

Campus Serre

1.0

Généré par Doxygen 1.7.6.1

Jeudi Juin 7 2018 20 :19 :19

## Table des matières

<b>1</b>	<b>Page principale du projet Campus Serre</b>	<b>2</b>
1.1	Introduction . . . . .	2
1.2	Table des matières . . . . .	2
<b>2</b>	<b>Changelog</b>	<b>2</b>
<b>3</b>	<b>Configuration</b>	<b>5</b>
<b>4</b>	<b>Manuel d'installation</b>	<b>5</b>
<b>5</b>	<b>Recette IR</b>	<b>5</b>
<b>6</b>	<b>Base de données SQLite</b>	<b>6</b>
<b>7</b>	<b>A propos</b>	<b>6</b>
<b>8</b>	<b>Licence GPL</b>	<b>7</b>
<b>9</b>	<b>Liste des choses à faire</b>	<b>7</b>
<b>10</b>	<b>Documentation des classes</b>	<b>7</b>
10.1	Référence de la classe BaseDeDonnees . . . . .	7
10.1.1	Description détaillée . . . . .	9
10.1.2	Documentation des constructeurs et destructeur . . . . .	9
10.1.3	Documentation des fonctions membres . . . . .	9
10.1.4	Documentation des données membres . . . . .	15
10.2	Référence de la classe GestionPort . . . . .	15
10.2.1	Description détaillée . . . . .	16
10.2.2	Documentation des constructeurs et destructeur . . . . .	16
10.2.3	Documentation des fonctions membres . . . . .	17
10.2.4	Documentation des données membres . . . . .	20
10.3	Référence de la classe IHMCampusSerre . . . . .	20
10.3.1	Description détaillée . . . . .	23
10.3.2	Documentation des constructeurs et destructeur . . . . .	23
10.3.3	Documentation des fonctions membres . . . . .	24
10.3.4	Documentation des données membres . . . . .	36

10.4	Référence de la structure Mesure . . . . .	37
10.4.1	Documentation des données membres . . . . .	37
10.5	Référence de la classe Telemetry . . . . .	38
10.5.1	Description détaillée . . . . .	41
10.5.2	Documentation des constructeurs et destructeur . . . . .	41
10.5.3	Documentation des fonctions membres . . . . .	41
10.5.4	Documentation des données membres . . . . .	48
<b>11</b>	<b>Documentation des fichiers</b>	<b>48</b>
11.1	Référence du fichier basededonnees.cpp . . . . .	49
11.1.1	Description détaillée . . . . .	49
11.2	Référence du fichier basededonnees.h . . . . .	49
11.2.1	Description détaillée . . . . .	49
11.3	Référence du fichier Changelog.dox . . . . .	49
11.4	Référence du fichier gestionPort.cpp . . . . .	49
11.4.1	Description détaillée . . . . .	49
11.5	Référence du fichier gestionPort.h . . . . .	50
11.5.1	Description détaillée . . . . .	50
11.5.2	Documentation des macros . . . . .	50
11.6	Référence du fichier IHMCampusSerre.cpp . . . . .	50
11.6.1	Description détaillée . . . . .	50
11.7	Référence du fichier IHMCampusSerre.h . . . . .	51
11.7.1	Description détaillée . . . . .	51
11.7.2	Documentation des macros . . . . .	51
11.8	Référence du fichier main.cpp . . . . .	51
11.8.1	Description détaillée . . . . .	51
11.8.2	Documentation des fonctions . . . . .	51
11.9	Référence du fichier README.dox . . . . .	52
11.10	Référence du fichier telemetry.cpp . . . . .	52
11.10.1	Description détaillée . . . . .	52
11.11	Référence du fichier telemetry.h . . . . .	52
11.11.1	Description détaillée . . . . .	53

# 1 Page principale du projet Campus Serre

## 1.1 Introduction

Le projet Campus Serre permet la gestion d'une serre au moyen de différents capteurs le tout géré par une IHM.

## 1.2 Table des matières

- [Configuration](#)
- [Manuel d'installation](#)
- [Changelog](#)
- [Recette IR](#)
- [Base de données SQLite](#)
- [A propos](#)
- [Licence GPL](#)

Dépôt SVN : <https://svn.riouxsvn.com/campus-serre>

# 2 Changelog

-----  
r30 | ldisario | 2018-05-25 16 :55 :42 +0200 (ven. 25 mai 2018) | 1 ligne

Ajout de la version 1

-----  
r29 | ldisario | 2018-05-22 09 :07 :09 +0200 (mar. 22 mai 2018) | 1 ligne

Ajout de l'affichage de l'état de la vanne dans l'onglet Télémétrie et correction de faute d'orthographe

-----  
r28 | ldisario | 2018-05-22 08 :18 :57 +0200 (mar. 22 mai 2018) | 1 ligne

Ajout du protocole de l'envoi de la trame pour passer le chauffage, vanne et ouvrant en automatique

-----  
r27 | ldisario | 2018-04-23 13 :57 :10 +0200 (lun. 23 avril 2018) | 1 ligne

Ajout d'un tri de trame dans receptionnerDonnees

-----  
r26 | ldisario | 2018-04-23 11 :17 :22 +0200 (lun. 23 avril 2018) | 1 ligne

Ajout de l'affichage de l'état du chauffage dans l'ihm

r25 | ldisario | 2018-04-23 10 :27 :28 +0200 (lun. 23 avril 2018) | 1 ligne

Ajout de la méthode commanderChauffage, Ajout de protocole d'envoi de la trame du chauffage

r24 | ldisario | 2018-04-23 10 :26 :11 +0200 (lun. 23 avril 2018) | 1 ligne

Ajout de la méthode commanderChauffage, Ajout de protocole d'envoi de la trame du chauffage

r23 | tvaira | 2018-04-19 19 :11 :59 +0200 (jeu. 19 avril 2018) | 1 ligne

Mise à jour du simulateur avec le nouveau protocole

r22 | ldisario | 2018-04-18 17 :47 :47 +0200 (mer. 18 avril 2018) | 1 ligne

Ajout de commentaire

r21 | ldisario | 2018-04-18 17 :47 :11 +0200 (mer. 18 avril 2018) | 1 ligne

Ajout de commentaire

r20 | ldisario | 2018-04-18 16 :51 :19 +0200 (mer. 18 avril 2018) | 1 ligne

Ajout de la fonction initialiserBDD, ajout des unités des consignes dans l'onglet Culture de l'ihm

r19 | ldisario | 2018-04-18 15 :51 :20 +0200 (mer. 18 avril 2018) | 1 ligne

Ajout du protocole d'envoi de trame, création de la méthode envoyerSeuils, envoyerConsignes, ajout des consignes dans l'ihm, création des fonctions recupererSeuils, recupererConsignes et creerIHM

r18 | tvaira | 2018-04-17 11 :01 :51 +0200 (mar. 17 avril 2018) | 1 ligne

Verification des TODO

r17 | tvaira | 2018-04-16 09 :07 :36 +0200 (lun. 16 avril 2018) | 1 ligne

Verification des TODO

r16 | ldisario | 2018-04-04 17 :54 :23 +0200 (mer. 04 avril 2018) | 1 ligne

Modification du Protocole, finalisation de la méthode afficherOngletHistorique ainsi que modifierSeuils de la classe [IHM Campus Serre](#)

r15 | ldisario | 2018-04-01 10 :17 :09 +0200 (dim. 01 avril 2018) | 1 ligne

Initialisation de la base de donnees

-----  
r14 | tvaira | 2018-03-31 19 :53 :58 +0200 (sam. 31 mars 2018) | 1 ligne

Retour sur la revue 2

-----  
r13 | tvaira | 2018-03-24 09 :40 :09 +0100 (sam. 24 mars 2018) | 1 ligne

Ajout parametrage Doxygen

-----  
r12 | tvaira | 2018-03-23 17 :04 :30 +0100 (ven. 23 mars 2018) | 1 ligne

Verification des commentaires Doxygen

-----  
r11 | tvaira | 2018-03-23 16 :16 :46 +0100 (ven. 23 mars 2018) | 1 ligne

-----  
r10 | ldisario | 2018-03-22 18 :06 :54 +0100 (jeu. 22 mars 2018) | 1 ligne

Restructuration du Projet

-----  
r9 | ldisario | 2018-03-21 17 :34 :43 +0100 (mer. 21 mars 2018) | 1 ligne

Ajout de la base de donnees SQLITE

-----  
r8 | ldisario | 2018-03-21 17 :26 :07 +0100 (mer. 21 mars 2018) | 2 lignes

Ajout de la classe [BaseDeDonnees](#)

-----  
r7 | ldisario | 2018-03-21 16 :31 :06 +0100 (mer. 21 mars 2018) | 1 ligne

Ajout du protocole de communication

-----  
r6 | ldisario | 2018-03-21 16 :29 :48 +0100 (mer. 21 mars 2018) | 1 ligne

Ajout du fichier IHM manquant

-----  
r5 | ldisario | 2018-03-21 16 :27 :51 +0100 (mer. 21 mars 2018) | 1 ligne

Ajout du projet Campus Serre : réception, extraction de toute la trame , et affichage de 5 champs de la trame

-----  
r4 | ldisario | 2018-03-21 16 :11 :32 +0100 (mer. 21 mars 2018) | 1 ligne

Ajout du projet Campus Serre : réception, extraction de toute la trame , et affichage de 5 champs de la trame

-----  
r3 | ldisario | 2018-03-21 16 :03 :18 +0100 (mer. 21 mars 2018) | 1 ligne

Ajout du projet Campus Serre : réception, extraction de toute la trame , et affichage de 5 champs de la trame

-----  
r2 | tvaira | 2018-02-03 11 :10 :25 +0100 (sam. 03 févr. 2018) | 1 ligne

Ajout initial (tv)

-----  
r1 | www-data | 2018-02-03 11 :00 :51 +0100 (sam. 03 févr. 2018) | 1 ligne

Creating initial repository structure

### 3 Configuration

Poste de développement :

- Distribution : Ubuntu 12.04.5 LTS
- OS : GNU/Linux
- Noyau : Linux
- Version : 3.8.0-44-generic
- Machine : x86\_64
- Processeur : Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
- Mémoire RAM : 8129984 kB

Liste des paquets Qt nécessaires :

- libqt4-sql libqt4-sql-sqlite libqtgui4 libqtcore4

Modules Xbee : voir dossier

### 4 Manuel d'installation

Fabrication de l'exécutable :

- `qmake`
- `make`

### 5 Recette IR

Étudiant 3 : Di Sario Laurent

- La base de données est fonctionnelle et complétée
- Le système est paramétrable
- Les appareils sont pilotables
- Les informations de la serre sont consultables

## 6 Base de données SQLite

```
pragma foreign_keys = on ;
```

```
CREATE TABLE seuils ( "idSeuils" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL , "temperatureMin" REAL NOT NULL, "temperatureMax" REAL NOT NULL, "vitesseMax" REAL ) ;
```

```
CREATE TABLE consignes ( "idConsignes" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL , "temperatureJourMin" REAL NOT NULL, "temperatureJourMax" REAL NOT NULL, "temperatureNuitMin" REAL NOT NULL, "temperatureNuitMax" REAL NOT NULL, "hygrometrieSolMin" REAL, "hygrometrieSolMax" REAL, "hygrometrieAirMin" REAL, "hygrometrieAirMax" REAL ) ;
```

```
CREATE TABLE cultures ( "idCulture" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL , "nom" VARCHAR(255) NOT NULL , "description" TEXT NULL , "idConsignes" INTEGER NOT NULL , "idSeuils" INTEGER NOT NULL , CONSTRAINT fk_cultures_1 FOREIGN KEY (idSeuils) REFERENCES seuils (idSeuils) ON DELETE CASCADE, CONSTRAINT fk_cultures_2 FOREIGN KEY (idConsignes) REFERENCES consignes (idConsignes) ON DELETE CASCADE ) ;
```

```
CREATE TABLE mesures ( "idMesure" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL , "date" DATE NOT NULL, "heure" TIME NOT NULL, "temperatureAir" FLOAT NOT NULL, "temperatureSol" REAL, "hygrometrieAir" REAL, "hygrometrieSol" REAL, "vitesseVent" REAL, "directionVent" INTEGER, "ouvrant" INTEGER, "vanne" - INTEGER, "chauffage" INTEGER, "idCulture" INTEGER NOT NULL , CONSTRAINT fk_mesures_1 FOREIGN KEY (idCulture) REFERENCES cultures (idCulture) ON DELETE CASCADE ) ;
```

```
INSERT INTO seuils(temperatureMin,temperatureMax,vitesseMax) VALUES(-2';30';95') ; INSERT INTO consignes(temperatureJourMin,temperatureJourMax,temperatureNuitMin,temperatureNuitMax,hygrometrieSolMin,hygrometrieSolMax,hygrometrieAirMin,hygrometrieAirMax) VALUES('18';22';10';13';";70';80') ; INSERT INTO cultures(nom,description,idConsignes,idSeuils) VALUES('Fraise Ciflorette','Variété de printemps','1','1') ;
```

## 7 A propos

Auteur

Di Sario <[disario.laurent@gmail.com](mailto:disario.laurent@gmail.com)>



Version

1.0

Date

2018

## 8 Licence GPL

This program is free software ; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation ; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY ; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program ; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

## 9 Liste des choses à faire

Membre [IHM Campus Serre](#) : [actualiser](#) ()  
gérer l'onglet commande des appareils

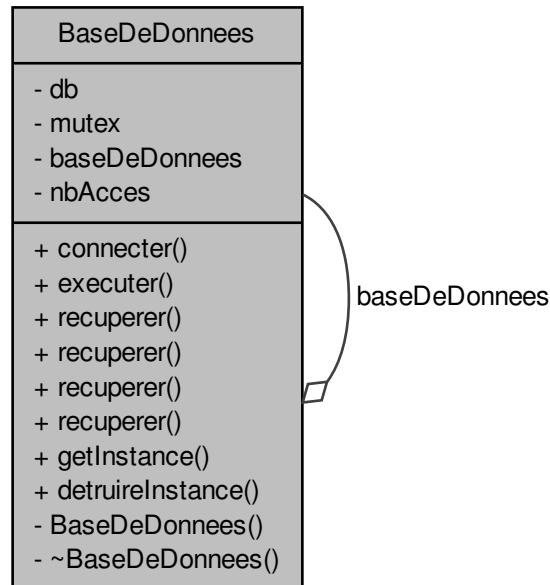
## 10 Documentation des classes

### 10.1 Référence de la classe BaseDeDonnees

Gestion la base de données SQLite.

```
#include <basededonnees.h>
```

Graphe de collaboration de BaseDeDonnees :



#### Fonctions membres publiques

- bool [connecter](#) (QString nomBase)
- 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](#) ()
- static void [detruireInstance](#) ()

#### Fonctions membres privées

- [BaseDeDonnees](#) ()
- [~BaseDeDonnees](#) ()

## Attributs privés

- QSqlDatabase [db](#)
- QMutex [mutex](#)

## Attributs privés statiques

- static [BaseDeDonnees](#) \* [baseDeDonnees](#) = NULL
- static int [nbAcces](#) = 0

## 10.1.1 Description détaillée

## Auteur

Thierry VAIRA, Di Sario

## Version

1.0

## 10.1.2 Documentation des constructeurs et destructeur

## 10.1.2.1 BaseDeDonnees : :BaseDeDonnees ( ) [private]

Références [db](#).

Référencé par [getInstance\(\)](#).

```
{  
    #ifdef DEBUG_BASEDEDONNEES  
    qDebug() << "<BaseDeDonnees::BaseDeDonnees()>";  
    #endif  
    db = QSqlDatabase::addDatabase("SQLITE");  
}
```

## 10.1.2.2 BaseDeDonnees : :~BaseDeDonnees ( ) [private]

```
{  
    #ifdef DEBUG_BASEDEDONNEES  
    qDebug() << "<BaseDeDonnees::~~BaseDeDonnees()>";  
    #endif  
}
```

## 10.1.3 Documentation des fonctions membres

10.1.3.1 bool BaseDeDonnees : :connecter ( QString *nomBase* )

Références [db](#), et [mutex](#).

Référencé par [IHMCampusSerre : :initialiserBDD\(\)](#).

```
{  
    QMutexLocker verrou(&mutex);  
    if(!db.isOpen())
```

```

{
    db.setDatabaseName(nomBase);

    if(db.open())
    {
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::connecter()>
connexion réussie à %1").arg(db.databaseName());
        #endif

        return true;
    }
    else
    {
        qDebug() << QString::fromUtf8("<BaseDeDonnees::connecter()> erreur :
impossible de se connecter à la base de données !");

        QMessageBox::critical(0, QString::fromUtf8("Campus Serre"),
        QString::fromUtf8("Impossible de se connecter à la base de données !"));

        return false;
    }
}
else
    return true;
}

```

### 10.1.3.2 void BaseDeDonnees : :detruiereInstance ( ) [static]

Références [baseDeDonnees](#), et [nbAcces](#).

Référencé par [IHMCampusSerre : :~IHMCampusSerre\(\)](#).

```

{
    // instance ?
    if(baseDeDonnees != NULL)
    {
        nbAcces--;
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << "<BaseDeDonnees::detruiereInstance()> nbAcces restants = " <
        < nbAcces;
        #endif
        // dernier ?
        if(nbAcces == 0)
            delete baseDeDonnees;
    }
}

```

### 10.1.3.3 bool BaseDeDonnees : :executer ( QString requete )

pour les requêtes UPDATE, INSERT et DELETE

Références [db](#), et [mutex](#).

Référencé par [IHMCampusSerre : :modifierConsignes\(\)](#), et [IHMCampusSerre : :modifierSeuils\(\)](#).

```

{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES

```

```

        qDebug() << QString::fromUtf8("<BaseDeDonnees::executer()> retour %1
pour la requete : %2").arg(QString::number(retour)).arg(requete);
    #endif
    if(retour)
    {
        return true;
    }
    else
    {
        qDebug() << QString::fromUtf8("<BaseDeDonnees::executer()> erreur :
%1 pour la requête %2").arg(r.lastError().text()).arg(requete);

        return false;
    }
}
else
    return false;
}

```

#### 10.1.3.4 BaseDeDonnees \* BaseDeDonnees : getInstance ( ) [static]

Références [BaseDeDonnees\(\)](#), [baseDeDonnees](#), et [nbAcces](#).

Référencé par [IHMCampusSerre : initialiserBDD\(\)](#).

```

{
    if(baseDeDonnees == NULL)
        baseDeDonnees = new BaseDeDonnees();

    nbAcces++;
    #ifdef DEBUG_BASEDEDONNEES
    qDebug() << "<BaseDeDonnees::getInstance()> nbAcces = " << nbAcces;
    #endif

    return baseDeDonnees;
}

```

#### 10.1.3.5 bool BaseDeDonnees : recuperer ( QString requete, QString & donnees )

pour les requêtes SELECT 1 -> 1

Références [db](#), et [mutex](#).

Référencé par [IHMCampusSerre : recupererConsignes\(\)](#), et [IHMCampusSerre : recupererSeuils\(\)](#).

```

{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
QString)> retour %1 pour la requete : %2").arg(QString::number(retour)).arg(requete);
        #endif
        if(retour)
        {
            // on se positionne sur l'enregistrement
            r.first();

            // on vérifie l'état de l'enregistrement retourné
            if(!r.isValid())

```

```

    {
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("
<BaseDeDonnees::recuperer(QString, QString)> résultat non valide !");
        #endif
        return false;
    }

    // on récupère sous forme de QString la valeur du champ
    if(r.isNull(0))
    {
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("
<BaseDeDonnees::recuperer(QString, QString)> résultat vide !");
        #endif
        return false;
    }
    donnees = r.value(0).toString();
    #ifdef DEBUG_BASEDEDONNEES
    qDebug() << "<BaseDeDonnees::recuperer(QString, QString)>
enregistrement -> " << donnees;
    #endif
    return true;
}
else
{
    qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
QString)> erreur : %1 pour la requête %2").arg(r.lastError().text()).arg(requete)
;

    return false;
}
}
else
    return false;
}

```

#### 10.1.3.6 bool BaseDeDonnees::recuperer ( QString *requete*, QStringList & *donnees* )

pour les requêtes SELECT 1 -> 1..\*

Références [db](#), et [mutex](#).

```

{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
QStringList)> retour %1 pour la requete : %2").arg(QString::number(retour)).arg(
requete);
        #endif
        if(retour)
        {
            // on se positionne sur l'enregistrement
            r.first();

            // on vérifie l'état de l'enregistrement retourné
            if(!r.isValid())
            {
                #ifdef DEBUG_BASEDEDONNEES
                qDebug() << QString::fromUtf8("
<BaseDeDonnees::recuperer(QString, QStringList)> résultat non valide !");
                #endif
                return false;
            }
        }
    }
}

```

```

        // on récupère sous forme de QString la valeur de tous les champs
        sélectionnés
        // et on les stocke dans une liste de QString
        for(int i=0;i<r.record().count();i++)
            if(!r.isNull(i))
                donnees << r.value(i).toString();
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << "<BaseDeDonnees::recuperer(QString, QStringList)>
enregistrement -> " << donnees;
        #endif
        return true;
    }
    else
    {
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
QStringList)> erreur : %1 pour la requête %2").arg(r.lastError().text()).arg(
requete);
        return false;
    }
}
else
    return false;
}

```

#### 10.1.3.7 bool BaseDeDonnees : :recuperer ( QString *requete*, QVector< QString > & *donnees* )

pour les requêtes SELECT 1..\* -> 1

Références [db](#), et [mutex](#).

```

{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;
    QString data;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
QVector<QString>> retour %1 pour la requete : %2").arg(QString::number(retour)).arg(
requete);
        #endif
        if(retour)
        {
            // pour chaque enregistrement
            while ( r.next() )
            {
                // on récupère sous forme de QString la valeur du champs
                sélectionné
                data = r.value(0).toString();

                #ifdef DEBUG_BASEDEDONNEES
                //qDebug() << "<BaseDeDonnees::recuperer(QString,
QVector<QString>> enregistrement -> " << data;
                #endif

                // on stocke l'enregistrement dans le QVector
                donnees.push_back(data);
            }
            #ifdef DEBUG_BASEDEDONNEES
            qDebug() << "<BaseDeDonnees::recuperer(QString, QVector<QString>>
enregistrement -> " << donnees;
            #endif
            return true;
        }
        else
        {

```

```

        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
        QVector<QString>> erreur : %1 pour la requête %2").arg(r.lastError().text()).arg
        (requete);

        return false;
    }
}
else
    return false;
}

```

#### 10.1.3.8 bool BaseDeDonnees : :recuperer ( QString *requete*, QVector< QStringList > & *donnees* )

pour les requêtes SELECT 1..\* -> 1..\*

Références [db](#), et [mutex](#).

```

{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;
    QStringList data;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
        QVector<QStringList>> retour %1 pour la requete : %2").arg(QString::number(retour)).
        arg(requete);
        #endif
        if(retour)
        {
            // pour chaque enregistrement
            while ( r.next() )
            {
                // on récupère sous forme de QString la valeur de tous les
                champs sélectionnés
                // et on les stocke dans une liste de QString
                for(int i=0;i<r.record().count();i++)
                    data << r.value(i).toString();

                #ifdef DEBUG_BASEDEDONNEES
                //qDebug() << "<BaseDeDonnees::recuperer(QString,
                QVector<QStringList>> enregistrement -> " << data;
                /*for(int i=0;i<r.record().count();i++)
                    qDebug() << r.value(i).toString();*/
                #endif

                // on stocke l'enregistrement dans le QVector
                donnees.push_back(data);

                // on efface la liste de QString pour le prochain
                enregistrement
                data.clear();
            }
            #ifdef DEBUG_BASEDEDONNEES
            qDebug() << "<BaseDeDonnees::recuperer(QString,
            QVector<QStringList>> enregistrement -> " << donnees;
            #endif
            return true;
        }
        else
        {
            qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
            QVector<QStringList>> erreur : %1 pour la requête %2").arg(r.lastError().text()).
            arg(requete);

            return false;
        }
    }
}

```



```

    }
    else
        return false;
}

```

#### 10.1.4 Documentation des données membres

**10.1.4.1 BaseDeDonnees \* BaseDeDonnees : :baseDeDonnees = NULL**  
[static, private]

Référencé par [destruireInstance\(\)](#), et [getInstance\(\)](#).

**10.1.4.2 QSqlDatabase BaseDeDonnees : :db** [private]

Référencé par [BaseDeDonnees\(\)](#), [connecter\(\)](#), [executer\(\)](#), et [recuperer\(\)](#).

**10.1.4.3 QMutex BaseDeDonnees : :mutex** [private]

Référencé par [connecter\(\)](#), [executer\(\)](#), et [recuperer\(\)](#).

**10.1.4.4 int BaseDeDonnees : :nbAcces = 0** [static, private]

Référencé par [destruireInstance\(\)](#), et [getInstance\(\)](#).

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

- [basededonnees.h](#)
- [basededonnees.cpp](#)

## 10.2 Référence de la classe GestionPort

Communique avec la serre via un port série virtuel.

```
#include <gestionPort.h>
```

#### Connecteurs publics

- void [receptionnerDonnees](#) ()  
*Réception des trames de la serre.*

#### Signaux

- void [nouvelleTrame](#) (QString trame)

#### Fonctions membres publiques

- [GestionPort](#) (QObject \*parent=0)  
*Constructeur : crée, paramètre et ouvre le port.*
- [~GestionPort](#) ()  
*Destructeur.*
- int [envoyerDonnees](#) (const QString &trame)

- *Envoie des trames a la serre.*  
bool [estOuvert](#) ()  
*Vérification si le port est ouvert.*

#### Fonctions membres privées

- void [creerPort](#) ()  
*Création du port.*
- void [parametrerPort](#) ()  
*Paramétrage du port.*
- void [ouvrirPort](#) ()  
*Ouverture du port.*
- bool [verifierTrame](#) (QString trame)  
*Vérifie si la trame passée en paramètre est une trame PCS complete lisible.*
- void [fermerPort](#) ()  
*Fermeture du port.*

#### Attributs privés

- QTextSerialPort \* [port](#)  
*relation vers la classe QTextSerialPort*

#### 10.2.1 Description détaillée

##### Auteur

Di Sario

##### Version

1.0

##### Date

Mercredi 21 février 2018

#### 10.2.2 Documentation des constructeurs et destructeur

##### 10.2.2.1 GestionPort : :GestionPort ( QObject \* *parent* = 0 )

##### Paramètres

<i>parent</i>	Adresse de l'objet parent (sinon 0)
---------------	-------------------------------------

Références [creerPort\(\)](#), [ouvrirPort\(\)](#), et [parametrerPort\(\)](#).

```

                                : QObject (parent)
{
    creerPort ();
    parametrerPort ();
    ouvrirPort ();
}

```

### 10.2.2.2 GestionPort : :~GestionPort ( )

```
{  
}
```

### 10.2.3 Documentation des fonctions membres

#### 10.2.3.1 void GestionPort : :creerPort ( ) [private]

Références [PORT](#), et [port](#).

Référencé par [GestionPort\(\)](#).

```
{  
    port = new QextSerialPort(QLatin1String(PORT), QextSerialPort::EventDriven,  
                             this);  
}
```

#### 10.2.3.2 int GestionPort : :envoyerDonnees ( const QString & trame )

Références [port](#).

Référencé par [IHMCampusSerre : :commanderChauffage\(\)](#), [IHMCampusSerre : :commanderOuvrant\(\)](#), [IHMCampusSerre : :commanderVanne\(\)](#), [IHMCampusSerre : :envoyerConsignes\(\)](#), et [IHMCampusSerre : :envoyerSeuils\(\)](#).

```
{  
    if (port == NULL || !port->isOpen())  
    {  
        return -1;  
    }  
  
    qDebug() << Q_FUNC_INFO << trame;  
    return port->write(trame.toLatin1());  
}
```

#### 10.2.3.3 bool GestionPort : :estOuvert ( )

Références [port](#).

Référencé par [IHMCampusSerre : :IHMCampusSerre\(\)](#).

```
{  
    if (port != NULL)  
    {  
        return port->isOpen();  
    }  
    return false;  
}
```

#### 10.2.3.4 void GestionPort : :fermerPort ( ) [private]

Références [port](#).

```
{
    if(port->isOpen())
    {
        port->close();
        disconnect(port, SIGNAL(readyRead()), this, SLOT(recevoir()));
    }
}
```

#### 10.2.3.5 void GestionPort : :nouvelleTrame ( QString trame ) [signal]

Référencé par [receptionnerDonnees\(\)](#).

#### 10.2.3.6 void GestionPort : :ouvrirPort ( ) [private]

Références [PORT](#), [port](#), et [receptionnerDonnees\(\)](#).

Référencé par [GestionPort\(\)](#).

```
{
    port->open(QIODevice::ReadWrite | QIODevice::Unbuffered);
    if(port->isOpen())
    {
        connect(port, SIGNAL(readyRead()), this, SLOT(receptionnerDonnees()));
    }
    else
    {
        qDebug() << Q_FUNC_INFO << "Port" << QLatin1String(PORT) << "non ouvert";
    }
}
```

#### 10.2.3.7 void GestionPort : :parametrerPort ( ) [private]

Références [port](#).

Référencé par [GestionPort\(\)](#).

```
{
    port->setBaudRate(BAUD9600);
    port->setDataBits(DATA_8);
    port->setStopBits(STOP_1);
}
```

#### 10.2.3.8 void GestionPort : :receptionnerDonnees ( ) [slot]

Références [nouvelleTrame\(\)](#), [port](#), et [verifierTrame\(\)](#).

Référencé par [ouvrirPort\(\)](#).

```
{
    QByteArray donneesRecues;
    while(port->bytesAvailable())
    {
        donneesRecues += port->readAll();
        usleep(100000);
    }

    QStringList listeDonnees;
    QString donnees(donneesRecues);
}
```

```

listeDonnees = donnees.split("$",QString::SkipEmptyParts);

for(int i = 0; i < listeDonnees.length(); i++)
{
    qDebug() << Q_FUNC_INFO << '$' + listeDonnees.at(i);
    if(verifierTrame('$' + listeDonnees.at(i)))
    {
        emit nouvelleTrame('$' + listeDonnees.at(i));
    }
}
}

```

### 10.2.3.9 bool GestionPort : :verifierTrame ( QString trame ) [private]

#### Paramètres

<i>trame</i>	QString La trame à verifier
--------------	-----------------------------

#### Renvoie

true si la trame est valide sinon false

Référencé par [receptionnerDonnees\(\)](#).

```

{
    QString checksum;
    const QString debutTrame = "$";
    const QString typeTrame = "PCS";
    const QString debutChecksum = "*";

    /*
    //Exemple de trame
    //QString trame = "$PCS.000.235.220.800.600.020.001.025.000.001*";
    */
    // phrase vide ?
    if(trame.length() != 0)
    {
        // est-ce une phrase PCS ?
        if(trame.startsWith(debutTrame))
        {
            // est-ce la bonne phrase ?
            if(trame.startsWith(debutTrame + typeTrame))
            {
                // y-a-t-il un checksum ?
                if(trame.contains(debutChecksum))
                {
                    checksum = trame.section(debutChecksum, 1, 1);
                    qDebug() << Q_FUNC_INFO << "checksum : 0x" << checksum;
                    return 1;
                }
                else
                {
                    qDebug() << Q_FUNC_INFO << "Attention : il n'y a pas de
checksum dans cette phrase !";
                    return 0;
                }
            }
            else
            {
                qDebug() << Q_FUNC_INFO << "Erreur : ce n'est pas une trame PCS
!";
                return 0;
            }
        }
        else
        {
            qDebug() << Q_FUNC_INFO << "Erreur : ce n'est pas une trame PCS !";
            return 0;
        }
    }
    else

```

```
qDebug() << Q_FUNC_INFO << "Erreur : phrase vide !";  
return 0;  
}
```

#### 10.2.4 Documentation des données membres

##### 10.2.4.1 QTextSerialPort\* GestionPort : :port [private]

Référencé par [creerPort\(\)](#), [envoyerDonnees\(\)](#), [estOuvert\(\)](#), [fermerPort\(\)](#), [ouvrirPort\(\)](#), [parametrerPort\(\)](#), et [receptionnerDonnees\(\)](#).

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

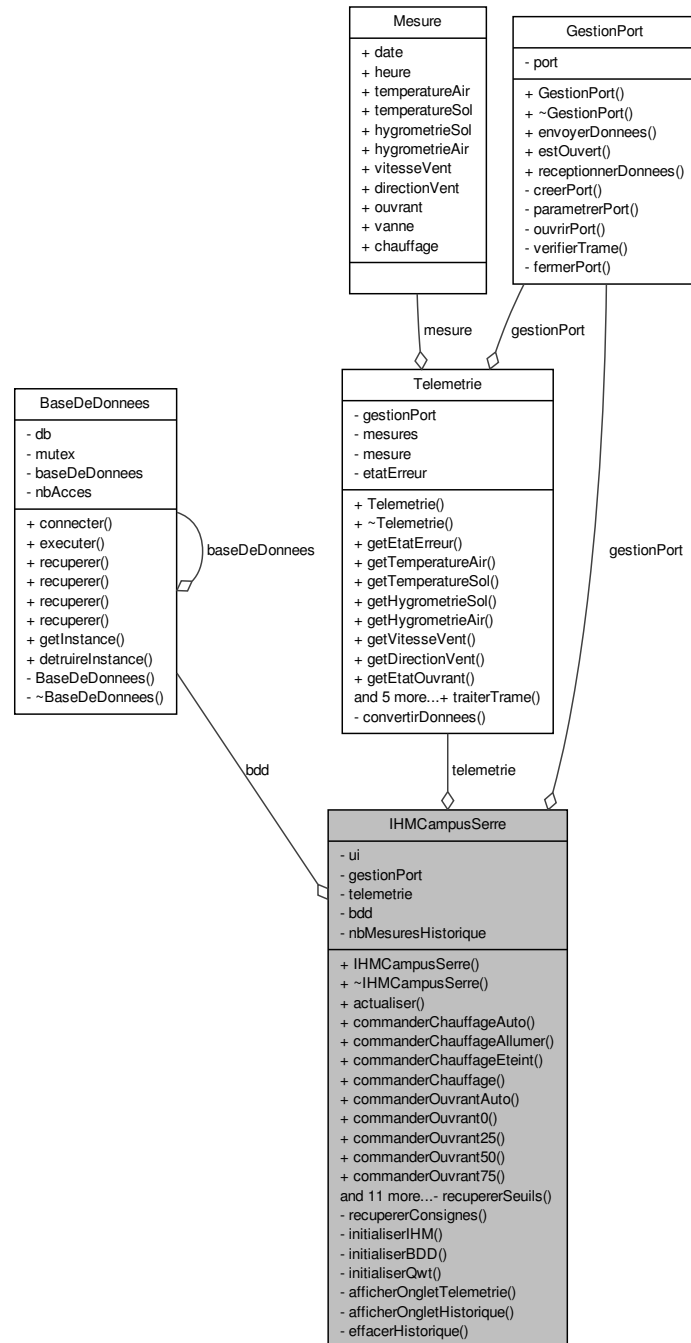
- [gestionPort.h](#)
- [gestionPort.cpp](#)

### 10.3 Référence de la classe IHMCampusSerre

La fenêtre principale de l'application.

```
#include <IHMCampusSerre.h>
```

Graphe de collaboration de IHMCampusSerre :



## Connecteurs publics

- void [actualiser](#) ()  
*Rafrachit l'affichage des données dans l'IHM.*
- void [commanderChauffageAuto](#) ()  
*Créer la commande pour passer le chauffage en automatique.*
- void [commanderChauffageAllumer](#) ()  
*Créer la commande pour allumer le chauffage.*
- void [commanderChauffageEteint](#) ()  
*Créer la commande pour éteindre le chauffage.*
- void [commanderChauffage](#) (int chauffage)  
*Créer la trame et la passe à la méthode envoyerDonnees.*
- void [commanderOuvrantAuto](#) ()  
*Créer la commande pour passer l'ouvrant en automatique.*
- void [commanderOuvrant0](#) ()  
*Créer la commande pour fermer l'ouvrant.*
- void [commanderOuvrant25](#) ()  
*Créer la commande pour ouvrir l'ouvrant à 25%.*
- void [commanderOuvrant50](#) ()  
*Créer la commande pour ouvrir l'ouvrant à 50%.*
- void [commanderOuvrant75](#) ()  
*Créer la commande pour ouvrir l'ouvrant à 75μ*
- void [commanderOuvrant100](#) ()  
*Créer la commande pour ouvrir l'ouvrant à 100%.*
- void [commanderOuvrant](#) (int ouverture)  
*Créer la trame et la passe à la méthode envoyerDonnees.*
- void [commanderVanneAuto](#) ()  
*Créer la commande pour passer la vanne en automatique.*
- void [commanderVanneOuvert](#) ()  
*Créer la commande pour ouvrir la vanne.*
- void [commanderVanneFermer](#) ()  
*Créer la commande pour fermer la vanne.*
- void [commanderVanne](#) (int vanne)  
*Créer la trame et la passe à la méthode envoyerDonnees.*
- void [selectionnerOnglet](#) (int onglet)
- void [modifierSeuils](#) ()  
*Enregistre les nouveaux seuils dans la base de donnée et envoie les nouveaux seuils.*
- void [modifierConsignes](#) ()  
*Enregistre les nouvelles consignes dans la base de donnée.*
- void [envoyerSeuils](#) (int seuilTempMin, int seuilTempMax, int seuilVitesseMax)  
*Créer la trame et la passe à la méthode envoyerDonnees.*
- void [envoyerConsignes](#) (int consigneTempMinJour, int consigneTempMaxJour, int consigneTempMinNuit, int consigneTempMaxNuit, int consigneHygroMinSol, int consigneHygroMaxSol, int consigneHygroMinAir, int consigneHygroMaxAir)  
*Créer la trame et la passe à la méthode envoyerDonnees.*
- void [modifierNbMesures](#) (int n)  
*Modifie le nombre de mesure afficher dans l'historique.*

## Fonctions membres publiques

- [IHMCampusSerre](#) (QWidget \*parent=0)  
*Lance l'ihm, initialise la base de donnée, récupère les seuils et les consignes dans la base de donnée.*
- [~IHMCampusSerre](#) ()  
*Destructeur.*



## Fonctions membres privées

- void [recupererSeuils](#) ()  
*Récupère les seuils de la base de donnée et les affiche dans l'onglet Culture.*
- void [recupererConsignes](#) ()  
*Récupère les consignes de la base de donnée et les affiche dans l'onglet Culture.*
- void [initialiserIHM](#) ()  
*initialise les widgets Qwt (thermomètre, ...)*
- void [initialiserBDD](#) ()  
*Initialise la base de donnée et s'y connecte.*
- void [initialiserQwt](#) ()  
*Initialise le compteur de vitesse et le compas.*
- void [afficherOngletTelemetrie](#) ()  
*Affiche les mesures dans l'onglet Télémétrie.*
- void [afficherOngletHistorique](#) ()  
*Affiche les valeurs dans l'onglet historique.*
- void [effacerHistorique](#) ()  
*Efface l'historique.*

## Attributs privés

- Ui : :IHMCampusSerre \* [ui](#)
- [GestionPort](#) \* [gestionPort](#)
- [Telemetrie](#) \* [telemetrie](#)
- [BaseDeDonnees](#) \* [bdd](#)
- int [nbMesuresHistorique](#)

## 10.3.1 Description détaillée

## Auteur

Di Sario

## Version

1.0

## Date

Mercredi 21 février 2018

## 10.3.2 Documentation des constructeurs et destructeur

10.3.2.1 IHMCampusSerre : :IHMCampusSerre ( QWidget \* *parent* = 0 )  
[explicit]

## Paramètres

<i>parent</i>	
---------------	--

Références [GestionPort](#) : [:estOuvert\(\)](#), [gestionPort](#), [initialiserBDD\(\)](#), [initialiserIHM\(\)](#), [recupererConsignes\(\)](#), [recupererSeuils\(\)](#), [telemetrie](#), et [ui](#).

```
QMainWindow (parent) ,
```

```

    ui(new Ui::IHMCampusSerre)
{
    ui->setupUi(this);

    initialiserBDD();

    recupererSeuils();
    recupererConsignes();

    gestionPort = new GestionPort(this);
    telemetrie = new Telemetrie(gestionPort, this);

    initialiserIHM();

    if(!gestionPort->estOuvert())
        QMessageBox::critical(0, QString::fromUtf8("Campus Serre"),
            QString::fromUtf8("Le port de communication n'est pas ouvert !"));
}

```

#### 10.3.2.2 IHMCampusSerre : ~IHMCampusSerre ( )

Références [BaseDeDonnees : :destruireInstance\(\)](#), et [ui](#).

```

{
    BaseDeDonnees::destruireInstance();
    delete ui;
}

```

### 10.3.3 Documentation des fonctions membres

#### 10.3.3.1 IHMCampusSerre : :actualiser ( ) [slot]

**A faire** gérer l'onglet commande des appareils

Références [afficherOngletHistorique\(\)](#), [afficherOngletTelemetrie\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```

{
    // En fonction de l'onglet sélectionné
    if(ui->onglets->currentIndex() == 0)
    {
        afficherOngletTelemetrie();
        afficherOngletHistorique();
    }
    if(ui->onglets->currentIndex() == 3)
        afficherOngletHistorique();
}

```

#### 10.3.3.2 IHMCampusSerre : :afficherOngletHistorique ( ) [private]

Références [effacerHistorique\(\)](#), [Telemetrie : :getMesures\(\)](#), [nbMesuresHistorique](#), [telemetrie](#), et [ui](#).

Référencé par [actualiser\(\)](#), et [modifierNbMesures\(\)](#).

```

{
    effacerHistorique();
}

```

```

// Récupérer les mesures (dans Telemetrie)
QVector<Mesure> mesures = telemetrie->getMesures();

// les insérer dans le tableau
QTableWidgetItem *itemDate, *itemHeure, *itemHygroAir, *itemHygroSol, *
    itemVitesseVent, *itemDirectionVent, *itemOuvrant, *itemVanne, *itemChauffage, *
    itemTemperatureAir, *itemTemperatureSol;
int nbMesures = 0;
for(int i = mesures.count()-1; i > -1; i--)
{
    itemDate = new QTableWidgetItem(mesures.at(i).date);
    itemDate->setFlags(Qt::ItemIsEnabled);
    itemDate->setTextAlignment(Qt::AlignHCenter);

    itemHeure = new QTableWidgetItem(mesures.at(i).heure);
    itemHeure->setFlags(Qt::ItemIsEnabled);
    itemHeure->setTextAlignment(Qt::AlignHCenter);

    itemHygroSol = new QTableWidgetItem(QString::number(mesures.at(i).
        hygrometrieSol));
    itemHygroSol->setFlags(Qt::NoItemFlags);
    itemHygroSol->setTextAlignment(Qt::AlignHCenter);

    itemHygroAir = new QTableWidgetItem(QString::number(mesures.at(i).
        hygrometrieAir));
    itemHygroAir->setFlags(Qt::NoItemFlags);
    itemHygroAir->setTextAlignment(Qt::AlignHCenter);

    itemVitesseVent = new QTableWidgetItem(QString::number(mesures.at(i).
        vitesseVent));
    itemVitesseVent->setFlags(Qt::NoItemFlags);
    itemVitesseVent->setTextAlignment(Qt::AlignHCenter);

    itemDirectionVent = new QTableWidgetItem(QString::number(mesures.at(i).
        directionVent));
    itemDirectionVent->setFlags(Qt::NoItemFlags);
    itemDirectionVent->setTextAlignment(Qt::AlignHCenter);

    itemOuvrant = new QTableWidgetItem(QString::number(mesures.at(i).
        ouvrant));
    itemOuvrant->setFlags(Qt::NoItemFlags);
    itemOuvrant->setTextAlignment(Qt::AlignHCenter);

    itemVanne = new QTableWidgetItem(QString::number(mesures.at(i).vanne));
    itemVanne->setFlags(Qt::NoItemFlags);
    itemVanne->setTextAlignment(Qt::AlignHCenter);

    itemChauffage = new QTableWidgetItem(QString::number(mesures.at(i).
        chauffage));
    itemChauffage->setFlags(Qt::NoItemFlags);
    itemChauffage->setTextAlignment(Qt::AlignHCenter);

    itemTemperatureAir = new QTableWidgetItem(QString::number(mesures.at(i).
        temperatureAir));
    itemTemperatureAir->setFlags(Qt::NoItemFlags);
    itemTemperatureAir->setTextAlignment(Qt::AlignHCenter);

    itemTemperatureSol = new QTableWidgetItem(QString::number(mesures.at(i).
        temperatureSol));
    itemTemperatureSol->setFlags(Qt::NoItemFlags);
    itemTemperatureSol->setTextAlignment(Qt::AlignHCenter);

    int nb = ui->tableauMesures->rowCount();
    ui->tableauMesures->setRowCount(++nb);
    ui->tableauMesures->setItem(nb-1, 0, itemDate);
    ui->tableauMesures->setItem(nb-1, 1, itemHeure);
    ui->tableauMesures->setItem(nb-1, 2, itemHygroSol);
    ui->tableauMesures->setItem(nb-1, 3, itemHygroAir);
    ui->tableauMesures->setItem(nb-1, 4, itemVitesseVent);
    ui->tableauMesures->setItem(nb-1, 5, itemDirectionVent);
    ui->tableauMesures->setItem(nb-1, 6, itemOuvrant);
    ui->tableauMesures->setItem(nb-1, 7, itemVanne);
    ui->tableauMesures->setItem(nb-1, 8, itemChauffage);
    ui->tableauMesures->setItem(nb-1, 9, itemTemperatureAir);
}

```

```

        ui->tableauMesures->setItem(nb-1, 10, itemTemperatureSol);

        ui->tableauMesures->scrollToItem(ui->tableauMesures->item(0, 1));
        ui->tableauMesures->setCurrentCell(0, 1);

        ++nbMesures;
        if (nbMesures == nbMesuresHistorique)
            break;
    }
}

```

#### 10.3.3.3 IHMCampusSerre : :afficherOngletTelemetrie ( ) [private]

Affichage des données dans l'onglet [Telemetrie](#).

Références [Telemetrie : :getDirectionVent\(\)](#), [Telemetrie : :getEtatChauffage\(\)](#), [Telemetrie : :getEtatOuvrant\(\)](#), [Telemetrie : :getEtatVanne\(\)](#), [Telemetrie : :getHygrometrieAir\(\)](#), [Telemetrie : :getHygrometrieSol\(\)](#), [Telemetrie : :getTemperatureAir\(\)](#), [Telemetrie : :getTemperatureSol\(\)](#), [Telemetrie : :getVitesseVent\(\)](#), [telemetrie](#), et [ui](#).

Référencé par [actualiser\(\)](#).

```

{
    ui->temperatureAir->display(telemetrie->getTemperatureAir());
    ui->temperatureSol->display(telemetrie->getTemperatureSol());
    ui->hygrometrieAir->display(telemetrie->getHygrometrieAir());
    ui->hygrometrieSol->display(telemetrie->getHygrometrieSol());
    ui->ouvertureOuvrant->setValue(telemetrie->getEtatOuvrant());
    ui->etatChauffage->setValue(telemetrie->getEtatChauffage());
    ui->etatVanne->setValue(telemetrie->getEtatVanne());
    ui->Compass->setValue(telemetrie->getDirectionVent());
    ui->Vitesse->setValue(telemetrie->getVitesseVent());
}

```

#### 10.3.3.4 IHMCampusSerre : :commanderChauffage ( int *chauffage* ) [slot]

##### Paramètres

<i>chauffage</i>	int la commande a envoyer
------------------	---------------------------

Références [GestionPort : :envoyerDonnees\(\)](#), et [gestionPort](#).

Référencé par [commanderChauffageAllumer\(\)](#), [commanderChauffageAuto\(\)](#), et [commanderChauffageEteint\(\)](#).

```

{
    QString trame = QString("$PCS.1.%1.*").arg(chauffage, 3, 10, QChar('0'));
    gestionPort->envoyerDonnees(trame);
}

```

#### 10.3.3.5 IHMCampusSerre : :commanderChauffageAllumer ( ) [slot]

Références [commanderChauffage\(\)](#), et [ui](#).

```

{
    commanderChauffage(001);
    ui->boutonChauffageAuto->setDefault(0);
    ui->boutonChauffageAllumer->setDefault(1);
    ui->boutonChauffageEteint->setDefault(0);
}

```

## 10.3.3.6 IHMCampusSerre : :commanderChauffageAuto ( ) [slot]

Références [commanderChauffage\(\)](#), et [ui](#).

```
{
    commanderChauffage(111);
    ui->boutonChauffageAuto->setDefault(1);
    ui->boutonChauffageAllumer->setDefault(0);
    ui->boutonChauffageEteint->setDefault(0);
}
```

## 10.3.3.7 IHMCampusSerre : :commanderChauffageEteint ( ) [slot]

Références [commanderChauffage\(\)](#), et [ui](#).

```
{
    commanderChauffage(000);
    ui->boutonChauffageAuto->setDefault(0);
    ui->boutonChauffageAllumer->setDefault(0);
    ui->boutonChauffageEteint->setDefault(1);
}
```

## 10.3.3.8 IHMCampusSerre : :commanderOuvrant ( int ouverture ) [slot]

## Paramètres

<i>ouverture</i>	int la commande a envoyer
------------------	---------------------------

Références [GestionPort : :envoyerDonnees\(\)](#), et [gestionPort](#).

Référencé par [commanderOuvrant0\(\)](#), [commanderOuvrant100\(\)](#), [commanderOuvrant25\(\)](#), [commanderOuvrant50\(\)](#), [commanderOuvrant75\(\)](#), et [commanderOuvrantAuto\(\)](#).

```
{
    QString trame = QString("$PCS.2.%1.*").arg(ouverture, 3, 10, QChar('0'));
    gestionPort->envoyerDonnees(trame);
}
```

## 10.3.3.9 IHMCampusSerre : :commanderOuvrant0 ( ) [slot]

Références [commanderOuvrant\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```
{
    commanderOuvrant(0);
    ui->boutonOuvrant0->setDefault(1);
    ui->boutonOuvrant25->setDefault(0);
    ui->boutonOuvrant50->setDefault(0);
    ui->boutonOuvrant75->setDefault(0);
    ui->boutonOuvrant100->setDefault(0);
    ui->boutonAutoOuvrant->setDefault(0);
    ui->boutonManuelOuvrant->setDefault(1);
}
```

**10.3.3.10 IHMCampusSerre : :commanderOuvrant100 ( ) [slot]**

Références [commanderOuvrant\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```
{
    commanderOuvrant(100);
    ui->boutonOuvrant0->setDefault(0);
    ui->boutonOuvrant25->setDefault(0);
    ui->boutonOuvrant50->setDefault(0);
    ui->boutonOuvrant75->setDefault(0);
    ui->boutonOuvrant100->setDefault(1);
    ui->boutonAutoOuvrant->setDefault(0);
    ui->boutonManuelOuvrant->setDefault(1);
}
```

**10.3.3.11 IHMCampusSerre : :commanderOuvrant25 ( ) [slot]**

Références [commanderOuvrant\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```
{
    commanderOuvrant(25);
    ui->boutonOuvrant0->setDefault(0);
    ui->boutonOuvrant25->setDefault(1);
    ui->boutonOuvrant50->setDefault(0);
    ui->boutonOuvrant75->setDefault(0);
    ui->boutonOuvrant100->setDefault(0);
    ui->boutonAutoOuvrant->setDefault(0);
    ui->boutonManuelOuvrant->setDefault(1);
}
```

**10.3.3.12 IHMCampusSerre : :commanderOuvrant50 ( ) [slot]**

Références [commanderOuvrant\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```
{
    commanderOuvrant(50);
    ui->boutonOuvrant0->setDefault(0);
    ui->boutonOuvrant25->setDefault(0);
    ui->boutonOuvrant50->setDefault(1);
    ui->boutonOuvrant75->setDefault(0);
    ui->boutonOuvrant100->setDefault(0);
    ui->boutonAutoOuvrant->setDefault(0);
    ui->boutonManuelOuvrant->setDefault(1);
}
```

**10.3.3.13 IHMCampusSerre : :commanderOuvrant75 ( ) [slot]**

Références [commanderOuvrant\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```
{
    commanderOuvrant(75);
    ui->boutonOuvrant0->setDefault(0);
    ui->boutonOuvrant25->setDefault(0);
}
```

```

    ui->boutonOuvrant50->setDefault(0);
    ui->boutonOuvrant75->setDefault(1);
    ui->boutonOuvrant100->setDefault(0);
    ui->boutonAutoOuvrant->setDefault(0);
    ui->boutonManuelOuvrant->setDefault(1);
}

```

#### 10.3.3.14 IHMCampusSerre : :commanderOuvrantAuto ( ) [slot]

Références [commanderOuvrant\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```

{
    commanderOuvrant(111);
    ui->boutonOuvrant0->setDefault(0);
    ui->boutonOuvrant25->setDefault(0);
    ui->boutonOuvrant50->setDefault(0);
    ui->boutonOuvrant75->setDefault(0);
    ui->boutonOuvrant100->setDefault(0);
    ui->boutonAutoOuvrant->setDefault(1);
    ui->boutonManuelOuvrant->setDefault(0);
}

```

#### 10.3.3.15 IHMCampusSerre : :commanderVanne ( int *vanne* ) [slot]

Paramètres

<i>vanne</i>	int la commande a envoyer
--------------	---------------------------

Références [GestionPort : :envoyerDonnees\(\)](#), et [gestionPort](#).

Référencé par [commanderVanneAuto\(\)](#), [commanderVanneFermer\(\)](#), et [commanderVanneOuvert\(\)](#).

```

{
    QString trame = QString("$PCS.3.%1.*").arg(vanne, 3, 10, QChar('0'));
    qDebug() << trame;
    gestionPort->envoyerDonnees(trame);
}

```

#### 10.3.3.16 IHMCampusSerre : :commanderVanneAuto ( ) [slot]

Références [commanderVanne\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```

{
    commanderVanne(111);
    ui->boutonAutoVanne->setDefault(1);
    ui->boutonOuvertureVanne->setDefault(0);
    ui->boutonFermetureVanne->setDefault(0);
}

```

#### 10.3.3.17 IHMCampusSerre : :commanderVanneFermer ( ) [slot]

Références [commanderVanne\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```
{
    commanderVanne(0);
    ui->boutonAutoVanne->setDefault(0);
    ui->boutonOuvertureVanne->setDefault(0);
    ui->boutonFermetureVanne->setDefault(1);
}
```

#### 10.3.3.18 IHMCampusSerre : :commanderVanneOuvert ( ) [slot]

Références [commanderVanne\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```
{
    commanderVanne(1);
    ui->boutonAutoVanne->setDefault(0);
    ui->boutonOuvertureVanne->setDefault(1);
    ui->boutonFermetureVanne->setDefault(0);
}
```

#### 10.3.3.19 IHMCampusSerre : :effacerHistorique ( ) [private]

Références [ui](#).

Référencé par [afficherOngletHistorique\(\)](#).

```
{
    int nb = ui->tableauMesures->rowCount();
    // on efface les lignes du tableau une par une
    for(int i = 0; i < nb; i++)
    {
        ui->tableauMesures->removeRow(0);
    }
}
```

#### 10.3.3.20 IHMCampusSerre : :envoyerConsignes ( int consigneTempMinJour, int consigneTempMaxJour, int consigneTempMinNuit, int consigneTempMaxNuit, int consigneHygroMinSol, int consigneHygroMaxSol, int consigneHygroMinAir, int consigneHygroMaxAir ) [slot]

##### Paramètres

<i>consigne-TempMin-Jour</i>	int la nouvelle valeur pour la température minimum du jour
<i>consigne-TempMax-Jour</i>	int la nouvelle valeur pour la température maximum du jour
<i>consigne-TempMin-Nuit</i>	int la nouvelle valeur pour la température minimum de la nuit
<i>consigne-TempMax-Nuit</i>	int la nouvelle valeur pour la température maximum de la nuit
<i>consigne-HygroMinSol</i>	int la nouvelle valeur pour l'hygrométrie minimum du sol



<i>consigne-HygroMax-Sol</i>	int la nouvelle valeur pour l'hygrométrie maximum du sol
<i>consigne-HygroMinAir</i>	int la nouvelle valeur pour l'hygrométrie minimum de l'air
<i>consigne-HygroMax-Air</i>	int la nouvelle valeur pour l'hygrométrie maximum de l'air

Références [GestionPort : :envoyerDonnees\(\)](#), et [gestionPort](#).

Référencé par [modifierConsignes\(\)](#).

```
{
    QString trame = QString("$PCS.5.%1.%2.%3.%4.%5.%6.%7.%8*").arg(
        consigneTempMinJour, 3, 10, QChar('0')).arg(consigneTempMaxJour, 3, 10, QChar('0')).arg(
        consigneTempMinNuit, 3, 10, QChar('0')).arg(consigneTempMaxNuit, 3, 10, QChar('0')).
        arg(consigneHygroMinSol, 3, 10, QChar('0')).arg(consigneHygroMaxSol, 3, 10,
        QChar('0')).arg(consigneHygroMinAir, 3, 10, QChar('0')).arg(consigneHygroMaxAir, 3,
        10, QChar('0'));
    qDebug() << trame;
    gestionPort->envoyerDonnees(trame);
}
```

#### 10.3.3.21 IHMCampusSerre : :envoyerSeuils ( int *seuilTempMin*, int *seuilTempMax*, int *seuilVitesseMax* ) [slot]

##### Paramètres

<i>seuilTemp-Min</i>	int la nouvelle valeur pour la température minimum
<i>seuilTemp-Max</i>	int la nouvelle valeur pour la température maximum
<i>seuilVitesse-Max</i>	int la nouvelle valeur pour la vitesse maximum

Références [GestionPort : :envoyerDonnees\(\)](#), et [gestionPort](#).

Référencé par [modifierSeuils\(\)](#).

```
{
    QString trame = QString("$PCS.4.%1.%2.%3.*").arg(seuilTempMin, 3, 10, QChar
        ('0')).arg(seuilTempMax, 3, 10, QChar('0')).arg(seuilVitesseMax, 3, 10, QChar(
        '0'));
    qDebug() << Q_FUNC_INFO << trame;
    gestionPort->envoyerDonnees(trame);
}
```

#### 10.3.3.22 IHMCampusSerre : :initialiserBDD ( ) [private]

Références [bdd](#), [BaseDeDonnees : :connecter\(\)](#), et [BaseDeDonnees : :getInstance\(\)](#).

Référencé par [IHMCampusSerre\(\)](#).

```
{
```

```

bdd = BaseDeDonnees::getInstance();
QString database = qApp->applicationDirPath() + "/" + qApp->applicationName
() + ".db";
bdd->connecter(database);
}

```

### 10.3.3.23 IHMCampusSerre :initialiserIHM( ) [private]

Références [actualiser\(\)](#), [commanderOuvrant0\(\)](#), [commanderOuvrant100\(\)](#), [commanderOuvrant25\(\)](#), [commanderOuvrant50\(\)](#), [commanderOuvrant75\(\)](#), [commanderOuvrantAuto\(\)](#), [commanderVanneAuto\(\)](#), [commanderVanneFermer\(\)](#), [commanderVanneOuvert\(\)](#), [initialiserQwt\(\)](#), [modifierConsignes\(\)](#), [modifierNbMesures\(\)](#), [modifierSeuils\(\)](#), [nbMesuresHistorique](#), [selectionnerOnglet\(\)](#), [telemetrie](#), et [ui](#).

Référencé par [IHMCampusSerre\(\)](#).

```

{
    nbMesuresHistorique = ui->spinBoxNbMesures->value();
    ui->tableauMesures->resizeColumnsToContents();
    ui->tableauMesures->verticalHeader()->setHidden(true);
    QHeaderView * headerView = ui->tableauMesures->horizontalHeader();
    headerView->setResizeMode(QHeaderView::Stretch);

    QAction *actionQuitter = new QAction("&Quitter", this);
    actionQuitter->setShortcut(QKeySequence(QKeySequence::Quit)); // Ctrl+Q
    addAction(actionQuitter);
    connect(actionQuitter, SIGNAL(triggered()), qApp, SLOT(quit()));

    connect(ui->onglets, SIGNAL(currentChanged(int)), this, SLOT(
        selectionnerOnglet(int)));
    connect(ui->boutonAutoOuvrant, SIGNAL(clicked()), this, SLOT(
        commanderOuvrantAuto()));
    connect(ui->boutonOuvrant0, SIGNAL(clicked()), this, SLOT(commanderOuvrant0
        ()));
    connect(ui->boutonOuvrant25, SIGNAL(clicked()), this, SLOT(
        commanderOuvrant25()));
    connect(ui->boutonOuvrant50, SIGNAL(clicked()), this, SLOT(
        commanderOuvrant50()));
    connect(ui->boutonOuvrant75, SIGNAL(clicked()), this, SLOT(
        commanderOuvrant75()));
    connect(ui->boutonOuvrant100, SIGNAL(clicked()), this, SLOT(
        commanderOuvrant100()));
    connect(ui->boutonAutoVanne, SIGNAL(clicked()), this, SLOT(commanderVanneAuto
        ()));
    connect(ui->boutonOuvertureVanne, SIGNAL(clicked()), this, SLOT(
        commanderVanneOuvert()));
    connect(ui->boutonFermetureVanne, SIGNAL(clicked()), this, SLOT(
        commanderVanneFermer()));
    connect(ui->EditTempMin, SIGNAL(editingFinished()), this, SLOT(
        modifierSeuils()));
    connect(ui->EditTempMax, SIGNAL(editingFinished()), this, SLOT(
        modifierSeuils()));
    connect(ui->EditVitMax, SIGNAL(editingFinished()), this, SLOT(modifierSeuils
        ()));
    connect(ui->EditHygroMinAir, SIGNAL(editingFinished()), this, SLOT(
        modifierConsignes()));
    connect(ui->EditHygroMinSol, SIGNAL(editingFinished()), this, SLOT(
        modifierConsignes()));
    connect(ui->EditHygroMaxAir, SIGNAL(editingFinished()), this, SLOT(
        modifierConsignes()));
    connect(ui->EditHygroMaxSol, SIGNAL(editingFinished()), this, SLOT(
        modifierConsignes()));
    connect(ui->EditTempMinNuit, SIGNAL(editingFinished()), this, SLOT(
        modifierConsignes()));
    connect(ui->EditTempMinJour, SIGNAL(editingFinished()), this, SLOT(
        modifierConsignes()));
    connect(ui->EditTempMaxNuit, SIGNAL(editingFinished()), this, SLOT(
        modifierConsignes()));
    connect(ui->EditTempMaxJour, SIGNAL(editingFinished()), this, SLOT(

```

```

        modifierConsignes());
connect(ui->spinBoxNbMesures, SIGNAL(valueChanged(int)), this, SLOT(
    modifierNbMesures(int)));
connect(telemetrie, SIGNAL(nouvellesDonnees()), this, SLOT(actualiser()));

initialiserQwt();

// Affichage Telemetrie
ui->onglets->setCurrentIndex(0);
}

```

#### 10.3.3.24 IHMCampusSerre : :initialiserQwt ( ) [private]

initialise les widgets Qwt (thermomètre, ...)

Références [ui](#).

Référéncé par [initialiserIHM\(\)](#).

```

{
    // Largeur de l'ombre
    ui->Compass->setLineWidth(10);
    //qwtDirection->setLineWidth(0);
    // Type d'ombre
    ui->Compass->setFrameShadow(QwtCompass::Sunken);
    //qwtDirection->setFrameShadow(QwtCompass::Raised);
    //qwtDirection->setFrameShadow(QwtCompass::Plain);

    // La ligne de graduation
    //qwtDirection->setScaleComponents( QwtAbstractScaleDraw::Backbone );
    // Les étiquettes de graduation
    //qwtDirection->setScaleComponents( QwtAbstractScaleDraw::Labels );
    // Les graduations
    //qwtDirection->setScaleComponents( QwtAbstractScaleDraw::Ticks );
    // Plusieurs
    ui->Compass->setScaleComponents( QwtAbstractScaleDraw::Backbone |
        QwtAbstractScaleDraw::Ticks | QwtAbstractScaleDraw::Labels );

    // Les étiquettes de graduation :
    /*QMap<double, QString> map;
    for ( double d = 0.0; d < 360.0; d += 60.0 )
    {
        QString label;
        label.sprintf("%.0f", d);
        map.insert(d, label);
    }
    qwtDirection->setLabelMap(map);*/
    // ou :
    QMap<double, QString> map;
    map.insert(0.0, "N");
    map.insert(45.0, "NE");
    map.insert(90.0, "E");
    map.insert(135.0, "SE");
    map.insert(180.0, "S");
    map.insert(225.0, "SO");
    map.insert(270.0, "O");
    map.insert(315.0, "NO");
    ui->Compass->setLabelMap(map);

    // Échelle
    ui->Compass->setScaleTicks(0, 0, 3);
    ui->Compass->setScale(0, 0, 5.0);

    // Aiguille
    ui->Compass->setNeedle(new QwtCompassMagnetNeedle(
        QwtCompassMagnetNeedle::ThinStyle));

    // Origine
    ui->Compass->setOrigin(270.0); // au Nord
}

```

```

// Modification des couleurs
QPalette p1 = ui->Compass->palette();
p1.setColor(QPalette::Base, QColor(219, 219, 219)); // Fond extérieur
p1.setColor(QPalette::WindowText, QColor(231, 231, 231)); // Fond intérieur
p1.setColor(QPalette::Text, QColor(70, 70, 255)); // Texte et graduations
ui->Compass->setPalette(p1);

ui->Compass->setReadOnly(true);

// L'origine
ui->Vitesse->setOrigin(135.0);
ui->Vitesse->setScaleComponents( QwtAbstractScaleDraw::Backbone |
    QwtAbstractScaleDraw::Ticks | QwtAbstractScaleDraw::Labels );
// Longueur de la graduation
ui->Vitesse->setScaleTicks(0, 4, 8);
// Taille du cadran
ui->Vitesse->setScaleArc(0.0, 270.0);
// Intervalle de graduation
ui->Vitesse->setScale(-1, 2, 20);
ui->Vitesse->setRange(0.0, 240.0);
// Espacement entre l'étiquette et la graduation
ui->Vitesse->scaleDraw()->setSpacing(8);

ui->Vitesse->setReadOnly(true);
ui->Vitesse->scaleDraw()->setPenWidth(1);
ui->Vitesse->setLineWidth(4);
ui->Vitesse->setFrameShadow(QwtDial::Sunken);

// Aiguille
QwtDialSimpleNeedle *needle = new QwtDialSimpleNeedle(
    QwtDialSimpleNeedle::Arrow, true, Qt::red, QColor(Qt::gray).light(130));
ui->Vitesse->setNeedle(needle);

// Modification des couleurs
QPalette p2 = ui->Vitesse->palette();
p2.setColor(QPalette::Text, QColor(70, 70, 255));
p2.setColor(QPalette::Foreground, QColor(200, 200, 200));
p2.setColor(QPalette::Background, QColor(150, 150, 150));
ui->Vitesse->setPalette(p2);

// Fixer les valeurs initiales
ui->Compass->setValue(0.0); // Donc le Nord
ui->Vitesse->setValue(0.0);
}

```

### 10.3.3.25 IHMCampusSerre : modifierConsignes ( ) [slot]

Références [bdd](#), [envoyerConsignes\(\)](#), [BaseDeDonnees : executer\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    int nouvelleConsigneHygroMinSol = ui->EditHygroMinSol->text().toInt();
    int nouvelleConsigneHygroMaxSol = ui->EditHygroMaxSol->text().toInt();
    int nouvelleConsigneHygroMinAir = ui->EditHygroMinAir->text().toInt();
    int nouvelleConsigneHygroMaxAir = ui->EditHygroMaxAir->text().toInt();
    int nouvelleConsigneTempMinJour = ui->EditTempMinJour->text().toInt();
    int nouvelleConsigneTempMaxJour = ui->EditTempMaxJour->text().toInt();
    int nouvelleConsigneTempMinNuit = ui->EditTempMinNuit->text().toInt();
    int nouvelleConsigneTempMaxNuit = ui->EditTempMaxNuit->text().toInt();
    QString requete = "UPDATE consignes SET temperatureJourMin =" +
        QString::number(nouvelleConsigneTempMinJour) + " ,temperatureJourMax =" +
        QString::number(nouvelleConsigneTempMaxJour) + " ,temperatureNuitMin =" + QString::number(
        nouvelleConsigneTempMinNuit) + " ,temperatureNuitMax =" + QString::number(
        nouvelleConsigneTempMaxNuit) + " ,hygrometrieSolMin =" + QString::number(
        nouvelleConsigneHygroMinSol) + " ,hygrometrieSolMax =" + QString::number(
        nouvelleConsigneHygroMaxSol) + " ,HygrometrieAirMin =" + QString::number(
        nouvelleConsigneHygroMinAir) + " ,hygrometrieAirMax =" + QString::number(
        nouvelleConsigneHygroMaxAir) + " WHERE idConsignes = '1'";
    bdd->executer(requete);
}

```

```

envoyerConsignes(nouvelleConsigneTempMinJour,nouvelleConsigneTempMaxJour,
nouvelleConsigneTempMinNuit,nouvelleConsigneTempMaxNuit,
nouvelleConsigneHygroMinSol,nouvelleConsigneHygroMaxSol,nouvelleConsigneHygroMinAir,
nouvelleConsigneHygroMaxAir);
}

```

### 10.3.3.26 IHMCampusSerre : modifierNbMesures ( int n ) [slot]

Références [afficherOngletHistorique\(\)](#), et [nbMesuresHistorique](#).

Référencé par [initialiserIHM\(\)](#).

```

{
    nbMesuresHistorique = n;
    afficherOngletHistorique();
}

```

### 10.3.3.27 IHMCampusSerre : modifierSeuils ( ) [slot]

Références [bdd](#), [envoyerSeuils\(\)](#), [BaseDeDonnees : :executer\(\)](#), et [ui](#).

Référencé par [initialiserIHM\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;

    int nouveauSeuilTempMin = ui->EditTempMin->text().toInt();
    int nouveauSeuilTempMax = ui->EditTempMax->text().toInt();
    int nouveauSeuilVitesseMax = ui->EditVitMax->text().toInt();
    QString requete = "UPDATE seuils SET temperatureMin = '" + QString::number(
        nouveauSeuilTempMin) + "' ,temperatureMax = '" + QString::number(
        nouveauSeuilTempMax) + "' ,vitesseMax = '" + QString::number(nouveauSeuilVitesseMax) + "' WHERE
        idSeuils = '1'";
    bdd->executer(requete);
    envoyerSeuils(nouveauSeuilTempMin, nouveauSeuilTempMax,
        nouveauSeuilVitesseMax);
}

```

### 10.3.3.28 IHMCampusSerre : recupererConsignes ( ) [private]

Références [bdd](#), [BaseDeDonnees : :recuperer\(\)](#), et [ui](#).

Référencé par [IHMCampusSerre\(\)](#).

```

{
    QStringList consignes;
    QString requeteConsignes = "select * from consignes";
    bool retourConsignes = bdd->recuperer(requeteConsignes, consignes);
    if(retourConsignes)
    {
        ui->EditTempMinJour->setText(consignes.at(1));
        ui->EditTempMaxJour->setText(consignes.at(2));
        ui->EditTempMinNuit->setText(consignes.at(3));
        ui->EditTempMaxNuit->setText(consignes.at(4));
        ui->EditHygroMinSol->setText(consignes.at(5));
        ui->EditHygroMaxSol->setText(consignes.at(6));
        ui->EditHygroMinAir->setText(consignes.at(7));
        ui->EditHygroMaxAir->setText(consignes.at(8));
    }
}

```

**10.3.3.29 IHMCampusSerre :recupererSeuils ( ) [private]**

Références [bdd](#), [BaseDeDonnees :recuperer\(\)](#), et [ui](#).

Référencé par [IHMCampusSerre\(\)](#).

```
{
    QStringList seuils;
    QString requete = "select * from seuils";
    bool retour = bdd->recuperer(requete, seuils);
    if (retour)
    {
        ui->EditTempMin->setText(seuils.at(1));
        ui->EditTempMax->setText(seuils.at(2));
        ui->EditVitMax->setText(seuils.at(3));
    }
}
```

**10.3.3.30 IHMCampusSerre :selectionnerOnglet ( int onglet ) [slot]**

Référencé par [initialiserIHM\(\)](#).

```
{
}
```

**10.3.4 Documentation des données membres****10.3.4.1 BaseDeDonnees\* IHMCampusSerre :bdd [private]**

relation vers la classe [BaseDeDonnees](#)

Référencé par [initialiserBDD\(\)](#), [modifierConsignes\(\)](#), [modifierSeuils\(\)](#), [recupererConsignes\(\)](#), et [recupererSeuils\(\)](#).

**10.3.4.2 GestionPort\* IHMCampusSerre :gestionPort [private]**

relation vers la classe [GestionPort](#)

Référencé par [commanderChauffage\(\)](#), [commanderOuvrant\(\)](#), [commanderVanne\(\)](#), [envoyerConsignes\(\)](#), [envoyerSeuils\(\)](#), et [IHMCampusSerre\(\)](#).

**10.3.4.3 int IHMCampusSerre :nbMesuresHistorique [private]**

nombre de mesures à afficher dans l'historique

Référencé par [afficherOngletHistorique\(\)](#), [initialiserIHM\(\)](#), et [modifierNbMesures\(\)](#).

**10.3.4.4 Telemetrie\* IHMCampusSerre :telemetrie [private]**

relation vers la classe [Telemetrie](#)

Référencé par [afficherOngletHistorique\(\)](#), [afficherOngletTelemetrie\(\)](#), [IHMCampusSerre\(\)](#), et [initialiserIHM\(\)](#).

#### 10.3.4.5 Ui : :IHMCampusSerre\* IHMCampusSerre : :ui [private]

relation vers la classe IHM

Référéncé par [actualiser\(\)](#), [afficherOngletHistorique\(\)](#), [afficherOngletTelemetrie\(\)](#), [commanderChauffageAllumer\(\)](#), [commanderChauffageAuto\(\)](#), [commanderChauffage-Eteint\(\)](#), [commanderOuvrant0\(\)](#), [commanderOuvrant100\(\)](#), [commanderOuvrant25\(\)](#), [commanderOuvrant50\(\)](#), [commanderOuvrant75\(\)](#), [commanderOuvrantAuto\(\)](#), [commander-VanneAuto\(\)](#), [commanderVanneFermer\(\)](#), [commanderVanneOuvert\(\)](#), [effacer-Historique\(\)](#), [IHMCampusSerre\(\)](#), [initialiserIHM\(\)](#), [initialiserQwt\(\)](#), [modifierConsignes\(\)](#), [modifierSeuils\(\)](#), [recupererConsignes\(\)](#), [recupererSeuils\(\)](#), et [~IHMCampusSerre\(\)](#).

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

- [IHMCampusSerre.h](#)
- [IHMCampusSerre.cpp](#)

## 10.4 Référence de la structure Mesure

```
#include <telemetrie.h>
```

### Attributs publics

- QString [date](#)
- QString [heure](#)
- double [temperatureAir](#)
- double [temperatureSol](#)
- double [hygrometrieSol](#)
- double [hygrometrieAir](#)
- double [vitesseVent](#)
- int [directionVent](#)
- int [ouvrant](#)
- bool [vanne](#)
- bool [chauffage](#)

### 10.4.1 Documentation des données membres

#### 10.4.1.1 bool Mesure : :chauffage

Référéncé par [Telemetrie : :convertirDonnees\(\)](#), [Telemetrie : :enregistrerMesure\(\)](#), et [Telemetrie : :getEtatChauffage\(\)](#).

#### 10.4.1.2 QString Mesure : :date

Référéncé par [Telemetrie : :enregistrerMesure\(\)](#).

#### 10.4.1.3 int Mesure : :directionVent

Référéncé par [Telemetrie : :convertirDonnees\(\)](#), [Telemetrie : :enregistrerMesure\(\)](#), et [Telemetrie : :getDirectionVent\(\)](#).

**10.4.1.4 QString Mesure : :heure**

Référencé par [Telemetrie : :enregistrerMesure\(\)](#).

**10.4.1.5 double Mesure : :hygrometrieAir**

Référencé par [Telemetrie : :convertirDonnees\(\)](#), [Telemetrie : :enregistrerMesure\(\)](#), et [Telemetrie : :getHygrometrieAir\(\)](#).

**10.4.1.6 double Mesure : :hygrometrieSol**

Référencé par [Telemetrie : :convertirDonnees\(\)](#), [Telemetrie : :enregistrerMesure\(\)](#), et [Telemetrie : :getHygrometrieSol\(\)](#).

**10.4.1.7 int Mesure : :ouvrant**

Référencé par [Telemetrie : :convertirDonnees\(\)](#), [Telemetrie : :enregistrerMesure\(\)](#), et [Telemetrie : :getEtatOuvrant\(\)](#).

**10.4.1.8 double Mesure : :temperatureAir**

Référencé par [Telemetrie : :convertirDonnees\(\)](#), [Telemetrie : :enregistrerMesure\(\)](#), et [Telemetrie : :getTemperatureAir\(\)](#).

**10.4.1.9 double Mesure : :temperatureSol**

Référencé par [Telemetrie : :convertirDonnees\(\)](#), [Telemetrie : :enregistrerMesure\(\)](#), et [Telemetrie : :getTemperatureSol\(\)](#).

**10.4.1.10 bool Mesure : :vanne**

Référencé par [Telemetrie : :convertirDonnees\(\)](#), [Telemetrie : :enregistrerMesure\(\)](#), et [Telemetrie : :getEtatVanne\(\)](#).

**10.4.1.11 double Mesure : :vitesseVent**

Référencé par [Telemetrie : :convertirDonnees\(\)](#), [Telemetrie : :enregistrerMesure\(\)](#), et [Telemetrie : :getVitesseVent\(\)](#).

La documentation de cette structure a été générée à partir du fichier suivant :

– [telemetrie.h](#)

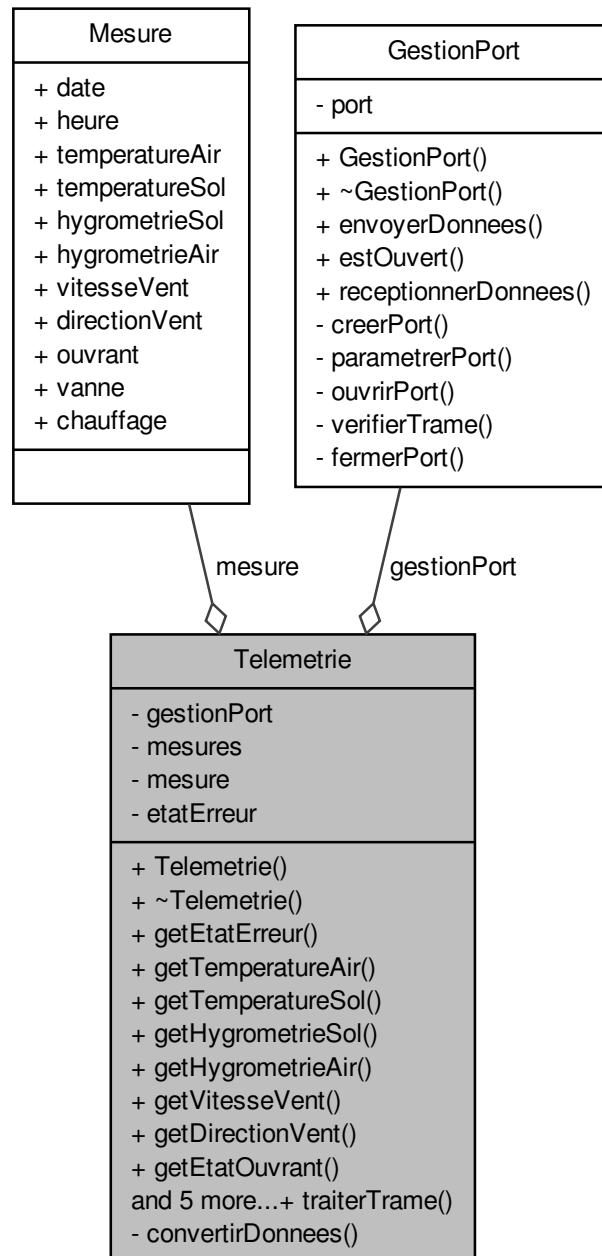
**10.5 Référence de la classe Telemetrie**

Communique avec la serre via un port série virtuel.

```
#include <telemetrie.h>
```



Graphe de collaboration de Telemetrie :



### Connecteurs publics

- void `traiterTrame` (QString trame)  
*Récupère la trame et la passe à la méthode `convertirDonnees()`*

### Signaux

- void `nouvellesDonnees` ()

### Fonctions membres publiques

- `Telemetry` (`GestionPort *gestionPort=0`, `QObject *parent=0`)  
*Constructeur.*
- `~Telemetry` ()  
*Destructeur.*
- bool `getEtatErreur` ()  
*Récupère la valeur l'état de l'erreur.*
- double `getTemperatureAir` ()  
*Récupère la valeur de la température de l'air.*
- double `getTemperatureSol` ()  
*Récupère la valeur de la température du sol.*
- double `getHygrometrieSol` ()  
*Récupère la valeur de la l'hygrométrie du sol.*
- double `getHygrometrieAir` ()  
*Récupère la valeur de la l'hygrométrie de l'air.*
- double `getVitesseVent` ()  
*Récupère la valeur de la vitesse du vent.*
- int `getDirectionVent` ()  
*Récupère la valeur de la direction du vent.*
- int `getEtatOuvrant` ()  
*Récupère la valeur de l'ouverture de l'ouvrant.*
- bool `getEtatVanne` ()  
*Récupère la valeur de l'état de la vanne.*
- bool `getEtatChauffage` ()  
*Récupère la valeur de l'état du chauffage.*
- void `enregistrerMesure` ()  
*Stock les valeurs dans un conteneur.*
- `Mesure` `getMesure` ()
- `QVector< Mesure >` `getMesures` ()
- void `purgerMesures` ()

### Fonctions membres privées

- bool `convertirDonnees` (QString trame)  
*Convertit les trames afin d'avoir les valeurs des mesures.*

### Attributs privés

- `GestionPort * gestionPort`  
*relation vers la classe `GestionPort`*
- `QVector< Mesure > mesures`  
*stockage des mesures*
- `Mesure mesure`  
*mesure courante des données de la serre*
- bool `etatErreur`

*indique une erreur*

### 10.5.1 Description détaillée

#### Auteur

Di Sario

#### Version

1.0

#### Date

Jeudi 22 mars 2018

### 10.5.2 Documentation des constructeurs et destructeur

#### 10.5.2.1 Telemetrie : :Telemetrie ( GestionPort \* gestionPort = 0, QObject \* parent = 0 )

##### Paramètres

<i>gestionPort</i>	GestionPort* adresse de l'objet <a href="#">GestionPort</a>
<i>parent</i>	QObject Adresse de l'objet parent (sinon 0)

Références [traiterTrame\(\)](#).

```
                                : QObject (
                                parent)
{
    if (gestionPort != 0)
        connect (gestionPort, SIGNAL(nouvelleTrame(QString)), this, SLOT(
            traiterTrame(QString));
}
```

#### 10.5.2.2 Telemetrie : :~Telemetrie ( )

```
{
}
```

### 10.5.3 Documentation des fonctions membres

#### 10.5.3.1 bool Telemetrie : :convertirDonnees ( QString trame ) [private]

Convertit les trames afin d'avoir les valeurs de la température de l'air et du sol ainsi que l'hygrométrie de l'air et du sol, la vitesse du vent ainsi que sa direction, l'état de l'ouvrant, la vanne et si il y a une erreur.

##### Paramètres

<i>trame</i>	QString La trame a convertir
--------------	------------------------------

## Renvoi

true si les données ont été converties

Références `Mesure : :chauffage`, `Mesure : :directionVent`, `etatErreur`, `Mesure : :hygrometrieAir`, `Mesure : :hygrometrieSol`, `mesure`, `Mesure : :ouvrant`, `Mesure : :temperatureAir`, `Mesure : :temperatureSol`, `Mesure : :vanne`, et `Mesure : :vitesseVent`.

Référencé par `traiterTrame()`.

```
{
    qDebug() << Q_FUNC_INFO << trame;
    // trame -> $PCS.000.235.220.080.060.020.001.025.000.001*
    /* champErreur -> 000 (Après le 1er ".")
    champTemperatureAir -> 235 (Après le 2eme ".")
    champTemperatureSol -> 220 (Après le 3eme ".")
    champHygrometrieSol -> 080 (Après le 4eme ".")
    champHygrometrieAir -> 050 (Après le 5eme ".")
    champVitesseVent -> 020 (Après le 6eme ".")
    champDirectionVent -> 050 (Après le 7eme ".")
    champOuvrant -> 025 (Après le 8eme ".")
    champVanne -> 000 (Après le 9eme ".")
    champChauffage -> 001 (après le 10eme ".")
    */
    QString champErreur = trame.section('.',1,1);

    if(champErreur == "000")
    {
        etatErreur = false;
    }
    else etatErreur = true;

    //qDebug() << "champErreur : " << champErreur;
    //qDebug() << "etatErreur : " << etatErreur;

    QString champTemperatureAir = trame.section('.',2,2);

    QString nombreEntier = champTemperatureAir;
    QString decimale = champTemperatureAir;
    nombreEntier.remove(2,3);
    decimale.remove(0,2);

    QString temperatureAir = nombreEntier + "," + decimale;
    mesure.temperatureAir = temperatureAir.toDouble();

    //qDebug() << "champTemperatureAir : " << champTemperatureAir;
    //qDebug() << "temperatureAir : " << temperatureAir.toDouble();

    QString champTemperatureSol = trame.section('.',3,3);

    nombreEntier = champTemperatureSol;
    decimale = champTemperatureSol;
    nombreEntier.remove(2,3);
    decimale.remove(0,2);

    QString temperatureSol = nombreEntier + "," + decimale;
    mesure.temperatureSol = temperatureSol.toDouble();

    //qDebug() << "champTemperatureSol : " << champTemperatureSol;
    //qDebug() << "temperatureSol : " << temperatureSol.toDouble();

    QString champHygrometrieSol = trame.section('.',4,4);

    QString verification1 = champHygrometrieSol;
    QString verification2 = champHygrometrieSol;
    QString hygrometrieSol;
    verification1.remove(2,2);
    verification2.remove(1,2);
    if(verification1 == "00")
    {
        champHygrometrieSol.remove(0,2);
        hygrometrieSol = champHygrometrieSol;
    }
}
```

```

    }
    else if (verification2 == "0")
    {
        champHygrometrieSol.remove(0,1);
        hygrometrieSol = champHygrometrieSol;
    }
    else hygrometrieSol = champHygrometrieSol;

    mesure.hygrometrieSol = hygrometrieSol.toDouble();

    //qDebug() << "champHygrometrieSol : " << champHygrometrieSol;
    //qDebug() << "hygrometrieSol : " << hygrometrieSol.toDouble();

    QString champHygrometrieAir = trame.section('.',5,5);

    verification1 = champHygrometrieAir;
    verification2 = champHygrometrieAir;
    QString hygrometrieAir;
    verification1.remove(2,2);
    verification2.remove(1,2);
    if(verification1 == "00")
    {
        champHygrometrieAir.remove(0,2);
        hygrometrieAir = champHygrometrieAir;
    }
    else if (verification2 == "0")
    {
        champHygrometrieAir.remove(0,1);
        hygrometrieAir = champHygrometrieAir;
    }
    else hygrometrieAir = champHygrometrieAir;

    mesure.hygrometrieAir = hygrometrieAir.toDouble();

    //qDebug() << "champHygrometrieAir : " << champHygrometrieAir;
    //qDebug() << "hygrometrieAir : " << hygrometrieAir.toDouble();

    QString champVitesseVent = trame.section('.',6,6);

    //qDebug() << "champVitesseVent : " << champVitesseVent;

    verification1 = champVitesseVent;
    verification2 = champVitesseVent;
    QString vitesseVent;
    verification1.remove(2,2);
    verification2.remove(1,2);
    if(verification1 == "00")
    {
        champVitesseVent.remove(0,2);
        vitesseVent = champVitesseVent;
    }
    else if (verification2 == "0")
    {
        champVitesseVent.remove(0,1);
        vitesseVent = champVitesseVent;
    }
    else vitesseVent = champVitesseVent;
    mesure.vitesseVent = vitesseVent.toDouble();

    //qDebug() << "verification1 et 2 : " << verification1 << "," <<
        verification2;
    //qDebug() << "vitesseVent : " << vitesseVent;

    QString champDirectionVent = trame.section('.',7,7);

    QString directionVent = champDirectionVent;
    mesure.directionVent = directionVent.toInt();

    //qDebug() << "champDirectionVent : " << champDirectionVent;
    //qDebug() << "directionVent : " << directionVent;

    QString champOuvrant = trame.section('.',8,8);
    QString etatOuvrant = champOuvrant;
    mesure.ouvrant = etatOuvrant.toInt();

```

```

//qDebug() << "champOuvrant : " << champOuvrant;
//qDebug() << "etatOuvrant : " << etatOuvrant;

QString champVanne = trame.section('.',9,9);

if(champVanne == "000")
{
    mesure.vanne = false;
}
else mesure.vanne = true;

//qDebug() << "champVanne : " << champVanne;
//qDebug() << "etatVanne : " << mesure.vanne;

QString champChauffage = trame.section('.',10,10);

if(champChauffage == "000")
{
    mesure.chauffage = false;
}
else mesure.chauffage = true;

return true;
}

```

### 10.5.3.2 Telemetrie : :enregistrerMesure ( )

Références [Mesure : :chauffage](#), [Mesure : :date](#), [Mesure : :directionVent](#), [getDirectionVent\(\)](#), [getEtatChauffage\(\)](#), [getEtatOuvrant\(\)](#), [getEtatVanne\(\)](#), [getHygrometrieAir\(\)](#), [getHygrometrieSol\(\)](#), [getTemperatureAir\(\)](#), [getTemperatureSol\(\)](#), [getVitesseVent\(\)](#), [Mesure : :heure](#), [Mesure : :hygrometrieAir](#), [Mesure : :hygrometrieSol](#), [mesure](#), [mesures](#), [Mesure : :ouvrant](#), [Mesure : :temperatureAir](#), [Mesure : :temperatureSol](#), [Mesure : :vanne](#), et [Mesure : :vitesseVent](#).

Référencé par [traiterTrame\(\)](#).

```

{
    Mesure mesure;
    QDateTime horodatage = QDateTime::currentDateTime();

    mesure.date = horodatage.toString("dd/MM/yyyy");
    mesure.heure = horodatage.toString("hh:mm:ss");

    mesure.temperatureAir = getTemperatureAir();
    mesure.temperatureSol = getTemperatureSol();
    mesure.hygrometrieAir = getHygrometrieAir();
    mesure.hygrometrieSol = getHygrometrieSol();
    mesure.vitesseVent = getVitesseVent();
    mesure.directionVent = getDirectionVent();
    mesure.ouvrant = getEtatOuvrant();
    mesure.vanne = getEtatVanne();
    mesure.chauffage = getEtatChauffage();

    mesures.push_back(mesure);
}

```

### 10.5.3.3 int Telemetrie : :getDirectionVent ( )

Renvoie

int

Références [Mesure : :directionVent](#), et [mesure](#).

Référencé par [IHMCampusSerre : :afficherOngletTelemetrie\(\)](#), et [enregistrerMesure\(\)](#).

```
{  
    return mesure.directionVent;  
}
```

#### 10.5.3.4 bool Telemetrie : :getEtatChauffage ( )

Renvoie

bool

Références [Mesure : :chauffage](#), et [mesure](#).

Référencé par [IHMCampusSerre : :afficherOngletTelemetrie\(\)](#), et [enregistrerMesure\(\)](#).

```
{  
    return mesure.chauffage;  
}
```

#### 10.5.3.5 bool Telemetrie : :getEtatErreur ( )

Renvoie

bool

Références [etatErreur](#).

```
{  
    return etatErreur;  
}
```

#### 10.5.3.6 int Telemetrie : :getEtatOuvrant ( )

Renvoie

int

Références [mesure](#), et [Mesure : :ouvrant](#).

Référencé par [IHMCampusSerre : :afficherOngletTelemetrie\(\)](#), et [enregistrerMesure\(\)](#).

```
{  
    return mesure.ouvrant;  
}
```

#### 10.5.3.7 bool Telemetrie : :getEtatVanne ( )

Renvoie

bool

Références [mesure](#), et [Mesure : :vanne](#).

Référencé par [IHMCampusSerre : :afficherOngletTelemetrie\(\)](#), et [enregistrerMesure\(\)](#).

```
{  
    return mesure.vanne;  
}
```

#### 10.5.3.8 double Telemetrie : :getHygrometrieAir ( )

Renvoie

double

Références [Mesure : :hygrometrieAir](#), et [mesure](#).

Référencé par [IHMCampusSerre : :afficherOngletTelemetrie\(\)](#), et [enregistrerMesure\(\)](#).

```
{  
    return mesure.hygrometrieAir;  
}
```

#### 10.5.3.9 double Telemetrie : :getHygrometrieSol ( )

Renvoie

double

Références [Mesure : :hygrometrieSol](#), et [mesure](#).

Référencé par [IHMCampusSerre : :afficherOngletTelemetrie\(\)](#), et [enregistrerMesure\(\)](#).

```
{  
    return mesure.hygrometrieSol;  
}
```

#### 10.5.3.10 Telemetrie : :getMesure ( )

Renvoie

[Mesure](#)

Références [mesure](#).

```
{  
    return mesure;  
}
```

#### 10.5.3.11 Telemetrie : :getMesures ( )



Renvoie

QVector<Mesure>

Références [mesures](#).

Référencé par [IHMCampusSerre : :afficherOngletHistorique\(\)](#).

```
{  
    return mesures;  
}
```

#### 10.5.3.12 double Telemetrie : :getTemperatureAir ( )

Renvoie

double

Références [mesure](#), et [Mesure : :temperatureAir](#).

Référencé par [IHMCampusSerre : :afficherOngletTelemetrie\(\)](#), et [enregistrerMesure\(\)](#).

```
{  
    return mesure.temperatureAir;  
}
```

#### 10.5.3.13 double Telemetrie : :getTemperatureSol ( )

Renvoie

double

Références [mesure](#), et [Mesure : :temperatureSol](#).

Référencé par [IHMCampusSerre : :afficherOngletTelemetrie\(\)](#), et [enregistrerMesure\(\)](#).

```
{  
    return mesure.temperatureSol;  
}
```

#### 10.5.3.14 double Telemetrie : :getVitesseVent ( )

Renvoie

double

Références [mesure](#), et [Mesure : :vitesseVent](#).

Référencé par [IHMCampusSerre : :afficherOngletTelemetrie\(\)](#), et [enregistrerMesure\(\)](#).

```
{  
    return mesure.vitesseVent;  
}
```

#### 10.5.3.15 void Telemetrie : :nouvellesDonnees ( ) [signal]

Référencé par [traiterTrame\(\)](#).

### 10.5.3.16 Telemetrie : :purgerMesures ( )

Références [mesures](#).

```
{
    mesures.clear();
}
```

### 10.5.3.17 void Telemetrie : :traiterTrame ( QString *trame* ) [slot]

Paramètres

<i>trame</i>	
--------------	--

Références [convertirDonnees\(\)](#), [enregistrerMesure\(\)](#), et [nouvellesDonnees\(\)](#).

Référencé par [Telemetrie\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO << trame;

    if(convertirDonnees(trame))
    {
        enregistrerMesure();

        emit nouvellesDonnees();
    }
}
```

## 10.5.4 Documentation des données membres

### 10.5.4.1 bool Telemetrie : :etatErreur [private]

Référencé par [convertirDonnees\(\)](#), et [getEtatErreur\(\)](#).

### 10.5.4.2 GestionPort\* Telemetrie : :gestionPort [private]

### 10.5.4.3 Mesure Telemetrie : :mesure [private]

Référencé par [convertirDonnees\(\)](#), [enregistrerMesure\(\)](#), [getDirectionVent\(\)](#), [getEtat-Chauffage\(\)](#), [getEtatOuvrant\(\)](#), [getEtatVanne\(\)](#), [getHygrometrieAir\(\)](#), [getHygrometrie-Sol\(\)](#), [getMesure\(\)](#), [getTemperatureAir\(\)](#), [getTemperatureSol\(\)](#), et [getVitesseVent\(\)](#).

### 10.5.4.4 QVector<Mesure> Telemetrie : :mesures [private]

Référencé par [enregistrerMesure\(\)](#), [getMesures\(\)](#), et [purgerMesures\(\)](#).

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

- [telemetrie.h](#)
- [telemetrie.cpp](#)

## 11 Documentation des fichiers

## 11.1 Référence du fichier basededonnees.cpp

Définition de la classe [GestionPort](#).

```
#include "basededonnees.h" #include <QDebug> #include <Q-  
MessageBox>
```

### 11.1.1 Description détaillée

#### Auteur

Thierry VAIRA, Di Sario

#### Version

1.0

## 11.2 Référence du fichier basededonnees.h

```
#include <QObject> #include <QtSql/QtSql> #include <QtSql-  
Database> #include <QMutex>
```

#### Classes

- class [BaseDeDonnees](#)  
*Gestion la base de données SQLite.*

### 11.2.1 Description détaillée

## 11.3 Référence du fichier Changelog.dox

## 11.4 Référence du fichier gestionPort.cpp

Définition de la classe [GestionPort](#).

```
#include "gestionPort.h" #include <unistd.h> #include <Q-  
StringList> #include <QDebug>
```

### 11.4.1 Description détaillée

#### Auteur

Di Sario

#### Version

1.0

## Date

Mercredi 21 février 2018

## 11.5 Référence du fichier gestionPort.h

```
#include "qextserialport.h" #include <QString>
```

## Classes

- class [GestionPort](#)  
*Communique avec la serre via un port série virtuel.*

## Macros

- #define [PORT](#) "/dev/ttyUSB0"  
*Fichier périphérique par défaut.*

## 11.5.1 Description détaillée

## 11.5.2 Documentation des macros

11.5.2.1 #define [PORT](#) "/dev/ttyUSB0"

Référencé par [GestionPort : :creerPort\(\)](#), et [GestionPort : :ouvrirPort\(\)](#).

## 11.6 Référence du fichier IHMCampusSerre.cpp

Définition de la classe [IHMCampusSerre](#).

```
#include "IHMCampusSerre.h" #include "ui_IHMCampusSerre.-  
h" #include "gestionPort.h" #include "telemetrie.h" #include  
"basededonnees.h" #include <QMessageBox> #include <Q-  
Debug> #include <qwt_compass.h> #include <qwt_compass_-  
rose.h> #include <qwt_dial_needle.h>
```

## 11.6.1 Description détaillée

## Auteur

Di Sario

## Version

1.0

## Date

Jeudi 22 mars 2018

## 11.7 Référence du fichier IHMCampusSerre.h

```
#include <QMainWindow>    #include <QTimer>    #include <Q-  
TableWidget>
```

### Classes

- class [IHMCampusSerre](#)  
*La fenêtre principale de l'application.*

### Macros

- #define [MIN\\_TEMP](#) 20
- #define [MAX\\_TEMP](#) 30
- #define [MAX\\_VIT](#) 90

#### 11.7.1 Description détaillée

#### 11.7.2 Documentation des macros

##### 11.7.2.1 #define MAX\_TEMP 30

##### 11.7.2.2 #define MAX\_VIT 90

##### 11.7.2.3 #define MIN\_TEMP 20

## 11.8 Référence du fichier main.cpp

Programme principal.

```
#include <QApplication> #include "IHMCampusSerre.h"
```

### Fonctions

- int [main](#) (int argc, char \*argv[])  
*Programme principal : Crée et affiche la fenêtre principale de l'application.*

#### 11.8.1 Description détaillée

Crée et affiche la fenêtre principale de l'application

#### 11.8.2 Documentation des fonctions

##### 11.8.2.1 main ( int argc, char \* argv[] )

## Paramètres

<i>argc</i>	
<i>argv[]</i>	

## Renvoie

int

```
{
    QApplication a(argc, argv);
    a.setApplicationName("projetCampusSerre");

    IHMCampusSerre w;
    w.show();

    return a.exec();
}
```

## 11.9 Référence du fichier README.dox

## 11.10 Référence du fichier telemetry.cpp

Définition de la classe [Telemetry](#).

```
#include "telemetry.h" #include <QDateTime> #include <Q-
Debug>
```

## 11.10.1 Description détaillée

## Auteur

Di Sario

## Version

1.0

## Date

Jeudi 22 mars 2018

## 11.11 Référence du fichier telemetry.h

```
#include "gestionPort.h" #include <QString> #include <Q-
Vector>
```

## Classes

- struct [Mesure](#)
- class [Telemetry](#)  
*Communique avec la serre via un port série virtuel.*

**11.11.1 Description détaillée**

## Index

- ~BaseDeDonnees
  - BaseDeDonnees, 8
- ~GestionPort
  - GestionPort, 15
- ~IHMCampusSerre
  - IHMCampusSerre, 23
- ~Telemetrie
  - Telemetrie, 40
- BaseDeDonnees, 6
  - ~BaseDeDonnees, 8
  - BaseDeDonnees, 8
  - baseDeDonnees, 14
  - BaseDeDonnees, 8
  - connecter, 8
  - db, 14
  - destruireInstance, 9
  - executer, 9
  - getInstance, 10
  - mutex, 14
  - nbAcces, 14
  - recuperer, 10–13
- Changelog.dox, 48
- GestionPort, 14
  - ~GestionPort, 15
  - GestionPort, 15
  - creerPort, 16
  - envoyerDonnees, 16
  - estOuvert, 16
  - fermerPort, 16
  - GestionPort, 15
  - nouvelleTrame, 17
  - ouvrirPort, 17
  - parametrerPort, 17
  - port, 19
  - receptionnerDonnees, 17
  - verifierTrame, 18
- IHMCampusSerre, 19
  - ~IHMCampusSerre, 23
  - IHMCampusSerre, 22
  - actualiser, 23
  - afficherOngletHistorique, 23
  - afficherOngletTelemetrie, 25
  - bdd, 35
  - commanderChauffage, 25
  - commanderChauffageAllumer, 25
  - commanderChauffageAuto, 25
  - commanderChauffageEteint, 26
  - commanderOuvrant, 26
  - commanderOuvrant0, 26
  - commanderOuvrant100, 26
  - commanderOuvrant25, 27
  - commanderOuvrant50, 27
  - commanderOuvrant75, 27
  - commanderOuvrantAuto, 28
  - commanderVanne, 28
  - commanderVanneAuto, 28
  - commanderVanneFermer, 28
  - commanderVanneOuvert, 29
  - effacerHistorique, 29
  - envoyerConsignes, 29
  - envoyerSeuils, 30
  - gestionPort, 35
  - IHMCampusSerre, 22
  - initialiserBDD, 30
  - initialiserIHM, 31
  - initialiserQwt, 32
  - modifierConsignes, 33
  - modifierNbMesures, 34
  - modifierSeuils, 34
  - nbMesuresHistorique, 35
  - recupererConsignes, 34
  - recupererSeuils, 34
  - selectionnerOnglet, 35
  - telemetrie, 35
  - ui, 35
- IHMCampusSerre.cpp, 49
- IHMCampusSerre.h, 50
  - MAX\_TEMP, 50
  - MAX\_VIT, 50
  - MIN\_TEMP, 50
- MAX\_TEMP
  - IHMCampusSerre.h, 50
- MAX\_VIT
  - IHMCampusSerre.h, 50
- MIN\_TEMP
  - IHMCampusSerre.h, 50
- Mesure, 36
  - chauffage, 36
  - date, 36
  - directionVent, 36
  - heure, 36
  - hygrometrieAir, 37



- hygrometrieSol, [37](#)
- ouvrant, [37](#)
- temperatureAir, [37](#)
- temperatureSol, [37](#)
- vanne, [37](#)
- vitesseVent, [37](#)
- PORT
  - gestionPort.h, [49](#)
- README.dox, [51](#)
- Telemetrie, [37](#)
  - ~Telemetrie, [40](#)
  - Telemetrie, [40](#)
  - convertirDonnees, [40](#)
  - enregistrerMesure, [43](#)
  - etatErreur, [47](#)
  - gestionPort, [47](#)
  - getDirectionVent, [43](#)
  - getEtatChauffage, [44](#)
  - getEtatErreur, [44](#)
  - getEtatOuvrant, [44](#)
  - getEtatVanne, [44](#)
  - getHygrometrieAir, [45](#)
  - getHygrometrieSol, [45](#)
  - getMesure, [45](#)
  - getMesures, [45](#)
  - getTemperatureAir, [46](#)
  - getTemperatureSol, [46](#)
  - getVitesseVent, [46](#)
  - mesure, [47](#)
  - mesures, [47](#)
  - nouvellesDonnees, [46](#)
  - purgerMesures, [46](#)
  - Telemetrie, [40](#)
  - traiterTrame, [47](#)
- actualiser
  - IHMCampusSerre, [23](#)
- afficherOngletHistorique
  - IHMCampusSerre, [23](#)
- afficherOngletTelemetrie
  - IHMCampusSerre, [25](#)
- baseDeDonnees
  - BaseDeDonnees, [14](#)
- basededonnees.cpp, [48](#)
- basededonnees.h, [48](#)
- bdd
  - IHMCampusSerre, [35](#)
- chauffage
  - Mesure, [36](#)
  - commanderChauffage
    - IHMCampusSerre, [25](#)
  - commanderChauffageAllumer
    - IHMCampusSerre, [25](#)
  - commanderChauffageAuto
    - IHMCampusSerre, [25](#)
  - commanderChauffageEteint
    - IHMCampusSerre, [26](#)
  - commanderOuvrant
    - IHMCampusSerre, [26](#)
  - commanderOuvrant0
    - IHMCampusSerre, [26](#)
  - commanderOuvrant100
    - IHMCampusSerre, [26](#)
  - commanderOuvrant25
    - IHMCampusSerre, [27](#)
  - commanderOuvrant50
    - IHMCampusSerre, [27](#)
  - commanderOuvrant75
    - IHMCampusSerre, [27](#)
  - commanderOuvrantAuto
    - IHMCampusSerre, [28](#)
  - commanderVanne
    - IHMCampusSerre, [28](#)
  - commanderVanneAuto
    - IHMCampusSerre, [28](#)
  - commanderVanneFermer
    - IHMCampusSerre, [28](#)
  - commanderVanneOuvert
    - IHMCampusSerre, [29](#)
  - connecter
    - BaseDeDonnees, [8](#)
  - convertirDonnees
    - Telemetrie, [40](#)
  - creerPort
    - GestionPort, [16](#)
  - date
    - Mesure, [36](#)
  - db
    - BaseDeDonnees, [14](#)
  - destruireInstance
    - BaseDeDonnees, [9](#)
  - directionVent
    - Mesure, [36](#)
  - effacerHistorique
    - IHMCampusSerre, [29](#)
  - enregistrerMesure

- Telemetrie, [43](#)
- envoyerConsignes
  - IHMCampusSerre, [29](#)
- envoyerDonnees
  - GestionPort, [16](#)
- envoyerSeuils
  - IHMCampusSerre, [30](#)
- estOuvert
  - GestionPort, [16](#)
- etatErreur
  - Telemetrie, [47](#)
- executer
  - BaseDeDonnees, [9](#)
- fermerPort
  - GestionPort, [16](#)
- gestionPort
  - IHMCampusSerre, [35](#)
  - Telemetrie, [47](#)
- gestionPort.cpp, [48](#)
- gestionPort.h, [49](#)
  - PORT, [49](#)
- getDirectionVent
  - Telemetrie, [43](#)
- getEtatChauffage
  - Telemetrie, [44](#)
- getEtatErreur
  - Telemetrie, [44](#)
- getEtatOuvrant
  - Telemetrie, [44](#)
- getEtatVanne
  - Telemetrie, [44](#)
- getHygrometrieAir
  - Telemetrie, [45](#)
- getHygrometrieSol
  - Telemetrie, [45](#)
- getInstance
  - BaseDeDonnees, [10](#)
- getMesure
  - Telemetrie, [45](#)
- getMesures
  - Telemetrie, [45](#)
- getTemperatureAir
  - Telemetrie, [46](#)
- getTemperatureSol
  - Telemetrie, [46](#)
- getVitesseVent
  - Telemetrie, [46](#)
- heure
- Mesure, [36](#)
- hygrometrieAir
  - Mesure, [37](#)
- hygrometrieSol
  - Mesure, [37](#)
- initialiserBDD
  - IHMCampusSerre, [30](#)
- initialiserIHM
  - IHMCampusSerre, [31](#)
- initialiserQwt
  - IHMCampusSerre, [32](#)
- main
  - main.cpp, [50](#)
- main.cpp, [50](#)
  - main, [50](#)
- measure
  - Telemetrie, [47](#)
- mesures
  - Telemetrie, [47](#)
- modifierConsignes
  - IHMCampusSerre, [33](#)
- modifierNbMesures
  - IHMCampusSerre, [34](#)
- modifierSeuils
  - IHMCampusSerre, [34](#)
- mutex
  - BaseDeDonnees, [14](#)
- nbAcces
  - BaseDeDonnees, [14](#)
- nbMesuresHistorique
  - IHMCampusSerre, [35](#)
- nouvelleTrame
  - GestionPort, [17](#)
- nouvellesDonnees
  - Telemetrie, [46](#)
- ouvrant
  - Mesure, [37](#)
- ouvrirPort
  - GestionPort, [17](#)
- parametrerPort
  - GestionPort, [17](#)
- port
  - GestionPort, [19](#)
- purgerMesures
  - Telemetrie, [46](#)

---

- receptionnerDonnees
  - GestionPort, [17](#)
- recuperer
  - BaseDeDonnees, [10–13](#)
- recupererConsignes
  - IHMCampusSerre, [34](#)
- recupererSeuils
  - IHMCampusSerre, [34](#)
  
- selectionnerOnglet
  - IHMCampusSerre, [35](#)
  
- telemetrie
  - IHMCampusSerre, [35](#)
- telemetrie.cpp, [51](#)
- telemetrie.h, [51](#)
- temperatureAir
  - Mesure, [37](#)
- temperatureSol
  - Mesure, [37](#)
- traiterTrame
  - Telemetrie, [47](#)
  
- ui
  - IHMCampusSerre, [35](#)
  
- vanne
  - Mesure, [37](#)
- verifierTrame
  - GestionPort, [18](#)
- vitesseVent
  - Mesure, [37](#)