.8.4.1 valeurAvancerPrecedente

```
QString Deplacement::valeurAvancerPrecedente  [private]
```

Référencé par rouleAvantArriere().

8.8.4.2 valeurTournerPrecedente

```
QString Deplacement::valeurTournerPrecedente  [private]
```

Référencé par tourneDroiteGauche().

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

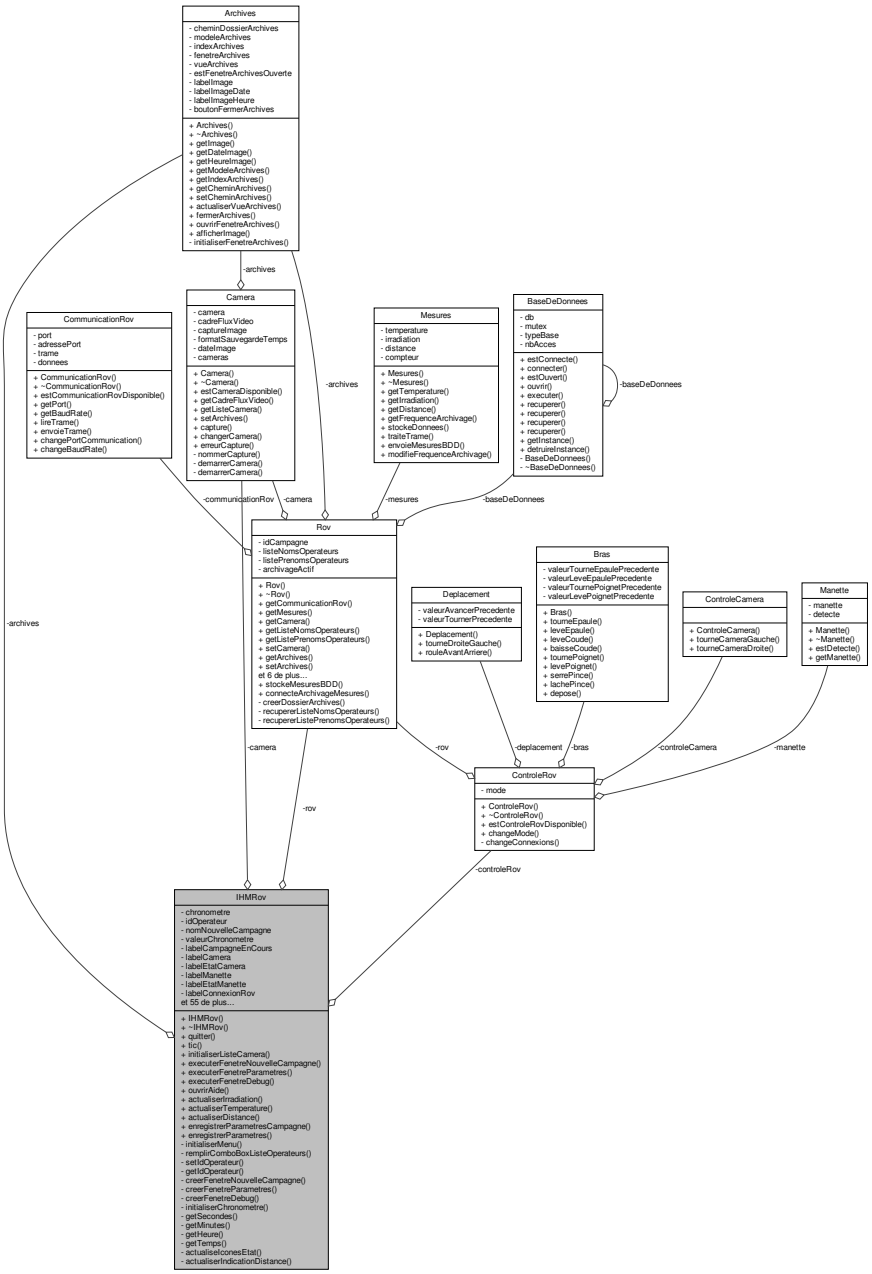
— deplacement.h
— deplacement.cpp

8.9 Référence de la classe IHMRov

Déclaration de la classe Rov.

```
#include <ihmrov.h>
```

Graphe de collaboration de IHMRov :



Connecteurs publics

- void quitter ()
  Permet de fermer l'application.
- void tic ()
  Actualise l'affichage du temps chaque seconde et actualise l'état des icones de rov, manette, et camera.
- void initialiserListeCamera ()
  Ajoute les caméras détectées dans une liste déroulante.
- void executerFenetreNouvelleCampagne ()
  Slot permettant de creer une nouvelle campagne.
- void executerFenetreParametres ()
  Affiche la fenêtre Paramètres.
- void executerFenetreDebug ()
  Affiche la fenêtre de Debug.
- void ouvrirAide ()
  ouvrir une fenetre informative sur l'application.

— void actualiserIrradiation (double irradiation)
Permet d'actualiser l'affichage de l'irradiation sur l'IHM.
— void actualiserTemperature (double temperature)
Permet d'actualiser l'affichage de la temperature sur l'IHM.
— void actualiserDistance (double distance)
Permet d'actualiser l'affichage de la distance sur l'IHM.
— void enregistrerParametresCampagne ()
Methode émettant l'ordre d'enregistrer les paramètres des la campagne.
— void enregistrerParametres ()
Applique les paramètres choisis par l'utilisateur suite à la fenêtre Paramètres.

Signaux

— void creationCampagne ()
— void nouvelleFrequenceArchivage (int)
— void nouveauPortCom (QString)
— void nouveauBaudRate (QString)
— void parametresSauvegardes ()

Fonctions membres publiques

— IHMRov (QWidget ∗parent=nullptr)
Constructeur de la classe IHMRov.
— ∼IHMRov ()
Destructeur de la classe IHMRov.

Fonctions membres privées

— void initialiserMenu ()
initialise la barre de menu
— void remplirComboBoxListeOperateurs ()
Méthode permettant de remplir le ComboBox de la liste des opérateurs au démarrage de l'IHM.
— void setIdOperateur (int idOperateur)
Mutateur de l'attribut idOperateur.
— int getIdOperateur ()
Accesseur de l'attribut idOperateur.
— void creerFenetreNouvelleCampagne ()
Méthode permettant d'initialiser la fenetre de création d'une nouvelle campagne.
— void creerFenetreParametres ()
Crée la fenêtre de Paramètres.
— void creerFenetreDebug ()
Crée la fenêtre de Debug.
— void initialiserChronometre ()
Démarre le chronomètre au lancement de l'application.
— long getSecondes ()
Retourne les secondes depuis le lancement de l'application.
— long getMinutes ()
Retourne les minutes depuis le lancement de l'application.
— long getHeure ()
Retourne les heures depuis le lancement de l'application.
— QString getTemps ()
IHMRov : :getTemps.
— void actualiseIconesEtat ()
Met à jour l'icone d'état de la communication dans l'IHM.
— void actualiserIndicationDistance (double distance)
Permet d'actualiser l'indicateur de la distance sur l'IHM.

Attributs privés

— Camera ∗ camera
association vers la caméra
— ControleRov ∗ controleRov
agrégation du contrôle du Rov
— Archives ∗ archives

association vers Archives
— Rov ∗ rov
association vers le Rov
— QTimer ∗ chronometre
timer pour chronomètrer la campagne
— int idOperateur
id de l'opérateur de la campagne encours
— QString nomNouvelleCampagne
nom de la campagne en cours
— long valeurChronometre
temps
— QLabel ∗ labelCampagneEnCours
— QLabel ∗ labelCamera
— QLabel ∗ labelEtatCamera
— QLabel ∗ labelManette
— QLabel ∗ labelEtatManette
— QLabel ∗ labelConnexionRov
— QLabel ∗ labelEtatConnexionRov
— QLabel ∗ labelChronometre
— QLabel ∗ labelCameras
— QLabel ∗ labelCameraDeconnectee
— QStackedWidget ∗ widgetEmpilement
— QLabel ∗ labelIconeRadiation
— QLabel ∗ labelMesureRadiation
— QwtThermo ∗ barRadiation
— QLabel ∗ labelIconeTemperature
— QLabel ∗ labelMesureTemperature
— QwtThermo ∗ barTemperature
— QLabel ∗ labelIconeDistance
— QLabel ∗ labelMesureDistance
— QLabel ∗ labelIndicationDistance
— QComboBox ∗ listeCamerasDispo
— QPushButton ∗ boutonQuitter
— QPushButton ∗ boutonArchives
— QPushButton ∗ boutonCapture
— QMenuBar ∗ barreMenu
— QMenu ∗ menuFichier
— QMenu ∗ menuAide
— QMenu ∗ menuOutils
— QAction ∗ actionNouvelleCampagne
— QAction ∗ actionParametre
— QAction ∗ actionDebug
— QAction ∗ actionAide
— QDialog ∗ fenetreDebug
— QLabel ∗ labelDebug
— QPlainTextEdit ∗ plainTextEditDebug
— QPushButton ∗ boutonEnvoyees
— QPushButton ∗ boutonRecues
— QPushButton ∗ boutonSQL
— QPushButton ∗ boutonEtats
— QDialog ∗ fenetreParametres
— QLabel ∗ labelArchivageMesures
— QCheckBox ∗ checkboxArchivage
— QLabel ∗ labelIntervalArchivage
— QSlider ∗ sliderIntervalArchivage
— QSpinBox ∗ spinBoxIntervalArchivage
— QLabel ∗ labelAppareils
— QComboBox ∗ comboboxAppareils
— QLabel ∗ labelBaudRate
— QComboBox ∗ comboboxBaudRate
— QDialog ∗ fenetreNouvelleCampagne
— QDir ∗ dossierNouvelleCampagne
— QPushButton ∗ boutonValider
— QPushButton ∗ boutonAnnuler
— QLabel ∗ labelCampagne
— QLabel ∗ labelNomCampagne
— QLineEdit ∗ lineEditNomCampagne
— QLabel ∗ labelDescriptionCampagne
— QLineEdit ∗ lineEditDescriptionCampagne
— QLabel ∗ labelOperateur
— QComboBox ∗ comboBoxListeOperateurs
— QMessageBox ∗ messageBoxAide

8.9.1 Description détaillée

Auteur

Nicolas BOFFREDO & Jacques REYNIER

Version

1.1

Date

Jeudi 13 Juin 2019

8.9.2 Documentation des constructeurs et destructeur

8.9.2.1 IHMRov()

```
IHMRov::IHMRov (
            QWidget * parent = nullptr )
```

Références actualiserDistance(), actualiserIrradiation(), actualiserTemperature(), APPLICATION_TITRE, archives, barRadiation, barreMenu, barTemperature, boutonArchives, boutonCapture, boutonQuitter, camera, controleRov, creerFenetreDebug(), creerFenetreNouvelleCampagne(), creerFenetreParametres(), executerFenetreNouvelleCampagne(), executerFenetreParametres(), Camera : :getCadreFluxVideo(), Rov : :getMesures(), initialiserChronometre(), initialiserListeCamera(), initialiserMenu(), labelCamera, labelCameraDeconnectee, labelCameras, labelCampagneEnCours, labelChronometre, labelConnexionRov, labelEtatCamera, labelEtatConnexionRov, labelEtatManette, labelIconeDistance, labelIconeRadiation, labelIconeTemperature, labelIndicationDistance, labelManette, labelMesureDistance, labelMesureRadiation, labelMesureTemperature, listeCamerasDispo, quitter(), rov, Camera : :setArchives(), Rov : :setArchives(), Rov : :setCamera(), et widgetEmpilement.

```
00024     : QDialog(parent)
00025 {
00026     qDebug() << Q_FUNC_INFO;
00027
00028     this->rov = new Rov(this);
00029
00030     this->creerFenetreNouvelleCampagne();
00031     this->creerFenetreParametres();
00032     this->creerFenetreDebug();
00033
00034     camera = new Camera(this);
00035     controleRov = new ControleRov(this, rov);
00036     archives = new Archives(this);
00037
00038     // met en place les associations
00039     rov->setCamera(camera);
00040     rov->setArchives(archives);
00041     camera->setArchives(archives);
00042
00043     const int largeurMAX = qApp->desktop()->availableGeometry(this).width();
00044     const int hauteurMAX = qApp->desktop()->availableGeometry(this).height();
00045     const int largeurVideo = largeurMAX/4 + largeurMAX/2;
00046     const int hauteurVideo = hauteurMAX - hauteurMAX/8;
00047
00048     this->initialiserMenu();
00049
00050     connect(rov->getMesures(), SIGNAL(irradiationActualisee(double)), this, SLOT(
      actualiserIrradiation(double)));
00051     connect(rov->getMesures(), SIGNAL(temperatureActualisee(double)), this, SLOT(
      actualiserTemperature(double)));
00052     connect(rov->getMesures(), SIGNAL(distanceActualisee(double)), this, SLOT(
      actualiserDistance(double)));
00053
00054     // Créer l'IHM
00055     // fixer le titre de la fenêtre
00056     this->setWindowTitle(APPLICATION_TITRE);
00057
00058     // créer les widgets
00059     labelCampagneEnCours = new QLabel(QString::fromUtf8("Campagne : aucune"), this);
00060     labelConnexionRov = new QLabel(QString::fromUtf8("Rov"), this);
00061     labelEtatConnexionRov = new QLabel(this);
00062     labelCamera = new QLabel(QString::fromUtf8("Caméra"), this);
00063     labelEtatCamera = new QLabel(this);
```

```
00064     labelManette = new QLabel(QString::fromUtf8("Manette"), this);
00065     labelEtatManette = new QLabel(this);
00066     labelChronometre = new QLabel(this);
00067     labelCameras = new QLabel(QString::fromUtf8("Caméras disponibles : "), this);
00068     labelCameras->setAlignment(Qt::AlignRight);
00069     labelCameraDeconnectee = new QLabel(this);
00070     widgetEmpilement = new QStackedWidget(this);
00071
00072     QFont policeLabel;
00073     policeLabel.setPointSize(18);
00074     labelChronometre->setFont(policeLabel);
00075
00076     // créer les widgets du layout des mesures
00077     labelIconeRadiation = new QLabel("&Radiation (µSv/h)", this);
00078     labelMesureRadiation = new QLabel("", this);
00079     barRadiation = new QwtThermo();
00080
00081     labelIconeTemperature = new QLabel(QString::fromUtf8("&Température (°C)"), this);
00082     labelMesureTemperature = new QLabel("", this);
00083     barTemperature = new QwtThermo();
00084
00085     labelIconeDistance = new QLabel(QString::fromUtf8("&Distance (cm)"), this);
00086     labelMesureDistance = new QLabel("", this);
00087     labelIndicationDistance = new QLabel(this);
00088
00089     // créer les widgets du layout boutons
00090     listeCamerasDispo = new QComboBox();
00091     listeCamerasDispo->setLayoutDirection(Qt::RightToLeft);
00092     listeCamerasDispo->setEditable(true);
00093     listeCamerasDispo->lineEdit()->setReadOnly(true);
00094     listeCamerasDispo->lineEdit()->setAlignment(Qt::AlignRight);
00095     initialiserListeCamera();
00096
00097     boutonCapture = new QPushButton("&Capturer", this);
00098     boutonArchives = new QPushButton("&Archives", this);
00099     boutonQuitter = new QPushButton("&Quitter", this);
00100
00101     // créer les layout
00102     QHBoxLayout *hFLayoutPrincipal = new QHBoxLayout;
00103     QHBoxLayout *hLayoutPrincipal = new QHBoxLayout;
00104     QVBoxLayout *vLayoutVideo = new QVBoxLayout;
00105     QVBoxLayout *vLayoutBoutonsEtMesures = new QVBoxLayout;
00106
00107     // layout des états
00108     QVBoxLayout *vLayoutEtats = new QVBoxLayout;
00109     QHBoxLayout *hLayoutEtatCamera = new QHBoxLayout;
00110     QHBoxLayout *hLayoutEtatManette = new QHBoxLayout;
00111     QHBoxLayout *hLayoutEtatRov = new QHBoxLayout;
00112
00113     // layout boutons
00114     QVBoxLayout *vLayoutChoixCamera = new QVBoxLayout;
00115     QVBoxLayout *vLayoutBoutons = new QVBoxLayout;
00116
00117     // layout des mesures
00118     QVBoxLayout *vLayoutMesures = new QVBoxLayout;
00119     QHBoxLayout *hLayoutMesuresRadiation = new QHBoxLayout;
00120     QHBoxLayout *hLayoutMesuresTemperature = new QHBoxLayout;
00121     QHBoxLayout *hLayoutMesuresDistance = new QHBoxLayout;
00122     QVBoxLayout *vLayoutIndicationDistance = new QVBoxLayout;
00123
00124     // layout menu
00125     QHBoxLayout *hLayoutMenu = new QHBoxLayout;
00126     hLayoutMenu->setMenuBar(barreMenu);
00127     hFLayoutPrincipal->addLayout(hLayoutMenu);
00128
00129     // positionner les widgets dans les layouts
00130
00131     // le layout du flux vidéo
00132     vLayoutVideo->addWidget(labelCampagneEnCours);
00133     vLayoutVideo->addWidget(widgetEmpilement);
00134
00135     widgetEmpilement->setFixedWidth(largeurVideo);
00136     widgetEmpilement->setFixedHeight(hauteurVideo);
00137     widgetEmpilement->addWidget(labelCameraDeconnectee);
00138     widgetEmpilement->addWidget(camera->getCadreFluxVideo());
00139     labelCameraDeconnectee->setPixmap(QPixmap(":/logos/Images/camera_deconnectee.png"
      ));
00140
00141     vLayoutVideo->addWidget(labelChronometre);
00142
00143     // le layout principal
00144     hLayoutPrincipal->setContentsMargins(0, 15, 0, 0);
00145     hLayoutPrincipal->addLayout(vLayoutVideo);
00146     hLayoutPrincipal->addLayout(vLayoutBoutonsEtMesures);
00147
00148     // le layout des boutons et des mesures
00149     vLayoutBoutonsEtMesures->addLayout(vLayoutMesures);
00150     vLayoutBoutonsEtMesures->addLayout(vLayoutEtats);
00151     vLayoutBoutonsEtMesures->addLayout(vLayoutChoixCamera);
00152     vLayoutBoutonsEtMesures->addLayout(vLayoutBoutons);
00153
```

```
00154     // le layout des mesures
00155     vLayoutMesures->addLayout(hLayoutMesuresRadiation);
00156     vLayoutMesures->addLayout(hLayoutMesuresTemperature);
00157     vLayoutMesures->addLayout(hLayoutMesuresDistance);
00158
00159     // layout radiation
00160     hLayoutMesuresRadiation->addStretch();
00161     hLayoutMesuresRadiation->addWidget(labelIconeRadiation);
00162     labelIconeRadiation->setPixmap(QPixmap(":/logos/Images/logo_radioactive.png"));
00163     hLayoutMesuresRadiation->addWidget(labelMesureRadiation);
00164     labelMesureRadiation->setText("Radiation (µSv/h) : ");
00165     hLayoutMesuresRadiation->addWidget(barRadiation);
00166     barRadiation->setOrientation(Qt::Vertical);
00167     barRadiation->setScale(0, 4);
00168     barRadiation->setValue(0.0);
00169
00170     // layout temperature
00171     hLayoutMesuresTemperature->addStretch();
00172     hLayoutMesuresTemperature->addWidget(labelIconeTemperature);
00173     labelIconeTemperature->setPixmap(QPixmap(":/logos/Images/logo_temperature.png"));
00174     hLayoutMesuresTemperature->addWidget(labelMesureTemperature);
00175     labelMesureTemperature->setText("Temperature (°C) : ");
00176     hLayoutMesuresTemperature->addWidget(barTemperature);
00177     barTemperature->setOrientation(Qt::Vertical);
00178     barTemperature->setScale(-20,60);
00179     barTemperature->setValue(0.0);
00180
00181     // layout distance
00182     hLayoutMesuresDistance->addStretch();
00183     hLayoutMesuresDistance->addWidget(labelIconeDistance);
00184     labelIconeDistance->setPixmap(QPixmap(":/logos/Images/logo_distance.png"));
00185     hLayoutMesuresDistance->addWidget(labelMesureDistance);
00186     labelMesureDistance->setText("Distance (cm) : ");
00187     hLayoutMesuresDistance->addLayout(vLayoutIndicationDistance);
00188     vLayoutIndicationDistance->addWidget(labelIndicationDistance);
00189     labelIndicationDistance->setPixmap(QPixmap(":/logos/Images/distance_defaut.png")
    );
00190
00191     // le layout des états
00192     hLayoutEtatCamera->addStretch();
00193     hLayoutEtatCamera->addWidget(labelCamera);
00194     hLayoutEtatCamera->addWidget(labelEtatCamera);
00195     hLayoutEtatManette->addStretch();
00196     hLayoutEtatManette->addWidget(labelManette);
00197     hLayoutEtatManette->addWidget(labelEtatManette);
00198     hLayoutEtatRov->addStretch();
00199     hLayoutEtatRov->addWidget(labelConnexionRov);
00200     hLayoutEtatRov->addWidget(labelEtatConnexionRov);
00201     vLayoutEtats->addLayout(hLayoutEtatManette);
00202     vLayoutEtats->addLayout(hLayoutEtatCamera);
00203     vLayoutEtats->addLayout(hLayoutEtatRov);
00204
00205     // le layout du choix de la caméra
00206     vLayoutChoixCamera->setAlignment(Qt::AlignRight);
00207     vLayoutChoixCamera->addWidget(labelCameras);
00208     vLayoutChoixCamera->addWidget(listeCamerasDispo);
00209
00210     // le layout des boutons
00211     vLayoutBoutons->addStretch();
00212     vLayoutBoutons->addWidget(boutonCapture);
00213     vLayoutBoutons->addWidget(boutonArchives);
00214     vLayoutBoutons->addWidget(boutonQuitter);
00215
00216     hFLayoutPrincipal->addLayout(hLayoutPrincipal);
00217
00218     setLayout(hFLayoutPrincipal);
00219     //resize(largeurMAX, hauteurMAX-30);
00220     setFixedSize(largeurMAX, hauteurMAX-30);
00221     setWindowFlags(Qt::Dialog | Qt::WindowCloseButtonHint);
00222
00223     // connecter signals/slots
00224     connect(boutonQuitter, SIGNAL(clicked()), this, SLOT(quitter()));
00225     connect(boutonArchives, SIGNAL(clicked()), archives, SLOT(ouvrirFenetreArchives()
    ));
00226     connect(boutonCapture, SIGNAL(clicked()), camera, SLOT(capture()));
00227     connect(listeCamerasDispo, SIGNAL(currentIndexChanged(const QString)),
    camera, SLOT(changerCamera(QString)));
00228
00229     this->executerFenetreNouvelleCampagne();
00230     this->executerFenetreParametres();
00231     this->initialiserChronometre();
00232
00233     /*if(!message.isEmpty())
00234         QMessageBox::critical(nullptr, QString::fromUtf8(APPLICATION_TITRE), message);*/
00235 }
```

8.9.2.2 ~IHMRov()

```
IHMRov::~IHMRov ( )
```

Références BaseDeDonnees : :detruireInstance().

```
00241 {
00242     BaseDeDonnees::detruireInstance();
00243     qDebug() << Q_FUNC_INFO;
00244 }
```

8.9.3 Documentation des fonctions membres

8.9.3.1 actualiseIconesEtat()

```
void IHMRov::actualiseIconesEtat ( )  [private]
```

Références boutonCapture, camera, controleRov, Camera : :estCameraDisponible(), CommunicationRov : :estCommunicationRov↩ Disponible(), ControleRov : :estControleRovDisponible(), Rov : :getCommunicationRov(), labelEtatCamera, labelEtatConnexionRov, labelEtatManette, rov, WIDGET_CAMERA_DISPONIBLE, WIDGET_CAMERA_INDISPONIBLE, et widgetEmpilement.

Référencé par tic().

```
00252 {
00253     QString message;
00254     if(rov->getCommunicationRov()->
      estCommunicationRovDisponible())
00255     {
00256         labelEtatConnexionRov->setPixmap(QPixmap(":/logos/Images/on.png"));
00257     }
00258     else
00259     {
00260         message += QString::fromUtf8("Aucune communication avec le Rov !\n");
00261         labelEtatConnexionRov->setPixmap(QPixmap(":/logos/Images/off.png"));
00262     }
00263     if(camera->estCameraDisponible())
00264     {
00265         labelEtatCamera->setPixmap(QPixmap(":/logos/Images/on.png"));
00266         widgetEmpilement->setCurrentIndex(
      WIDGET_CAMERA_DISPONIBLE);
00267     }
00268     else
00269     {
00270         message += QString::fromUtf8("Aucune caméra détectée !\n");
00271         labelEtatCamera->setPixmap(QPixmap(":/logos/Images/off.png"));
00272         boutonCapture->setEnabled(false);
00273         widgetEmpilement->setCurrentIndex(
      WIDGET_CAMERA_INDISPONIBLE);
00274     }
00275     if(controleRov->estControleRovDisponible())
00276     {
00277         labelEtatManette->setPixmap(QPixmap(":/logos/Images/on.png"));
00278     }
00279     else
00280     {
00281         message += QString::fromUtf8("Aucune manette détectée !\n");
00282         labelEtatManette->setPixmap(QPixmap(":/logos/Images/off.png"));
00283     }
00284 }
```

8.9.3.2 actualiserDistance

```
void IHMRov::actualiserDistance (
             double distance )  [slot]
```

S'actualise à chaque reception de mesure

Paramètres

distance	double

Références actualiserIndicationDistance(), et labelMesureDistance.

Référencé par IHMRov().

```
00408 {
00409     if(distance >= 0)
00410     {
00411         labelMesureDistance->setText("Distance (cm) : " + QString::number(distance));
00412         actualiserIndicationDistance(distance);
00413     }
00414     else
00415     {
00416         labelMesureDistance->setText("Distance (cm) : < 5");
00417         actualiserIndicationDistance(distance);
00418     }
00419
00420 }
```

8.9.3.3 actualiserIndicationDistance()

```
void IHMRov::actualiserIndicationDistance (
             double distance )  [private]
```

S'actualise à chaque reception de mesure

Paramètres

distance	double

Références DISTANCE_LOIN, DISTANCE_PROCHE, et labelIndicationDistance.

Référencé par actualiserDistance().

```
00429 {
00430     if (distance > DISTANCE_LOIN)
00431         labelIndicationDistance->setPixmap(QPixmap(":/logos/Images/distance_loin.png
      "));
00432
00433     else if ((distance <= DISTANCE_LOIN) && (distance >=
      DISTANCE_PROCHE))
00434         labelIndicationDistance->setPixmap(QPixmap("
      :/logos/Images/distance_proche.png"));
00435
00436     else if (distance < DISTANCE_PROCHE)
00437         labelIndicationDistance->setPixmap(QPixmap("
      :/logos/Images/distance_danger.png"));
00438 }
```

8.9.3.4 actualiserIrradiation

```
void IHMRov::actualiserIrradiation (
             double irradiation )  [slot]
```

S'actualise à chaque reception de mesure

Paramètres

irradiation	double

Références barRadiation, et labelMesureRadiation.

Référencé par IHMRov().

```
00384 {
00385     labelMesureRadiation->setText("Radiation (µSv/h) : " + QString::number(irradiation)
      );
00386     barRadiation->setValue(int(irradiation));
00387 }
```

8.9.3.5 actualiserTemperature

```
void IHMRov::actualiserTemperature (
            double temperature )  [slot]
```

S'actualise à chaque reception de mesure

Paramètres

temperature	double

Références barTemperature, et labelMesureTemperature.

Référencé par IHMRov().

```
00396 {
00397     labelMesureTemperature->setText("Temperature (°C) : " + QString::number(
      temperature));
00398     barTemperature->setValue(int(temperature));
00399 }
```

8.9.3.6 creationCampagne

```
void IHMRov::creationCampagne ( )  [signal]
```

Référencé par creerFenetreNouvelleCampagne(), et enregistrerParametresCampagne().

8.9.3.7 creerFenetreDebug()

```
void IHMRov::creerFenetreDebug ( )  [private]
```

Références boutonAnnuler, boutonEnvoyees, boutonEtats, boutonRecues, boutonSQL, fenetreDebug, labelDebug, et plainTextEdit↩ Debug.

Référencé par IHMRov().

```
00497 {
00498     const int largeurFenetreDebug = qApp->desktop()->availableGeometry(this).width() / 3;
00499     const int hauteurFenetreDebug = qApp->desktop()->availableGeometry(this).height() / 4;
00500
00501     fenetreDebug = new QDialog();
00502     boutonEnvoyees = new QPushButton("Envoi");
00503     boutonRecues = new QPushButton("Reception");
00504     boutonSQL = new QPushButton("SQL");
00505     boutonEtats = new QPushButton("Status");
00506     boutonAnnuler = new QPushButton("&Retour");
00507
00508     labelDebug = new QLabel("En cours de développement.");
00509     plainTextEditDebug = new QPlainTextEdit("Test");
00510
00511     QVBoxLayout *vLayoutFenetreDebug = new QVBoxLayout;
00512     QHBoxLayout *hLayoutEntete = new QHBoxLayout;
00513     QVBoxLayout *vLayoutSelection = new QVBoxLayout;
00514     QHBoxLayout *hLayoutBoutonRetour = new QHBoxLayout;
00515
00516     fenetreDebug->setWindowTitle("Mode Debug");
00517     fenetreDebug->setFixedSize(largeurFenetreDebug, hauteurFenetreDebug);
00518
00519     fenetreDebug->setLayout(vLayoutFenetreDebug);
00520
00521     vLayoutFenetreDebug->addLayout(hLayoutEntete);
00522     //vLayoutFenetreDebug->addLayout(vLayoutSelection);
00523     vLayoutFenetreDebug->addLayout(hLayoutBoutonRetour);
00524
00525     hLayoutEntete->addWidget(labelDebug);
00526
00527     vLayoutSelection->addWidget(boutonEnvoyees);
00528     vLayoutSelection->addWidget(boutonRecues);
00529     vLayoutSelection->addWidget(boutonSQL);
00530     vLayoutSelection->addWidget(boutonEtats);
00531
00532     hLayoutBoutonRetour->addWidget(boutonAnnuler);
00533
00534     connect(boutonAnnuler, SIGNAL(clicked()), fenetreDebug, SLOT(reject()));
00535 }
```

8.9.3.8 creerFenetreNouvelleCampagne()

```
void IHMRov::creerFenetreNouvelleCampagne ( )  [private]
```

Références boutonAnnuler, boutonValider, comboBoxListeOperateurs, creationCampagne(), enregistrerParametresCampagne(), fenetreNouvelleCampagne, labelCampagne, labelDescriptionCampagne, labelNomCampagne, labelOperateur, lineEditDescription↩Campagne, lineEditNomCampagne, et remplirComboBoxListeOperateurs().

Référencé par IHMRov().

```
00576 {
00577     const int largeurFenetreNouvelleCampagne = qApp->desktop()->availableGeometry(this).width() / 3;
00578     const int hauteurFenetreNouvelleCampagne = qApp->desktop()->availableGeometry(this).height() / 4;
00579
00580     // Les widgets
00581     fenetreNouvelleCampagne = new QDialog(this);
00582     boutonValider = new QPushButton(QString::fromUtf8("&Créer"));
00583     boutonAnnuler = new QPushButton(QString::fromUtf8("&Annuler"));
00584
00585     // Campagne
00586     labelCampagne = new QLabel;
00587     labelNomCampagne = new QLabel;
00588     lineEditNomCampagne = new QLineEdit;
00589     labelDescriptionCampagne = new QLabel;
00590     lineEditDescriptionCampagne = new QLineEdit;
00591
00592     // Operateur
00593     labelOperateur = new QLabel;
00594     comboBoxListeOperateurs = new QComboBox;
00595
00596     // Les layouts
00597     QVBoxLayout *vLayoutFenetreNouvelleCampagne = new QVBoxLayout;
00598     QHBoxLayout *hLayoutFenetreCampagne = new QHBoxLayout;
00599     QVBoxLayout *vLayoutCampagne = new QVBoxLayout;
00600     QVBoxLayout *vLayoutOperateur = new QVBoxLayout;
00601     QHBoxLayout *hLayoutNomCampagne = new QHBoxLayout;
00602     QHBoxLayout *hLayoutDescriptionCampagne = new QHBoxLayout;
00603     QHBoxLayout *hLayoutBoutonsCampagne = new QHBoxLayout;
00604
```

```
00605     fenetreNouvelleCampagne->setWindowTitle("Démarrer une nouvelle campagne");
00606     fenetreNouvelleCampagne->setFixedSize(largeurFenetreNouvelleCampagne,
      hauteurFenetreNouvelleCampagne);
00607     fenetreNouvelleCampagne->setLayout(vLayoutFenetreNouvelleCampagne);
00608
00609     // Campagne
00610     labelCampagne->setText("Campagne");
00611     labelNomCampagne->setText("Nom : ");
00612     labelDescriptionCampagne->setText("Description : ");
00613     hLayoutNomCampagne->addWidget(labelNomCampagne);
00614     hLayoutNomCampagne->addWidget(lineEditNomCampagne);
00615     hLayoutDescriptionCampagne->addWidget(labelDescriptionCampagne);
00616     hLayoutDescriptionCampagne->addWidget(lineEditDescriptionCampagne);
00617     vLayoutCampagne->addWidget(labelCampagne);
00618     vLayoutCampagne->addLayout(hLayoutNomCampagne);
00619     vLayoutCampagne->addLayout(hLayoutDescriptionCampagne);
00620     vLayoutCampagne->addStretch();
00621
00622     // Operateur
00623     labelOperateur->setText(QString::fromUtf8("Opérateur"));
00624     vLayoutOperateur->addWidget(labelOperateur);
00625     vLayoutOperateur->addWidget(comboBoxListeOperateurs);
00626     vLayoutOperateur->addStretch();
00627     remplirComboBoxListeOperateurs();
00628
00629     hLayoutFenetreCampagne->addLayout(vLayoutCampagne);
00630     hLayoutFenetreCampagne->addSpacing(10);
00631     hLayoutFenetreCampagne->addLayout(vLayoutOperateur);
00632
00633     // Boutons
00634     hLayoutBoutonsCampagne->addWidget(boutonValider);
00635     hLayoutBoutonsCampagne->addWidget(boutonAnnuler);
00636     hLayoutBoutonsCampagne->addStretch();
00637
00638     vLayoutFenetreNouvelleCampagne->addLayout(hLayoutFenetreCampagne);
00639     vLayoutFenetreNouvelleCampagne->addStretch();
00640     vLayoutFenetreNouvelleCampagne->addLayout(hLayoutBoutonsCampagne);
00641
00642     // Les connexions
00643     connect(boutonValider, SIGNAL(clicked()), this, SLOT(
      enregistrerParametresCampagne()));
00644     connect(this, SIGNAL(creationCampagne()),
      fenetreNouvelleCampagne, SLOT(accept()));
00645     connect(boutonAnnuler, SIGNAL(clicked()),
      fenetreNouvelleCampagne, SLOT(reject()));
00646 }
```

8.9.3.9 creerFenetreParametres()

```
void IHMRov::creerFenetreParametres ( )  [private]
```

Références boutonAnnuler, boutonValider, CommunicationRov : :changeBaudRate(), CommunicationRov : :changePort↩ Communication(), checkboxArchivage, comboboxAppareils, comboboxBaudRate, enregistrerParametres(), fenetreParametres, CommunicationRov : :getBaudRate(), Rov : :getCommunicationRov(), Mesures : :getFrequenceArchivage(), Rov : :getMesures(), CommunicationRov : :getPort(), labelAppareils, labelArchivageMesures, labelBaudRate, labelIntervalArchivage, Mesures : :modifie↩ FrequenceArchivage(), nouveauBaudRate(), nouveauPortCom(), nouvelleFrequenceArchivage(), parametresSauvegardes(), rov, sliderIntervalArchivage, et spinBoxIntervalArchivage.

Référencé par IHMRov().

```
00693 {
00694     const int largeurFenetreParametres = qApp->desktop()->availableGeometry(this).width() / 3;
00695     const int hauteurFenetreParametres = qApp->desktop()->availableGeometry(this).height() / 4;
00696
00697     fenetreParametres = new QDialog();
00698     boutonValider = new QPushButton("&Valider");
00699     boutonAnnuler = new QPushButton("&Retour");
00700
00701     labelArchivageMesures = new QLabel("Archivage Mesures :");
00702     checkboxArchivage = new QCheckBox;
00703     checkboxArchivage->setChecked(true);
00704
00705     labelIntervalArchivage = new QLabel("Période d'archivage des mesures (sec) :");
00706     sliderIntervalArchivage = new QSlider(Qt::Horizontal);
00707     sliderIntervalArchivage->setRange(12, 120);
00708     sliderIntervalArchivage->setSliderPosition(rov->
      getMesures()->getFrequenceArchivage());
00709     spinBoxIntervalArchivage = new QSpinBox;
```

```
00710     spinBoxIntervalArchivage->setRange(12, 120);
00711     spinBoxIntervalArchivage->setValue(rov->
      getMesures()->getFrequenceArchivage());
00712
00713     labelAppareils = new QLabel("Port :");
00714     comboboxAppareils = new QComboBox;
00715     for(int i = 0; i < 4; i++)
00716         comboboxAppareils->addItem("/dev/ttyUSB" + QString::number(i));
00717
00718     comboboxAppareils->addItem("/dev/ttyACM0");
00719     comboboxAppareils->setCurrentText(rov->
      getCommunicationRov()->getPort());
00720
00721     labelBaudRate = new QLabel("Baudrate : ");
00722     comboboxBaudRate = new QComboBox;
00723     QStringList listeBaudRate = {"1200", "2400", "4800", "9600", "19200", "38400", "57600", "115200"};
00724     comboboxBaudRate->addItems(listeBaudRate);
00725     comboboxBaudRate->setCurrentText(rov->getCommunicationRov()->
      getBaudRate());
00726
00727     // LAYOUTS
00728     QVBoxLayout *vLayoutFenetreParametres = new QVBoxLayout;
00729     QHBoxLayout *hLayoutArchivageActif = new QHBoxLayout;
00730     QHBoxLayout *hLayoutIntervalArchivage = new QHBoxLayout;
00731     QHBoxLayout *hLayoutAppareils = new QHBoxLayout;
00732     QHBoxLayout *hLayoutBaudRate = new QHBoxLayout;
00733     QHBoxLayout *hLayoutValiderAnnuler = new QHBoxLayout;
00734
00735     fenetreParametres->setWindowTitle("Paramètres de la campagne");
00736     fenetreParametres->setFixedSize(largeurFenetreParametres, hauteurFenetreParametres);
00737
00738     fenetreParametres->setLayout(vLayoutFenetreParametres);
00739
00740     vLayoutFenetreParametres->addLayout(hLayoutArchivageActif);
00741     vLayoutFenetreParametres->addLayout(hLayoutIntervalArchivage);
00742     vLayoutFenetreParametres->addLayout(hLayoutAppareils);
00743     vLayoutFenetreParametres->addLayout(hLayoutBaudRate);
00744     vLayoutFenetreParametres->addLayout(hLayoutValiderAnnuler);
00745
00746     // Contenu
00747     hLayoutArchivageActif->addWidget(labelArchivageMesures);
00748     hLayoutArchivageActif->addWidget(checkboxArchivage);
00749
00750     hLayoutIntervalArchivage->addWidget(labelIntervalArchivage);
00751     hLayoutIntervalArchivage->addWidget(spinBoxIntervalArchivage);
00752     hLayoutIntervalArchivage->addWidget(sliderIntervalArchivage);
00753
00754     hLayoutAppareils->addWidget(labelAppareils);
00755     hLayoutAppareils->addWidget(comboboxAppareils);
00756
00757     hLayoutBaudRate->addWidget(labelBaudRate);
00758     hLayoutBaudRate->addWidget(comboboxBaudRate);
00759
00760     hLayoutValiderAnnuler->addWidget(boutonValider);
00761     hLayoutValiderAnnuler->addWidget(boutonAnnuler);
00762
00763     // CONNEXIONS
00764     connect(spinBoxIntervalArchivage, SIGNAL(valueChanged(int)),
      sliderIntervalArchivage, SLOT(setValue(int)));
00765     connect(sliderIntervalArchivage, SIGNAL(valueChanged(int)),
      spinBoxIntervalArchivage, SLOT(setValue(int)));
00766
00767     connect(checkboxArchivage, SIGNAL(toggled(bool)),
      sliderIntervalArchivage, SLOT(setEnabled(bool)));
00768     connect(checkboxArchivage, SIGNAL(toggled(bool)),
      spinBoxIntervalArchivage, SLOT(setEnabled(bool)));
00769
00770     connect(boutonValider, SIGNAL(clicked()), this, SLOT(
      enregistrerParametres()));
00771     connect(this, SIGNAL(parametresSauvegardes()),
      fenetreParametres, SLOT(accept()));
00772     connect(boutonAnnuler, SIGNAL(clicked()), fenetreParametres, SLOT(reject(
      )));
00773
00774     connect(this, &IHMRov::nouvelleFrequenceArchivage,
      rov->getMesures(), &Mesures::modifieFrequenceArchivage);
00775     connect(this, &IHMRov::nouveauPortCom, rov->
      getCommunicationRov(), &
      CommunicationRov::changePortCommunication);
00776     connect(this, &IHMRov::nouveauBaudRate, rov->
      getCommunicationRov(), &CommunicationRov::changeBaudRate
      );
00777 }
```

8.9.3.10 enregistrerParametres

```
void IHMRov::enregistrerParametres ( )  [slot]
```

Références APPLICATION_TITRE, checkboxArchivage, comboboxAppareils, comboboxBaudRate, CommunicationRov : :estCommunicationRovDisponible(), Rov : :getCommunicationRov(), nouveauBaudRate(), nouveauPortCom(), nouvelleFrequenceArchivage(), parametresSauvegardes(), rov, Rov : :setArchivageActif(), et sliderIntervalArchivage.

Référencé par creerFenetreParametres().

```
00798 {
00799     qDebug() << Q_FUNC_INFO;
00800
00801     emit nouveauPortCom(comboboxAppareils->currentText());
00802
00803     rov->setArchivageActif(checkboxArchivage->isChecked());
00804
00805     emit nouvelleFrequenceArchivage(
      sliderIntervalArchivage->value());
00806
00807     emit nouveauBaudRate(comboboxBaudRate->currentText());
00808
00809     if(!rov->getCommunicationRov()->
      estCommunicationRovDisponible())
00810         QMessageBox::critical(nullptr, QString::fromUtf8(APPLICATION_TITRE),
      QString::fromUtf8("Impossible de se connecter au port choisi !" ));
00811     else
00812     {
00813         emit parametresSauvegardes();
00814     }
00815 }
```

8.9.3.11 enregistrerParametresCampagne

```
void IHMRov::enregistrerParametresCampagne ( )  [slot]
```

Références APPLICATION_TITRE, comboBoxListeOperateurs, creationCampagne(), Rov : :creerNouvelleCampagne(), idOperateur, labelCampagneEnCours, lineEditDescriptionCampagne, lineEditNomCampagne, et rov.

Référencé par creerFenetreNouvelleCampagne().

```
00342 {
00343     qDebug() << Q_FUNC_INFO;
00344     QString idOperateur = QString::number(comboBoxListeOperateurs->
      currentIndex() + 1);
00345
00346     if(lineEditNomCampagne->text().isEmpty())
00347     {
00348         QMessageBox::critical(nullptr, QString::fromUtf8(APPLICATION_TITRE),
      QString::fromUtf8("Il faut donner un nom à la campagne !"));
00349         return;
00350     }
00351     else if(rov->creerNouvelleCampagne(
      lineEditNomCampagne->text(), lineEditDescriptionCampagne->
      text(), idOperateur))
00352     {
00353         labelCampagneEnCours->setText("Campagne : " +
      lineEditNomCampagne->text());
00354         emit creationCampagne();
00355     }
00356     else
00357     {
00358         QMessageBox::critical(nullptr, QString::fromUtf8(APPLICATION_TITRE),
      QString::fromUtf8("Impossible de créer la campagne !"));
00359         qDebug() << Q_FUNC_INFO << lineEditNomCampagne->text() <<
      lineEditDescriptionCampagne->text() << idOperateur;
00360         return;
00361     }
00362 }
```

8.9.3.12 executerFenetreDebug

```
void IHMRov::executerFenetreDebug ( )  [slot]
```

Références fenetreDebug.

Référencé par initialiserMenu().

```
00483 {
00484     qDebug() << Q_FUNC_INFO;
00485     int retour = fenetreDebug->exec();
00486     qDebug() << Q_FUNC_INFO << retour;
00487     if(retour == QDialog::Rejected)
00488     {
00489     }
00490 }
```

8.9.3.13 executerFenetreNouvelleCampagne

```
void IHMRov::executerFenetreNouvelleCampagne ( )  [slot]
```

Connecté à l'action "Nouvelle Campagne" dans le menu "Campagne"

Références fenetreNouvelleCampagne.

Référencé par IHMRov(), et initialiserMenu().

```
00328 {
00329     qDebug() << Q_FUNC_INFO;
00330     int retour = fenetreNouvelleCampagne->exec();
00331     qDebug() << Q_FUNC_INFO << retour;
00332     if(retour == QDialog::Rejected)
00333     {
00334     }
00335 }
```

8.9.3.14 executerFenetreParametres

```
void IHMRov::executerFenetreParametres ( )  [slot]
```

Références fenetreParametres.

Référencé par IHMRov(), et initialiserMenu().

```
00784 {
00785     qDebug() << Q_FUNC_INFO;
00786     int retour = fenetreParametres->exec();
00787     qDebug() << Q_FUNC_INFO << retour;
00788     if(retour == QDialog::Rejected)
00789     {
00790     }
00791 }
```

8.9.3.15 getHeure()

```
long IHMRov::getHeure ( )  [private]
```

Références valeurChronometre.

Référencé par getTemps().

```
00684 {
00685     return valeurChronometre/3600;
00686 }
```

8.9.3.16 getIdOperateur()

```
int IHMRov::getIdOperateur ( )  [private]
```

Références idOperateur.

```
00567 {
00568     return idOperateur;
00569 }
```

8.9.3.17 getMinutes()

```
long IHMRov::getMinutes ( )  [private]
```

Références valeurChronometre.

Référencé par getTemps().

```
00675 {
00676     return (valeurChronometre%3600)/60;
00677 }
```

8.9.3.18 getSecondes()

```
long IHMRov::getSecondes ( )  [private]
```

Références valeurChronometre.

Référencé par getTemps().

```
00666 {
00667     return valeurChronometre%60;
00668 }
```

8.9.3.19 getTemps()

```
QString IHMRov::getTemps ( )  [private]
```

Renvoie

QString le temps formaté "hh :mm :ss"

Références getHeure(), getMinutes(), et getSecondes().

Référencé par tic().

```
00823 {
00824     QString heure, minutes, secondes;
00825
00826    if (getHeure() < 10)
00827       heure = "0" + QString::number(getHeure());
00828    else heure = QString::number(getHeure());
00829
00830    if (getMinutes() < 10)
00831       minutes = "0" + QString::number(getMinutes());
00832    else minutes = QString::number(getMinutes());
00833
00834    if (getSecondes() < 10)
00835       secondes = "0" + QString::number(getSecondes());
00836    else secondes = QString::number(getSecondes());
00837
00838    QString temps = heure + ":" + minutes + ":" + secondes;
00839    return temps;
00840 }
```

8.9.3.20 initialiserChronometre()

```
void IHMRov::initialiserChronometre ( )  [private]
```

Références chronometre, PERIODE, tic(), et valeurChronometre.

Référencé par IHMRov().

```
00653 {
00654     this->valeurChronometre = 0;
00655     QTimer *chronometre = new QTimer(this);
00656     connect(chronometre, SIGNAL(timeout()), this, SLOT(tic()));
00657
00658     chronometre->start(PERIODE);
00659 }
```

8.9.3.21 initialiserListeCamera

```
void IHMRov::initialiserListeCamera ( )  [slot]
```

Références camera, Camera : :getListeCamera(), et listeCamerasDispo.

Référencé par IHMRov().

```
00312 {
00313     QList<QCameraInfo> listeCamera = camera->getListeCamera();
00314     for(int i = 0; i < listeCamera.count(); i++)
00315     {
00316         listeCamerasDispo->addItem(listeCamera[i].description() + " (" + listeCamera[i].
      deviceName() + ")");
00317         listeCamerasDispo->setSizeAdjustPolicy(QComboBox::AdjustToContents);
00318         listeCamerasDispo->setFixedWidth(listeCamerasDispo->sizeHint().
      width()+10);
00319     }
00320 }
```

8.9.3.22 initialiserMenu()

```
void IHMRov::initialiserMenu ( )  [private]
```

Références actionAide, actionDebug, actionNouvelleCampagne, actionParametre, barreMenu, executerFenetreDebug(), executer↩FenetreNouvelleCampagne(), executerFenetreParametres(), menuAide, menuFichier, menuOutils, et ouvrirAide().

Référencé par IHMRov().

```
00445 {
00446     barreMenu = new QMenuBar(this);
00447     menuFichier = new QMenu(QString::fromUtf8("Fichier"), barreMenu);
00448     menuOutils = new QMenu(QString::fromUtf8("Outils"), barreMenu);
00449     menuAide = new QMenu(QString::fromUtf8("À propos"), barreMenu);
00450 
00451     actionNouvelleCampagne = new QAction(QString::fromUtf8("Nouvelle campagne"), this
      );
00452     actionParametre = new QAction(QString::fromUtf8("Parametres"), this);
00453     actionDebug = new QAction(QString::fromUtf8("Debug"), this);
00454     actionAide = new QAction(QString::fromUtf8("À propos"), this);
00455 
00456     actionNouvelleCampagne->setShortcuts(QKeySequence::New);
00457     actionNouvelleCampagne->setStatusTip(QString::fromUtf8("Créer une nouvelle
       campagne"));
00458     connect(actionNouvelleCampagne, SIGNAL(triggered()), this, SLOT(
      executerFenetreNouvelleCampagne()));
00459 
00460     actionParametre->setShortcut(tr("Ctrl+p"));
00461     connect(actionParametre, SIGNAL(triggered()), this, SLOT(
      executerFenetreParametres()));
00462 
00463     actionDebug->setShortcut(tr("Ctrl+d"));
00464     connect(actionDebug, SIGNAL(triggered()), this, SLOT(
      executerFenetreDebug()));
00465 
00466     actionAide->setShortcut(QKeySequence::HelpContents);
00467     connect(actionAide, SIGNAL(triggered()), this, SLOT(ouvrirAide()));
00468 
00469     barreMenu->addMenu(menuFichier);
00470     barreMenu->addMenu(menuOutils);
00471     barreMenu->addMenu(menuAide);
00472     menuFichier->addAction(actionNouvelleCampagne);
00473     menuFichier->addAction(actionParametre);
00474     menuOutils->addAction(actionDebug);
00475     menuAide->addAction(actionAide);
00476 }
```

8.9.3.23 nouveauBaudRate

```
void IHMRov::nouveauBaudRate (
            QString  )  [signal]
```

Référencé par creerFenetreParametres(), et enregistrerParametres().

8.9.3.24 nouveauPortCom

```
void IHMRov::nouveauPortCom (
            QString  )  [signal]
```

Référencé par creerFenetreParametres(), et enregistrerParametres().

8.9.3.25 nouvelleFrequenceArchivage

```
void IHMRov::nouvelleFrequenceArchivage (
            int  )  [signal]
```

Référencé par creerFenetreParametres(), et enregistrerParametres().

8.9.3.26 ouvrirAide

```
void IHMRov::ouvrirAide ( )  [slot]
```

Références APPLICATION_INFORMATIONS, APPLICATION_TITRE, et messageBoxAide.

Référencé par initialiserMenu().

```
00369 {
00370     messageBoxAide = new QMessageBox(this);
00371     messageBoxAide->setWindowTitle("Aide");
00372     messageBoxAide->setText(APPLICATION_TITRE);
00373     messageBoxAide->setInformativeText(APPLICATION_INFORMATIONS);
00374     messageBoxAide->exec();
00375 }
```

8.9.3.27 parametresSauvegardes

```
void IHMRov::parametresSauvegardes ( )  [signal]
```

Référencé par creerFenetreParametres(), et enregistrerParametres().

8.9.3.28 quitter

```
void IHMRov::quitter ( )  [slot]
```

Référencé par IHMRov().

```
00291 {
00292     close();
00293 }
```

8.9.3.29 remplirComboBoxListeOperateurs()

```
void IHMRov::remplirComboBoxListeOperateurs ( )  [private]
```

Références comboBoxListeOperateurs, Rov : :getListeNomsOperateurs(), Rov : :getListePrenomsOperateurs(), et rov.

Référencé par creerFenetreNouvelleCampagne().

```
00542 {
00543     qDebug() << Q_FUNC_INFO;
00544     QString operateur;
00545
00546     for(int i=0; i < rov->getListeNomsOperateurs().size(); i++)
00547     {
00548         operateur = rov->getListeNomsOperateurs()[i] + " " +
      rov->getListePrenomsOperateurs()[i];
00549         comboBoxListeOperateurs->addItem(operateur);
00550     }
00551 }
```

8.9.3.30 setIdOperateur()

```
void IHMRov::setIdOperateur (
            int idOperateur )  [private]
```

Références idOperateur.

```
00558 {
00559     this->idOperateur = idOperateur;
00560 }
```

8.9.3.31 tic

```
void IHMRov::tic ( )  [slot]
```

Références actualiseIconesEtat(), getTemps(), labelChronometre, et valeurChronometre.

Référencé par initialiserChronometre().

```
00300 {
00301     valeurChronometre++;
00302     labelChronometre->setText(this->getTemps());
00303
00304     actualiseIconesEtat();
00305 }
```

8.9.4 Documentation des données membres

8.9.4.1 actionAide

```
QAction* IHMRov::actionAide  [private]
```

Référencé par initialiserMenu().

8.9.4.2 actionDebug

```
QAction* IHMRov::actionDebug  [private]
```

Référencé par initialiserMenu().

8.9.4.3 actionNouvelleCampagne

```
QAction* IHMRov::actionNouvelleCampagne  [private]
```

Référencé par initialiserMenu().

8.9.4.4 actionParametre

```
QAction* IHMRov::actionParametre  [private]
```

Référencé par initialiserMenu().

8.9.4.5 archives

```
Archives* IHMRov::archives  [private]
```

Référencé par IHMRov().

8.9.4.6 barRadiation

```
QwtThermo* IHMRov::barRadiation  [private]
```

Référencé par actualiserIrradiation(), et IHMRov().

8.9.4.7 barreMenu

```
QMenuBar* IHMRov::barreMenu  [private]
```

Référencé par IHMRov(), et initialiserMenu().

8.9.4.8 barTemperature

```
QwtThermo* IHMRov::barTemperature  [private]
```

Référencé par actualiserTemperature(), et IHMRov().

8.9.4.9 boutonAnnuler

```
QPushButton* IHMRov::boutonAnnuler  [private]
```

Référencé par creerFenetreDebug(), creerFenetreNouvelleCampagne(), et creerFenetreParametres().

8.9.4.10 boutonArchives

```
QPushButton* IHMRov::boutonArchives  [private]
```

Référencé par IHMRov().

8.9.4.11 boutonCapture

```
QPushButton* IHMRov::boutonCapture  [private]
```

Référencé par actualiseIconesEtat(), et IHMRov().

8.9.4.12 boutonEnvoyees

```
QPushButton* IHMRov::boutonEnvoyees  [private]
```

Référencé par creerFenetreDebug().

8.9.4.13 boutonEtats

```
QPushButton* IHMRov::boutonEtats  [private]
```

Référencé par creerFenetreDebug().

8.9.4.14 boutonQuitter

```
QPushButton* IHMRov::boutonQuitter  [private]
```

Référencé par IHMRov().

8.9.4.15 boutonRecues

```
QPushButton* IHMRov::boutonRecues  [private]
```

Référencé par creerFenetreDebug().

8.9.4.16 boutonSQL

```
QPushButton* IHMRov::boutonSQL  [private]
```

Référencé par creerFenetreDebug().

8.9.4.17 boutonValider

```
QPushButton* IHMRov::boutonValider  [private]
```

Référencé par creerFenetreNouvelleCampagne(), et creerFenetreParametres().

8.9.4.18 camera

```
Camera* IHMRov::camera  [private]
```

Référencé par actualiseIconesEtat(), IHMRov(), et initialiserListeCamera().

8.9.4.19 checkboxArchivage

```
QCheckBox* IHMRov::checkboxArchivage  [private]
```

Référencé par creerFenetreParametres(), et enregistrerParametres().

8.9.4.20 chronometre

```
QTimer* IHMRov::chronometre  [private]
```

Référencé par initialiserChronometre().

8.9.4.21 comboboxAppareils

```
QComboBox* IHMRov::comboboxAppareils  [private]
```

Référencé par creerFenetreParametres(), et enregistrerParametres().

8.9.4.22 comboboxBaudRate

```
QComboBox* IHMRov::comboboxBaudRate  [private]
```

Référencé par creerFenetreParametres(), et enregistrerParametres().

8.9.4.23 comboBoxListeOperateurs

```
QComboBox* IHMRov::comboBoxListeOperateurs  [private]
```

Référencé par creerFenetreNouvelleCampagne(), enregistrerParametresCampagne(), et remplirComboBoxListeOperateurs().

8.9.4.24 controleRov

```
ControleRov* IHMRov::controleRov  [private]
```

Référencé par actualiseIconesEtat(), et IHMRov().

8.9.4.25 dossierNouvelleCampagne

```
QDir* IHMRov::dossierNouvelleCampagne  [private]
```

8.9.4.26 fenetreDebug

```
QDialog* IHMRov::fenetreDebug  [private]
```

Référencé par creerFenetreDebug(), et executerFenetreDebug().

8.9.4.27 fenetreNouvelleCampagne

```
QDialog* IHMRov::fenetreNouvelleCampagne  [private]
```

Référencé par creerFenetreNouvelleCampagne(), et executerFenetreNouvelleCampagne().

8.9.4.28 fenetreParametres

```
QDialog* IHMRov::fenetreParametres  [private]
```

Référencé par creerFenetreParametres(), et executerFenetreParametres().

8.9.4.29 idOperateur

```
int IHMRov::idOperateur  [private]
```

Référencé par enregistrerParametresCampagne(), getIdOperateur(), et setIdOperateur().

8.9.4.30 labelAppareils

```
QLabel* IHMRov::labelAppareils  [private]
```

Référencé par creerFenetreParametres().

8.9.4.31 labelArchivageMesures

```
QLabel* IHMRov::labelArchivageMesures  [private]
```

Référencé par creerFenetreParametres().

8.9.4.32 labelBaudRate

```
QLabel* IHMRov::labelBaudRate  [private]
```

Référencé par creerFenetreParametres().

8.9.4.33 labelCamera

```
QLabel* IHMRov::labelCamera  [private]
```

Référencé par IHMRov().

8.9.4.34 labelCameraDeconnectee

```
QLabel* IHMRov::labelCameraDeconnectee  [private]
```

Référencé par IHMRov().

8.9.4.35 labelCameras

```
QLabel* IHMRov::labelCameras  [private]
```

Référencé par IHMRov().

8.9.4.36 labelCampagne

```
QLabel* IHMRov::labelCampagne  [private]
```

Référencé par creerFenetreNouvelleCampagne().

8.9.4.37 labelCampagneEnCours

```
QLabel* IHMRov::labelCampagneEnCours  [private]
```

Référencé par enregistrerParametresCampagne(), et IHMRov().

8.9.4.38 labelChronometre

```
QLabel* IHMRov::labelChronometre  [private]
```

Référencé par IHMRov(), et tic().

8.9.4.39 labelConnexionRov

```
QLabel* IHMRov::labelConnexionRov  [private]
```

Référencé par IHMRov().

8.9.4.40 labelDebug

```
QLabel* IHMRov::labelDebug  [private]
```

Référencé par creerFenetreDebug().

8.9.4.41 labelDescriptionCampagne

```
QLabel* IHMRov::labelDescriptionCampagne  [private]
```

Référencé par creerFenetreNouvelleCampagne().

8.9.4.42 labelEtatCamera

```
QLabel* IHMRov::labelEtatCamera  [private]
```

Référencé par actualiseIconesEtat(), et IHMRov().

8.9.4.43 labelEtatConnexionRov

```
QLabel* IHMRov::labelEtatConnexionRov  [private]
```

Référencé par actualiseIconesEtat(), et IHMRov().

8.9.4.44 labelEtatManette

```
QLabel* IHMRov::labelEtatManette  [private]
```

Référencé par actualiseIconesEtat(), et IHMRov().

8.9.4.45 labelIconeDistance

```
QLabel* IHMRov::labelIconeDistance  [private]
```

Référencé par IHMRov().

8.9.4.46 labelIconeRadiation

```
QLabel* IHMRov::labelIconeRadiation  [private]
```

Référencé par IHMRov().

8.9.4.47 labelIconeTemperature

```
QLabel* IHMRov::labelIconeTemperature  [private]
```

Référencé par IHMRov().

8.9.4.48 labelIndicationDistance

```
QLabel* IHMRov::labelIndicationDistance  [private]
```

Référencé par actualiserIndicationDistance(), et IHMRov().

8.9.4.49 labelIntervalArchivage

```
QLabel* IHMRov::labelIntervalArchivage  [private]
```

Référencé par creerFenetreParametres().

8.9.4.50 labelManette

```
QLabel* IHMRov::labelManette  [private]
```

Référencé par IHMRov().

8.9.4.51 labelMesureDistance

```
QLabel* IHMRov::labelMesureDistance  [private]
```

Référencé par actualiserDistance(), et IHMRov().

8.9.4.52 labelMesureRadiation

```
QLabel* IHMRov::labelMesureRadiation  [private]
```

Référencé par actualiserIrradiation(), et IHMRov().

8.9.4.53 labelMesureTemperature

```
QLabel* IHMRov::labelMesureTemperature  [private]
```

Référencé par actualiserTemperature(), et IHMRov().

8.9.4.54 labelNomCampagne

```
QLabel* IHMRov::labelNomCampagne  [private]
```

Référencé par creerFenetreNouvelleCampagne().

8.9.4.55 labelOperateur

```
QLabel* IHMRov::labelOperateur  [private]
```

Référencé par creerFenetreNouvelleCampagne().

8.9.4.56 lineEditDescriptionCampagne

```
QLineEdit* IHMRov::lineEditDescriptionCampagne  [private]
```

Référencé par creerFenetreNouvelleCampagne(), et enregistrerParametresCampagne().

8.9.4.57 lineEditNomCampagne

```
QLineEdit* IHMRov::lineEditNomCampagne  [private]
```

Référencé par creerFenetreNouvelleCampagne(), et enregistrerParametresCampagne().

8.9.4.58 listeCamerasDispo

```
QComboBox* IHMRov::listeCamerasDispo  [private]
```

Référencé par IHMRov(), et initialiserListeCamera().

8.9.4.59 menuAide

```
QMenu* IHMRov::menuAide  [private]
```

Référencé par initialiserMenu().

8.9.4.60 menuFichier

```
QMenu* IHMRov::menuFichier  [private]
```

Référencé par initialiserMenu().

8.9.4.61 menuOutils

```
QMenu* IHMRov::menuOutils  [private]
```

Référencé par initialiserMenu().

8.9.4.62 messageBoxAide

```
QMessageBox* IHMRov::messageBoxAide  [private]
```

Référencé par ouvrirAide().

8.9.4.63 nomNouvelleCampagne

```
QString IHMRov::nomNouvelleCampagne  [private]
```

8.9.4.64 plainTextEditDebug

```
QPlainTextEdit* IHMRov::plainTextEditDebug  [private]
```

Référencé par creerFenetreDebug().

8.9.4.65 rov

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

Référencé par actualiseIconesEtat(), creerFenetreParametres(), enregistrerParametres(), enregistrerParametresCampagne(), IHM↩Rov(), et remplirComboBoxListeOperateurs().

8.9.4.66 sliderIntervalArchivage

```
QSlider* IHMRov::sliderIntervalArchivage  [private]
```

Référencé par creerFenetreParametres(), et enregistrerParametres().

8.9.4.67 spinBoxIntervalArchivage

```
QSpinBox* IHMRov::spinBoxIntervalArchivage  [private]
```

Référencé par creerFenetreParametres().

8.9.4.68 valeurChronometre

```
long IHMRov::valeurChronometre  [private]
```

Référencé par getHeure(), getMinutes(), getSecondes(), initialiserChronometre(), et tic().

8.9.4.69 widgetEmpilement

```
QStackedWidget* IHMRov::widgetEmpilement  [private]
```

Référencé par actualiseIconesEtat(), et IHMRov().

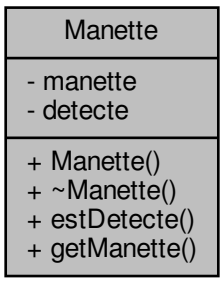
La documentation de cette classe a été générée à partir des fichiers suivants :
- ihmrov.h
- ihmrov.cpp

8.10 Référence de la classe Manette

Déclaration de la classe Manette. Liaison avec la manette, permettant d'en recevoir les signaux.

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

Graphe de collaboration de Manette :



Fonctions membres publiques

- Manette (QObject *parent=nullptr)
  Constructeur de la classe GestionnaireManette. La connexion entre les évènements de la manette et les méthodes cibles sont réalisées ici.
- ~Manette ()
- bool estDetecte () const
  estDetecte
- QGamepad * getManette ()
  getManette

Attributs privés

- QGamepad * manette
  Contient la manette actuellement connectée.
- bool detecte
  Manette connectée, ou non.

8.10.1 Description détaillée

Auteur

REYNIER Jacques & Nicolas BOFFREDO

Version

1.1

Date

Jeudi 13 Juin 2019

8.10.2 Documentation des constructeurs et destructeur

8.10.2.1 Manette()

```
Manette::Manette (
            QObject * parent = nullptr )  [explicit]
```

Paramètres

parent	

Références detecte, et manette.

```
00025                                : QObject(parent), manette(nullptr),
      detecte(false)
00026 {
00027     #ifndef QT_NO_DEBUG_OUTPUT
00028     QLoggingCategory::setFilterRules(QStringLiteral("qt.gamepad.debug=true"));
00029     #endif
00030
00031     auto manettes = QGamepadManager::instance()->connectedGamepads();
00032
00033     if (manettes.isEmpty())
00034     {
00035         qDebug() << Q_FUNC_INFO << "Aucune manette détectée !";
00036         detecte = false;
00037     }
00038     else
00039     {
00040         manette = new QGamepad(*manettes.begin(), this);
00041         detecte = true;
00042     }
00043 }
```

8.10.2.2 ~Manette()

```
Manette::~Manette ( )
```

```
00046 {
00047
00048 }
```

8.10.3 Documentation des fonctions membres

8.10.3.1 estDetecte()

```
bool Manette::estDetecte ( ) const
```

Renvoie l'état de la manette (connectée, ou non).

Renvoie

detecte bool Etat de l'attribut detecte de l'objet.
detecte bool Indique si une manette est connectée ou non.

Références detecte.

Référencé par ControleRov : :ControleRov(), et ControleRov : :estControleRovDisponible().

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

8.10.3.2 getManette()

```
QGamepad * Manette::getManette ( )
```

Renvoie la manette actuellement connectée.

Renvoie

manette QGamepad Manette actuellement connectée.
manette QGamepad∗ Manette connectée.

Références manette.

Référencé par ControleRov : :changeConnexions(), et ControleRov : :ControleRov().

```
00066 {
00067     return manette;
00068 }
```

8.10.4 Documentation des données membres

8.10.4.1 detecte

```
bool Manette::detecte  [private]
```

Référencé par estDetecte(), et Manette().

8.10.4.2 manette

```
QGamepad* Manette::manette  [private]
```

Référencé par getManette(), et Manette().

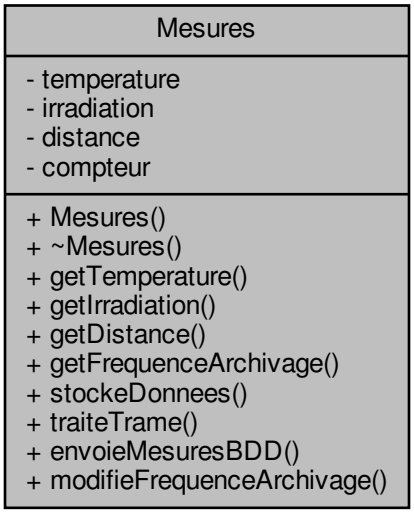
La documentation de cette classe a été générée à partir des fichiers suivants :
- manette.h
- manette.cpp

8.11 Référence de la classe Mesures

Déclaration de la classe Mesures. Gestion des mesures des capteurs (température, irradiation, et distance).

```
#include <mesures.h>
```

Graphe de collaboration de Mesures :



Connecteurs publics

- void traiteTrame (QString trame)
  Vérifie la validité, et découpe la trame reçue.
- void envoieMesuresBDD ()
  Envoie un signal toutes les 30 secondes contenant la température et l'irradiation.
- void modifieFrequenceArchivage (int)
  Modifie l'interval du timer correspondant à l'archivage des mesures dans la BDD.

Signaux

- void irradiationActualisee (double)
  Signal émis lorsqu'une nouvelle valeur d'irradiation est reçue.
- void temperatureActualisee (double)
  Signal émis lorsqu'une nouvelle valeur de température est reçue.
- void distanceActualisee (double)
  Signal émis lorsqu'une nouvelle valeur de distance est reçue.
- void mesuresBDDPrete (double temperature, double irradiation)
  Signal émis toutes les x secondes visant à stocker les valeurs de température et d'irradiation (en argument) dans la BDD.

Fonctions membres publiques

- Mesures (QObject ∗parent=nullptr)
  Constructeur de la classe Mesures.
- ∼Mesures ()
  Destructeur de la classe Mesures.
- double getTemperature () const
  Retourne la température stockée dans l'objet.
- double getIrradiation () const
  Retourne le taux d'irradiation stocké dans l'objet.
- double getDistance () const
  Renvoie le dernier relevé du capteur de proximité.
- int getFrequenceArchivage () const
  Renvoie la fréquence d'archivage des données.
- void stockeDonnees (QString type, QString donnee)
  Stocke la donnee passée en argument.

Attributs privés

- double temperature
  Dernière température relevée.
- double irradiation
  Dernier taux d'irradiation relevé.
- double distance
  Dernière mesure du capteur de proximité relevé.
- QTimer ∗ compteur
  Compteur envoyant toutes les x secondes un signal timeout.

8.11.1 Description détaillée

Auteur

REYNIER Jacques & Nicolas BOFFREDO

Version

1.1

Date

Jeudi 13 Juin 2019

8.11.2 Documentation des constructeurs et destructeur

8.11.2.1 Mesures()

```
Mesures::Mesures (
            QObject * parent = nullptr )  [explicit]
```

Crée un compteur émettant un signal toutes les 30 secondes (modifiable par l'utilisateur), permettant l'envoi des mesures dans la BDD.

Paramètres

parent	QObject*

Références compteur, et envoieMesuresBDD().

```
00023                                : QObject(parent), temperature(-999.0),
      irradiation(-999.0), distance(-999.0)
00024 {
00025     compteur = new QTimer(this);
00026     connect(compteur, SIGNAL(timeout()), this, SLOT(envoieMesuresBDD()));
00027     compteur->start(30000);
00028 }
```

8.11.2.2 ~Mesures()

```
Mesures::~Mesures ( )
```

```
00035 {
00036
00037 }
```

8.11.3 Documentation des fonctions membres

8.11.3.1 distanceActualisee

```
void Mesures::distanceActualisee (
             double  )  [signal]
```

Référencé par stockeDonnees().

8.11.3.2 envoieMesuresBDD

```
void Mesures::envoieMesuresBDD ( )  [slot]
```

Envoie un signal comprenant les mesures de température et d'irradiation à destination de la BDD, émis toutes les x secondes (fréquence d'archivage).

Références irradiation, mesuresBDDPrete(), et temperature.

Référencé par Mesures().

```
00150 {
00151     emit mesuresBDDPrete(this->temperature, this->
      irradiation);
00152 }
```

8.11.3.3 getDistance()

```
Mesures::getDistance ( ) const
```

Renvoie

distance double Dernier relevé du capteur de proximité.

Références distance.

```
00065 {
00066     return this->distance;
00067 }
```

8.11.3.4 getFrequenceArchivage()

```
int Mesures::getFrequenceArchivage ( ) const
```

Renvoie la fréquence d'archivage des mesures dans la BDD en secondes.

Renvoie

frequence int Fréquence d'archivage en secondes.

Références compteur.

Référencé par IHMRov : :creerFenetreParametres().

```
00075 {
00076     return (compteur->interval() / 1000);
00077 }
```

8.11.3.5 getIrradiation()

```
Mesures::getIrradiation ( ) const
```

Renvoie le dernier taux d'irradiation reçu.

Renvoie

irradiation int Dernier taux d'irradiation relevé.

Références irradiation.

```
00055 {
00056     return this->irradiation;
00057 }
```

8.11.3.6 getTemperature()

```
Mesures::getTemperature ( ) const
```

Renvoie la dernière température reçue.

Renvoie

temperature int Dernière température relevée.

Références temperature.

```
00045 {
00046     return this->temperature;
00047 }
```

8.11.3.7 irradiationActualisee

```
void Mesures::irradiationActualisee (
            double  )  [signal]
```

Référencé par stockeDonnees().

8.11.3.8 mesuresBDDPrete

```
void Mesures::mesuresBDDPrete (
            double temperature,
            double irradiation )  [signal]
```

Référencé par envoieMesuresBDD().

8.11.3.9 modifieFrequenceArchivage

```
void Mesures::modifieFrequenceArchivage (
            int frequence )  [slot]
```

Modifie la fréquence.

Paramètres

frequence	

Références compteur.

Référencé par IHMRov : :creerFenetreParametres().

```
00160 {
00161     compteur->setInterval(frequence * 1000);
00162 }
```

8.11.3.10 stockeDonnees()

```
void Mesures::stockeDonnees (
            QString type,
            QString donnee )
```

Stocke les donnees passés en argument dans l'objet mesures.

Paramètres

type	QString Type de donnée à stocker.
donnee	QString Valeur de la donnée à stocker.

Références distance, distanceActualisee(), irradiation, irradiationActualisee(), temperature, et temperatureActualisee().

Référencé par traiteTrame().

```
00125 {
00126     if (type == "irradiation")
00127     {
00128         this->irradiation = donnee.toDouble();
00129         emit irradiationActualisee(this->irradiation);
00130     }
00131
00132     if (type == "temperature")
00133     {
00134         this->temperature = donnee.toDouble();
00135         emit temperatureActualisee(this->temperature);
00136     }
00137
00138     if (type == "distance")
00139     {
00140         this->distance = donnee.toDouble();
00141         emit distanceActualisee(this->distance);
00142     }
00143 }
```

8.11.3.11 temperatureActualisee

```
void Mesures::temperatureActualisee (
            double  )  [signal]
```

Référencé par stockeDonnees().

8.11.3.12 traiteTrame

```
void Mesures::traiteTrame (
            QString trame )  [slot]
```

Paramètres

trame	QString Trame reçue.

Références stockeDonnees().

```
00085 {
```

```
00086     bool trameValide = true;
00087
00088     if (trame.startsWith("$") && trame.endsWith("\n"))
00089     {
00090         trame.remove(QChar('$'));
00091         trame.remove(QChar('\n'));
00092
00093         for(int i = 1; i < trame.length(); i++)     // Parcours la trame et vérifie si le contenu après le
      caractère de type est bien un chiffre (filtre les erreurs exemple : "$TX4\n").
00094         {
00095             if(!trame[i].isDigit() && trame[i] != "." && trame[i] != "-")
00096                 trameValide = false;
00097         }
00098     }
00099     else
00100         trameValide = false;
00101
00102     if(trameValide)
00103     {
00104         if(trame.startsWith('T'))
00105             stockeDonnees("temperature", trame.remove('T'));
00106         else if(trame.startsWith('R'))
00107             stockeDonnees("irradiation", trame.remove('R'));
00108         else if(trame.startsWith('D'))
00109             stockeDonnees("distance", trame.remove('D'));
00110         else
00111             trameValide = false;
00112     }
00113
00114     if(!trameValide)
00115         qDebug() << Q_FUNC_INFO << "ERREUR ! Trame invalide";
00116 }
```

8.11.4 Documentation des données membres

8.11.4.1 compteur

```
QTimer* Mesures::compteur  [private]
```

Référencé par getFrequenceArchivage(), Mesures(), et modifieFrequenceArchivage().

8.11.4.2 distance

```
double Mesures::distance  [private]
```

Référencé par getDistance(), et stockeDonnees().

8.11.4.3 irradiation

```
double Mesures::irradiation  [private]
```

Référencé par envoieMesuresBDD(), getIrradiation(), et stockeDonnees().

8.11.4.4 temperature

```
double Mesures::temperature  [private]
```

Référencé par envoieMesuresBDD(), getTemperature(), et stockeDonnees().

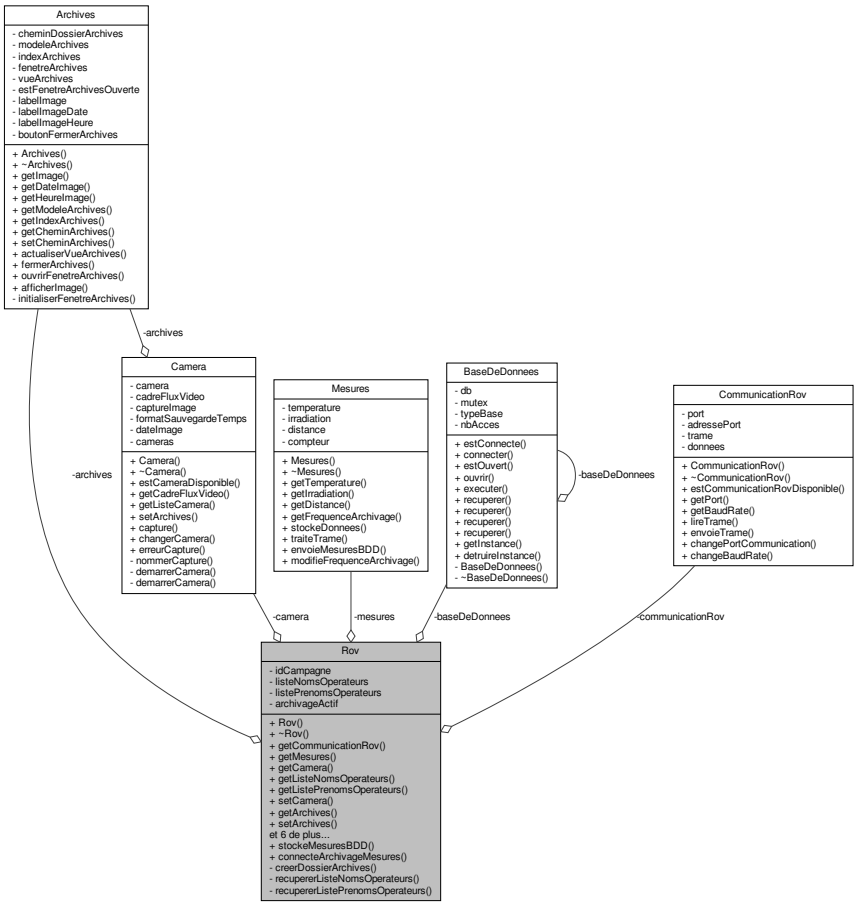
La documentation de cette classe a été générée à partir des fichiers suivants :
- mesures.h
- mesures.cpp

8.12 Référence de la classe Rov

Déclaration de la classe Rov. Gestion des liaisons entre les différentes classes, et paramètres de la campagne et de la BDD.

`#include <rov.h>`

Graphe de collaboration de Rov :



Connecteurs publics

— void stockeMesuresBDD (double temperature, double irradiation)
Stocke les mesures dans la table mesures.
— void connecteArchivageMesures (bool archivageActif)
Active, ou désactive l'archivage des mesures dans la BDD.

Fonctions membres publiques

— Rov (QObject *parent=nullptr)
Constructeur de la classe Rov.
— ~Rov ()
Destructeur de la classe Rov.
— CommunicationRov * getCommunicationRov () const
Renvoie l'objet réalisant la connexion entre le rov et le programme.
— Mesures * getMesures () const
Renvoie l'objet contenant les mesures et les méthodes affiliées.
— Camera * getCamera () const
Renvoie l'objet Camera.

- QVector< QString > getListeNomsOperateurs ()
  Accesseur retournant la liste des noms des opérateurs.
- QVector< QString > getListePrenomsOperateurs ()
  Accesseur retournant la liste des noms des opérateurs.
- void setCamera (Camera *camera)
  Accesseur affectant une instance de la classe Camera.
- Archives * getArchives () const
  Accesseur retournant une instance de la classe Archives.
- void setArchives (Archives *archives)
  Accesseur affectant une instance de la classe Archives.
- void setIdCampagne (QString idCampagne)
  Remplace l'idCampagne.
- QString getIdCampagne ()
  Renvoie idCampagne.
- bool creerNouvelleCampagne (QString nom, QString description, QString idOperateur)
  Crée une nouvelle campagne.
- bool creerDossiersNouvelleCampagne (QString nomNouvelleCampagne)
  Crée un dossier correspondant au nom de la campagne créée.
- void setArchivageActif (bool)
  Modifie la valeur de archivageActif.
- bool getArchivageActif () const
  Retourne l'état de archivageActif.

Fonctions membres privées

- bool creerDossierArchives (QString cheminDossierCampagne)
  Crée le dossier des archives correspondant à la campagne créée.
- void recupererListeNomsOperateurs ()
  Méthode permettant de récuperer la liste des noms opérateurs.
- void recupererListePrenomsOperateurs ()
  Méthode permettant de récuperer la liste des prenoms opérateurs.

Attributs privés

- CommunicationRov * communicationRov
  Communication via le port série avec le rov.
- Mesures * mesures
  Les mesures des capteurs.
- Camera * camera
  La caméra.
- Archives * archives
  Les archives.
- BaseDeDonnees * baseDeDonnees
  La base de données.
- QString idCampagne
  Numéro d'id de la campagne en cours.
- QVector< QString > listeNomsOperateurs
  Liste des noms des opérateurs.
- QVector< QString > listePrenomsOperateurs
  Liste des prenoms des opérateurs.
- bool archivageActif
  L'archivage des mesures est demandé.

8.12.1 Description détaillée

Auteur

REYNIER Jacques & Nicolas BOFFREDO

Version

1.1

Date

Jeudi 13 Juin 2019

8.12.2 Documentation des constructeurs et destructeur

8.12.2.1 Rov()

```
Rov::Rov (
            QObject * parent = nullptr )  [explicit]
```

Instancie des objets Archives, CommunicationRov, Mesures, et BaseDeDonnees. Se connecte à la BDD, et connecte les trames reçues à l'objet Mesures.

Paramètres

parent	

Références archivageActif, archives, baseDeDonnees, communicationRov, connecteArchivageMesures(), BaseDeDonnees : :estOuvert(), BaseDeDonnees : :getInstance(), mesures, BaseDeDonnees : :ouvrir(), recupererListeNomsOperateurs(), et recupererListePrenomsOperateurs().

```
00024                       : QObject(parent), communicationRov(nullptr),
      mesures(nullptr), camera(nullptr), listeNomsOperateurs(0),
      listePrenomsOperateurs(0), archivageActif(1)
00025 {
00026     qDebug() << Q_FUNC_INFO;
00027     archives = new Archives(this);
00028     communicationRov = new CommunicationRov(this);
00029     mesures = new Mesures(this);
00030     baseDeDonnees = BaseDeDonnees::getInstance("QSQLITE");
00031
00032     baseDeDonnees->ouvrir("rovnet.sqlite");
00033     if(baseDeDonnees->estOuvert())
00034         qDebug() << Q_FUNC_INFO << "ouverture réussie BD";
00035     else
00036         qDebug() << Q_FUNC_INFO << "echec ouverture BD";
00037
00038     // Connexions
00039     connect(communicationRov, SIGNAL(trameRecue(QString)),
      mesures, SLOT(traiteTrame(QString)));
00040
00041     connecteArchivageMesures(archivageActif);
00042
00043     recupererListeNomsOperateurs();
00044     recupererListePrenomsOperateurs();
00045 }
```

8.12.2.2 ~Rov()

```
Rov::~Rov ( )
```

Références BaseDeDonnees : :detruireInstance().

```
00052 {
00053     BaseDeDonnees::detruireInstance();
00054     qDebug() << Q_FUNC_INFO;
00055 }
```

8.12.3 Documentation des fonctions membres

8.12.3.1 connecteArchivageMesures

```
void Rov::connecteArchivageMesures (
            bool archivageActif )  [slot]
```

Active ou désactive l'archivage des mesures dans la BDD.

Paramètres

archivageActif	bool nouvel état d'activation.

Références mesures, et stockeMesuresBDD().

Référencé par Rov(), et setArchivageActif().

```
00085 {
00086     if(archivageActif)
00087         connect(mesures, SIGNAL(mesuresBDDPrete(double, double)), this, SLOT(
      stockeMesuresBDD(double, double)));
00088     else
00089         disconnect(mesures, SIGNAL(mesuresBDDPrete(double, double)), this, SLOT(
      stockeMesuresBDD(double, double)));
00090 }
```

8.12.3.2 creerDossierArchives()

```
bool Rov::creerDossierArchives (
            QString cheminDossierCampagne )  [private]
```

Références archives, et Archives : :setCheminArchives().

Référencé par creerDossiersNouvelleCampagne().

```
00247 {
00248     qDebug() << Q_FUNC_INFO;
00249     QDir dossierArchives(cheminDossierCampagne);
00250     if(dossierArchives.mkdir("Archives"))
00251     {
00252         QString cheminDossierArchives = cheminDossierCampagne + "/Archives/";
00253         qDebug() << Q_FUNC_INFO << archives << cheminDossierArchives;
00254         archives->setCheminArchives(cheminDossierArchives);
00255         return true;
00256     }
00257     return false;
00258 }
```

8.12.3.3 creerDossiersNouvelleCampagne()

```
bool Rov::creerDossiersNouvelleCampagne (
            QString nomNouvelleCampagne )
```

Références creerDossierArchives().

Référencé par creerNouvelleCampagne().

```
00231 {
00232     QDir dossierApplication(QApplication::applicationDirPath());
00233     if(dossierApplication.mkdir(nomNouvelleCampagne))
00234     {
00235         QString cheminDossierCampagne = QApplication::applicationDirPath() + "/" + nomNouvelleCampagne;
00236         qDebug() << Q_FUNC_INFO << cheminDossierCampagne;
00237         return creerDossierArchives(cheminDossierCampagne);
00238     }
00239     return false;
00240 }
```

8.12.3.4 creerNouvelleCampagne()

```
bool Rov::creerNouvelleCampagne (
            QString nom,
            QString description,
            QString idOperateur )
```

Paramètres

nom	QString
description	QString
idOperateur	QString

Renvoie

bool

Références archives, baseDeDonnees, communicationRov, creerDossiersNouvelleCampagne(), CommunicationRov : :est↩ CommunicationRovDisponible(), BaseDeDonnees : :executer(), Archives : :getCheminArchives(), idCampagne, BaseDeDonnees↩ : :recuperer(), et setIdCampagne().

Référencé par IHMRov : :enregistrerParametresCampagne().

```
00178 {
00179     if(communicationRov->estCommunicationRovDisponible())
00180     {
00181         if(creerDossiersNouvelleCampagne(nom))
00182         {
00183             QString cheminArchives = archives->getCheminArchives();
00184             QString requete = "INSERT INTO 'campagnes'(nom, description, date, cheminArchives, idOperateur)
       VALUES ('" + nom + "', '" + description + "', datetime('now', 'localtime'), '" + cheminArchives + "', '" +
      idOperateur + "')";
00185             bool retour = baseDeDonnees->executer(requete);
00186             if(!retour)
00187             {
00188                 return false;
00189             }
00190
00191             QString idCampagne;
00192             bool requeteRecupIdCampagne = baseDeDonnees->recuperer("SELECT idCampagne
       FROM campagnes WHERE cheminArchives = '" + cheminArchives + "';", idCampagne);
00193             if(requeteRecupIdCampagne)
00194             {
00195                 setIdCampagne(idCampagne);
00196             }
00197             else
00198             {
00199                 return false;
00200             }
00201             return true;
00202         }
00203         else
00204             return false;
00205     }
00206     else
00207         return false;
00208 }
```

8.12.3.5 getArchivageActif()

```
bool Rov::getArchivageActif ( ) const
```

Indique si l'archivage des mesures dans la BDD est activé ou non.

Renvoie

bool archivageActif état d'activation de l'archivage des mesures.

Références archivageActif.

```
00075 {
00076     return this->archivageActif;
00077 }
```

8.12.3.6 getArchives()

```
Archives * Rov::getArchives ( ) const
```

Renvoie

une instance de la classe *Archives*

Références archives.

```
00128 {
00129     return archives;
00130 }
```

8.12.3.7 getCamera()

```
Camera * Rov::getCamera ( ) const
```

Accesseur retournant une instance de la classe Camera.

Renvoie

une instance de la classe *Camera*

Références camera.

```
00118 {
00119     return camera;
00120 }
```

8.12.3.8 getCommunicationRov()

```
CommunicationRov * Rov::getCommunicationRov ( ) const
```

Accesseur retournant une instance de la classe CommunicationRov.

Renvoie

une instance de la classe *CommunicationRov*

Références communicationRov.

Référencé par IHMRov : :actualiseIconesEtat(), ControleRov : :ControleRov(), IHMRov : :creerFenetreParametres(), et IHMRov↩ : :enregistrerParametres().

```
00098 {
00099     return this->communicationRov;
00100 }
```

8.12.3.9 getIdCampagne()

```
QString Rov::getIdCampagne ( )
```

Accesseur retournant l'id de la campagne en cours.

Renvoie

un *QString* correspondant à l'id de la campagne en cours.

Références idCampagne.

```
00165 {
00166     return this->idCampagne;
00167 }
```

8.12.3.10 getListeNomsOperateurs()

```
QVector< QString > Rov::getListeNomsOperateurs ( )
```

Renvoie

un *QVector* de *QString*, la liste des noms des opérateurs

Références listeNomsOperateurs.

Référencé par IHMRov : :remplirComboBoxListeOperateurs().

```
00283 {
00284     return listeNomsOperateurs;
00285 }
```

8.12.3.11 getListePrenomsOperateurs()

```
QVector< QString > Rov::getListePrenomsOperateurs ( )
```

Renvoie

un *QVector* de *QString*, la liste des noms des opérateurs

Références listePrenomsOperateurs.

Référencé par IHMRov : :remplirComboBoxListeOperateurs().

```
00310 {
00311     return listePrenomsOperateurs;
00312 }
```

8.12.3.12 getMesures()

```
Mesures * Rov::getMesures ( ) const
```

Accesseur retournant une instance de la classe Mesures.

Renvoie

une instance de la classe *Mesures*

Références mesures.

Référencé par IHMRov : :creerFenetreParametres(), et IHMRov : :IHMRov().

```
00108 {
00109     return mesures;
00110 }
```

8.12.3.13 recupererListeNomsOperateurs()

```
void Rov::recupererListeNomsOperateurs ( )  [private]
```

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

Référencé par Rov().

```
00265 {
00266     QString requete = "SELECT nom FROM operateurs";
00267     bool reussi = baseDeDonnees->recuperer(requete,
      listeNomsOperateurs);
00268     qDebug() << requete << listeNomsOperateurs;
00269
00270     if(reussi)
00271         qDebug() << Q_FUNC_INFO << listeNomsOperateurs;
00272
00273     else
00274         qDebug() << Q_FUNC_INFO << "erreur SQL" << requete;
00275 }
```

8.12.3.14 recupererListePrenomsOperateurs()

```
void Rov::recupererListePrenomsOperateurs ( )  [private]
```

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

Référencé par Rov().

```
00292 {
00293     QString requete = "SELECT prenom FROM operateurs";
00294     bool reussi = baseDeDonnees->recuperer(requete,
      listePrenomsOperateurs);
00295     qDebug() << requete << listePrenomsOperateurs;
00296
00297     if(reussi)
00298         qDebug() << Q_FUNC_INFO << listePrenomsOperateurs;
00299
00300     else
00301         qDebug() << Q_FUNC_INFO << "erreur SQL" << requete;
00302 }
```

8.12.3.15 setArchivageActif()

```
void Rov::setArchivageActif (
            bool archivageActif )
```

Rend l'archivage des mesures actif, ou non.

Paramètres

archivageActif	bool vrai si l'archivage des mesures est demandé, sinon faux.

Références archivageActif, et connecteArchivageMesures().

Référencé par IHMRov : :enregistrerParametres().

```
00063 {
00064     qDebug() << Q_FUNC_INFO << archivageActif;
00065     this->archivageActif = archivageActif;
00066     connecteArchivageMesures(archivageActif);
00067 }
```

8.12.3.16 setArchives()

```
void Rov::setArchives (
            Archives * archives )
```

Références archives.

Référencé par IHMRov : :IHMRov().

```
00146 {
00147     this->archives = archives;
00148 }
```

8.12.3.17 setCamera()

```
void Rov::setCamera (
            Camera * camera )
```

Références camera.

Référencé par IHMRov : :IHMRov().

```
00137 {
00138     this->camera = camera;
00139 }
```

8.12.3.18 setIdCampagne()

```
void Rov::setIdCampagne (
            QString idCampagne )
```

Accesseur affectant l'id de la campagne en cours.

Références idCampagne.

Référencé par creerNouvelleCampagne().

```
00155 {
00156     this->idCampagne = idCampagne;
00157 }
```

8.12.3.19 stockeMesuresBDD

```
void Rov::stockeMesuresBDD (
            double temperature,
            double irradiation )  [slot]
```

Stocke les mesures dans la BDD.

Paramètres

temperature	double
irradiation	double

Références baseDeDonnees, BaseDeDonnees : :executer(), et idCampagne.

Référencé par connecteArchivageMesures().

```
00217 {
00218     QString requete = "INSERT INTO mesures VALUES(datetime('now', 'localtime'), " + QString::number(
      temperature) + ", " + QString::number(irradiation) + ", " + idCampagne + ");";
00219     bool reussi = baseDeDonnees->executer(requete);
00220     qDebug() << requete;
00221
00222     if (!reussi)
00223         qDebug() << Q_FUNC_INFO << "ERREUR ! Echec de l'envoi des mesures dans la BDD !";
00224 }
```

8.12.4 Documentation des données membres

8.12.4.1 archivageActif

```
bool Rov::archivageActif  [private]
```

Référencé par getArchivageActif(), Rov(), et setArchivageActif().

8.12.4.2 archives

```
Archives* Rov::archives  [private]
```

Référencé par creerDossierArchives(), creerNouvelleCampagne(), getArchives(), Rov(), et setArchives().

8.12.4.3 baseDeDonnees

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

Référencé par creerNouvelleCampagne(), recupererListeNomsOperateurs(), recupererListePrenomsOperateurs(), Rov(), et stocke↩ MesuresBDD().

8.12.4.4 camera

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

Référencé par getCamera(), et setCamera().

8.12.4.5 communicationRov

```
CommunicationRov* Rov::communicationRov  [private]
```

Référencé par creerNouvelleCampagne(), getCommunicationRov(), et Rov().

8.12.4.6 idCampagne

```
QString Rov::idCampagne  [private]
```

Référencé par creerNouvelleCampagne(), getIdCampagne(), setIdCampagne(), et stockeMesuresBDD().

8.12.4.7 listeNomsOperateurs

```
QVector<QString> Rov::listeNomsOperateurs  [private]
```

Référencé par getListeNomsOperateurs(), et recupererListeNomsOperateurs().

8.12.4.8 listePrenomsOperateurs

```
QVector<QString> Rov::listePrenomsOperateurs  [private]
```

Référencé par getListePrenomsOperateurs(), et recupererListePrenomsOperateurs().

8.12.4.9 mesures

```
Mesures* Rov::mesures  [private]
```

Référencé par connecteArchivageMesures(), getMesures(), et Rov().

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

9 Documentation des fichiers

9.1 Référence du fichier archives.cpp

Définition de la classe Archives.

```
#include "archives.h"
#include <QDebug>
```

9.1.1 Description détaillée

Auteur

Nicolas BOFFREDO

Version

1.1

9.2 Référence du fichier archives.h

Déclaration de la classe Archives.

```
#include <QApplication>
#include <QObject>
#include <QDialog>
#include <QDir>
#include <QtWidgets>
#include <QListView>
#include <QFileSystemModel>
```

Classes

— class Archives
Déclaration de la classe Archives.

Macros

— #define LARGEUR_IMAGE 640
— #define HAUTEUR_IMAGE 480
— #define HAUTEUR_INFORMATIONS HAUTEUR_IMAGE/2
— #define LARGEUR_VUE_ARCHIVES LARGEUR_IMAGE/2
— #define LARGEUR_MAX LARGEUR_IMAGE + LARGEUR_ARCHIVES
— #define HAUTEUR_MAX HAUTEUR_IMAGE + HAUTEUR_INFORMATIONS

9.2.1 Description détaillée

Auteur

Nicolas BOFFREDO

Version

1.1

Date

Mercredi 12 Juin 2019

9.2.2 Documentation des macros

9.2.2.1 HAUTEUR_IMAGE

```
#define HAUTEUR_IMAGE 480
```

9.2.2.2 HAUTEUR_INFORMATIONS

```
#define HAUTEUR_INFORMATIONS HAUTEUR_IMAGE/2
```

9.2.2.3 HAUTEUR_MAX

```
#define HAUTEUR_MAX HAUTEUR_IMAGE + HAUTEUR_INFORMATIONS
```

9.2.2.4 LARGEUR_IMAGE

```
#define LARGEUR_IMAGE 640
```

9.2.2.5 LARGEUR_MAX

```
#define LARGEUR_MAX LARGEUR_IMAGE + LARGEUR_ARCHIVES
```

9.2.2.6 LARGEUR_VUE_ARCHIVES

```
#define LARGEUR_VUE_ARCHIVES LARGEUR_IMAGE/2
```

9.3 Référence du fichier basededonnees.cpp

Définition de la classe BaseDeDonnees.

```
#include "basededonnees.h"
#include <QDebug>
#include <QMessageBox>
```

9.3.1 Description détaillée

Auteur

Thierry Vaira

Version

1.1

9.4 Référence du fichier basededonnees.h

Déclaration de la classe BaseDeDonnees.

```
#include <QObject>
#include <QtSql/QtSql>
#include <QSqlDatabase>
#include <QMutex>
#include <QString>
```

Classes

— class BaseDeDonnees
Déclaration de la classe BaseDeDonnees.

Macros

— #define DEBUG_BASEDEDONNEES
— #define BDD_HOSTNAME "localhost"
— #define BDD_USERNAME "root"
— #define BDD_PASSWORD "password"

9.4.1 Description détaillée

Auteur

Thierry VAIRA

Version

1.1

9.4.2 Documentation des macros

9.4.2.1 BDD_HOSTNAME

```
#define BDD_HOSTNAME "localhost"
```

9.4.2.2 BDD_PASSWORD

```
#define BDD_PASSWORD "password"
```

9.4.2.3 BDD_USERNAME

```
#define BDD_USERNAME "root"
```

9.4.2.4 DEBUG_BASEDEDONNEES

```
#define DEBUG_BASEDEDONNEES
```

9.5 Référence du fichier bras.cpp

Définition de la classe Bras.

```
#include "bras.h"
#include <cmath>
#include <QDebug>
```

9.5.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Juin 2019

9.6 Référence du fichier bras.h

Déclaration de la classe Bras. Réceptionne les signaux de la manette destiné aux mouvements du bras, et émet les trames correspondantes.

```
#include <QObject>
```

Classes

— class Bras
Déclaration de la classe Bras. Réceptionne les signaux de la manette destiné aux mouvements du bras, et émet les trames correspondantes.

9.6.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Juin 2019

9.7 Référence du fichier camera.cpp

Définition de la classe Camera.

```
#include "camera.h"
#include <QApplication>
#include <QDebug>
#include <QString>
#include <QTimer>
#include <QCameraViewfinder>
```

9.7.1 Description détaillée

Auteur

Nicolas BOFFREDO

Version

1.1

9.8 Référence du fichier camera.h

Déclaration de la classe Camera.

```
#include "archives.h"
#include <QObject>
#include <QString>
#include <QDateTime>
#include <QCamera>
#include <QCameraInfo>
#include <QCameraViewfinder>
#include <QCameraImageCapture>
```

Classes

— class Camera
Déclaration de la classe Camera.

Macros

— #define LARGEUR_RESOLUTION_IMAGE 1280
— #define HAUTEUR_RESOLUTION_IMAGE 720

9.8.1 Description détaillée

Auteur

Nicolas BOFFREDO

Version

1.1

Date

Jeudi 13 Juin 2019

9.8.2 Documentation des macros

9.8.2.1 HAUTEUR_RESOLUTION_IMAGE

```
#define HAUTEUR_RESOLUTION_IMAGE 720
```

9.8.2.2 LARGEUR_RESOLUTION_IMAGE

```
#define LARGEUR_RESOLUTION_IMAGE 1280
```

9.9 Référence du fichier Changelog.md

9.10 Référence du fichier communicationrov.cpp

Définition de la classe CommunicationRov.

```
#include "communicationrov.h"
#include <QDebug>
```

9.10.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Mars 2019

9.11 Référence du fichier communicationrov.h

Déclaration de la classe CommunicationRov. Gère la communication entre le Rov et le Rov.

```
#include <QObject>
#include <QtSerialPort/QSerialPort>
```

Classes

— class CommunicationRov
Déclaration de la classe CommunicationRov. Gère la communication entre le Rov et le Rov.

9.11.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Juin 2019

9.12 Référence du fichier controlecamera.cpp

Définition de la classe ControleCamera.

```
#include "controlecamera.h"
#include <cmath>
#include <QDebug>
```

9.12.1 Description détaillée

Auteur

Nicolas BOFFREDO

Version

1.1

9.13 Référence du fichier controlecamera.h

Déclaration de la classe ControleCamera.

```
#include <QObject>
```

Classes

— class ControleCamera
Déclaration de la classe ControleCamera.

9.13.1 Description détaillée

Auteur

Nicolas BOFFREDO

Version

1.1

Date

Jeudi 13 Juin 2019

9.14 Référence du fichier controlerov.cpp

Définition de la classe ControleRov.

```
#include "controlerov.h"
#include <QLoggingCategory>
#include <QtGamepad/QGamepad>
#include "deplacement.h"
#include "bras.h"
#include "controlecamera.h"
```

9.14.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Juin 2019

9.15 Référence du fichier controlerov.h

Déclaration de la classe ControleRov. Permet le contrôle des éléments du rov, en reliant la manette aux méthodes de déplacement.

```
#include <QObject>
#include "manette.h"
#include "deplacement.h"
#include "bras.h"
#include "controlecamera.h"
#include "rov.h"
```

Classes

— class ControleRov
Déclaration de la classe ControleRov. Permet le contrôle des éléments du rov, en reliant la manette aux méthodes de déplacement.

Macros

— #define MODE_DEPLACEMENT 0
— #define MODE_BRAS 1

9.15.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Juin 2019

9.15.2 Documentation des macros

9.15.2.1 MODE_BRAS

```
#define MODE_BRAS 1
```

Référencé par ControleRov : :changeConnexions(), et ControleRov : :changeMode().

9.15.2.2 MODE_DEPLACEMENT

```
#define MODE_DEPLACEMENT 0
```

Référencé par ControleRov : :changeConnexions(), et ControleRov : :changeMode().

9.16 Référence du fichier deplacement.cpp

Définition de la classe Deplacement.

```
#include "deplacement.h"
#include <cmath>
#include <QDebug>
```

9.16.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Juin 2019

9.17 Référence du fichier deplacement.h

Déclaration de la classe Deplacement. Réceptionne les signaux de la manette destiné aux déplacements du Rov, et émet les trames correspondantes.

```
#include <QObject>
```

Classes

— class Deplacement
Déclaration de la classe Deplacement. Réceptionne les signaux de la manette destiné aux déplacements du Rov, et émet les trames correspondantes.

9.17.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

Date

Jeudi 13 Juin 2019

9.18 Référence du fichier ihmrov.cpp

Définition de la classe IHMRov.

```
#include "ihmrov.h"
#include "camera.h"
#include "controlerov.h"
#include "rov.h"
#include "basededonnees.h"
#include <QMessageBox>
```

9.18.1 Description détaillée

Auteur

Nicolas BOFFREDO & Jacques REYNIER

Version

1.1

9.19 Référence du fichier ihmrov.h

Déclaration de la classe Rov.

```
#include "camera.h"
#include "controlerov.h"
#include "rov.h"
#include <QtWidgets>
#include <QComboBox>
#include <QDialog>
#include <QTimer>
#include <QApplication>
#include <QListView>
#include <QFileSystemModel>
#include <QStackedWidget>
#include <QDebug>
#include <QMenuBar>
#include <QMenu>
#include <qwt_thermo.h>
#include <QMainWindow>
```

Classes

— class IHMRov
Déclaration de la classe Rov.

Macros

— #define APPLICATION_TITRE "Projet Rov'net - BTS SN IR 2019 (E6.2)"
— #define APPLICATION_INFORMATIONS "Version : 1.0 \nJacques Reynier & Boffredo Nicolas"
— #define PERIODE 1000
— #define WIDGET_CAMERA_INDISPONIBLE 0
— #define WIDGET_CAMERA_DISPONIBLE 1
— #define DISTANCE_LOIN 40
— #define DISTANCE_PROCHE 25

9.19.1 Description détaillée

Auteur

Nicolas BOFFREDO & Jacques REYNIER

Version

1.1

Date

Jeudi 13 Juin 2019

9.19.2 Documentation des macros

9.19.2.1 APPLICATION_INFORMATIONS

```
#define APPLICATION_INFORMATIONS "Version : 1.0 \nJacques Reynier & Boffredo Nicolas"
```

Référencé par IHMRov : :ouvrirAide().

9.19.2.2 APPLICATION_TITRE

```
#define APPLICATION_TITRE "Projet Rov'net - BTS SN IR 2019 (E6.2)"
```

Référencé par IHMRov : :enregistrerParametres(), IHMRov : :enregistrerParametresCampagne(), IHMRov : :IHMRov(), et IHMRov↩ : :ouvrirAide().

9.19.2.3 DISTANCE_LOIN

```
#define DISTANCE_LOIN 40
```

Référencé par IHMRov : :actualiserIndicationDistance().

9.19.2.4 DISTANCE_PROCHE

```
#define DISTANCE_PROCHE 25
```

Référencé par IHMRov : :actualiserIndicationDistance().

9.19.2.5 PERIODE

```
#define PERIODE 1000
```

Référencé par IHMRov : :initialiserChronometre().

9.19.2.6 WIDGET_CAMERA_DISPONIBLE

```
#define WIDGET_CAMERA_DISPONIBLE 1
```

Référencé par IHMRov : :actualiseIconesEtat().

9.19.2.7 WIDGET_CAMERA_INDISPONIBLE

```
#define WIDGET_CAMERA_INDISPONIBLE 0
```

Référencé par IHMRov : :actualiseIconesEtat().

9.20 Référence du fichier INSTALL.md

9.21 Référence du fichier main.cpp

Programme principal Rov'net.

```
#include "ihmrov.h"
#include <QApplication>
```

Fonctions

— int main (int argc, char ∗argv[ ])

9.21.1 Description détaillée

Crée et affiche la fenêtre principale de l'application Rov'net

Auteur

Nicolas BOFFREDO nboffredo@gmail.com
Jacques REYNIER reynier.jacques@gmail.com

Version

1.1

9.21.2 Documentation des fonctions

9.21.2.1 main()

```
main (
            int argc,
            char * argv[] )
```

Paramètres

argc	
argv[ ]	

Renvoie

int

```
00024 {
00025     QApplication a(argc, argv);
00026
00027     IHMRov ihm;
00028     ihm.show();
00029
00030     return a.exec();
00031 }
```

9.22 Référence du fichier manette.cpp

Définition de la classe Manette.

```
#include "manette.h"
#include "deplacement.h"
#include "communicationrov.h"
#include <QLoggingCategory>
#include <QtGamepad/QGamepad>
#include <QDebug>
```

9.22.1 Description détaillée

Auteur

REYNIER Jacques

Version

1.1

9.23 Référence du fichier manette.h

Déclaration de la classe Manette. Liaison avec la manette, permettant d'en recevoir les signaux.

```
#include <QtCore/QObject>
#include <QtCore/QTimer>
#include "rov.h"
#include "deplacement.h"
#include "bras.h"
```

Classes

— class Manette
Déclaration de la classe Manette. Liaison avec la manette, permettant d'en recevoir les signaux.

9.23.1 Description détaillée

Auteur

REYNIER Jacques & Nicolas BOFFREDO

Version

1.1

Date

Jeudi 13 Juin 2019

9.24 Référence du fichier mesures.cpp

Définition de la classe Mesures.

```
#include "mesures.h"
#include <QDebug>
#include <QTimer>
```

9.24.1 Description détaillée

Auteur

REYNIER Jacques & Nicolas BOFFREDO

Version

1.1

9.25 Référence du fichier mesures.h

Déclaration de la classe Mesures. Gestion des mesures des capteurs (température, irradiation, et distance).

```
#include <QObject>
#include <QTimer>
#include "communicationrov.h"
```

Classes

— class Mesures
Déclaration de la classe Mesures. Gestion des mesures des capteurs (température, irradiation, et distance).

9.25.1 Description détaillée

Auteur

REYNIER Jacques & Nicolas BOFFREDO

Version

1.1

Date

Jeudi 13 Juin 2019

9.26 Référence du fichier README.md

9.27 Référence du fichier rov.cpp

Définition de la classe Rov. Gestion des liaisons entre les différentes classes, et paramètres de la campagne et de la BDD.

```
#include "rov.h"
#include "basededonnees.h"
#include <QDebug>
```

9.27.1 Description détaillée

Auteur

REYNIER Jacques & Nicolas BOFFREDO

Version

1.1

9.28 Référence du fichier rov.h

Déclaration de la classe Rov. Gestion des liaisons entre les différentes classes, et paramètres de la campagne et de la BDD.

```
#include <QObject>
#include "communicationrov.h"
#include "mesures.h"
#include "camera.h"
#include "archives.h"
```

Classes

— class Rov
Déclaration de la classe Rov. Gestion des liaisons entre les différentes classes, et paramètres de la campagne et de la BDD.

9.28.1 Description détaillée

Auteur

REYNIER Jacques & Nicolas BOFFREDO

Version

1.1

Date

Jeudi 13 Juin 2019

Index