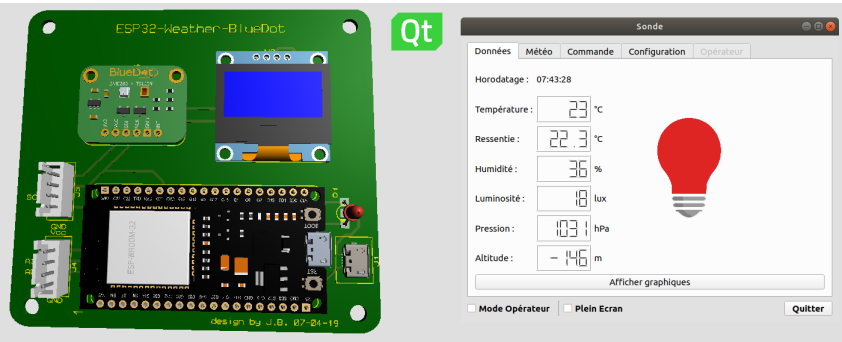
Mini-projet Qt Sonde ESP32 Weather

version 4.1



Fabien BOUNOIR
Ethan VILLESSECHE

Table des matières

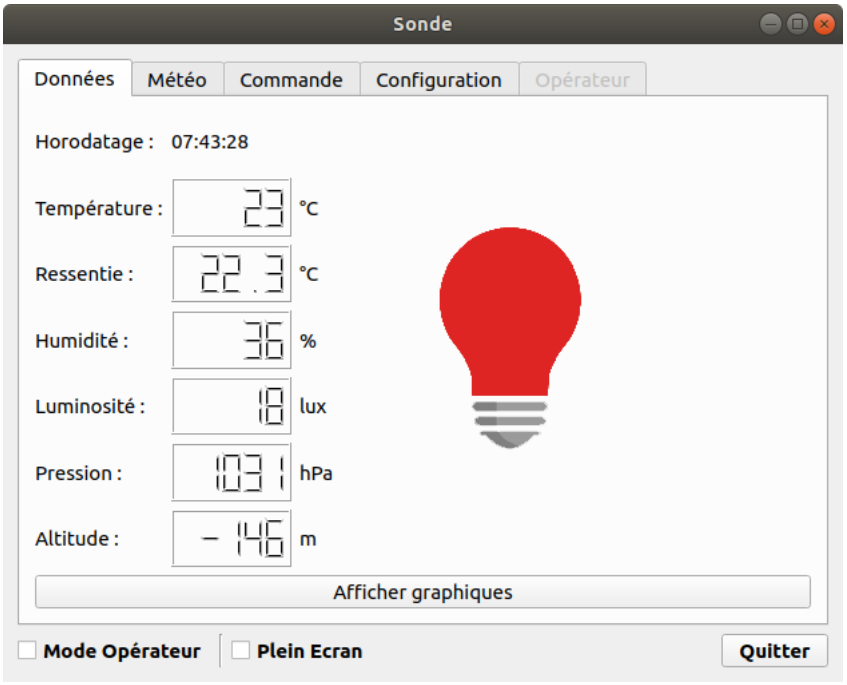
1 Mini-projet Qt Sonde ESP32

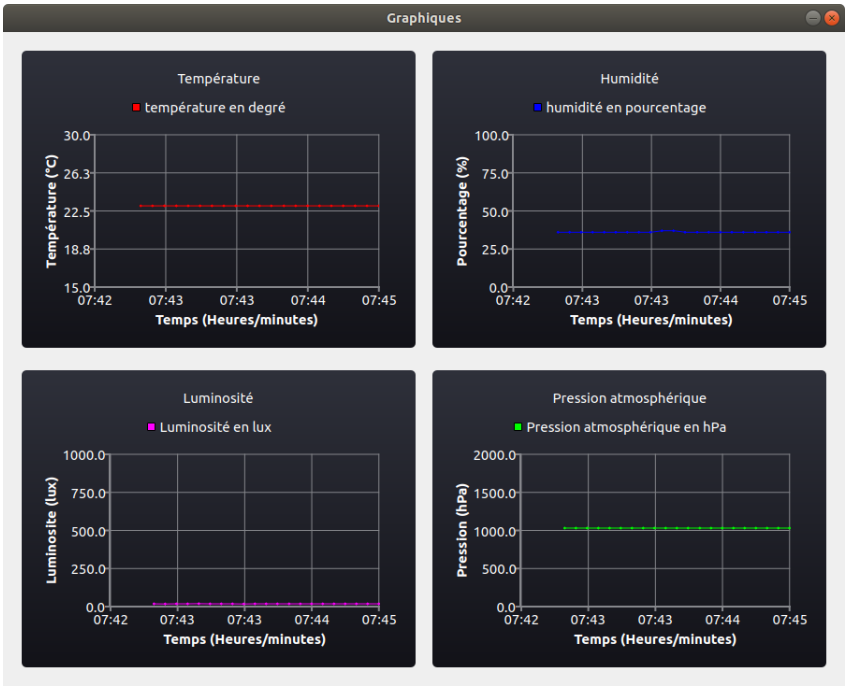
1.1 Table des matières

Programme Qt

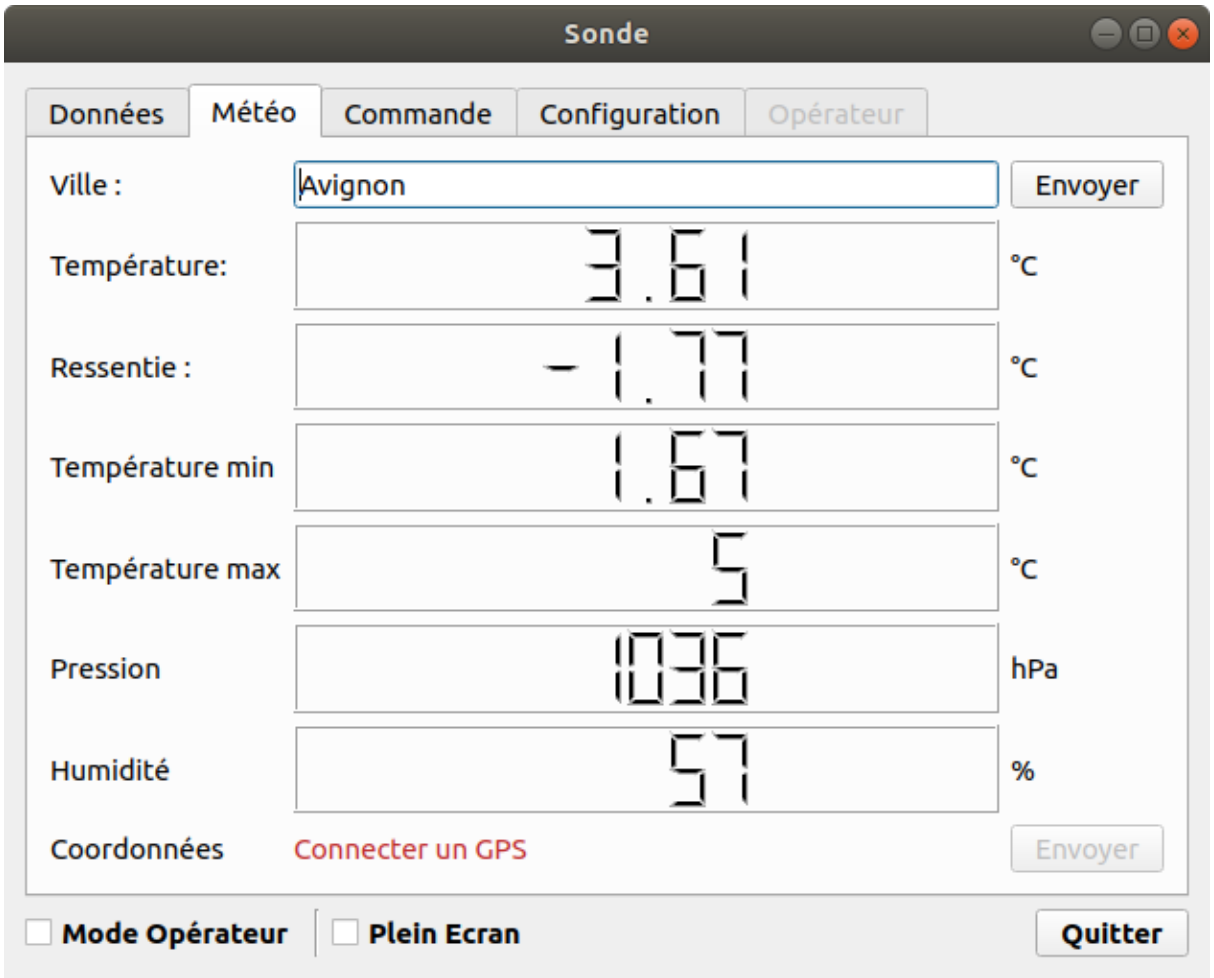
Le programme, réalisé avec le framework Qt 5.11.2, permet de communiquer avec une sonde équipée de différents capteurs.

L'IHM affiche les informations des capteurs dans l'onglet Données. Il est possible de consulter les relevés de mesures sous forme de graphiques.

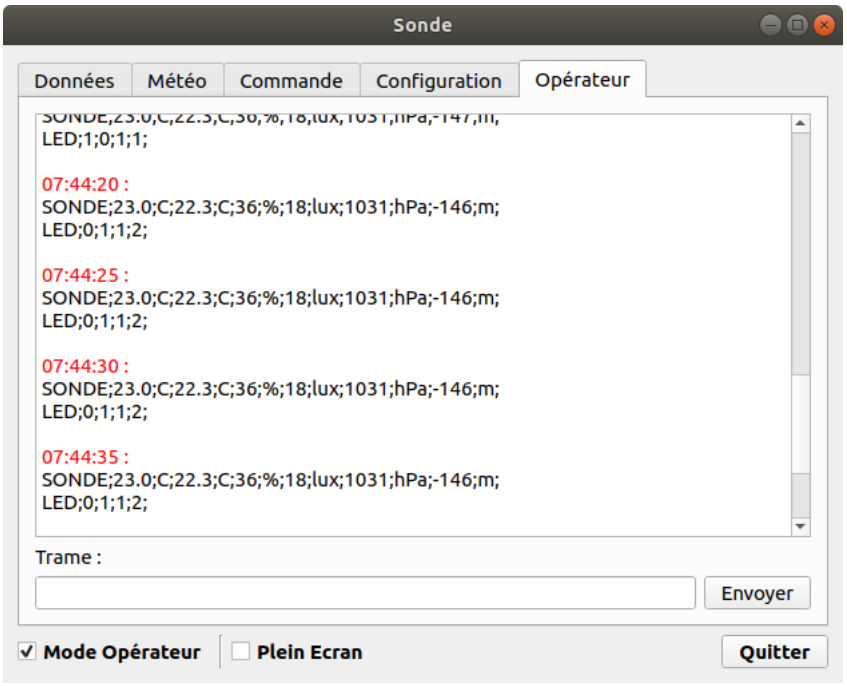




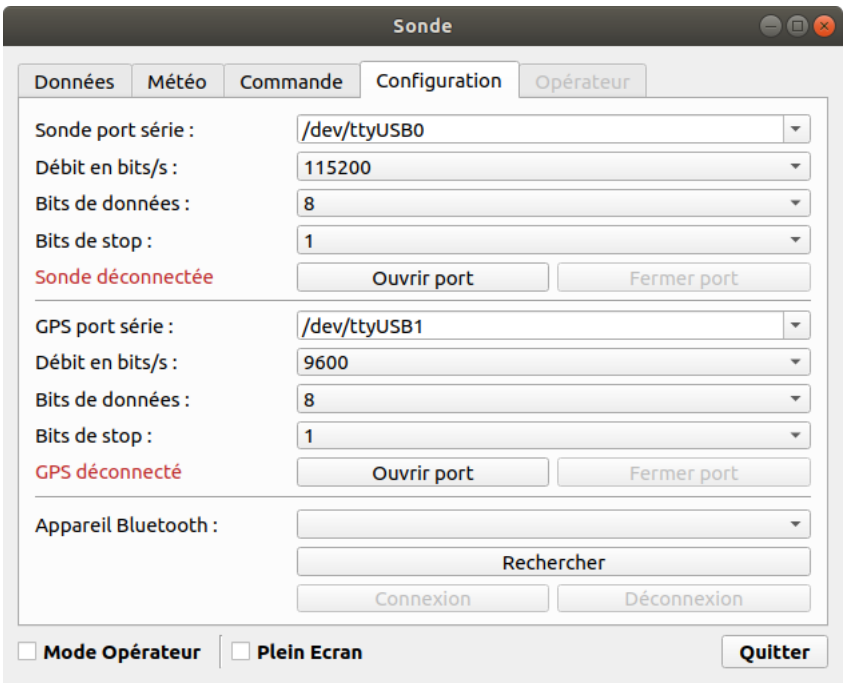
Il est possible de consulter les données météos d'une ville dont il est possible de saisir le nom. Un affichage des coordonnées GPS est dipsonible.



Tous les échanges de trame s'affichent dans l'onglet Opérateur où il est également possible de piloter la led manuellement selon le protocole.



La communication se fera au choix de l'utilisateur, soit par liaison série soit par communication Bluetooth. La communication via WiFi n'est pas implémentée dans ce programme.

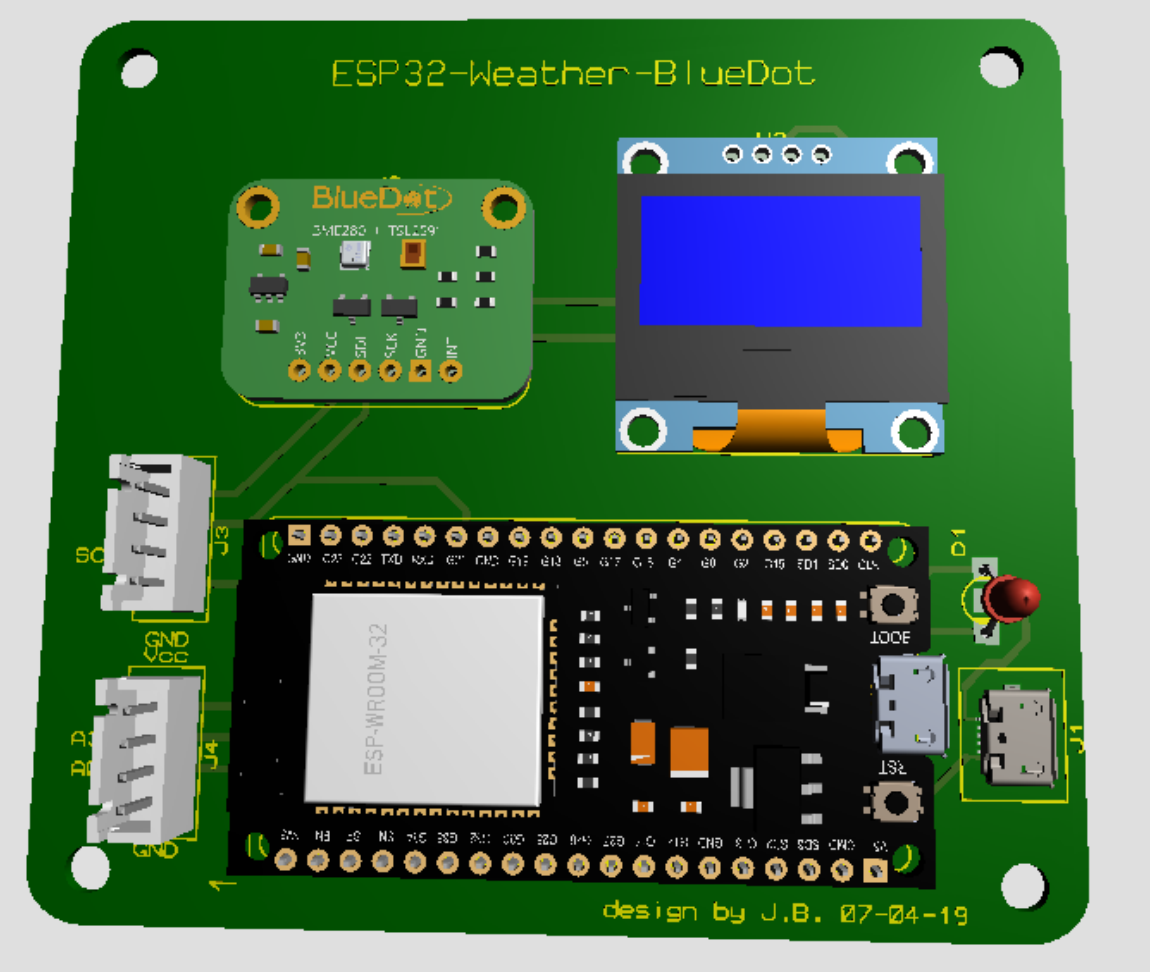


Sonde ESP32-Weather

La carte ESP32-Weather est une sonde construite autour d'un ESP32 et équipée d'un module **BlueDot** I2C, qui intègre un capteur d'éclairement lumineux **TSL 2591** et un capteur **BM↩ E280** (température, humidité et pression atmosphétrique), et d'une Led Bicolore. Les mesures sont affichées périodiquement sur l'écran **OLED** de la carte.

La sonde communique aussi via le WiFi, le Bluetooth et la liaison série. Le même protocole est utilisé pour les trois modes de communication. L'écran de la sonde affiche l'adresse IP et le numéro de port utilisés pour une communication WiFi et l'adresse MAC de l'interface Bluetooth.

La carte a été réalisée par des étudiants d'EC et le programme de l'ESP32 par un professeur.



Auteurs

Fabien Bounoir (IR) bounoirfabien@gmail.com

Ethan Villesseche (IR) villesseche.ethan@gmail.com

2 Changelog

r61 | fbounoir | 2020-01-17 21 :58 :51 +0100 (ven. 17 janv. 2020) | 1 ligne

ajout de la documentation pour le tags 4.0

r60 | fbounoir | 2020-01-17 21 :40 :46 +0100 (ven. 17 janv. 2020) | 1 ligne

création du tag 4.0

r59 | fbounoir | 2020-01-17 21 :35 :06 +0100 (ven. 17 janv. 2020) | 1 ligne

finalisation 4.0

r58 | fbounoir | 2020-01-16 23 :50 :36 +0100 (jeu. 16 janv. 2020) | 1 ligne

ajout requete pour le meteo avec coordonner Gps

r57 | fbounoir | 2020-01-15 23 :31 :28 +0100 (mer. 15 janv. 2020) | 1 ligne

creation des different graphique (temperature,humiditer,luminosite,pression)

r56 | fbounoir | 2020-01-15 20 :35 :47 +0100 (mer. 15 janv. 2020) | 1 ligne

ajout diagramme temperature / humidite / pression / luminosite

r55 | fbounoir | 2020-01-15 16 :40 :57 +0100 (mer. 15 janv. 2020) | 1 ligne

creation fenetre graphique

r54 | fbounoir | 2020-01-14 21 :51 :04 +0100 (mar. 14 janv. 2020) | 1 ligne

ajout de la determination de la localisation au demarrage du logiciel pour actualiser les données de meteo

r53 | fbounoir | 2020-01-13 19 :42 :22 +0100 (lun. 13 janv. 2020) | 1 ligne

correction orthographe

r52 | fbounoir | 2020-01-10 16 :32 :14 +0100 (ven. 10 janv. 2020) | 1 ligne

ajout de la documentation 3.0

r51 | fbounoir | 2020-01-10 16 :27 :39 +0100 (ven. 10 janv. 2020) | 1 ligne

creation du tag 3.0

r50 | fbounoir | 2020-01-10 16 :25 :24 +0100 (ven. 10 janv. 2020) | 1 ligne

correction de bug

r49 | fbounoir | 2020-01-10 11 :22 :04 +0100 (ven. 10 janv. 2020) | 1 ligne

communication avec l'esp 32 ebn bluetooth

r48 | fbounoir | 2020-01-09 22 :33 :09 +0100 (jeu. 09 janv. 2020) | 1 ligne

possibiliter de scanner appareil disponible

r47 | fbounoir | 2020-01-09 17 :41 :42 +0100 (jeu. 09 janv. 2020) | 1 ligne

correction bug recuperation donnée api Openweather

r46 | fbounoir | 2020-01-08 21 :56 :05 +0100 (mer. 08 janv. 2020) | 1 ligne

decomposition du Json recu et affichage dans l'Ihm

r45 | fbounoir | 2020-01-08 19 :06 :41 +0100 (mer. 08 janv. 2020) | 1 ligne

ajout get et set classe Meteo

r44 | fbounoir | 2020-01-08 16 :21 :47 +0100 (mer. 08 janv. 2020) | 1 ligne

requete api

r43 | fbounoir | 2020-01-08 11 :18 :02 +0100 (mer. 08 janv. 2020) | 1 ligne

ajout image led

r42 | fbounoir | 2020-01-03 13 :15 :58 +0100 (ven. 03 janv. 2020) | 1 ligne

mise a jour TODO

r41 | fbounoir | 2020-01-03 12 :19 :24 +0100 (ven. 03 janv. 2020) | 1 ligne

rectification remarque 2.1

r40 | fbounoir | 2019-12-30 11 :22 :10 +0100 (lun. 30 déc. 2019) | 1 ligne

correction README

r39 | fbounoir | 2019-12-30 11 :17 :25 +0100 (lun. 30 déc. 2019) | 1 ligne

ajout de la documentation tags 2.1

r38 | fbounoir | 2019-12-30 11 :13 :23 +0100 (lun. 30 déc. 2019) | 1 ligne

création du tag 2.1

r37 | fbounoir | 2019-12-30 11 :10 :41 +0100 (lun. 30 déc. 2019) | 1 ligne

creation documentation

r36 | fbounoir | 2019-12-29 22 :12 :55 +0100 (dim. 29 déc. 2019) | 1 ligne

ajout enregistrement configuration port dans un fichier INI

r35 | fbounoir | 2019-12-23 21 :16 :06 +0100 (lun. 23 déc. 2019) | 1 ligne

correction probleme nom methode

r34 | fbounoir | 2019-12-23 19 :05 :39 +0100 (lun. 23 déc. 2019) | 1 ligne

les qDebug ne s'affiche plus en release

r33 | fbounoir | 2019-12-23 15 :57 :50 +0100 (lun. 23 déc. 2019) | 1 ligne

ajout documentation version 2.0

r32 | fbounoir | 2019-12-23 15 :52 :40 +0100 (lun. 23 déc. 2019) | 1 ligne

creation du tag 2.0

r31 | fbounoir | 2019-12-23 15 :48 :57 +0100 (lun. 23 déc. 2019) | 1 ligne

mise a jour pour rendu 2.0

r30 | fbounoir | 2019-12-23 15 :44 :45 +0100 (lun. 23 déc. 2019) | 1 ligne

correction bug affichage

r29 | fbounoir | 2019-12-22 19 :25 :55 +0100 (dim. 22 déc. 2019) | 1 ligne

ajout choisir port de communication

r28 | fbounoir | 2019-12-22 17 :23 :44 +0100 (dim. 22 déc. 2019) | 1 ligne

ajout possibiliter controler Led

r27 | fbounoir | 2019-12-22 12 :07 :16 +0100 (dim. 22 déc. 2019) | 1 ligne

modification Ihm onglet Operateur pour changer etat Led

r26 | fbounoir | 2019-12-22 11 :03 :09 +0100 (dim. 22 déc. 2019) | 1 ligne

ajout possibiliter d'ouvrir et fermer port

r25 | fbounoir | 2019-12-21 12 :20 :37 +0100 (sam. 21 déc. 2019) | 1 ligne

ajout horodatage

r24 | fbounoir | 2019-12-20 16 :07 :11 +0100 (ven. 20 déc. 2019) | 1 ligne

ajout documentation de devellopement

r23 | fbounoir | 2019-12-20 16 :02 :44 +0100 (ven. 20 déc. 2019) | 1 ligne

creation du tag 1.0

r22 | fbounoir | 2019-12-20 16 :00 :08 +0100 (ven. 20 déc. 2019) | 1 ligne

mise a jour fichier Doxyfile

r21 | fbounoir | 2019-12-20 12 :45 :47 +0100 (ven. 20 déc. 2019) | 1 ligne

ajout commentaire de code

r20 | fbounoir | 2019-12-20 12 :37 :50 +0100 (ven. 20 déc. 2019) | 1 ligne

ajout du fichier doxyfile

r19 | evillesseche | 2019-12-20 11 :45 :30 +0100 (ven. 20 déc. 2019) | 1 ligne

mise a jour du roadbook

r18 | fbounoir | 2019-12-20 11 :45 :11 +0100 (ven. 20 déc. 2019) | 1 ligne

modification Ihm

r17 | fbounoir | 2019-12-19 21 :17 :14 +0100 (jeu. 19 déc. 2019) | 1 ligne

ajout commentaire de code

r16 | fbounoir | 2019-12-19 15 :44 :09 +0100 (jeu. 19 déc. 2019) | 1 ligne

traitement de la trame

r15 | fbounoir | 2019-12-18 22 :51 :44 +0100 (mer. 18 déc. 2019) | 1 ligne

getter et setter classe Esp32

r14 | fbounoir | 2019-12-18 22 :50 :36 +0100 (mer. 18 déc. 2019) | 1 ligne

decomposition trame

r13 | fbounoir | 2019-12-18 16 :26 :12 +0100 (mer. 18 déc. 2019) | 1 ligne

initialisation port

r12 | evillesseche | 2019-12-18 16 :25 :13 +0100 (mer. 18 déc. 2019) | 1 ligne

mise a jour du roadbook

r11 | evillesseche | 2019-12-18 14 :50 :19 +0100 (mer. 18 déc. 2019) | 1 ligne

declaration des attribus de la classe transmission

r10 | fbounoir | 2019-12-18 14 :42 :13 +0100 (mer. 18 déc. 2019) | 1 ligne

creation methode configurerPort

r9 | evillesseche | 2019-12-18 14 :32 :39 +0100 (mer. 18 déc. 2019) | 1 ligne

declaration des attribut de la classe esp32

r8 | fbounoir | 2019-12-18 14 :15 :05 +0100 (mer. 18 déc. 2019) | 1 ligne

correction liaison entre classe

r7 | fbounoir | 2019-12-18 11 :46 :36 +0100 (mer. 18 déc. 2019) | 1 ligne

liaison entre classe

r6 | fbounoir | 2019-12-13 13 :54 :18 +0100 (ven. 13 déc. 2019) | 1 ligne

ajout onglet configuration

r5 | fbounoir | 2019-12-13 13 :40 :49 +0100 (ven. 13 déc. 2019) | 1 ligne

creation lhm

r4 | fbounoir | 2019-12-13 12 :37 :28 +0100 (ven. 13 déc. 2019) | 1 ligne

ajout fichier Qt design

r3 | fbounoir | 2019-12-12 16 :20 :12 +0100 (jeu. 12 déc. 2019) | 1 ligne

mise a jour ROADBOOK

r2 | fbounoir | 2019-12-12 15 :58 :42 +0100 (jeu. 12 déc. 2019) | 1 ligne

creation classe lhm

r1 | www-data | 2019-12-10 17 :46 :26 +0100 (mar. 10 déc. 2019) | 1 ligne

Creating initial repository structure

3 README

Nom : Mini-projet Qt Sonde ESP32 (BTS SN-IR La Salle Avignon)

Numéro de version : 4.1

Auteurs

Fabien Bounoir (IR) bounoirfabien@gmail.com

Ethan Villesseche (IR) villesseche.ethan@gmail.com

Description

Le programme a été réalisé avec le framework Qt 5.11.2.

Fichier `.pro` :

```
QT       += core gui \
            serialport network \
            bluetooth \
            positioning \
            charts

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = Sonde
TEMPLATE = app

CONFIG += c++11

SOURCES += \
        main.cpp \
        ihm.cpp \
    transmission.cpp \
    esp32.cpp \
    meteo.cpp \
    graphique.cpp \
    gps.cpp

HEADERS += \
        ihm.h \
    transmission.h \
    esp32.h \
    meteo.h \
    graphique.h \
    gps.h

FORMS += \
        ihm.ui

CONFIG(release, debug|release):DEFINES+=QT_NO_DEBUG_OUTPUT
```

Protocole

```
SONDE;TEMPERATURE;UNITE;RESSENTIE;UNITE;HUMIDITE;UNITE;ECLAIREMENT;UNITE;PRESSION;UNITE;ALTITUDE;UNITE;\n
LED;ETAT LED ROUGE;ETAT LED VERTE;ETAT;COULEUR;\n
```

Exemple :

```
SONDE;20.8;C;20.0;C;41;%;997;lux;1007;hPa;52;m;\n -> Température 20,8 °C, Ressentie 20 °C, Humidité 41 %,
       un éclairement de 997 lux, une pression atmosphérique de 1007 hPa et d'une altitude évaluée à 52 m
LED;1;0;1;1;\n -> Le Led est allumée en rouge
```

Remarques :

- Les valeurs de température sont précisées au dixième de degré.
- Un booléen est égal à 0 pour false et 1 pour true.
- Les codes de couleur pour la Led sont :
  - Aucune (éteinte) = 0
  - Rouge = 1
  - Verte = 2
  - Orange = 3

Les clients connectés ont la possibilité d'envoyer une requête pour commander la led :

```
SET LED commande\n
```

Le champ commande peut prendre les valeurs suivantes :

```
SET LED ON\n -> allume la Led dans sa couleur courante
SET LED OFF\n -> éteint la Led
SET LED 0\n -> éteint la Led
SET LED 1\n -> allume la Led en rouge
SET LED 2\n -> allume la Led en vert
SET LED 3\ -> allume la Led en orange
SET LED ROUGE\n -> allume la Led en rouge
SET LED VERT\n -> allume la Led en vert
SET LED VERTE\n -> allume la Led en vert
SET LED ORANGE\n -> allume la Led en orange
```

Remarque : la requête est insensible à la casse.

4 A propos

Auteur

Fabien Bounoir (IR) bounoirfabien@gmail.com
Ethan Villesseche (IR) villesseche.ethan@gmail.com

Version

4.0

Date

2020

5 Licence GPL

6 Liste des choses à faire

Classe Transmission
Implémenter la communication WiFi avec la sonde

7 Documentation des espaces de nommage

7.1 Référence de l'espace de nommage Ui

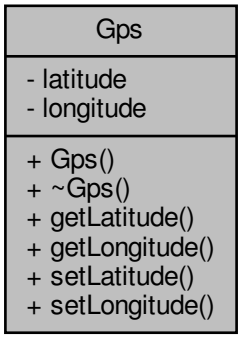
8 Documentation des classes

8.1 Référence de la classe Gps

Declaration de la classe Gps.

```
#include "gps.h"
```

Graphe de collaboration de Gps :



Fonctions membres publiques

- Gps ()
  constructeur de la classe Gps
- ~Gps ()
  destructeur de la classe Gps
- double getLatitude () const
  retourne la valeur de la latitude
- double getLongitude () const
  retourne la valeur de la longitude
- void setLatitude (double latitude)
  modifie la valeur de la latitude
- void setLongitude (double longitude)
  modifie la valeur de la longitude

Attributs privés

- double latitude
  variable qui stocke la latitude
- double longitude
  variable qui stocke la longitude

8.1.1 Documentation des constructeurs et destructeur

8.1.1.1 Gps()

```
Gps::Gps ( )
```

```
00020         : latitude(0.), longitude(0.)
00021 {
00022
00023 }
```

8.1.1.2 ~Gps()

```
Gps::~Gps ( )
```

constructeur de la classe GPS

```
00031 {
00032
00033 }
```

8.1.2 Documentation des fonctions membres

8.1.2.1 getLatitude()

```
double Gps::getLatitude ( ) const
```

destructeur de la classe GPS

Renvoie

double

Références latitude.

Référencé par Ihm : :actualiserDonneeGps(), et Ihm : :on_pushButtonEnvoyerCoordonnee_↩clicked().

```
00053 {
00054     return latitude;
00055 }
```

8.1.2.2 getLongitude()

```
double Gps::getLongitude ( ) const
```

recuperer la latitude

Renvoie

double

Références longitude.

Référencé par Ihm : :actualiserDonneeGps(), et Ihm : :on_pushButtonEnvoyerCoordonnee_↩clicked().

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

8.1.2.3 setLatitude()

```
void Gps::setLatitude (
            double latitude )
```

recuperer la longitune

Paramètres

latitude	

Références latitude.

Référencé par Transmission : :decomposerDonneeGps().

```
00064 {
00065     this->latitude = latitude;
00066 }
```

8.1.2.4 setLongitude()

```
void Gps::setLongitude (
            double longitude )
```

fixe la valeur de la latitude

Paramètres

longitude	

Références longitude.

Référencé par Transmission : :decomposerDonneeGps().

```
00075 {
00076     this->longitude = longitude;
00077 }
```

8.1.3 Documentation des données membres

8.1.3.1 latitude

`double Gps::latitude [private]`

fixe la valeur de la longitude

Référencé par getLatitude(), et setLatitude().

8.1.3.2 longitude

`double Gps::longitude [private]`

Référencé par getLongitude(), et setLongitude().

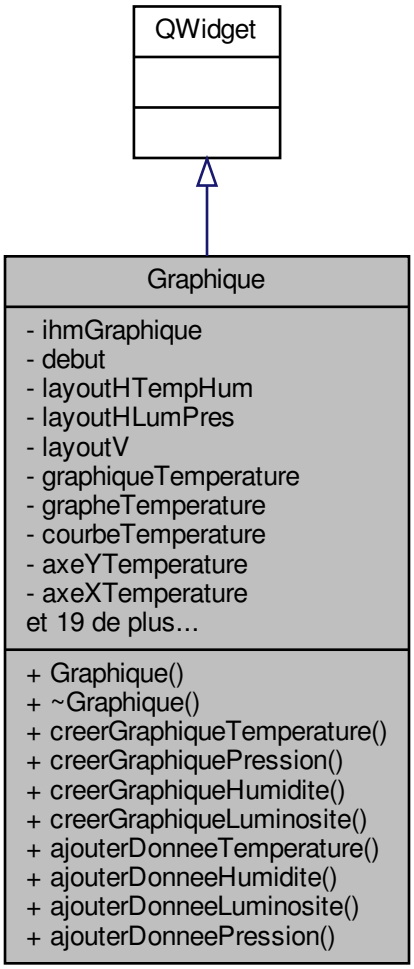
La documentation de cette classe a été générée à partir des fichiers suivants :
- gps.h
- gps.cpp

8.2 Référence de la classe Graphique

Declaration de la classe Graphique.

`#include "graphique.h"`

Graphe de collaboration de Graphique :



Fonctions membres publiques

- Graphique (QWidget *parent=nullptr)
  le constructeur de la classe Graphique
- ~Graphique ()
  le destructeur de la classe Graphique
- void creerGraphiqueTemperature ()
  methode qui cree le graphique pour la temperature
- void creerGraphiquePression ()
  methode qui cree le graphique pour la pression atmosphérique
- void creerGraphiqueHumidite ()
  methode qui cree le graphique pour l'humidité
- void creerGraphiqueLuminosite ()
  methode qui cree le graphique pour la luminosité
- void ajouterDonneeTemperature (double temperature)
  Méthode appelée pour ajouter des valeurs au graphique températures.
- void ajouterDonneeHumidite (int humidite)
  Méthode appelée pour ajouter des valeurs au graphique humidité

— void ajouterDonneeLuminosite (int luminosite)
Méthode appelée pour ajouter des valeurs au graphique luminosité
— void ajouterDonneePression (int pression)
Méthode appelée pour ajouter des valeurs au graphique pression.

Attributs privés

— QWidget * ihmGraphique
le widget qui contient les graphiques
— QDateTime debut
objet date utilisé pour l'axe temps des graphiques
— QHBoxLayout * layoutHTempHum
layout qui contient le graphique temperature et humidité
— QHBoxLayout * layoutHLumPres
layout qui contient le graphique luminosité et pression
— QVBoxLayout * layoutV
layout qui contient les deux layout (layoutHTempHum / layoutHLumPres)
— QChartView * graphiqueTemperature
un widget pour afficher le graphe temperature
— QChart * grapheTemperature
la représentation du graphe temperature
— QLineSeries * courbeTemperature
les données temperatures
— QValueAxis * axeYTemperature
l'axe Y temperature
— QDateTimeAxis * axeXTemperature
l'axe X temperature
— QChartView * graphiqueLuminosite
un widget pour afficher le graphe luminosité
— QChart * grapheLuminosite
la représentation du graphe luminosité
— QLineSeries * courbeLuminosite
les données de luminosité
— QValueAxis * axeYLuminosite
l'axe Y luminosité
— QDateTimeAxis * axeXLuminosite
l'axe X lumonisité
— QChartView * graphiquePression
un widget pour afficher le graphe pression
— QChart * graphePression
la représentation du graphe pression
— QLineSeries * courbePression
les données de pression
— QValueAxis * axeYPression
l'axe Y pression
— QDateTimeAxis * axeXPression
l'axe X pression
— QValueAxis * axeYHumidite
l'axe Y humidite
— QDateTimeAxis * axeXHumidite
l'axe X humidite
— QChart * grapheHumidite
la représentation du graphe humidité
— QChartView * graphiqueHumidite
un widget pour afficher un graphe humidité
— QLineSeries * courbeHumidite
les données d'humidité

— double axeYTemperatureMax
variable qui contient la valeur max de l'axe temperature
— double axeYTemperatureMin
variable qui contient la valeur min de l'axe temperature
— int axeYPressionMax
variable qui contient la valeur max de l'axe pression
— int axeYLuminositeMax
variable qui contient la valeur max de l'axe luminosite

8.2.1 Documentation des constructeurs et destructeur

8.2.1.1 Graphique()

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

Références creerGraphiqueHumidite(), creerGraphiqueLuminosite(), creerGraphiquePression(), creerGraphiqueTemperature(), debut, ihmGraphique, layoutHLumPres, layoutHTempHum, et layoutV.

```
00020                                    : QWidget(parent), axeYTemperatureMax(30),
      axeYTemperatureMin(15), axeYPressionMax(2000),
      axeYLuminositeMax(1000)
00021 {
00022     setWindowTitle("Graphiques");
00023     setFixedSize(900,700);
00024     isFullScreen();
00025     debut = QDateTime::currentDateTime();
00026
00027     layoutV = new QVBoxLayout();
00028     layoutHTempHum = new QHBoxLayout();
00029     layoutHLumPres = new QHBoxLayout();
00030
00031     creerGraphiqueTemperature();
00032     creerGraphiqueHumidite();
00033     creerGraphiqueLuminosite();
00034     creerGraphiquePression();
00035
00036     layoutV->addItem(layoutHTempHum);
00037     layoutV->addItem(layoutHLumPres);
00038
00039     this->setLayout(layoutV);
00040
00041     ihmGraphique = new QWidget(this);
00042 }
```

8.2.1.2 ~Graphique()

```
Graphique::~Graphique ( )
```

constructeur de la classe graphique

Références ihmGraphique.

```
00050 {
00051     delete ihmGraphique;
00052 }
```

8.2.2 Documentation des fonctions membres

8.2.2.1 ajouterDonneeHumidite()

```
void Graphique::ajouterDonneeHumidite (
            int humidite )
```

fonction appelée pour ajouter des valeurs au graphique temperature

Paramètres

humidite	

Références axeXHumidite, courbeHumidite, debut, grapheHumidite, et graphiqueHumidite.

Référencé par Ihm : :actualiserDonnee().

```
00295 {
00296     QDateTime fin;
00297     fin = QDateTime::currentDateTime();
00298
00299     axeXHumidite->setMin(debut);
00300     courbeHumidite->append(fin.toMSecsSinceEpoch(), humidite);
00301     axeXHumidite->setMax(fin);
00302
00303     graphiqueHumidite->setChart(grapheHumidite);
00304
00305
00306 }
```

8.2.2.2 ajouterDonneeLuminosite()

```
void Graphique::ajouterDonneeLuminosite (
            int luminosite )
```

fonction appelée pour ajouter des valeurs au graphique humidite

Paramètres

luminosite	

Références axeXLuminosite, axeYLuminosite, axeYLuminositeMax, courbeLuminosite, debut, grapheLuminosite, et graphiqueLuminosite.

Référencé par Ihm : :actualiserDonnee().

```
00315 {
00316     QDateTime fin;
00317     fin = QDateTime::currentDateTime();
00318
00319     axeXLuminosite->setMin(debut);
00320     if(luminosite > axeYLuminositeMax)
00321     {
00322         axeYLuminosite->setRange(0, luminosite);
00323         axeYLuminositeMax = luminosite;
00324     }
```

```
00325     courbeLuminosite->append(fin.toMSecsSinceEpoch(), luminosite);
00326     axeXLuminosite->setMax(fin);
00327
00328     graphiqueLuminosite->setChart(grapheLuminosite);
00329     graphiqueLuminosite->update();
00330
00331 }
```

8.2.2.3 ajouterDonneePression()

```
void Graphique::ajouterDonneePression (
          int pression )
```

fonction appelée pour ajouter des valeurs au graphique luminosité

Paramètres

pression	

Références axeXPression, axeYPression, axeYPressionMax, courbePression, debut, graphePression, et graphiquePression.

Référencé par Ihm : :actualiserDonnee().

```
00340 {
00341     QDateTime fin;
00342     fin = QDateTime::currentDateTime();
00343
00344     axeXPression->setMin(debut);
00345     if(pression > axeYPressionMax)
00346     {
00347         axeYPression->setRange(0, pression);
00348         axeYPressionMax = pression;
00349     }
00350     courbePression->append(fin.toMSecsSinceEpoch(), pression);
00351     axeXPression->setMax(fin);
00352
00353     graphiquePression->setChart(grapheLuminosite);
00354     graphiquePression->update();
00355 }
```

8.2.2.4 ajouterDonneeTemperature()

```
void Graphique::ajouterDonneeTemperature (
          double temperature )
```

fonction appelée pour creer le graphique luminosité

Paramètres

temperature	

Références axeXTemperature, axeYTemperature, axeYTemperatureMax, axeYTemperatureMin, courbeTemperature, debut, grapheTemperature, et graphiqueTemperature.

Référencé par Ihm : :actualiserDonnee().

```
00266 {
00267     QDateTime fin;
00268     fin = QDateTime::currentDateTime();
00269
00270     axeXTemperature->setMin(debut);
00271     if(temperature > axeYTemperatureMax)
00272     {
00273         axeYTemperature->setRange(axeYTemperatureMin, temperature);
00274         axeYTemperatureMax = temperature;
00275     }
00276     else if(temperature < axeYTemperatureMin)
00277     {
00278         axeYTemperature->setRange(temperature, axeYTemperatureMax);
00279         axeYTemperatureMin = temperature;
00280     }
00281     courbeTemperature->append(fin.toMSecsSinceEpoch(), temperature);
00282     axeXTemperature->setMax(fin);
00283
00284     graphiqueTemperature->setChart(grapheTemperature);
00285
00286 }
```

8.2.2.5 creerGraphiqueHumidite()

`void Graphique::creerGraphiqueHumidite ( )`

fonction appelée pour creer le graphique pression

Références axeXHumidite, axeYHumidite, courbeHumidite, debut, grapheHumidite, graphique↩ Humidite, et layoutHTempHum.

Référencé par Graphique().

```
00111 {
00112
00113     // Les données
00114     courbeHumidite = new QLineSeries(this);
00115     courbeHumidite->setName(QString::fromUtf8("humidité en pourcentage"));
00116     courbeHumidite->setColor(Qt::blue);
00117
00118     // Le grapheHumidite
00119     grapheHumidite = new QChart();
00120     grapheHumidite->setTitle("Humidité");
00121     grapheHumidite->setTheme(QChart::ChartThemeDark);
00122     grapheHumidite->addSeries(courbeHumidite);
00123
00124     //QDateTimeAxis
00125     axeXHumidite = new QDateTimeAxis(this);
00126     axeXHumidite->setTickCount(5);
00127     axeXHumidite->setFormat("hh:mm");
00128     axeXHumidite->setMin(debut);
00129     axeXHumidite->setTitleText("Temps (Heures/minutes)");
00130
00131     grapheHumidite->addAxis(axeXHumidite, Qt::AlignBottom);
00132     courbeHumidite->attachAxis(axeXHumidite);
00133
00134     axeYHumidite = new QValueAxis(this);
00135     axeYHumidite->setRange(0, 100);
00136     axeYHumidite->setLabelFormat("%.1f");
00137     axeYHumidite->setTitleText(QString::fromUtf8("Pourcentage (%)"));
00138     grapheHumidite->addAxis(axeYHumidite, Qt::AlignLeft);
00139     courbeHumidite->setPointsVisible(true);
00140
00141     //afficher les points sur la courbe
00142     //courbeHumidite->setPointLabelsFormat("@yPoint %");
00143     //courbeHumidite->setPointLabelsVisible(true);
00144
00145     courbeHumidite->attachAxis(axeYHumidite);
00146
00147     // Le widget
```

```
00148     graphiqueHumidite = new QChartView(grapheHumidite);
00149     graphiqueHumidite->setRenderHint(QPainter::Antialiasing);
00150
00151     resize(300, 200);
00152
00153     layoutHTempHum->addWidget(graphiqueHumidite);
00154 }
```

8.2.2.6 creerGraphiqueLuminosite()

`void Graphique::creerGraphiqueLuminosite ( )`

fonction appelée pour creer le graphique humidite

Références axeXLuminosite, axeYLuminosite, axeYLuminositeMax, courbeLuminosite, debut, grapheLuminosite, graphiqueLuminosite, et layoutHLumPres.

Référencé par Graphique().

```
00162 {
00163
00164     // Les données
00165     courbeLuminosite = new QLineSeries(this);
00166     courbeLuminosite->setName(QString::fromUtf8("Luminosité en lux"));
00167     courbeLuminosite->setColor(Qt::magenta);
00168
00169
00170     // Le grapheLuminosite
00171     grapheLuminosite = new QChart();
00172     grapheLuminosite->setTitle("Luminosité");
00173     grapheLuminosite->setTheme(QChart::ChartThemeDark);
00174     grapheLuminosite->addSeries(courbeLuminosite);
00175
00176     //QDateTimeAxis
00177     axeXLuminosite = new QDateTimeAxis(this);
00178     axeXLuminosite->setTickCount(5);
00179     axeXLuminosite->setFormat("hh:mm");
00180     axeXLuminosite->setMin(debut);
00181     axeXLuminosite->setTitleText("Temps (Heures/minutes)");
00182
00183     grapheLuminosite->addAxis(axeXLuminosite, Qt::AlignBottom);
00184     courbeLuminosite->attachAxis(axeXLuminosite);
00185
00186     axeYLuminosite = new QValueAxis(this);
00187     axeYLuminosite->setRange(0, axeYLuminositeMax);
00188     axeYLuminosite->setLabelFormat("%.1f");
00189     axeYLuminosite->setTitleText(QString::fromUtf8("Luminosite (lux)"));
00190     grapheLuminosite->addAxis(axeYLuminosite, Qt::AlignLeft);
00191     courbeLuminosite->setPointsVisible(true);
00192
00193     //afficher les points sur la courbe
00194     //courbeLuminosite->setPointLabelsFormat("@yPoint Lux");
00195     //courbeLuminosite->setPointLabelsVisible(true);
00196
00197     courbeLuminosite->attachAxis(axeYLuminosite);
00198
00199     // Le widget
00200     graphiqueLuminosite = new QChartView(grapheLuminosite);
00201     graphiqueLuminosite->setRenderHint(QPainter::Antialiasing);
00202
00203     resize(300, 200);
00204
00205     layoutHLumPres->addWidget(graphiqueLuminosite);
00206 }
```

8.2.2.7 creerGraphiquePression()

`void Graphique::creerGraphiquePression ( )`

fonction appelée pour creer le graphique temperature

Références axeXPression, axeYPression, axeYPressionMax, courbePression, debut, graphePression, graphiquePression, et layoutHLumPres.

Référencé par Graphique().

```
00214 {
00215
00216     // Les données
00217     courbePression = new QLineSeries(this);
00218     courbePression->setName(QString::fromUtf8("Pression atmosphérique en hPa"));
00219     courbePression->setColor(Qt::green);
00220
00221     // Le graphePression
00222     graphePression = new QChart();
00223     graphePression->setTitle("Pression atmosphérique");
00224     graphePression->setTheme(QChart::ChartThemeDark);
00225     graphePression->addSeries(courbePression);
00226
00227     //QDateTimeAxis
00228     axeXPression = new QDateTimeAxis(this);
00229     axeXPression->setTickCount(5);
00230     axeXPression->setFormat("hh:mm");
00231     axeXPression->setMin(debut);
00232     axeXPression->setTitleText("Temps (Heures/minutes)");
00233
00234     graphePression->addAxis(axeXPression, Qt::AlignBottom);
00235     courbePression->attachAxis(axeXPression);
00236
00237     axeYPression = new QValueAxis(this);
00238     axeYPression->setRange(0, axeYPressionMax);
00239     axeYPression->setLabelFormat("%.1f");
00240     axeYPression->setTitleText(QString::fromUtf8("Pression (hPa)"));
00241     graphePression->addAxis(axeYPression, Qt::AlignLeft);
00242     courbePression->setPointsVisible(true);
00243
00244     //afficher les points sur la courbe
00245     //courbePression->setPointLabelsFormat("@yPoint hPa");
00246     //courbePression->setPointLabelsVisible(true);
00247
00248     courbePression->attachAxis(axeYPression);
00249
00250     // Le widget
00251     graphiquePression = new QChartView(graphePression);
00252     graphiquePression->setRenderHint(QPainter::Antialiasing);
00253
00254     resize(300, 200);
00255
00256     layoutHLumPres->addWidget(graphiquePression);
00257 }
```

8.2.2.8 creerGraphiqueTemperature()

`void Graphique::creerGraphiqueTemperature ( )`

destructeur de la classe graphique

Références axeXTemperature, axeYTemperature, axeYTemperatureMax, axeYTemperatureMin, courbeTemperature, debut, grapheTemperature, graphiqueTemperature, et layoutHTempHum.

Référencé par Graphique().

```
00060 {
00061
00062     // Les données
00063     courbeTemperature = new QLineSeries(this);
00064     courbeTemperature->setName(QString::fromUtf8("température en degré"));
00065     courbeTemperature->setColor(Qt::red);
00066
00067     // Le grapheTemperature
00068     grapheTemperature = new QChart();
00069     grapheTemperature->setTitle("Température");
00070     grapheTemperature->setTheme(QChart::ChartThemeDark);
00071     grapheTemperature->addSeries(courbeTemperature);
00072
00073     //QDateTimeAxis
00074     axeXTemperature = new QDateTimeAxis(this);
00075     axeXTemperature->setTickCount(5);
00076     axeXTemperature->setFormat("hh:mm");
00077     axeXTemperature->setMin(debut);
00078     axeXTemperature->setTitleText("Temps (Heures/minutes)");
00079
00080     grapheTemperature->addAxis(axeXTemperature, Qt::AlignBottom);
00081     courbeTemperature->attachAxis(axeXTemperature);
00082
00083     axeYTemperature = new QValueAxis(this);
00084     axeYTemperature->setRange(axeYTemperatureMin,
      axeYTemperatureMax);
00085     axeYTemperature->setLabelFormat("%.1f");
00086     axeYTemperature->setTitleText(QString::fromUtf8("Température (°C)"));
00087     grapheTemperature->addAxis(axeYTemperature, Qt::AlignLeft);
00088     courbeTemperature->setPointsVisible(true);
00089
00090     //afficher les points sur la courbe
00091     //courbeTemperature->setPointLabelsFormat("@yPoint °C");
00092     //courbeTemperature->setPointLabelsVisible(true);
00093
00094     courbeTemperature->attachAxis(axeYTemperature);
00095
00096     // Le widget
00097     graphiqueTemperature = new QChartView(grapheTemperature);
00098     graphiqueTemperature->setRenderHint(QPainter::Antialiasing);
00099
00100     resize(300, 200);
00101
00102     layoutHTempHum->addWidget(graphiqueTemperature);
00103 }
```

8.2.3 Documentation des données membres

8.2.3.1 axeXHumidite

`QDateTimeAxis* Graphique::axeXHumidite [private]`

Référencé par ajouterDonneeHumidite(), et creerGraphiqueHumidite().

8.2.3.2 axeXLuminosite

`QDateTimeAxis* Graphique::axeXLuminosite [private]`

Référencé par ajouterDonneeLuminosite(), et creerGraphiqueLuminosite().

8.2.3.3 axeXPression

`QDateTimeAxis* Graphique::axeXPression [private]`

Référencé par ajouterDonneePression(), et creerGraphiquePression().

8.2.3.4 axeXTemperature

```
QDateTimeAxis* Graphique::axeXTemperature [private]
```

Référencé par ajouterDonneeTemperature(), et creerGraphiqueTemperature().

8.2.3.5 axeYHumidite

```
QValueAxis* Graphique::axeYHumidite [private]
```

Référencé par creerGraphiqueHumidite().

8.2.3.6 axeYLuminosite

```
QValueAxis* Graphique::axeYLuminosite [private]
```

Référencé par ajouterDonneeLuminosite(), et creerGraphiqueLuminosite().

8.2.3.7 axeYLuminositeMax

```
int Graphique::axeYLuminositeMax [private]
```

Référencé par ajouterDonneeLuminosite(), et creerGraphiqueLuminosite().

8.2.3.8 axeYPression

```
QValueAxis* Graphique::axeYPression [private]
```

Référencé par ajouterDonneePression(), et creerGraphiquePression().

8.2.3.9 axeYPressionMax

```
int Graphique::axeYPressionMax [private]
```

Référencé par ajouterDonneePression(), et creerGraphiquePression().

8.2.3.10 axeYTemperature

```
QValueAxis* Graphique::axeYTemperature [private]
```

Référencé par ajouterDonneeTemperature(), et creerGraphiqueTemperature().

8.2.3.11 axeYTemperatureMax

```
double Graphique::axeYTemperatureMax [private]
```

Référencé par ajouterDonneeTemperature(), et creerGraphiqueTemperature().

8.2.3.12 axeYTemperatureMin

```
double Graphique::axeYTemperatureMin [private]
```

Référencé par ajouterDonneeTemperature(), et creerGraphiqueTemperature().

8.2.3.13 courbeHumidite

```
QLineSeries* Graphique::courbeHumidite [private]
```

Référencé par ajouterDonneeHumidite(), et creerGraphiqueHumidite().

8.2.3.14 courbeLuminosite

```
QLineSeries* Graphique::courbeLuminosite [private]
```

Référencé par ajouterDonneeLuminosite(), et creerGraphiqueLuminosite().

8.2.3.15 courbePression

```
QLineSeries* Graphique::courbePression [private]
```

Référencé par ajouterDonneePression(), et creerGraphiquePression().

8.2.3.16 courbeTemperature

```
QLineSeries* Graphique::courbeTemperature [private]
```

Référencé par ajouterDonneeTemperature(), et creerGraphiqueTemperature().

8.2.3.17 debut

```
QDateTime Graphique::debut [private]
```

Référencé par ajouterDonneeHumidite(), ajouterDonneeLuminosite(), ajouterDonneePression(), ajouterDonneeTemperature(), creerGraphiqueHumidite(), creerGraphiqueLuminosite(), creer↩ GraphiquePression(), creerGraphiqueTemperature(), et Graphique().

8.2.3.18 grapheHumidite

```
QChart* Graphique::grapheHumidite [private]
```

Référencé par ajouterDonneeHumidite(), et creerGraphiqueHumidite().

8.2.3.19 grapheLuminosite

```
QChart* Graphique::grapheLuminosite [private]
```

Référencé par ajouterDonneeLuminosite(), et creerGraphiqueLuminosite().

8.2.3.20 graphePression

```
QChart* Graphique::graphePression [private]
```

Référencé par ajouterDonneePression(), et creerGraphiquePression().

8.2.3.21 grapheTemperature

```
QChart* Graphique::grapheTemperature [private]
```

Référencé par ajouterDonneeTemperature(), et creerGraphiqueTemperature().

8.2.3.22 graphiqueHumidite

```
QChartView* Graphique::graphiqueHumidite [private]
```

Référencé par ajouterDonneeHumidite(), et creerGraphiqueHumidite().

8.2.3.23 graphiqueLuminosite

```
QChartView* Graphique::graphiqueLuminosite [private]
```

Référencé par ajouterDonneeLuminosite(), et creerGraphiqueLuminosite().

8.2.3.24 graphiquePression

```
QChartView* Graphique::graphiquePression [private]
```

Référencé par ajouterDonneePression(), et creerGraphiquePression().

8.2.3.25 graphiqueTemperature

```
QChartView* Graphique::graphiqueTemperature [private]
```

Référencé par ajouterDonneeTemperature(), et creerGraphiqueTemperature().

8.2.3.26 ihmGraphique

```
QWidget* Graphique::ihmGraphique [private]
```

fonction appelée pour ajouter des valeurs au graphique pression

Référencé par Graphique(), et ~Graphique().

8.2.3.27 layoutHLumPres

```
QHBoxLayout* Graphique::layoutHLumPres [private]
```

Référencé par creerGraphiqueLuminosite(), creerGraphiquePression(), et Graphique().

8.2.3.28 layoutHTempHum

```
QHBoxLayout* Graphique::layoutHTempHum [private]
```

Référencé par creerGraphiqueHumidite(), creerGraphiqueTemperature(), et Graphique().

8.2.3.29 layoutV

```
QVBoxLayout* Graphique::layoutV [private]
```

Référencé par Graphique().

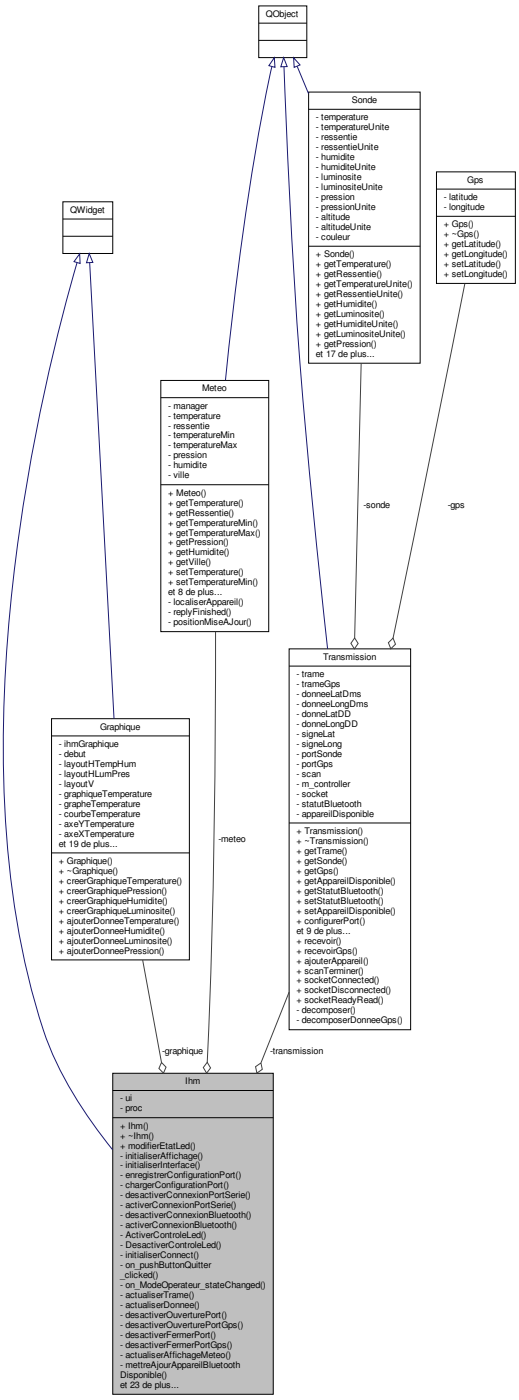
La documentation de cette classe a été générée à partir des fichiers suivants :
- graphique.h
- graphique.cpp

8.3 Référence de la classe Ihm

Declaration de la classe Ihm.

```
#include "ihm.h"
```

Graphe de collaboration de Ihm :



Fonctions membres publiques

— Ihm (QWidget *parent=nullptr)
constructeur de la classe Ihm
— ~Ihm ()
destructeur de la classe Ihm
— void modifierEtatLed ()
Méthode qui actualise l'etat de la led dans ihm et change le radiobutton activer en fontion de l'etat.

Connecteurs privés

- void on_pushButtonQuitter_clicked ()
  Méthode appelée lorsque le Bouton quitté est enclenché
- void on_ModeOperateur_stateChanged (int arg1)
  Méthode appelée lorsque le case à cocher Mode operateur est enclenché
- void actualiserTrame ()
  ajout la nouvelle trame recu dans le mode operateur
- void actualiserDonnee ()
- void desactiverOuverturePort ()
  desactive des bouton de l'Ihm
- void desactiverOuverturePortGps ()
  desactive des bouton de l'Ihm
- void desactiverFermerPort ()
  desactive des bouton de l'Ihm
- void desactiverFermerPortGps ()
  desactive des bouton de l'Ihm
- void actualiserAffichageMeteo ()
  fonction qui actualise les différente donné de la météo
- void mettreAjourAppareilBluetoothDisponible ()
  fonction qui met a jour les appareil bluetooth disponible
- void actualiserAffichageBluetooth ()
  fonction qui actualise l'affichage de la parti bluetooth
- void actualiserMessageStatutConnecterBluetooth ()
  fonction qui actualiser dans l'ihm le statut de la connexion avec l'appareil
- void actualiserMessageStatutDeconnecterBluetooth ()
  met à jour le message de statut pour afficher que l'appareil est déconnecter
- void actualiserMessageStatutErreurBluetooth ()
  met à jour le message de statut pour afficher une erreur
- void on_pushButtonOuvrirPort_clicked ()
  bouton qui ouvre le port serie
- void on_pushButtonFermerPort_clicked ()
  bouton qui ferme le port serie
- void on_pushButtonEnvoyerTrame_clicked ()
  recuperer le texte de la lineEdit dans l'onglet operateur pour l'envoyer par le port serie
- void on_radioButtonLedRouge_clicked ()
  envoit une trame pour allumer la led en rouge
- void on_radioButtonLedVert_clicked ()
  envoit une trame pour allumer la led en vert
- void on_radioButtonLedOrange_clicked ()
  envoit une trame pour allumer la led en orange
- void on_radioButtonLedOff_clicked ()
  envoit une trame pour eteindre la led
- void on_pushButtonVille_clicked ()
  fonction qui envoit la ville contenue dans la line edit
- void on_pushButtonScan_clicked ()
  fonction appelée quand le bouton scan est cliqué
- void on_pushButtonConnexion_clicked ()
  fonction appelée quand le bouton connexion est cliqué
- void on_pushButtonDeconnexion_clicked ()
  Méthode appelée quand on clique sur le bouton deconnexion elle réactive la possibilité de se connecter en port série.
- void on_pushButtonGraphique_clicked ()
  Méthode appelée dès que le push bouton graphique est enclenché
- void on_checkBoxPleinEcran_stateChanged (int arg1)
  Méthode appelée dès que l'etat du checkBox plein ecran change.
- void actualiserDonneeGps ()
  Méthode qui met a jour la latitude et longitude dans l'Ihm.

— void on_pushButtonEnvoyerCoordonnee_clicked ()
bouton qui actualise l'etat des données meteo grace à la latitude et longitude
— void on_pushButtonOuvrirPortGps_clicked ()
bouton qui permet d'ouvrir la communication pour le port serie du GPS
— void on_pushButtonFermerPortGps_clicked ()
bouton qui permet de fermer la communication pour le port serie du GPS
— void actualiserMessageStatutConnectionGps (QString message)
met à jour le statut de connexion du GPS
— void actualiserMessageStatutConnectionSonde (QString message)
met à jour le statut de connexion de la Sonde

Fonctions membres privées

— void initialiserAffichage ()
remet les lcdNumber a zero quand le port est fermer
— void initialiserInterface ()
initialise l'interface utilisateur
— void enregistrerConfigurationPort ()
enregistrer la configuration du port dans un fichier .ini
— void chargerConfigurationPort ()
charger la configuration du port depuis un fichier .ini
— void desactiverConnexionPortSerie ()
active des objet de l'interface quand la connexion bluetooth est activer
— void activerConnexionPortSerie ()
désactive des objet de l'interface quand la connexion bluetooth est activer
— void desactiverConnexionBluetooth ()
active des objet de l'interface quand la connexion bluetooth est activer
— void activerConnexionBluetooth ()
désactive des objet de l'interface quand la connexion bluetooth est activer
— void ActiverControleLed ()
active la possibilité de contrôler les leds
— void DesactiverControleLed ()
Désactiver la possibilité de contrôler les leds.
— void initialiserConnect ()
initialise les connects

Attributs privés

— Ui : :Ihm * ui
objet Ihm
— Transmission * transmission
objet transmission
— Meteo * meteo
objet meteo
— Graphique * graphique
objet graphique ;
— QProcess * proc
pointer proc

8.3.1 Documentation des constructeurs et destructeur

8.3.1.1 Ihm()

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

Paramètres

parent	

Références graphique, initialiserAffichage(), initialiserConnect(), initialiserInterface(), meteo, proc, transmission, et ui.

```
00024                       :
00025     QWidget(parent),
00026     ui(new Ui::Ihm)
00027 {
00028     ui->setupUi(this);
00029
00030     initialiserInterface();
00031     initialiserAffichage();
00032
00033     transmission = new Transmission(this);
00034     meteo = new Meteo(this);
00035     graphique = new Graphique();
00036     proc = new QProcess();
00037
00038     initialiserConnect();
00039 }
```

8.3.1.2 ~Ihm()

```
Ihm::~Ihm ( )
```

constructeur de la classe Ihm

Références enregistrerConfigurationPort(), graphique, et ui.

```
00047 {
00048     enregistrerConfigurationPort();
00049     delete graphique;
00050     delete ui;
00051 }
```

8.3.2 Documentation des fonctions membres

8.3.2.1 activerConnexionBluetooth()

```
void Ihm::activerConnexionBluetooth ( )  [private]
```

desactive la possibilité de connexion par bluetooth car une connexion par port serie est en cours

Références ui.

```
00663 {
00664     ui->comboBoxBluetooth->setDisabled(false);
00665     ui->pushButtonScan->setDisabled(false);
00666     ui->pushButtonConnexion->setDisabled(false);
00667     ui->pushButtonDeconnexion->setDisabled(false);
00668     ui->labelStatut->setText("");
00669 }
```

8.3.2.2 activerConnexionPortSerie()

`void Ihm::activerConnexionPortSerie ( ) [private]`

desactive la possibilité de connexion par port serie car une connexion bluetooth est en cours

Références ui.

```
00634 {
00635     ui->comboBoxAppareil->setDisabled(false);
00636     ui->comboBoxDebitBaud->setDisabled(false);
00637     ui->comboBoxBitsDonnees->setDisabled(false);
00638     ui->comboBoxBitsStop->setDisabled(false);
00639     ui->pushButtonOuvrirPort->setDisabled(false);
00640 }
```

8.3.2.3 ActiverControleLed()

`void Ihm::ActiverControleLed ( ) [private]`

active la possibilité de connexion par bluetooth car aucun connexion en cours

Références ui.

Référencé par actualiserMessageStatutConnecterBluetooth(), et desactiverOuverturePort().

```
00300 {
00301     ui->radioButtonLedOff->setDisabled(false);
00302     ui->radioButtonLedVert->setDisabled(false);
00303     ui->radioButtonLedRouge->setDisabled(false);
00304     ui->radioButtonLedOrange->setDisabled(false);
00305 }
```

8.3.2.4 actualiserAffichageBluetooth

`void Ihm::actualiserAffichageBluetooth ( ) [private], [slot]`

methode qui actualise les appareil disponible

Références ui.

Référencé par initialiserConnect().

```
00532 {
00533     ui->pushButtonScan->setEnabled(true);
00534     ui->pushButtonConnexion->setEnabled(true);
00535     ui->labelStatut->setText("<font color=\"#19B2AD\">Recherche terminée</font>");
00536 }
```

8.3.2.5 actualiserAffichageMeteo

`void Ihm::actualiserAffichageMeteo ( ) [private], [slot]`

Méthode appelée quand le port serie du GPS est fermé

Références Meteo : :getHumidite(), Meteo : :getPression(), Meteo : :getRessentie(), Meteo : :getTemperature(), Meteo : :getTemperatureMax(), Meteo : :getTemperatureMin(), Meteo : :getVille(), meteo, et ui.

Référencé par initialiserConnect().

```
00422 {
00423     ui->lineEditVille->setText(meteo->getVille());
00424     ui->lcdNumberTemperatureMeteo->display(meteo->getTemperature());
00425     ui->lcdNumberHumiditeMeteo->display(meteo->getHumidite());
00426     ui->lcdNumberRessentieMeteo->display(meteo->getRessentie());
00427     ui->lcdNumberPressionMeteo->display(meteo->getPression());
00428     ui->lcdNumberTempMinMeteo->display(meteo->getTemperatureMin());
00429     ui->lcdNumberTempMaxMeteo->display(meteo->getTemperatureMax());
00430 }
```

8.3.2.6 actualiserDonnee

`void Ihm::actualiserDonnee ( ) [private], [slot]`

met a jour l'affichage de la trame en mode Operateur lors du signal trameRecue()

Références Graphique : :ajouterDonneeHumidite(), Graphique : :ajouterDonneeLuminosite(), Graphique : :ajouterDonneePression(), Graphique : :ajouterDonneeTemperature(), Sonde : :getAltitude(), Sonde : :getAltitudeUnite(), Sonde : :getHumidite(), Sonde : :getHumiditeUnite(), Sonde : :getLuminosite(), Sonde : :getLuminositeUnite(), Sonde : :getPression(), Sonde : :getPressionUnite(), Sonde : :getRessentie(), Sonde : :getRessentieUnite(), Transmission : :getSonde(), Sonde : :getTemperature(), Sonde : :getTemperatureUnite(), graphique, modifierEtatLed(), transmission, et ui.

Référencé par initialiserConnect().

```
00246 {
00247     QDateTime maintenant = QDateTime::currentDateTime();
00248     QString heure = maintenant.toString("hh:mm:ss");
00249     ui->horodatage->setText(heure);
00250 
00251     ui->lcdNumberTemperature->display(transmission->getSonde()->
      getTemperature());
00252     ui->lcdNumberHumidite->display(transmission->getSonde()->
      getHumidite());
00253     ui->lcdNumberRessentie->display(transmission->getSonde()->
      getRessentie());
00254     ui->lcdNumberLuminosite->display(transmission->getSonde()->
      getLuminosite());
00255     ui->lcdNumberPression->display(transmission->getSonde()->
      getPression());
00256     ui->lcdNumberAltitude->display(transmission->getSonde()->
      getAltitude());
00257 
00258     ui->labelUniteTemperature->setText(transmission->getSonde()->
      getTemperatureUnite());
00259     ui->labelUniteRessenti->setText(transmission->getSonde()->
      getRessentieUnite());
00260     ui->labelUniteHumidite->setText(transmission->getSonde()->
      getHumiditeUnite());
00261     ui->labelUniteLuminosite->setText(transmission->getSonde()->
      getLuminositeUnite());
```

```
00262     ui->labelUnitePression->setText(transmission->getSonde()->
      getPressionUnite());
00263     ui->labelUniteAltitude->setText(transmission->getSonde()->
      getAltitudeUnite());
00264
00265     modifierEtatLed();
00266
00267     graphique->ajouterDonneeTemperature(
      transmission->getSonde()->getTemperature());
00268     graphique->ajouterDonneeHumidite(transmission->
      getSonde()->getHumidite());
00269     graphique->ajouterDonneeLuminosite(
      transmission->getSonde()->getLuminosite());
00270     graphique->ajouterDonneePression(transmission->
      getSonde()->getPression());
00271 }
```

8.3.2.7 actualiserDonneeGps

`void Ihm::actualiserDonneeGps ( ) [private], [slot]`

Références Transmission : :getGps(), Gps : :getLatitude(), Gps : :getLongitude(), transmission, et ui.

Référencé par initialiserConnect().

```
00705 {
00706     ui->labelValeurCoordonnee->setText(QString::number(transmission->
      getGps()->getLatitude()) + " " + QString::number(transmission->
      getGps()->getLongitude()));
00707     ui->pushButtonEnvoyerCoordonnee->setEnabled(true);
00708 }
```

8.3.2.8 actualiserMessageStatutConnecterBluetooth

`void Ihm::actualiserMessageStatutConnecterBluetooth ( ) [private], [slot]`

methode qui met à jour l'affichage des boutons (enable / disable)

Références ActiverControleLed(), Transmission : :getStatutBluetooth(), proc, transmission, et ui.

Référencé par initialiserConnect().

```
00558 {
00559     ui->labelStatut->setText("<font color=\"#f39c12\">" + transmission->
      getStatutBluetooth() + "</font>");
00560
00561     proc->execute("notify-send --icon=/usr/share/icons/hicolor/48x48/apps/bluetooth.png \"" +
      transmission->getStatutBluetooth() + "\" ");
00562
00563     ui->pushButtonConnexion->setEnabled(false);
00564     ui->pushButtonScan->setEnabled(false);
00565     ui->pushButtonDeconnexion->setEnabled(true);
00566     ui->lineEditEnvoyerTrame->setDisabled(false);
00567     ui->pushButtonEnvoyerTrame->setDisabled(false);
00568     //desactiverConnexionPortSerie();
00569     ActiverControleLed();
00570 }
```

8.3.2.9 actualiserMessageStatutConnectionGps

```
void Ihm::actualiserMessageStatutConnectionGps (
          QString message ) [private], [slot]
```

Références Transmission : :fermerPortGps(), transmission, et ui.

Référencé par initialiserConnect().

```
00731 {
00732     transmission->fermerPortGps();
00733     ui->labelStatutPortSerieGps->setText("<font color=\"#8e44ad\">" + message + "</font>");
00734 }
```

8.3.2.10 actualiserMessageStatutConnectionSonde

```
void Ihm::actualiserMessageStatutConnectionSonde (
          QString message ) [private], [slot]
```

Références Transmission : :fermerPort(), transmission, et ui.

Référencé par initialiserConnect().

```
00737 {
00738     transmission->fermerPort();
00739     ui->labelStatutPortSerieSonde->setText("<font color=\"#8e44ad\">" + message + "</font>");
00740 }
```

8.3.2.11 actualiserMessageStatutDeconnecterBluetooth

```
void Ihm::actualiserMessageStatutDeconnecterBluetooth ( ) [private], [slot]
```

methode qui met à jour le message de statut de la connexion bluetooth (appareil connecté)

Références DesactiverControleLed(), Transmission : :getStatutBluetooth(), initialiserAffichage(), proc, transmission, et ui.

Référencé par initialiserConnect().

```
00578 {
00579     ui->labelStatut->setText("<font color=\"#7329b9\">" + transmission->
      getStatutBluetooth() + "</font>");
00580
00581     proc->execute("notify-send --icon=/usr/share/icons/hicolor/48x48/apps/bluetooth.png \"" +
      transmission->getStatutBluetooth() + "\" ");
00582
00583     ui->pushButtonConnexion->setEnabled(true);
00584     ui->pushButtonScan->setEnabled(true);
00585     ui->pushButtonDeconnexion->setEnabled(false);
00586     ui->lineEditEnvoyerTrame->setDisabled(true);
00587     ui->pushButtonEnvoyerTrame->setDisabled(true);
00588     initialiserAffichage();
00589     DesactiverControleLed();
00590 }
```

8.3.2.12 actualiserMessageStatutErreurBluetooth

`void Ihm::actualiserMessageStatutErreurBluetooth ( ) [private], [slot]`

methode qui met à jour le message de statut de la connexion bluetooth (appareil deconnecté)

Références Transmission : :getStatutBluetooth(), transmission, et ui.

Référencé par initialiserConnect().

```
00598 {
00599     ui->labelStatut->setText("<font color=\"#7329b9\">" + transmission->
      getStatutBluetooth() + "</font>");
00600 }
```

8.3.2.13 actualiserTrame

`void Ihm::actualiserTrame ( ) [private], [slot]`

permet de passer en mode Operateur

Références Transmission : :getTrame(), transmission, et ui.

Référencé par initialiserConnect().

```
00200 {
00201     QDateTime maintenant = QDateTime::currentDateTime();
00202     QString heure = maintenant.toString("hh:mm:ss");
00203
00204     ui->textEditTrame->setTextColor(Qt::red);
00205     ui->textEditTrame->append(heure + " :");
00206     ui->textEditTrame->setTextColor(Qt::black);
00207     ui->textEditTrame->append(transmission->getTrame());
00208 }
```

8.3.2.14 chargerConfigurationPort()

`void Ihm::chargerConfigurationPort ( ) [private]`

enregistre la configuration du port dans un fichier .ini

Références ui.

Référencé par initialiserInterface().

```
00110 {
00111     QSettings configuration("Sonde.ini", QSettings::IniFormat);
00112
00113     ui->comboBoxAppareil->setCurrentText(configuration.value("PortSerieSonde/Appareil","/dev/ttyUSB0").
      toString());
00114     ui->comboBoxDebitBaud->setCurrentText(configuration.value("PortSerieSonde/DebitBauds","115200").
      toString());
00115     ui->comboBoxBitsDonnees->setCurrentText(configuration.value("PortSerieSonde/BitsDonnee","8").toString
      ());
00116     ui->comboBoxBitsStop->setCurrentText(configuration.value("PortSerieSonde/BitsStop","1").toString());
00117
00118     ui->comboBoxPortSerieGps->setCurrentText(configuration.value("PortSerieGps/Appareil","/dev/ttyUSB1").
      toString());
00119     ui->comboBoxDebitGps->setCurrentText(configuration.value("PortSerieGps/DebitBauds","9600").toString()
      );
00120     ui->comboBoxBitsDonneesGps->setCurrentText(configuration.value("PortSerieGps/BitsDonnee","8").
      toString());
00121     ui->comboBoxBitsStopGps->setCurrentText(configuration.value("PortSerieGps/BitsStop","1").toString());
00122 }
```

8.3.2.15 desactiverConnexionBluetooth()

`void Ihm::desactiverConnexionBluetooth ( ) [private]`

active la possibilité de connexion par port serie car aucun connexion en cours

Références ui.

```
00648 {
00649     ui->comboBoxBluetooth->setDisabled(true);
00650     ui->pushButtonScan->setDisabled(true);
00651     ui->pushButtonConnexion->setDisabled(true);
00652     ui->pushButtonDeconnexion->setDisabled(true);
00653     ui->labelStatut->setText("<font color=\"#63cb23\">Connexion en cours ...</font>");
00654
00655 }
```

8.3.2.16 desactiverConnexionPortSerie()

`void Ihm::desactiverConnexionPortSerie ( ) [private]`

charge la configuration du port depuis un fichier .ini

Références ui.

```
00619 {
00620     ui->comboBoxAppareil->setDisabled(true);
00621     ui->comboBoxDebitBaud->setDisabled(true);
00622     ui->comboBoxBitsDonnees->setDisabled(true);
00623     ui->comboBoxBitsStop->setDisabled(true);
00624     ui->pushButtonOuvrirPort->setDisabled(true);
00625     ui->pushButtonFermerPort->setDisabled(true);
00626 }
```

8.3.2.17 DesactiverControleLed()

`void Ihm::DesactiverControleLed ( ) [private]`

active le controle de la led quand une connexion est en cours

Références ui.

Référencé par actualiserMessageStatutDeconnecterBluetooth(), et desactiverFermerPort().

```
00354 {
00355     ui->radioButtonLedOff->setDisabled(true);
00356     ui->radioButtonLedVert->setDisabled(true);
00357     ui->radioButtonLedRouge->setDisabled(true);
00358     ui->radioButtonLedOrange->setDisabled(true);
00359 }
```

8.3.2.18 desactiverFermerPort

`void Ihm::desactiverFermerPort ( ) [private], [slot]`

Méthode appelée quand le port serie du GPS est ouvert

Références DesactiverControleLed(), initialiserAffichage(), et ui.

Référencé par initialiserConnect().

```
00368 {
00369     ui->pushButtonOuvrirPort->setDisabled(false);
00370     ui->pushButtonFermerPort->setDisabled(true);
00371     ui->lineEditEnvoyerTrame->setDisabled(true);
00372     ui->pushButtonEnvoyerTrame->setDisabled(true);
00373     initialiserAffichage();
00374     ui->labelStatutPortSerieSonde->setText("<font color=\"#bd2c2c\">Sonde déconnectée</font>");
00375     ui->comboBoxAppareil->setDisabled(false);
00376     ui->comboBoxBitsDonnees->setDisabled(false);
00377     ui->comboBoxBitsStop->setDisabled(false);
00378     ui->comboBoxDebitBaud->setDisabled(false);
00379     DesactiverControleLed();
00380 }
```

8.3.2.19 desactiverFermerPortGps

`void Ihm::desactiverFermerPortGps ( ) [private], [slot]`

Méthode appelée quand un port serie est fermé

Références ui.

Référencé par initialiserConnect().

```
00388 {
00389     ui->pushButtonOuvrirPortGps->setDisabled(false);
00390     ui->pushButtonFermerPortGps->setDisabled(true);
00391     ui->labelValeurCoordonnee->setText("<font color=\"#bd2c2c\">Connecter un GPS</font>");
00392     ui->labelStatutPortSerieGps->setText("<font color=\"#bd2c2c\">GPS déconnecté</font>");
00393     ui->comboBoxPortSerieGps->setDisabled(false);
00394     ui->comboBoxBitsDonneesGps->setDisabled(false);
00395     ui->comboBoxBitsStopGps->setDisabled(false);
00396     ui->comboBoxDebitGps->setDisabled(false);
00397 }
```

8.3.2.20 desactiverOuverturePort

`void Ihm::desactiverOuverturePort ( ) [private], [slot]`

met a jour les données dans l'onglet Données en fonction de la trame recue

Références ActiverControleLed(), et ui.

Référencé par initialiserConnect().

```
00313 {
00314     ui->pushButtonOuvrirPort->setDisabled(true);
00315     ui->pushButtonFermerPort->setDisabled(false);
00316     ui->lineEditEnvoyerTrame->setDisabled(false);
00317     ui->pushButtonEnvoyerTrame->setDisabled(false);
00318     ui->lineEditEnvoyerTrame->setDisabled(false);
00319     ui->pushButtonEnvoyerTrame->setDisabled(false);
```

```
00320     ui->comboBoxAppareil->setDisabled(true);
00321     ui->comboBoxBitsDonnees->setDisabled(true);
00322     ui->comboBoxBitsStop->setDisabled(true);
00323     ui->comboBoxDebitBaud->setDisabled(true);
00324     ui->labelStatutPortSerieSonde->setText("<font color=\"#3498db\">Sonde connectée</font>");
00325
00326     ActiverControleLed();
00327 }
```

8.3.2.21 desactiverOuverturePortGps

`void Ihm::desactiverOuverturePortGps ( ) [private], [slot]`

Méthode appelée quand un port serie de la Sonde est ouvert

Références ui.

Référencé par initialiserConnect().

```
00335 {
00336     ui->pushButtonOuvrirPortGps->setDisabled(true);
00337     ui->pushButtonFermerPortGps->setDisabled(false);
00338     ui->labelStatutPortSerieGps->setText("<font color=\"#3498db\">GPS connecté</font>");
00339
00340     ui->comboBoxPortSerieGps->setDisabled(true);
00341     ui->comboBoxBitsDonneesGps->setDisabled(true);
00342     ui->comboBoxBitsStopGps->setDisabled(true);
00343     ui->comboBoxDebitGps->setDisabled(true);
00344 }
```

8.3.2.22 enregistrerConfigurationPort()

`void Ihm::enregistrerConfigurationPort ( ) [private]`

désactive certains Boutons ne pouvant être utilisés

Références ui.

Référencé par ~Ihm().

```
00086 {
00087     QSettings configuration("Sonde.ini", QSettings::IniFormat);
00088
00089     configuration.beginGroup("PortSerieSonde");
00090     configuration.setValue("Appareil", ui->comboBoxAppareil->currentText());
00091     configuration.setValue("DebitBauds", ui->comboBoxDebitBaud->currentText());
00092     configuration.setValue("BitsDonnee", ui->comboBoxBitsDonnees->currentText());
00093     configuration.setValue("BitsStop", ui->comboBoxBitsStop->currentText());
00094     configuration.endGroup();
00095
00096     configuration.beginGroup("PortSerieGps");
00097     configuration.setValue("Appareil", ui->comboBoxPortSerieGps->currentText());
00098     configuration.setValue("DebitBauds", ui->comboBoxDebitGps->currentText());
00099     configuration.setValue("BitsDonnee", ui->comboBoxBitsDonneesGps->currentText());
00100     configuration.setValue("BitsStop", ui->comboBoxBitsStopGps->currentText());
00101     configuration.endGroup();
00102 }
```

8.3.2.23 initialiserAffichage()

```
void Ihm::initialiserAffichage ( )  [private]
```

Références LED_OFF, et ui.

Référencé par actualiserMessageStatutDeconnecterBluetooth(), desactiverFermerPort(), et Ihm().

```
00405 {
00406     ui->lcdNumberTemperature->display("----");
00407     ui->lcdNumberHumidite->display("----");
00408     ui->lcdNumberRessentie->display("----");
00409     ui->lcdNumberLuminosite->display("----");
00410     ui->lcdNumberPression->display("----");
00411     ui->lcdNumberAltitude->display("----");
00412
00413     ui->ImageEtatLed->setPixmap(QPixmap(LED_OFF));
00414 }
```

8.3.2.24 initialiserConnect()

```
void Ihm::initialiserConnect ( )  [private]
```

desactive le controle de la led car aucun connexion en cours

Références actualiserAffichageBluetooth(), actualiserAffichageMeteo(), actualiserDonnee(), actualiserDonneeGps(), actualiserMessageStatutConnecterBluetooth(), actualiserMessageStatut↩ ConnectionGps(), actualiserMessageStatutConnectionSonde(), actualiserMessageStatutDeconnecter↩ Bluetooth(), actualiserMessageStatutErreurBluetooth(), actualiserTrame(), desactiverFermer↩ Port(), desactiverFermerPortGps(), desactiverOuverturePort(), desactiverOuverturePortGps(), meteo, mettreAjourAppareilBluetoothDisponible(), et transmission.

Référencé par Ihm().

```
00059 {
00060     connect(transmission, SIGNAL(trameSondeRecue()), this, SLOT(
      actualiserTrame()));
00061     connect(transmission, SIGNAL(trameSondeRecue()), this, SLOT(
      actualiserDonnee()));
00062     connect(transmission, SIGNAL(trameGpsRecue()), this, SLOT(
      actualiserTrame()));
00063     connect(transmission, SIGNAL(trameGpsRecue()), this, SLOT(
      actualiserDonneeGps()));
00064
00065     connect(transmission, SIGNAL(portOuvert()), this, SLOT(
      desactiverOuverturePort()));
00066     connect(transmission, SIGNAL(portOuvertGps()), this, SLOT(
      desactiverOuverturePortGps()));
00067     connect(transmission, SIGNAL(portFerme()), this, SLOT(
      desactiverFermerPort()));
00068     connect(transmission, SIGNAL(portFermeGps()), this, SLOT(
      desactiverFermerPortGps()));
00069
00070     connect(meteo, SIGNAL(donnerMeteoMiseAJour()), this, SLOT(
      actualiserAffichageMeteo()));
00071     connect(transmission, SIGNAL(nouvelleAppareilDisponible()), this, SLOT(
      mettreAjourAppareilBluetoothDisponible()));
00072     connect(transmission, SIGNAL(scanfini()), this, SLOT(
      actualiserAffichageBluetooth()));
00073     connect(transmission, SIGNAL(connecter()), this, SLOT(
      actualiserMessageStatutConnecterBluetooth()));
00074     connect(transmission, SIGNAL(deconnecter()), this, SLOT(
      actualiserMessageStatutDeconnecterBluetooth()));
00075     connect(transmission, SIGNAL(socketErreur()), this, SLOT(
      actualiserMessageStatutErreurBluetooth()));
00076     connect(transmission, SIGNAL(erreurConnectionPortSerieGps(QString)), this, SLOT(
```

```
      actualiserMessageStatutConnectionGps(QString)));
00077     connect(transmission, SIGNAL(erreurConnectionPortSerieSonde(QString)), this, SLOT(
      actualiserMessageStatutConnectionSonde(QString)));
00078 }
```

8.3.2.25 initialiserInterface()

`void Ihm::initialiserInterface ( ) [private]`

reinitialise l'affichage des données

Références chargerConfigurationPort(), LED_OFF, ONGLET_MODE_OPERATEUR, et ui.

Référencé par Ihm().

```
00130 {
00131     ui->tabWidget->setTabEnabled(ONGLET_MODE_OPERATEUR,false);
00132     ui->textEditTrame->setReadOnly(true);
00133     ui->pushButtonFermerPort->setDisabled(true);
00134     ui->radioButtonLedOff->setDisabled(true);
00135     ui->radioButtonLedVert->setDisabled(true);
00136     ui->radioButtonLedRouge->setDisabled(true);
00137     ui->radioButtonLedOrange->setDisabled(true);
00138     ui->radioButtonLedOff->setChecked(true);
00139     ui->ImageEtatLed->setPixmap(QPixmap(LED_OFF));
00140     ui->labelStatutPortSerieSonde->setText("<font color=\"#bd2c2c\">Sonde déconnectée</font>");
00141     ui->labelStatutPortSerieGps->setText("<font color=\"#bd2c2c\">GPS déconnecté</font>");
00142
00143     ui->comboBoxAppareil->addItems(QStringList{"/dev/ttyUSB0", "/dev/ttyUSB1", "/dev/ttyUSB2", "
      /dev/ttyUSB3", "/dev/ttyS2", "/dev/ttyS3" });
00144     ui->comboBoxDebitBaud->addItems(QStringList{"1200", "1800", "2400", "4800", "9600", "19200", "38400",
       "57600", "115200", "230400", "460800"});
00145     ui->comboBoxBitsDonnees->addItems(QStringList{"5", "6", "7", "8"});
00146     ui->comboBoxBitsStop->addItems(QStringList{"1", "2"});
00147
00148     ui->comboBoxPortSerieGps->addItems(QStringList{"/dev/ttyUSB0", "/dev/ttyUSB1", "/dev/ttyUSB2", "
      /dev/ttyUSB3", "/dev/ttyACM0", "/dev/ttyACM2" });
00149     ui->comboBoxDebitGps->addItems(QStringList{"1200", "1800", "2400", "4800", "9600", "19200", "38400",
      "57600", "115200", "230400", "460800"});
00150     ui->comboBoxBitsDonneesGps->addItems(QStringList{"5", "6", "7", "8"});
00151     ui->comboBoxBitsStopGps->addItems(QStringList{"1", "2"});
00152
00153
00154     ui->pushButtonFermerPortGps->setDisabled(true);
00155     ui->pushButtonConnexion->setDisabled(true);
00156     ui->pushButtonDeconnexion->setDisabled(true);
00157     ui->lineEditEnvoyerTrame->setDisabled(true);
00158     ui->pushButtonEnvoyerTrame->setDisabled(true);
00159     ui->pushButtonEnvoyerCoordonnee->setEnabled(false);
00160     ui->labelValeurCoordonnee->setText("<font color=\"#bd2c2c\">Connecter un GPS</font>");
00161     chargerConfigurationPort();
00162
00163 }
```

8.3.2.26 mettreAjourAppareilBluetoothDisponible

`void Ihm::mettreAjourAppareilBluetoothDisponible ( ) [private], [slot]`

methode qui actualise les données de meteo reçues

Références Transmission : :getAppareilDisponible(), transmission, et ui.

Référencé par initialiserConnect().

```
00505 {
00506      ui->comboBoxBluetooth->clear();
00507      ui->comboBoxBluetooth->addItems(transmission->
      getAppareilDisponible());
00508 }
```

8.3.2.27 modifierEtatLed()

`Ihm::modifierEtatLed ( )`

Méthode qui actualise les données de L'ihm.

destructeur de la classe Ihm

Références Sonde : :getEtatLed(), Transmission : :getSonde(), LED_OFF, LED_ORANGE, LED_ROUGE, LED_VERT, transmission, ui, VALEUR_LED_OFF, VALEUR_LED_ORANGE, VALEUR_LED_ROUGE, et VALEUR_LED_VERT.

Référencé par actualiserDonnee().

```
00216 {
00217     switch (transmission->getSonde()->getEtatLed())
00218           {
00219              case VALEUR_LED_OFF:
00220                 ui->ImageEtatLed->setPixmap(QPixmap(LED_OFF));
00221                 ui->radioButtonLedOff->setChecked(true);
00222                 break;
00223              case VALEUR_LED_ROUGE:
00224                 ui->ImageEtatLed->setPixmap(QPixmap(LED_ROUGE));
00225                 ui->radioButtonLedRouge->setChecked(true);
00226                 break;
00227              case VALEUR_LED_VERT:
00228                 ui->ImageEtatLed->setPixmap(QPixmap(LED_VERT));
00229                 ui->radioButtonLedVert->setChecked(true);
00230                break;
00231              case VALEUR_LED_ORANGE:
00232                 ui->ImageEtatLed->setPixmap(QPixmap(LED_ORANGE));
00233                 ui->radioButtonLedOrange->setChecked(true);
00234                break;
00235              default:
00236                 ui->ImageEtatLed->setPixmap(QPixmap(LED_OFF));
00237                 ui->radioButtonLedOff->setChecked(true);
00238           }
00239 }
```

8.3.2.28 on_checkBoxPleinEcran_stateChanged

`void Ihm::on_checkBoxPleinEcran_stateChanged (`
`int arg1 ) [private], [slot]`

Paramètres

arg1	

```
00688 {
00689     if(arg1 == 2)
00690     {
00691         showFullScreen();
00692     }
```

```
00693     else
00694     {
00695         showNormal();
00696     }
00697 }
```

8.3.2.29 on_ModeOperateur_stateChanged

```
void Ihm::on_ModeOperateur_stateChanged (
            int arg1 ) [private], [slot]
```

action de fermer la widget

Paramètres

arg1	

Références MODE_OPERATEUR_ACTIVER, ONGLET_MODE_OPERATEUR, et ui.

```
00183 {
00184     if(arg1 == MODE_OPERATEUR_ACTIVER)
00185     {
00186         ui->tabWidget->setTabEnabled(ONGLET_MODE_OPERATEUR,true);
00187     }
00188     else
00189     {
00190         ui->tabWidget->setTabEnabled(ONGLET_MODE_OPERATEUR,false);
00191     }
00192 }
```

8.3.2.30 on_pushButtonConnexion_clicked

```
void Ihm::on_pushButtonConnexion_clicked ( ) [private], [slot]
```

Méthode appelée dès que le push bouton Scan est enclenché

Références Transmission : :connecterAppareilBluetooth(), transmission, et ui.

```
00544 {
00545     if(ui->comboBoxBluetooth->currentText() != "")
00546     {
00547         ui->labelStatut->setText("<font color=\"#21618c\">Connexion en cours ...</font>");
00548         transmission->connecterAppareilBluetooth(
      ui->comboBoxBluetooth->currentText());
00549     }
00550 }
```

8.3.2.31 on_pushButtonDeconnexion_clicked

```
void Ihm::on_pushButtonDeconnexion_clicked ( ) [private], [slot]
```

Méthode appelée dès que le push bouton Connexion est enclenché

Références Transmission : :deconnecterAppareilBluetooth(), et transmission.

```
00608 {
00609     transmission->deconnecterAppareilBluetooth();
00610     //activerConnexionPortSerie();
00611 }
```

8.3.2.32 on_pushButtonEnvoyerCoordonnee_clicked

`void Ihm::on_pushButtonEnvoyerCoordonnee_clicked ( ) [private], [slot]`

Références Meteo : :creerUrlCoordonnerGps(), Transmission : :getGps(), Gps : :getLatitude(), Gps : :getLongitude(), meteo, et transmission.

```
00716 {
00717     meteo->creerUrlCoordonnerGps(transmission->
      getGps()->getLatitude(), transmission->getGps()->
      getLongitude());
00718 }
```

8.3.2.33 on_pushButtonEnvoyerTrame_clicked

`void Ihm::on_pushButtonEnvoyerTrame_clicked ( ) [private], [slot]`

Méthode appelée dès que le bouton Fermer port est enclenché

Références Transmission : :envoyerDonnees(), transmission, et ui.

```
00438 {
00439     if(ui->lineEditEnvoyerTrame->text() != "")
00440     {
00441         transmission->envoyerDonnees(ui->lineEditEnvoyerTrame->text());
00442     }
00443 }
```

8.3.2.34 on_pushButtonFermerPort_clicked

`void Ihm::on_pushButtonFermerPort_clicked ( ) [private], [slot]`

Méthode appelée dès que le bouton Ouvrir port est enclenché

Références Transmission : :fermerPort(), et transmission.

```
00290 {
00291     transmission->fermerPort();
00292 }
```

8.3.2.35 on_pushButtonFermerPortGps_clicked

`void Ihm::on_pushButtonFermerPortGps_clicked ( ) [private], [slot]`

Références Transmission : :fermerPortGps(), et transmission.

```
00726 {
00727     transmission->fermerPortGps();
00728 }
```

8.3.2.36 on_pushButtonGraphique_clicked

`void Ihm::on_pushButtonGraphique_clicked ( ) [private], [slot]`

Méthode appelée dès que le push bouton deconnexion est enclenché

Références graphique.

```
00677 {
00678     graphique->show();
00679 }
```

8.3.2.37 on_pushButtonOuvrirPort_clicked

`void Ihm::on_pushButtonOuvrirPort_clicked ( ) [private], [slot]`

methode qui met à jour le message de statut de la connexion bluetooth (erreur de connexion)

Références Transmission : :configurerPort(), transmission, et ui.

```
00280 {
00281     transmission->configurerPort(ui->comboBoxAppareil->currentText(),
      ui->comboBoxDebitBaud->currentText(), ui->comboBoxBitsDonnees->currentText(),
      ui->comboBoxBitsStop->currentText());
00282 }
```

8.3.2.38 on_pushButtonOuvrirPortGps_clicked

`void Ihm::on_pushButtonOuvrirPortGps_clicked ( ) [private], [slot]`

Références Transmission : :configurerPortGps(), transmission, et ui.

```
00721 {
00722     transmission->configurerPortGps(ui->comboBoxPortSerieGps->currentText(),
       ui->comboBoxDebitGps->currentText(), ui->comboBoxBitsDonneesGps->currentText(),
      ui->comboBoxBitsStopGps->currentText());
00723 }
```

8.3.2.39 on_pushButtonQuitter_clicked

`void Ihm::on_pushButtonQuitter_clicked ( ) [private], [slot]`

modifie l'etat de la led en fonction de la trame recue

Références graphique.

```
00171 {
00172     this->close();
00173     graphique->close();
00174 }
```

8.3.2.40 on_pushButtonScan_clicked

`void Ihm::on_pushButtonScan_clicked ( ) [private], [slot]`

Méthode appelée dès que le radio bouton Ville est enclenché

Références Transmission : :demarrerScan(), transmission, et ui.

```
00516 {
00517     ui->comboBoxBluetooth->clear();
00518     ui->pushButtonScan->setEnabled(false);
00519     ui->pushButtonConnexion->setEnabled(false);
00520     ui->pushButtonDeconnexion->setEnabled(false);
00521     ui->labelStatut->setText("<font color=\"#bd2c2c\">Recherche en cours ...</font>");
00522
00523     transmission->demarrerScan();
00524 }
```

8.3.2.41 on_pushButtonVille_clicked

`void Ihm::on_pushButtonVille_clicked ( ) [private], [slot]`

Méthode appelée dès que le radio bouton LedOff est enclenché

Références Meteo : :creerUrlVille(), meteo, et ui.

```
00492 {
00493     if(ui->lineEditVille->text() != "")
00494     {
00495         meteo->creerUrlVille(ui->lineEditVille->text());
00496     }
00497 }
```

8.3.2.42 on_radioButtonLedOff_clicked

`void Ihm::on_radioButtonLedOff_clicked ( ) [private], [slot]`

Méthode appelée dès que le radio bouton LedOrange est enclenché

Références Transmission : :envoyerDonnees(), et transmission.

```
00481 {
00482     transmission->envoyerDonnees("SET LED off");
00483 }
```

8.3.2.43 on_radioButtonLedOrange_clicked

`void Ihm::on_radioButtonLedOrange_clicked ( ) [private], [slot]`

Méthode appelée dès que le radio bouton LedVert est enclenché

Références Transmission : :envoyerDonnees(), et transmission.

```
00471 {
00472     transmission->envoyerDonnees("SET LED 3");
00473 }
```

8.3.2.44 on_radioButtonLedRouge_clicked

```
void Ihm::on_radioButtonLedRouge_clicked ( ) [private], [slot]
```

Méthode appelée dès que le bouton envoyer est enclenché

Références Transmission : :envoyerDonnees(), et transmission.

```
00451 {
00452     transmission->envoyerDonnees("SET LED 1");
00453 }
```

8.3.2.45 on_radioButtonLedVert_clicked

```
void Ihm::on_radioButtonLedVert_clicked ( ) [private], [slot]
```

Méthode appelée dès que le radio bouton LedRoude est enclenché

Références Transmission : :envoyerDonnees(), et transmission.

```
00461 {
00462     transmission->envoyerDonnees("SET LED 2");
00463 }
```

8.3.3 Documentation des données membres

8.3.3.1 graphique

```
Graphique* Ihm::graphique [private]
```

Référencé par actualiserDonnee(), Ihm(), on_pushButtonGraphique_clicked(), on_pushButton↩ Quitter_clicked(), et ∼Ihm().

8.3.3.2 meteo

```
Meteo* Ihm::meteo [private]
```

Référencé par actualiserAffichageMeteo(), Ihm(), initialiserConnect(), on_pushButtonEnvoyer↩ Coordonnee_clicked(), et on_pushButtonVille_clicked().

8.3.3.3 proc

```
QProcess* Ihm::proc [private]
```

Référencé par actualiserMessageStatutConnecterBluetooth(), actualiserMessageStatutDeconnecter↩ Bluetooth(), et Ihm().

8.3.3.4 transmission

```
Transmission* Ihm::transmission [private]
```

Référencé par actualiserDonnee(), actualiserDonneeGps(), actualiserMessageStatutConnecter↩ Bluetooth(), actualiserMessageStatutConnectionGps(), actualiserMessageStatutConnection↩ Sonde(), actualiserMessageStatutDeconnecterBluetooth(), actualiserMessageStatutErreur↩ Bluetooth(), actualiserTrame(), Ihm(), initialiserConnect(), mettreAjourAppareilBluetooth↩ Disponible(), modifierEtatLed(), on_pushButtonConnexion_clicked(), on_pushButtonDeconnexion↩ _clicked(), on_pushButtonEnvoyerCoordonnee_clicked(), on_pushButtonEnvoyerTrame_↩ clicked(), on_pushButtonFermerPort_clicked(), on_pushButtonFermerPortGps_clicked(), on↩ _pushButtonOuvrirPort_clicked(), on_pushButtonOuvrirPortGps_clicked(), on_pushButton↩ Scan_clicked(), on_radioButtonLedOff_clicked(), on_radioButtonLedOrange_clicked(), on_↩ radioButtonLedRouge_clicked(), et on_radioButtonLedVert_clicked().

8.3.3.5 ui

```
Ui::Ihm* Ihm::ui [private]
```

Référencé par activerConnexionBluetooth(), activerConnexionPortSerie(), ActiverControleLed(), actualiserAffichageBluetooth(), actualiserAffichageMeteo(), actualiserDonnee(), actualiserDonnee↩ Gps(), actualiserMessageStatutConnecterBluetooth(), actualiserMessageStatutConnectionGps(), actualiserMessageStatutConnectionSonde(), actualiserMessageStatutDeconnecterBluetooth(), actualiserMessageStatutErreurBluetooth(), actualiserTrame(), chargerConfigurationPort(), desactiver↩ ConnexionBluetooth(), desactiverConnexionPortSerie(), DesactiverControleLed(), desactiver↩ FermerPort(), desactiverFermerPortGps(), desactiverOuverturePort(), desactiverOuverturePort↩ Gps(), enregistrerConfigurationPort(), Ihm(), initialiserAffichage(), initialiserInterface(), mettre↩ AjourAppareilBluetoothDisponible(), modifierEtatLed(), on_ModeOperateur_stateChanged(), on_pushButtonConnexion_clicked(), on_pushButtonEnvoyerTrame_clicked(), on_pushButton↩ OuvrirPort_clicked(), on_pushButtonOuvrirPortGps_clicked(), on_pushButtonScan_clicked(), on_pushButtonVille_clicked(), et ~Ihm().

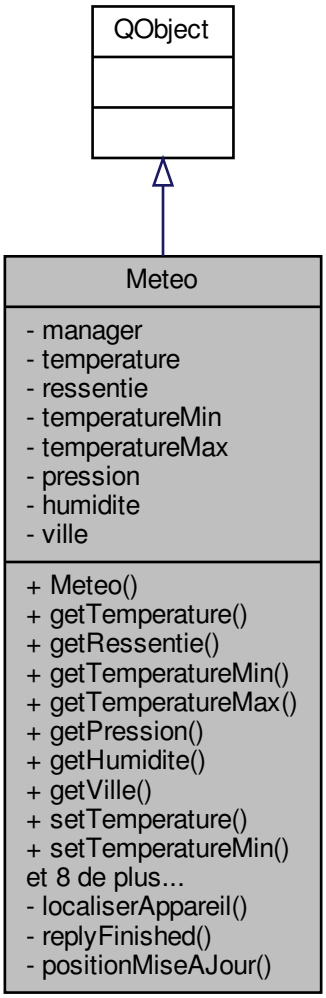
La documentation de cette classe a été générée à partir des fichiers suivants :
- ihm.h
- ihm.cpp

8.4 Référence de la classe Meteo

Declaration de la classe Meteo.

```
#include "meteo.h"
```

Graphe de collaboration de Meteo :



Signaux

— void donnerMeteoMiseAJour ()

Fonctions membres publiques

— Meteo (QObject *parent=nullptr)
constructeur de la classe Meteo
— double getTemperature () const
retourne la valeur de temperature
— double getRessentie () const
retourne la valeur de ressentie
— double getTemperatureMin () const
retourne la valeur de temperature Minimale en degre
— double getTemperatureMax () const
retourne la valeur de temperature Maximale en degre

- int getPression () const
  retourne la valeur de pression
- int getHumidite () const
  retourne la valeur de humidité
- QString getVille () const
  retourne la valeur de la ville
- void setTemperature (double temperature)
  modifie la variable température avec celle passée en paramètre
- void setTemperatureMin (double temperatureMin)
  modifie la variable températureMin avec celle passée en paramètre
- void setTemperatureMax (double temperatureMax)
  modifie la variable températureMax avec celle passée en paramètre
- void setRessentie (double ressentie)
  modifie la variable ressentie avec celle passée en paramètre
- void setPression (int pression)
  modifie la variable pression avec celle passée en paramètre
- void setHumidite (int humidite)
  modifie la variable humidite avec celle passée en paramètre
- void setVille (QString ville)
  modifie la variable ville avec celle passée en paramètre
- void recupererDonnerMeteo (QString URL)
  fonction qui fait une requête a l'api du site openweathermap en fonction de la ville choisie
- void creerUrlVille (QString ville)
  methode qui cree un Url pour l'api avec une longitude et latitude
- void creerUrlCoordonnerGps (double latitude, double longitude)
  methode qui cree un Url pour l'api avec une ville

Connecteurs privés

- void replyFinished (QNetworkReply *reply)
  Décompose le Json pour récupérer les différentes données.
- void positionMiseAJour (const QGeoPositionInfo &info)
  methode appelée quand la position est mise à jour

Fonctions membres privées

- void localiserAppareil ()
  fonction qui localise l'appareil

Attributs privés

- QNetworkAccessManager * manager
  objet manager
- double temperature
  variable qui stocke la temperature
- double ressentie
  variable qui stocke le ressentie
- double temperatureMin
  variable qui stocke a temperature minimale
- double temperatureMax
  variable qui stocke la temperature maximale
- int pression
  variable qui stocke la pression atmosphérique
- int humidite
  variable qui stocke l'humidité

— QString ville
variable qui stocke la ville

8.4.1 Documentation des constructeurs et destructeur

8.4.1.1 Meteo()

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

Paramètres

parent	

Références manager, et replyFinished().

```
00019                             : QObject(parent), temperature(0.),
      ressentie(0.), temperatureMin(0.), temperatureMax(0.),
      pression(0), humidite(0)
00020 {
00021     manager = new QNetworkAccessManager(this);
00022     connect(manager, SIGNAL(finished(QNetworkReply*)), this, SLOT(
      replyFinished(QNetworkReply*)));
00023
00024     //localiserAppareil();          //methode pour localiser sans GPS.
00025 }
```

8.4.2 Documentation des fonctions membres

8.4.2.1 creerUrlCoordonnerGps()

```
void Meteo::creerUrlCoordonnerGps (
            double latitude,
            double longitude )
```

cree l'URl pour l'API avec une ville

Paramètres

latitude	
longitude	

Références API_KEY, et recupererDonnerMeteo().

Référencé par Ihm : :on_pushButtonEnvoyerCoordonnee_clicked(), et positionMiseAJour().

```
00221 {
00222     recupererDonnerMeteo("http://api.openweathermap.org/data/2.5/weather?lat=" +
      QString::number(latitude) + "&lon=" + QString::number(longitude) + "&units=metric&APPID=" +
      API_KEY);
00223 }
```

8.4.2.2 creerUrlVille()

```
void Meteo::creerUrlVille (
           QString ville )
```

envoie une requete à l'API de openweather

Paramètres

ville	

Références API_KEY, et recupererDonnerMeteo().

Référencé par Ihm : :on_pushButtonVille_clicked().

```
00232 {
00233     recupererDonnerMeteo("http://api.openweathermap.org/data/2.5/weather?q=" +
      ville + ",FR&units=metric&APPID=" + API_KEY);
00234 }
```

8.4.2.3 donnerMeteoMiseAJour

```
void Meteo::donnerMeteoMiseAJour ( )  [signal]
```

Référencé par replyFinished().

8.4.2.4 getHumidite()

```
int Meteo::getHumidite ( ) const
```

recuperer la pression atmosphérique

Renvoie

int

Références humidite.

Référencé par Ihm : :actualiserAffichageMeteo().

```
00089 {
00090     return humidite;
00091 }
```

8.4.2.5 getPression()

```
int Meteo::getPression ( ) const
```

recuperer la temperature maximale

Renvoie

int

Références pression.

Référencé par Ihm : :actualiserAffichageMeteo().

```
00078 {
00079     return pression;
00080 }
```

8.4.2.6 getRessentie()

```
double Meteo::getRessentie ( ) const
```

recuperer la temperature

Renvoie

double

Références ressentie.

Référencé par Ihm : :actualiserAffichageMeteo().

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

8.4.2.7 getTemperature()

```
double Meteo::getTemperature ( ) const
```

Renvoie

double

Références temperature.

Référencé par Ihm : :actualiserAffichageMeteo().

```
00034 {
00035     return temperature;
00036 }
```

8.4.2.8 getTemperatureMax()

```
double Meteo::getTemperatureMax ( ) const
```

recuperer la temperature minimale

Renvoie

double

Références temperatureMax.

Référencé par Ihm : :actualiserAffichageMeteo().

```
00067 {
00068     return temperatureMax;
00069 }
```

8.4.2.9 getTemperatureMin()

```
double Meteo::getTemperatureMin ( ) const
```

recuperer le ressentie

Renvoie

double

Références temperatureMin.

Référencé par Ihm : :actualiserAffichageMeteo().

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

8.4.2.10 getVille()

```
QString Meteo::getVille ( ) const
```

recuperer l'humidité

Renvoie

QString

Références ville.

Référencé par Ihm : :actualiserAffichageMeteo().

```
00100 {
00101     return ville;
00102 }
```

8.4.2.11 localiserAppareil()

```
void Meteo::localiserAppareil ( )  [private]
```

methode qui localise l'appareil

Références positionMiseAJour().

```
00187 {
00188     QGeoPositionInfoSource *source = QGeoPositionInfoSource::createDefaultSource(this);
00189     if (source)
00190     {
00191         connect(source, SIGNAL(positionUpdated(QGeoPositionInfo)), this, SLOT(
      positionMiseAJour(QGeoPositionInfo)));
00192         source->startUpdates();
00193     }
00194     source->setUpdateInterval(2000);
00195 }
```

8.4.2.12 positionMiseAJour

```
void Meteo::positionMiseAJour (
           const QGeoPositionInfo & info )  [private], [slot]
```

fonction appelée quand un nouveau Json est recu

Paramètres

info	

Références creerUrlCoordonnerGps().

Référencé par localiserAppareil().

```
00204 {
00205     qDebug() << "Position Mise à jour :" << info.coordinate();
00206
00207     QGeoCoordinate coordonner = QGeoCoordinate();
00208     coordonner = info.coordinate();
00209
00210     creerUrlCoordonnerGps(coordonner.latitude(), coordonner.longitude());
00211 }
```

8.4.2.13 recupererDonnerMeteo()

```
void Meteo::recupererDonnerMeteo (
           QString URL )
```

modifier la valeur de la ville

Paramètres

URL	

Références manager.

Référencé par creerUrlCoordonnerGps(), et creerUrlVille().

```
00243 {
00244     QUrl url(URL);
00245
00246     QNetworkRequest request;
00247     request.setUrl(url);
00248
00249     request.setRawHeader("Accept", "application/json");
00250     qDebug() << Q_FUNC_INFO << request.url() << endl;
00251     manager->get(request);
00252
00253 }
```

8.4.2.14 replyFinished

```
void Meteo::replyFinished (
            QNetworkReply * reply )  [private], [slot]
```

signal emis quand le JSON est decomposé

Paramètres

reply	

Références donnerMeteoMiseAJour(), main(), setHumidite(), setPression(), setRessentie(), setTemperature(), setTemperatureMax(), setTemperatureMin(), et setVille().

Référencé par Meteo().

```
00262 {
00263     QString datas = reply->readAll();
00264
00265     QJsonDocument doc = QJsonDocument::fromJson(datas.toUtf8());
00266
00267     QJsonObject obj = doc.object();
00268
00269     QJsonObject main = obj.value(QString("main")).toObject();
00270
00271     setVille(obj.value(QString("name")).toString());
00272     setTemperature(main.value(QString("temp")).toDouble());
00273     setTemperatureMin(main.value(QString("temp_min")).toDouble());
00274     setTemperatureMax(main.value(QString("temp_max")).toDouble());
00275     setRessentie(main.value(QString("feels_like")).toDouble());
00276     setPression(main.value(QString("pressure")).toInt());
00277     setHumidite(main.value(QString("humidity")).toInt());
00278
00279     emit donnerMeteoMiseAJour();
00280
00281 }
```

8.4.2.15 setHumidite()

```
void Meteo::setHumidite (
            int humidite )
```

modifier la valeur de la pression

Paramètres

humidite	

Références humidite.

Référencé par replyFinished().

```
00166 {
00167     this->humidite = humidite;
00168 }
```

8.4.2.16 setPression()

```
void Meteo::setPression (
            int pression )
```

modifier la valeur de ressentie

Paramètres

pression	

Références pression.

Référencé par replyFinished().

```
00177 {
00178     this->pression = pression;
00179 }
```

8.4.2.17 setRessentie()

```
void Meteo::setRessentie (
            double ressentie )
```

modifier la valeur de la temperature maximale

Paramètres

ressentie	

Références ressentie.

Référencé par replyFinished().

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

8.4.2.18 setTemperature()

```
void Meteo::setTemperature (
            double temperature )
```

recuperer le nom de la ville

Paramètres

temperature	

Références temperature.

Référencé par replyFinished().

```
00122 {
00123     this->temperature = temperature;
00124 }
```

8.4.2.19 setTemperatureMax()

```
void Meteo::setTemperatureMax (
            double temperatureMax )
```

modifier la valeur de la temperature minimale

Paramètres

temperatureMax	

Références temperatureMax.

Référencé par replyFinished().

```
00133 {
00134     this->temperatureMax = temperatureMax;
00135 }
```

8.4.2.20 setTemperatureMin()

```
void Meteo::setTemperatureMin (
            double temperatureMin )
```

modifier la valeur de la temperature

Paramètres

temperatureMin	

Références temperatureMin.

Référencé par replyFinished().

```
00144 {
00145     this->temperatureMin = temperatureMin;
00146 }
```

8.4.2.21 setVille()

```
void Meteo::setVille (
            QString ville )
```

modifier la valeur de l'humidite

Paramètres

ville	

Références ville.

Référencé par replyFinished().

```
00112 {
00113     this->ville = ville;
00114 }
```

8.4.3 Documentation des données membres

8.4.3.1 humidite

```
int Meteo::humidite [private]
```

Référencé par getHumidite(), et setHumidite().

8.4.3.2 manager

```
QNetworkAccessManager* Meteo::manager [private]
```

cree l'URl pour l'API avec les coordonnée de longitude et latitude

Référencé par Meteo(), et recupererDonnerMeteo().

8.4.3.3 pression

```
int Meteo::pression [private]
```

Référencé par getPression(), et setPression().

8.4.3.4 ressentie

`double Meteo::ressentie [private]`

Référencé par getRessentie(), et setRessentie().

8.4.3.5 temperature

`double Meteo::temperature [private]`

fonction appelée quand la position est mise à jour

Référencé par getTemperature(), et setTemperature().

8.4.3.6 temperatureMax

`double Meteo::temperatureMax [private]`

Référencé par getTemperatureMax(), et setTemperatureMax().

8.4.3.7 temperatureMin

`double Meteo::temperatureMin [private]`

Référencé par getTemperatureMin(), et setTemperatureMin().

8.4.3.8 ville

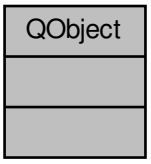
`QString Meteo::ville [private]`

Référencé par getVille(), et setVille().

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

8.5 Référence de la classe QObject

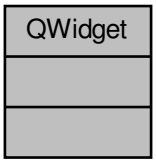
Graphe de collaboration de QObject :



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

8.6 Référence de la classe QWidget

Graphe de collaboration de QWidget :



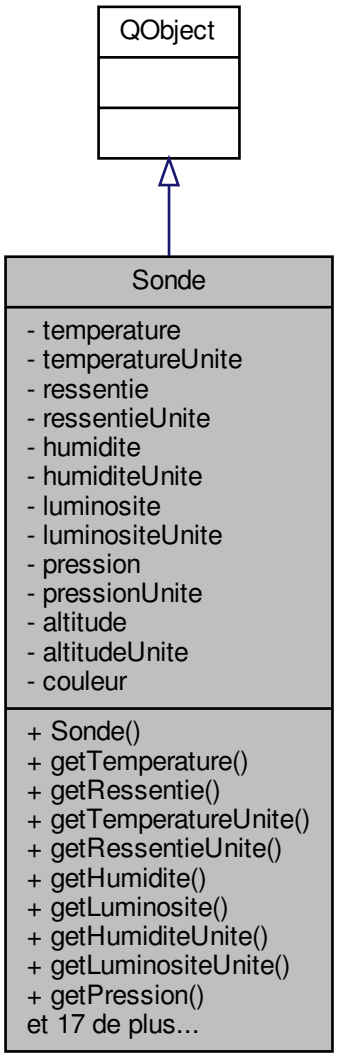
La documentation de cette classe a été générée à partir du fichier suivant :
— ihm.h

8.7 Référence de la classe Sonde

Declaration de la classe Sonde.

```
#include "sonde.h"
```

Graphe de collaboration de Sonde :



Fonctions membres publiques

- Sonde (QObject *parent=nullptr)
  contructeur de la classe Sonde
- double getTemperature () const
  retourne la valeur de temperature
- double getRessentie () const
  retourne la valeur de ressentie
- QString getTemperatureUnite () const
  retourne l'unité de la valeur de temperature
- QString getRessentieUnite () const
  retourne l'unité de la valeur de ressentie
- int getHumidite () const
  retourne la valeur d'humidité
- int getLuminosite () const

retourne la valeur de luminosité
- QString getHumiditeUnite () const
  retourne l'unité de la valeur d'humidite
- QString getLuminositeUnite () const
  retourne l'unité de la valeur de luminosite
- int getPression () const
  retourne la valeur de pression
- QString getPressionUnite () const
  retourne l'unité de la valeur de pression
- int getAltitude () const
  retourne la valeur de l'altitude
- QString getAltitudeUnite () const
  retourne l'unité de la valeur de l'altitude
- int getEtatLed () const
  retourne l'etat de la led
- void setTemperature (double temperature)
  modifie la variable température avec celle passée en paramètre
- void setRessentie (double ressentie)
  modifie la variable ressentie avec celle passée en paramètre
- void setTemperatureUnite (QString temperatureUnite)
  modifie la variable de l'unité utilisée pour la temperature
- void setRessentieUnite (QString ressentieUnite)
  modifie la variable de l'unité utilisée pour le ressentie
- void setHumidite (int humidite)
  modifie la variable de l'humidite
- void setLuminosite (int luminosite)
  modifie la variable de la luminosite
- void setHumiditeUnite (QString humiditeUnite)
  modifie la variable de l'unité utilisée pour l'humidite
- void setLuminositeUnite (QString luminositeUnite)
  modifie la variable de l'unité utilisée pour la luminosite
- void setPression (int pression)
  modifie la variable de la pression
- void setPressionUnite (QString pressionUnite)
  modifie la variable de l'unité utilisée pour la pression
- void setAltitude (int altitude)
  modifie la variable altitude passée en paramètre
- void setAltitudeUnite (QString altitudeUnite)
  modifie la variable de l'unité utilisée pour l'altitude
- void setCouleurLed (int couleur)
  modifie la variable de l'etat de la led

Attributs privés

- double temperature
  variable qui stocke la temperature
- QString temperatureUnite
  variable qui stocke l'unite de la temperature
- double ressentie
  variable qui stocke le ressentie
- QString ressentieUnite
  variable qui stocke l'unite de ressentie
- int humidite
  variable qui stocke l'humidite
- QString humiditeUnite
  variable qui stocke l'unite de l'humidite
- int luminosite

variable qui stocke la luminosite
- QString luminositeUnite
  variable qui stocke l'unite de la luminosite
- int pression
  variable qui stocke la pression atmosphérique
- QString pressionUnite
  variable qui stocke l'unite de pression atmosphérique
- int altitude
  variable qui stocke l'altitude
- QString altitudeUnite
  variable qui stocke l'unite de l'altitude
- int couleur
  variable qui stocke l'etat de la led (0 = eteint / 1 = rouge / 2 = vert / 3 = orange

8.7.1 Documentation des constructeurs et destructeur

8.7.1.1 Sonde()

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

Paramètres

parent	

```
00021                           : QObject(parent), temperature(0.),
      temperatureUnite("°C"), ressentie(0.), ressentieUnite("°C"),\
00022     humidite(0), humiditeUnite("%"), luminosite(0),
      luminositeUnite("Lux"), pression(0), pressionUnite("hPa"),
      altitude(0), altitudeUnite("Mètre"), couleur(0)
00023 {
00024
00025 }
```

8.7.2 Documentation des fonctions membres

8.7.2.1 getAltitude()

```
int Sonde::getAltitude ( ) const
```

recuperer l'unite de pression atmosphérique

Renvoie

int

Références altitude.

Référencé par Ihm : :actualiserDonnee().

```
00144 {
00145     return altitude;
00146 }
```

8.7.2.2 getAltitudeUnite()

```
QString Sonde::getAltitudeUnite ( ) const
```

recuperer l'altitude

Renvoie

QString

Références altitudeUnite.

Référencé par Ihm : :actualiserDonnee().

```
00155 {
00156     return altitudeUnite;
00157 }
```

8.7.2.3 getEtatLed()

```
int Sonde::getEtatLed ( ) const
```

recuperer l'unite d'altitude

Renvoie

int

Références couleur.

Référencé par Ihm : :modifierEtatLed().

```
00166 {
00167     return couleur;
00168 }
```

8.7.2.4 getHumidite()

```
int Sonde::getHumidite ( ) const
```

recuperer l'unite du resssentie

Renvoie

int

Références humidite.

Référencé par Ihm : :actualiserDonnee().

```
00078 {
00079     return humidite;
00080 }
```

8.7.2.5 getHumiditeUnite()

```
QString Sonde::getHumiditeUnite ( ) const
```

recuperer la luminosite

Renvoie

QString

Références humiditeUnite.

Référencé par Ihm : :actualiserDonnee().

```
00100 {
00101     return humiditeUnite;
00102 }
```

8.7.2.6 getLuminosite()

```
int Sonde::getLuminosite ( ) const
```

recuperer l'humidite

Renvoie

int

Références luminosite.

Référencé par Ihm : :actualiserDonnee().

```
00089 {
00090     return luminosite;
00091 }
```

8.7.2.7 getLuminositeUnite()

```
QString Sonde::getLuminositeUnite ( ) const
```

recuperer l'unite d'humidite

Renvoie

QString

Références luminositeUnite.

Référencé par Ihm : :actualiserDonnee().

```
00111 {
00112     return luminositeUnite;
00113 }
```

8.7.2.8 getPression()

```
int Sonde::getPression ( ) const
```

recuperer l'unite de luminosite

Renvoie

int

Références pression.

Référencé par Ihm : :actualiserDonnee().

```
00122 {
00123     return pression;
00124 }
```

8.7.2.9 getPressionUnite()

```
QString Sonde::getPressionUnite ( ) const
```

recuperer la pression atmosphérique

Renvoie

QString

Références pressionUnite.

Référencé par Ihm : :actualiserDonnee().

```
00133 {
00134     return pressionUnite;
00135 }
```

8.7.2.10 getRessentie()

```
double Sonde::getRessentie ( ) const
```

recuperer la temperature

Renvoie

double

Références ressentie.

Référencé par Ihm : :actualiserDonnee().

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

8.7.2.11 getRessentieUnite()

```
QString Sonde::getRessentieUnite ( ) const
```

recuperer l'unite de temperature

Renvoie

QString

Références ressentieUnite.

Référencé par Ihm : :actualiserDonnee().

```
00067 {
00068     return ressentieUnite;
00069 }
```

8.7.2.12 getTemperature()

```
double Sonde::getTemperature ( ) const
```

constructeur de la classe Sonde

Renvoie

double ...

Références temperature.

Référencé par Ihm : :actualiserDonnee().

```
00034 {
00035     return temperature;
00036 }
```

8.7.2.13 getTemperatureUnite()

```
QString Sonde::getTemperatureUnite ( ) const
```

recuperer le ressentie

Renvoie

QString

Références temperatureUnite.

Référencé par Ihm : :actualiserDonnee().

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

8.7.2.14 setAltitude()

```
void Sonde::setAltitude (
            int altitude )
```

modifier la valeur de l'unite de pression atmosphérique

Paramètres

altitude	

Références altitude.

Référencé par Transmission : :decomposer().

```
00287 {
00288     this->altitude = altitude;
00289 }
```

8.7.2.15 setAltitudeUnite()

```
void Sonde::setAltitudeUnite (
           QString altitudeUnite )
```

modifier la valeur d'altitude

Paramètres

altitudeUnite	

Références altitudeUnite.

Référencé par Transmission : :decomposer().

```
00298 {
00299     this->altitudeUnite = altitudeUnite;
00300 }
```

8.7.2.16 setCouleurLed()

```
void Sonde::setCouleurLed (
           int couleur )
```

modifier la valeur de l'unite altitude

Paramètres

couleur	

Références couleur.

Référencé par Transmission : :decomposer().

```
00308 {
00309     this->couleur = couleur;
00310 }
```

8.7.2.17 setHumidite()

```
void Sonde::setHumidite (
            int humidite )
```

modifier la valeur de l'uniter de ressentie

Paramètres

humidite	

Références humidite.

Référencé par Transmission : :decomposer().

```
00221 {
00222     this->humidite = humidite;
00223 }
```

8.7.2.18 setHumiditeUnite()

```
void Sonde::setHumiditeUnite (
            QString humiditeUnite )
```

modifier la valeur de luminosite

Paramètres

humiditeUnite	

Références humiditeUnite.

Référencé par Transmission : :decomposer().

```
00243 {
00244     this->humiditeUnite = humiditeUnite;
00245 }
```

8.7.2.19 setLuminosite()

```
void Sonde::setLuminosite (
            int luminosite )
```

modifier la valeur de l'humidite

Paramètres

luminosite	

Références luminosite.

Référencé par Transmission : :decomposer().

```
00232 {
00233     this->luminosite = luminosite;
00234 }
```

8.7.2.20 setLuminositeUnite()

```
void Sonde::setLuminositeUnite (
            QString luminositeUnite )
```

modifier la valeur de l'unite d'humidite

Paramètres

luminositeUnite	

Références luminositeUnite.

Référencé par Transmission : :decomposer().

```
00254 {
00255     this->luminositeUnite = luminositeUnite;
00256 }
```

8.7.2.21 setPression()

```
void Sonde::setPression (
            int pression )
```

modifier la valeur de l'unite de luminosite

Paramètres

pression	

Références pression.

Référencé par Transmission : :decomposer().

```
00265 {
00266     this->pression = pression;
00267 }
```

8.7.2.22 setPressionUnite()

```
void Sonde::setPressionUnite (
            QString pressionUnite )
```

modifier la valeur de la pression atmosphérique

Paramètres

pressionUnite	

Références pressionUnite.

Référencé par Transmission : :decomposer().

```
00276 {
00277     this->pressionUnite = pressionUnite;
00278 }
```

8.7.2.23 setRessentie()

```
void Sonde::setRessentie (
            double ressentie )
```

modifier la valeur de la temperature

Paramètres

ressentie	

Références ressentie.

Référencé par Transmission : :decomposer().

```
00188 {
00189     this->ressentie = ressentie;
00190 }
```

8.7.2.24 setRessentieUnite()

```
void Sonde::setRessentieUnite (
            QString ressentieUnite )
```

modifier la valeur de l'uniter de temperature

Paramètres

ressentieUnite	

Références ressentieUnite.

Référencé par Transmission : :decomposer().

```
00210 {
00211     this->ressentieUnite = "°" + ressentieUnite;
00212 }
```

8.7.2.25 setTemperature()

```
void Sonde::setTemperature (
            double temperature )
```

recuperer l'etat de la led

Paramètres

temperature	

Références temperature.

Référencé par Transmission : :decomposer().

```
00177 {
00178     this->temperature = temperature;
00179 }
```

8.7.2.26 setTemperatureUnite()

```
void Sonde::setTemperatureUnite (
            QString temperatureUnite )
```

modifier la valeur de ressentie

Paramètres

temperatureUnite	

Références temperatureUnite.

Référencé par Transmission : :decomposer().

```
00199 {
00200     this->temperatureUnite = "°" + temperatureUnite;
00201 }
```

8.7.3 Documentation des données membres

8.7.3.1 altitude

```
int Sonde::altitude [private]
```

Référencé par getAltitude(), et setAltitude().

8.7.3.2 altitudeUnite

```
QString Sonde::altitudeUnite [private]
```

Référencé par getAltitudeUnite(), et setAltitudeUnite().

8.7.3.3 couleur

```
int Sonde::couleur [private]
```

Référencé par getEtatLed(), et setCouleurLed().

8.7.3.4 humidite

```
int Sonde::humidite [private]
```

Référencé par getHumidite(), et setHumidite().

8.7.3.5 humiditeUnite

```
QString Sonde::humiditeUnite [private]
```

Référencé par getHumiditeUnite(), et setHumiditeUnite().

8.7.3.6 luminosite

```
int Sonde::luminosite [private]
```

Référencé par getLuminosite(), et setLuminosite().

8.7.3.7 luminositeUnite

```
QString Sonde::luminositeUnite [private]
```

Référencé par getLuminositeUnite(), et setLuminositeUnite().

8.7.3.8 pression

```
int Sonde::pression [private]
```

Référencé par getPression(), et setPression().

8.7.3.9 pressionUnite

```
QString Sonde::pressionUnite [private]
```

Référencé par getPressionUnite(), et setPressionUnite().

8.7.3.10 ressentie

```
double Sonde::ressentie [private]
```

Référencé par getRessentie(), et setRessentie().

8.7.3.11 ressentieUnite

```
QString Sonde::ressentieUnite [private]
```

Référencé par getRessentieUnite(), et setRessentieUnite().

8.7.3.12 temperature

```
double Sonde::temperature [private]
```

modifier l'etat de la led

Référencé par getTemperature(), et setTemperature().

8.7.3.13 temperatureUnite

```
QString Sonde::temperatureUnite [private]
```

Référencé par getTemperatureUnite(), et setTemperatureUnite().

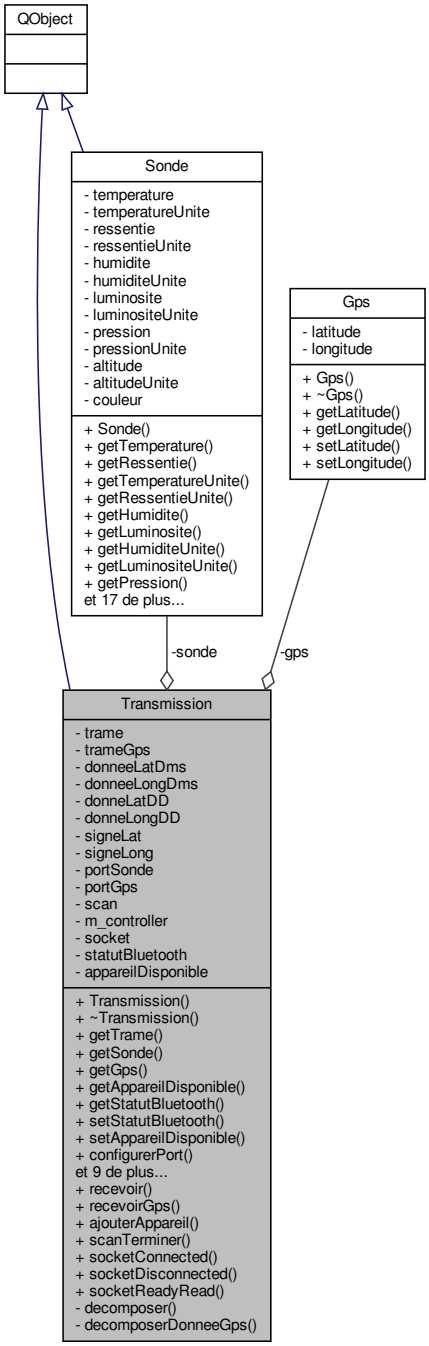
La documentation de cette classe a été générée à partir des fichiers suivants :
- sonde.h
- sonde.cpp

8.8 Référence de la classe Transmission

Declaration de la classe Transmission (Liaison série et Bluetooth)

```
#include "transmission.h"
```

Graphe de collaboration de Transmission :



Connecteurs publics

— void recevoir ()
Récupérer les données reçues sur le port série.
— void recevoirGps ()
Récupérer la trame émise par le Gps.
— void ajouterAppareil (const QBluetoothDeviceInfo &info)
ajoute les appareils trouvés dans une liste pour les afficher dans l'IHM
— void scanTerminer ()

fonction appelée quand le scan est fini
- void socketConnected ()
  fonction appelée quand l'appareil est connecté
- void socketDisconnected ()
  fonction appelée quand l'appareil est deconnecté
- void socketReadyRead ()
  methode appelée quand une trame est disponible

Signaux

- void trameSondeRecue ()
  signal émis quand une nouvelle trame de l'ESP32 est recue
- void trameGpsRecue ()
  signal émis quand une nouvelle trame GPS est recue
- void portOuvert ()
  signal émis quand le port série de la Sonde est ouvert
- void portOuvertGps ()
  signal émis quand le port serie du GPS est ouvert
- void portFerme ()
  signal émis quand le port serie de la Sonde est fermé
- void portFermeGps ()
  signal émis quand le port serie du GPS est fermé
- void nouvelleAppareilDisponible ()
  signal émis quand un nouvelle appareil est disponible
- void scanfini ()
  signal émis quand le scan est terminé
- void connecter ()
  signal émis quand l'appareil est connecté
- void deconnecter ()
  signal émis quand l'appareil est deconnecté
- void socketErreur ()
  signal émis quand il y a une erreur avec la communication
- void erreurConnectionPortSerieGps (QString message)
  signal émis quand il y a une erreur de connexion
- void erreurConnectionPortSerieSonde (QString message)
  signal émis quand il y a une de connexion

Fonctions membres publiques

- Transmission (QObject *parent=nullptr)
  constructeur de la classe Transmission
- ~Transmission ()
  destructeur de la classe Transmission
- QString getTrame () const
  retourne la trame stockée
- Sonde * getSonde () const
  retourne l'objet Sonde
- Gps * getGps () const
  retourne l'objet gps
- QStringList getAppareilDisponible () const
  retourne une liste des appareils bluetooth disponibles
- QString getStatutBluetooth () const
  retourne la statut de la connexion avec l'appareil bluetooth
- void setStatutBluetooth (QString statutBluetooth)
  permet de modifier le statut de la connexion avec l'appareil bluetooth
- void setAppareilDisponible (QString appareilDisponible)

ajoute un appareil a la liste des appareils disponibles
- void configurerPort (QString portCommunication, QString DebitBaud, QString BitsDonnees, QString BitsStop)
  configure le port serie
- void configurerPortGps (QString portCommunication, QString DebitBaud, QString BitsDonnees, QString BitsStop)
  configure le port serie pour le Gps
- void fermerPort ()
  cette methode ferme le port serie
- void fermerPortGps ()
  cette methode ferme le port serie
- void envoyerDonnees (QString envoyerTrame)
  envoyer les données par le port serie ou bluetooth suivant le mode choisie
- void ouvrirPort ()
  ouvrir le port série de la Sonde
- void ouvrirPortGps ()
  ouvrir le port série pour le GPS
- void demarrerScan ()
  cette fonction recherche les appareils disponibles
- void connecterAppareilBluetooth (QString appareil)
  Permet de se connecter à un appareil.
- void deconnecterAppareilBluetooth ()
  methode qui deconnecte l'appareil bluetooth connecté

Fonctions membres privées

- void decomposer ()
  Décompose la trame reçue.
- void decomposerDonneeGps ()
  Décompose la trame gps pour avoir la latitude et la longitude.

Attributs privés

- QString trame
  stockage de la trame de l'ESP32 recue
- QString trameGps
  stockage de la trame GPS recue
- QString donneeLatDms
  stockage des données de latitude en DMS
- QString donneeLongDms
  stockage des données de longitude en DMS
- double donneLatDD
  stockage des données de latitude en DD
- double donneLongDD
  stockage des données de longitude en DD
- QString signeLat
  stockage du signe de la latitude
- QString signeLong
  stockage du signe de la longitude
- Sonde * sonde
  objet sonde
- Gps * gps
  objet gps
- QSerialPort * portSonde
  objet portSonde
- QSerialPort * portGps

objet portGps
— QBluetoothDeviceDiscoveryAgent * scan
objet scan
— QLowEnergyController * m_controller
objet m_controller
— QBluetoothSocket * socket
objet socket
— QString statutBluetooth
stockage du statut de la connexion bluetooth
— QStringList appareilDisponible
stockage des appareil bluetooth disponible

8.8.1 Description détaillée

A faire Implémenter la communication WiFi avec la sonde

8.8.2 Documentation des constructeurs et destructeur

8.8.2.1 Transmission()

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

Paramètres

parent	

Références ajouterAppareil(), demarrerScan(), gps, portGps, portSonde, scan, scanTerminer(), socket, et sonde.

```
00021                                           : QObject(parent), trame(""),
      trameGps(""), donneeLatDms(""), donneeLongDms(""),
      donneLatDD(0.), donneLongDD(0.),\
00022   signeLat(""), signeLong("")
00023 {
00024     sonde = new Sonde(this);
00025     gps = new Gps();
00026     portSonde = new QSerialPort(this);
00027     portGps = new QSerialPort(this);
00028     scan = new QBluetoothDeviceDiscoveryAgent(this);
00029     socket = new QBluetoothSocket(QBluetoothServiceInfo::RfcommProtocol);
00030
00031     connect(scan, SIGNAL(deviceDiscovered(QBluetoothDeviceInfo)), this, SLOT(
      ajouterAppareil(QBluetoothDeviceInfo)));
00032     connect(scan, SIGNAL(finished()), this, SLOT(scanTerminer()));
00033
00034     demarrerScan();
00035 }
```

8.8.2.2 ~Transmission()

```
Transmission::~Transmission ( )
```

constructeur de la classe Transmission

Références deconnecterAppareilBluetooth(), fermerPort(), et gps.

```
00043 {
00044     this->fermerPort();
00045     delete gps;
00046     deconnecterAppareilBluetooth();
00047 }
```

8.8.3 Documentation des fonctions membres

8.8.3.1 ajouterAppareil

```
void Transmission::ajouterAppareil (
           const QBluetoothDeviceInfo & info ) [slot]
```

methode appelée quand une nouvelle trame du GPS et disponible

Paramètres

info	

Références appareilDisponible, et setAppareilDisponible().

Référencé par Transmission().

```
00411 {
00412     QString appareilDisponible = info.address().toString();
00413
00414     qDebug() << "Appareil Bluetooth trouvé :" << QString("%1 %2").arg(info.address().toString()).arg(info.
      name()) << endl;
00415
00416     setAppareilDisponible(appareilDisponible);
00417 }
```

8.8.3.2 configurerPort()

```
void Transmission::configurerPort (
           QString portCommunication,
           QString DebitBaud,
           QString BitsDonnees,
           QString BitsStop )
```

configure le port de communication

modifier la liste des appareils Bluetooth disponibles

Références ouvrirPort(), et portSonde.

Référencé par Ihm : :on_pushButtonOuvrirPort_clicked().

```
00175 {
00176     portSonde = new QSerialPort(portCommunication);
00177
00178     portSonde->setBaudRate(DebitBaud.toInt());
00179     portSonde->setDataBits(QSerialPort::DataBits(BitsDonnees.toInt()));
```

```
00180     portSonde->setParity(QSerialPort::NoParity);
00181     portSonde->setStopBits(QSerialPort::StopBits(BitsStop.toInt()));
00182     portSonde->setFlowControl(QSerialPort::NoFlowControl);
00183
00184     ouvrirPort();
00185 }
```

8.8.3.3 configurerPortGps()

```
void Transmission::configurerPortGps (
            QString portCommunication,
            QString DebitBaud,
            QString BitsDonnees,
            QString BitsStop )
```

configure le port de communication Gps

configurer le port de transmission de la sonde

Références ouvrirPortGps(), et portGps.

Référencé par Ihm : :on_pushButtonOuvrirPortGps_clicked().

```
00193 {
00194     portGps = new QSerialPort(portCommunication);
00195
00196     portGps->setBaudRate(DebitBaud.toInt());
00197     portGps->setDataBits(QSerialPort::DataBits(BitsDonnees.toInt()));
00198     portGps->setParity(QSerialPort::NoParity);
00199     portGps->setStopBits(QSerialPort::StopBits(BitsStop.toInt()));
00200     portGps->setFlowControl(QSerialPort::NoFlowControl);
00201
00202     ouvrirPortGps();
00203 }
```

8.8.3.4 connecter

```
void Transmission::connecter ( )  [signal]
```

Référencé par socketConnected().

8.8.3.5 connecterAppareilBluetooth()

```
void Transmission::connecterAppareilBluetooth (
            QString appareil )
```

demarrer la recherche de périphériques Bluetooth

Paramètres

appareil	

Références setStatutBluetooth(), socket, socketConnected(), socketDisconnected(), socketErreur(), et socketReadyRead().

Référencé par Ihm : :on_pushButtonConnexion_clicked().

```
00438 {
00439     connect(socket, SIGNAL(connected()), this, SLOT(socketConnected()));
00440     connect(socket, SIGNAL(disconnected()), this, SLOT(socketDisconnected()));
00441     connect(socket, SIGNAL(readyRead()), this, SLOT(socketReadyRead()));
00442     connect(socket, QOverload<QBluetoothSocket::SocketError>::of(&QBluetoothSocket::error),
00443         [=](QBluetoothSocket::SocketError error)
00444     {
00445         qDebug() <<__FUNCTION__ << error;
00446         setStatutBluetooth("Erreur");
00447         emit socketErreur();
00448     });
00449
00450     QBluetoothAddress adresse = QBluetoothAddress(appareil);
00451     QBluetoothUuid uuid = QBluetoothUuid(QBluetoothUuid::SerialPort);
00452     socket->connectToService(adresse, uuid);
00453     socket->open(QIODevice::ReadWrite);
00454 }
```

8.8.3.6 decomposer()

```
void Transmission::decomposer ( )  [private]
```

Références Sonde : :setAltitude(), Sonde : :setAltitudeUnite(), Sonde : :setCouleurLed(), Sonde : :setHumidite(), Sonde : :setHumiditeUnite(), Sonde : :setLuminosite(), Sonde : :setLuminositeUnite(), Sonde : :setPression(), Sonde : :setPressionUnite(), Sonde : :setRessentie(), Sonde : :setRessentieUnite(), Sonde : :setTemperature(), Sonde : :setTemperatureUnite(), sonde, trame, et trameSondeRecue().

Référencé par recevoir(), et socketReadyRead().

```
00245 {
00246     if(trame.startsWith("SONDE") && trame.endsWith("\r\n"))
00247     {
00248         sonde->setTemperature(trame.section(";",1,1).toDouble());
00249         sonde->setTemperatureUnite(trame.section(";",2,2));
00250         sonde->setRessentie(trame.section(";",3,3).toDouble());
00251         sonde->setRessentieUnite(trame.section(";",4,4));
00252         sonde->setHumidite(trame.section(";",5,5).toInt());
00253         sonde->setHumiditeUnite(trame.section(";",6,6));
00254         sonde->setLuminosite(trame.section(";",7,7).toInt());
00255         sonde->setLuminositeUnite(trame.section(";",8,8));
00256         sonde->setPression(trame.section(";",9,9).toInt());
00257         sonde->setPressionUnite(trame.section(";",10,10));
00258         sonde->setAltitude(trame.section(";",11,11).toInt());
00259         sonde->setAltitudeUnite(trame.section(";",12,12));
00260         sonde->setCouleurLed(trame.section(";",17,17).toInt());
00261     }
00262     emit trameSondeRecue();
00263 }
```

8.8.3.7 decomposerDonneeGps()

```
void Transmission::decomposerDonneeGps ( )  [private]
```

decomposition de la trame recue

Références donneeLatDms, donneeLongDms, donneLatDD, donneLongDD, gps, Gps : :setLatitude(), Gps : :setLongitude(), signeLat, signeLong, trameGps, et trameGpsRecue().

Référencé par recevoirGps().

```
00271 {
00272     if(trameGps.startsWith("$GPGGA") && trameGps.endsWith("\r\n"))
00273     {
00274         donneeLatDms = trameGps.section(",",2,2);
00275         donneeLongDms = trameGps.section(",",4,4);
00276         signeLat = trameGps.section(",",3,3);
00277         signeLong = trameGps.section(",",5,5);
00278 
00279         if(signeLat == "N")
00280         {
00281             donneLatDD = donneeLatDms.left(2).toDouble() +
      donneeLatDms.mid(2,2).toDouble()/60 + donneeLatDms.mid(5,2).toDouble()/3600;
00282             qDebug() << "latitude DD:" << donneLatDD << endl;
00283             gps->setLatitude(donneLatDD);
00284         }
00285         else if(signeLong == "S")
00286         {
00287             donneLatDD = donneeLatDms.left(2).toDouble() +
      donneeLatDms.mid(2,2).toDouble()/60 + donneeLatDms.mid(5,2).toDouble()/3600;
00288             qDebug() << "latitude DD:" << donneLatDD * -1 << endl;
00289             gps->setLatitude(donneLatDD * -1);
00290         }
00291 
00292         if(signeLong == "E")
00293         {
00294             donneLongDD = donneeLongDms.left(3).toDouble() +
      donneeLongDms.mid(2,2).toDouble()/60 + donneeLongDms.mid(5,2).toDouble()/3600;
00295             qDebug() << "longitude DD:" << donneLongDD << endl;
00296             gps->setLongitude(donneLongDD);
00297         }
00298         else if(signeLat == "W")
00299         {
00300             donneLongDD = donneeLongDms.left(3).toDouble() +
      donneeLongDms.mid(2,2).toDouble()/60 + donneeLongDms.mid(5,2).toDouble()/3600;
00301             qDebug() << "longitude DD:" << donneLongDD * -1 << endl;
00302             gps->setLongitude(donneLongDD * -1);
00303         }
00304     }
00305     emit trameGpsRecue();
00306 }
```

8.8.3.8 deconnecter

`void Transmission::deconnecter ( ) [signal]`

Référencé par socketDisconnected().

8.8.3.9 deconnecterAppareilBluetooth()

`void Transmission::deconnecterAppareilBluetooth ( )`

connecter un appareil bluetooth

Références socket.

Référencé par Ihm : :on_pushButtonDeconnexion_clicked(), et ~Transmission().

```
00462 {
00463     if (socket->isOpen())
00464         {
00465             socket->close();
00466         }
00467 }
```

8.8.3.10 demarrerScan()

```
void Transmission::demarrerScan ( )
```

Références appareilDisponible, et scan.

Référencé par Ihm : :on_pushButtonScan_clicked(), et Transmission().

```
00398 {
00399     qDebug() << "scan en cours..." << endl;
00400     appareilDisponible.clear();
00401     scan->start();
00402 }
```

8.8.3.11 envoyerDonnees()

```
void Transmission::envoyerDonnees (
           QString envoyerTrame )
```

fermer le port GPS de transmission

Paramètres

envoyerTrame	

Références portFerme(), portSonde, socket, et trame.

Référencé par Ihm : :on_pushButtonEnvoyerTrame_clicked(), Ihm : :on_radioButtonLedOff_↩ clicked(), Ihm : :on_radioButtonLedOrange_clicked(), Ihm : :on_radioButtonLedRouge_clicked(), et Ihm : :on_radioButtonLedVert_clicked().

```
00369 {
00370     if(portSonde->isOpen())
00371     {
00372         const char* trame = envoyerTrame.toStdString().c_str();
00373         portSonde->write(trame);
00374
00375         qDebug() << __FUNCTION__ << ": " << trame << endl;
00376     }
00377
00378     if(socket->isOpen())
00379     {
00380         const char* trame = envoyerTrame.toStdString().c_str();
00381         socket->write(trame);
00382
00383         qDebug() << __FUNCTION__ << ": " << trame << endl;
00384     }
00385
00386     if(!socket->isOpen() && !portSonde->isOpen())
00387     {
00388         emit portFerme();
00389     }
00390 }
```

8.8.3.12 erreurConnectionPortSerieGps

```
void Transmission::erreurConnectionPortSerieGps (
           QString message ) [signal]
```

Référencé par recevoirGps().

8.8.3.13 erreurConnectionPortSerieSonde

```
void Transmission::erreurConnectionPortSerieSonde (
          QString message )  [signal]
```

Référencé par recevoir().

8.8.3.14 fermerPort()

```
void Transmission::fermerPort ( )
```

configurer le port de transmission pour le GPS

Références portFerme(), et portSonde.

Référencé par Ihm : :actualiserMessageStatutConnectionSonde(), Ihm : :on_pushButtonFermer↩Port_clicked(), et ~Transmission().

```
00211 {
00212     if(portSonde->isOpen())
00213     {
00214         portSonde->close();
00215
00216         qDebug() << __FUNCTION__ << ": " <<"port Sonde fermer" <<endl;
00217
00218         emit portFerme();
00219     }
00220 }
```

8.8.3.15 fermerPortGps()

```
void Transmission::fermerPortGps ( )
```

fermer le port de transmission de la sonde

Références portFermeGps(), et portGps.

Référencé par Ihm : :actualiserMessageStatutConnectionGps(), et Ihm : :on_pushButtonFermer↩PortGps_clicked().

```
00228 {
00229     if(portGps->isOpen())
00230     {
00231         portGps->close();
00232
00233         qDebug() << __FUNCTION__ << ": " <<"port Gps fermer" <<endl;
00234
00235         emit portFermeGps();
00236     }
00237 }
```

8.8.3.16 getAppareilDisponible()

```
QStringList Transmission::getAppareilDisponible ( ) const
```

recuperer l'objet gps

Renvoie

QStringList

Références appareilDisponible.

Référencé par Ihm : :mettreAjourAppareilBluetoothDisponible().

```
00089 {
00090     return appareilDisponible;
00091 }
```

8.8.3.17 getGps()

```
Gps * Transmission::getGps ( ) const
```

recuperer l'objet sonde

Renvoie

gps

Références gps.

Référencé par Ihm : :actualiserDonneeGps(), et Ihm : :on_pushButtonEnvoyerCoordonnee_↩clicked().

```
00078 {
00079     return gps;
00080 }
```

8.8.3.18 getSonde()

```
Sonde * Transmission::getSonde ( ) const
```

recuperer la trame

Renvoie

sonde

Références sonde.

Référencé par Ihm : :actualiserDonnee(), et Ihm : :modifierEtatLed().

```
00067 {
00068     return sonde;
00069 }
```

8.8.3.19 getStatutBluetooth()

```
QString Transmission::getStatutBluetooth ( ) const
```

recuperer les appareils disponibles

Renvoie

QString

Références statutBluetooth.

Référencé par Ihm : :actualiserMessageStatutConnecterBluetooth(), Ihm : :actualiserMessage↩ StatutDeconnecterBluetooth(), et Ihm : :actualiserMessageStatutErreurBluetooth().

```
00100 {
00101     return statutBluetooth;
00102 }
```

8.8.3.20 getTrame()

```
QString Transmission::getTrame ( ) const
```

destructeur de la classe Transmission

Renvoie

QString

Références trame.

Référencé par Ihm : :actualiserTrame().

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

8.8.3.21 nouvelleAppareilDisponible

```
void Transmission::nouvelleAppareilDisponible ( )  [signal]
```

Référencé par scanTerminer().

8.8.3.22 ouvrirPort()

```
void Transmission::ouvrirPort ( )
```

envoyer des données sur le port serie

Références portOuvert(), portSonde, et recevoir().

Référencé par configurerPort().

```
00133 {
00134     portSonde->open(QIODevice::ReadWrite);
00135     if(portSonde->isOpen())
00136     {
00137         qDebug() << __FUNCTION__ << ": " <<"Le port est ouvert";
00138 
00139         connect(portSonde, SIGNAL(readyRead()), this, SLOT(recevoir()));
00140         emit portOuvert();
00141     }
00142     else
00143     {
00144         qDebug() << __FUNCTION__ << ": " << "Erreur, le port serie de la Sonde n'a pas pu être ouvert";
00145     }
00146 }
```

8.8.3.23 ouvrirPortGps()

`void Transmission::ouvrirPortGps ( )`

Références portGps, portOuvertGps(), et recevoirGps().

Référencé par configurerPortGps().

```
00154 {
00155     portGps->open(QIODevice::ReadWrite);
00156     if(portGps->isOpen())
00157     {
00158         qDebug() << __FUNCTION__ << ": " <<"Le port est ouvert";
00159 
00160         connect(portGps, SIGNAL(readyRead()), this, SLOT(recevoirGps()));
00161         emit portOuvertGps();
00162     }
00163     else
00164     {
00165         qDebug() << __FUNCTION__ << ": " << "Erreur, le port serie du GPS n'a pas pu être ouvert";
00166     }
00167 }
```

8.8.3.24 portFerme

`void Transmission::portFerme ( ) [signal]`

Référencé par envoyerDonnees(), et fermerPort().

8.8.3.25 portFermeGps

`void Transmission::portFermeGps ( ) [signal]`

Référencé par fermerPortGps().

8.8.3.26 portOuvert

`void Transmission::portOuvert ( ) [signal]`

Référencé par ouvrirPort().

8.8.3.27 portOuvertGps

`void Transmission::portOuvertGps ( ) [signal]`

Référencé par ouvrirPortGps().

8.8.3.28 recevoir

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

Références decomposer(), erreurConnectionPortSerieSonde(), portSonde, et trame.

Référencé par ouvrirPort().

```
00314 {
00315     QByteArray donnees;
00316 
00317         while(portSonde->waitForReadyRead(10))
00318         {
00319             donnees += portSonde->readAll();
00320         }
00321         trame = QString(donnees.data());
00322 
00323         if(trame.startsWith("SONDE") && trame.endsWith("\r\n"))
00324         {
00325             qDebug() << Q_FUNC_INFO << "trame Port serie reçu : " << trame << endl;
00326 
00327             this->decomposer();
00328         }else if(trame.startsWith("$GPGGA") && trame.endsWith("\r\n"))
00329         {
00330             qDebug() << Q_FUNC_INFO << "mauvais module connecté" << endl;
00331             emit erreurConnectionPortSerieSonde("Erreur Appareil Connecté");
00332         }
00333 
00334 }
```

8.8.3.29 recevoirGps

`void Transmission::recevoirGps ( ) [slot]`

methode appelée quand une nouvelle trame et disponible

Références decomposerDonneeGps(), erreurConnectionPortSerieGps(), portGps, et trameGps.

Référencé par ouvrirPortGps().

```
00342 {
00343     QByteArray donnees;
00344 
00345         while(portGps->waitForReadyRead(10))
00346         {
00347             donnees += portGps->readAll();
00348         }
00349         trameGps = QString(donnees.data());
00350 
00351         if(trameGps.startsWith("$GPGGA") && trameGps.endsWith("\r\n"))
00352         {
00353             qDebug() << Q_FUNC_INFO << "trame Gps reçu : " << trameGps << endl;
00354             this->decomposerDonneeGps();
00355         }else if(trameGps.startsWith("SONDE") && trameGps.endsWith("\r\n"))
00356         {
00357             qDebug() << Q_FUNC_INFO << "mauvais module connecté" << endl;
00358             emit erreurConnectionPortSerieGps("Erreur Appareil Connecté");
00359         }
00360 }
```

8.8.3.30 scanfini

```
void Transmission::scanfini ( )  [signal]
```

Référencé par scanTerminer().

8.8.3.31 scanTerminer

```
void Transmission::scanTerminer ( )  [slot]
```

methode appelée quand un nouveau appareil est disponible

Références nouvelleAppareilDisponible(), et scanfini().

Référencé par Transmission().

```
00425 {
00426     qDebug() << "scan terminé";
00427     emit scanfini();
00428     emit nouvelleAppareilDisponible();
00429 }
```

8.8.3.32 setAppareilDisponible()

```
void Transmission::setAppareilDisponible (
            QString appareilDisponible )
```

modifier le statut de la connexion Bluetooth

Paramètres

appareilDisponible	

Références appareilDisponible.

Référencé par ajouterAppareil().

```
00122 {
00123     this->appareilDisponible << appareilDisponible;
00124
00125 }
```

8.8.3.33 setStatutBluetooth()

```
void Transmission::setStatutBluetooth (
            QString statutBluetooth )
```

recuperer les message de statut de la connexion

Paramètres

statutBluetooth	

Références statutBluetooth.

Référencé par connecterAppareilBluetooth(), socketConnected(), et socketDisconnected().

```
00111 {
00112     this->statutBluetooth = statutBluetooth;
00113 }
```

8.8.3.34 socketConnected

`void Transmission::socketConnected ( ) [slot]`

methode appelée quand le scan est terminé

Références connecter(), setStatutBluetooth(), et socket.

Référencé par connecterAppareilBluetooth().

```
00475 {
00476     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Périphérique connecté ") +
      socket->peerName() + " [" + socket->peerAddress().toString() + "]";
00477     QString message = "connecter à " + socket->peerName();
00478     setStatutBluetooth(message);
00479     emit connecter();
00480 }
```

8.8.3.35 socketDisconnected

`void Transmission::socketDisconnected ( ) [slot]`

methode appelée quand l'appareil est connecté

Références deconnecter(), et setStatutBluetooth().

Référencé par connecterAppareilBluetooth().

```
00488 {
00489     qDebug() << Q_FUNC_INFO;
00490     QString message = "Périphérique déconnecté";
00491     qDebug() << message;
00492     setStatutBluetooth(message);
00493     emit deconnecter();
00494 }
```

8.8.3.36 socketErreur

`void Transmission::socketErreur ( ) [signal]`

Référencé par connecterAppareilBluetooth().

8.8.3.37 socketReadyRead

`void Transmission::socketReadyRead ( ) [slot]`

methode appelée quand l'appareil est deconnecté

Références decomposer(), socket, et trame.

Référencé par connecterAppareilBluetooth().

```
00502 {
00503     qDebug() << Q_FUNC_INFO;
00504     QByteArray donnees;
00505
00506     while (socket->bytesAvailable())
00507     {
00508         donnees += socket->readAll();
00509         usleep(150000); // cf. timeout
00510     }
00511     trame = QString(donnees);
00512     qDebug() << "Données bluetooth reçues :" << QString(donnees);
00513
00514     if(trame.startsWith("SONDE") && trame.endsWith("\r\n"))
00515     {
00516         decomposer();
00517     }
00518 }
```

8.8.3.38 trameGpsRecue

`void Transmission::trameGpsRecue ( ) [signal]`

Référencé par decomposerDonneeGps().

8.8.3.39 trameSondeRecue

`void Transmission::trameSondeRecue ( ) [signal]`

déconnecter l'appareil bluetooth

Référencé par decomposer().

8.8.4 Documentation des données membres

8.8.4.1 appareilDisponible

`QStringList Transmission::appareilDisponible [private]`

decomposition de la trame GPS recue

Référencé par ajouterAppareil(), demarrerScan(), getAppareilDisponible(), et setAppareil↩ Disponible().

8.8.4.2 donneeLatDms

```
QString Transmission::donneeLatDms [private]
```

Référencé par decomposerDonneeGps().

8.8.4.3 donneeLongDms

```
QString Transmission::donneeLongDms [private]
```

Référencé par decomposerDonneeGps().

8.8.4.4 donneLatDD

```
double Transmission::donneLatDD [private]
```

Référencé par decomposerDonneeGps().

8.8.4.5 donneLongDD

```
double Transmission::donneLongDD [private]
```

Référencé par decomposerDonneeGps().

8.8.4.6 gps

```
Gps* Transmission::gps [private]
```

Référencé par decomposerDonneeGps(), getGps(), Transmission(), et ~Transmission().

8.8.4.7 m_controller

```
QLowEnergyController* Transmission::m_controller [private]
```

8.8.4.8 portGps

```
QSerialPort* Transmission::portGps [private]
```

Référencé par configurerPortGps(), fermerPortGps(), ouvrirPortGps(), recevoirGps(), et Transmission().

8.8.4.9 portSonde

```
QSerialPort* Transmission::portSonde [private]
```

Référencé par configurerPort(), envoyerDonnees(), fermerPort(), ouvrirPort(), recevoir(), et Transmission().

8.8.4.10 scan

```
QBluetoothDeviceDiscoveryAgent* Transmission::scan [private]
```

Référencé par demarrerScan(), et Transmission().

8.8.4.11 signeLat

```
QString Transmission::signeLat [private]
```

Référencé par decomposerDonneeGps().

8.8.4.12 signeLong

```
QString Transmission::signeLong [private]
```

Référencé par decomposerDonneeGps().

8.8.4.13 socket

```
QBluetoothSocket* Transmission::socket [private]
```

Référencé par connecterAppareilBluetooth(), deconnecterAppareilBluetooth(), envoyerDonnees(), socketConnected(), socketReadyRead(), et Transmission().

8.8.4.14 sonde

```
Sonde* Transmission::sonde [private]
```

Référencé par decomposer(), getSonde(), et Transmission().

8.8.4.15 statutBluetooth

```
QString Transmission::statutBluetooth [private]
```

Référencé par getStatutBluetooth(), et setStatutBluetooth().

8.8.4.16 trame

```
QString Transmission::trame [private]
```

methode appelée quand une trame est disponible

Référencé par decomposer(), envoyerDonnees(), getTrame(), recevoir(), et socketReadyRead().

8.8.4.17 trameGps

```
QString Transmission::trameGps [private]
```

Référencé par decomposerDonneeGps(), et recevoirGps().

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

9 Documentation des fichiers

9.1 Référence du fichier Changelog.md

9.2 Référence du fichier gps.cpp

classe qui contient les valeurs du GPS (latitude / longitude)

```
#include "gps.h"
```

9.2.1 Description détaillée

Auteur

Bounoir Fabien
Villesseche Ethan

Version

4.1

9.3 Référence du fichier gps.h

Declaration de la classe Gps.

```
#include <QObject>
```

Classes

— class Gps
Declaration de la classe Gps.

9.3.1 Description détaillée

Version

4.1

Auteur

Bounoir Fabien
Villesseche Ethan

9.4 Référence du fichier graphique.cpp

classe qui gére les differents graphiques

```
#include "graphique.h"
```

9.4.1 Description détaillée

Auteur

Bounoir Fabien
Villesseche Ethan

Version

4.1

9.5 Référence du fichier graphique.h

Declaration de la classe Graphique.

```
#include <QObject>
#include <QtWidgets>
#include <QtCharts>
#include <QApplication>
```

Classes

— class Graphique
Declaration de la classe Graphique.

9.5.1 Description détaillée

Version

4.1

Auteur

Bounoir Fabien
Villesseche Ethan

9.6 Référence du fichier ihm.cpp

classe qui s'occupe de l'affichage dans l'ihm

```
#include "ihm.h"
#include "ui_ihm.h"
#include <QDebug>
```

9.6.1 Description détaillée

Auteur

Bounoir Fabien
Villesseche Ethan

Version

4.1

9.7 Référence du fichier ihm.h

Declaration de la classe Ihm.

```
#include "transmission.h"
#include "meteo.h"
#include "graphique.h"
#include <QWidget>
#include <QDateTime>
#include <QSettings>
#include <QApplication>
```

Classes

— class Ihm
Declaration de la classe Ihm.

Espaces de nommage

— Ui

Macros

— #define MODE_OPERATEUR_ACTIVER 2
— #define ONGLET_MODE_OPERATEUR 4
— #define LED_ROUGE "../Sonde/image/ledRouge.png"
— #define LED_VERT "../Sonde/image/ledVert.png"
— #define LED_ORANGE "../Sonde/image/ledOrange.png"
— #define LED_OFF "../Sonde/image/ledEteint.png"
— #define VALEUR_LED_OFF 0
— #define VALEUR_LED_ROUGE 1
— #define VALEUR_LED_VERT 2
— #define VALEUR_LED_ORANGE 3

9.7.1 Description détaillée

Version

4.1

Auteur

Bounoir Fabien
Villesseche Ethan

9.7.2 Documentation des macros

9.7.2.1 LED_OFF

```
#define LED_OFF "../Sonde/image/ledEteint.png"
```

Référencé par Ihm : :initialiserAffichage(), Ihm : :initialiserInterface(), et Ihm : :modifierEtatLed().

9.7.2.2 LED_ORANGE

```
#define LED_ORANGE "../Sonde/image/ledOrange.png"
```

Référencé par Ihm : :modifierEtatLed().

9.7.2.3 LED_ROUGE

```
#define LED_ROUGE "../Sonde/image/ledRouge.png"
```

Référencé par Ihm : :modifierEtatLed().

9.7.2.4 LED_VERT

```
#define LED_VERT "../Sonde/image/ledVert.png"
```

Référencé par Ihm : :modifierEtatLed().

9.7.2.5 MODE_OPERATEUR_ACTIVER

```
#define MODE_OPERATEUR_ACTIVER 2
```

Référencé par Ihm : :on_ModeOperateur_stateChanged().

9.7.2.6 ONGLET_MODE_OPERATEUR

```
#define ONGLET_MODE_OPERATEUR 4
```

Référencé par Ihm : :initialiserInterface(), et Ihm : :on_ModeOperateur_stateChanged().

9.7.2.7 VALEUR_LED_OFF

```
#define VALEUR_LED_OFF 0
```

Référencé par Ihm : :modifierEtatLed().

9.7.2.8 VALEUR_LED_ORANGE

```
#define VALEUR_LED_ORANGE 3
```

Référencé par Ihm : :modifierEtatLed().

9.7.2.9 VALEUR_LED_ROUGE

```
#define VALEUR_LED_ROUGE 1
```

Référencé par Ihm : :modifierEtatLed().

9.7.2.10 VALEUR_LED_VERT

```
#define VALEUR_LED_VERT 2
```

Référencé par Ihm : :modifierEtatLed().

9.8 Référence du fichier main.cpp

Programme principal Sonde.

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

Fonctions

— int main (int argc, char *argv[])

9.8.1 Description détaillée

Crée et affiche la fenêtre principale de l'application Sonde

Auteur

Bounoir Fabien
Villesseche Ethan

Version

4.1

9.8.2 Documentation des fonctions

9.8.2.1 main()

```
main (
            int argc,
            char * argv[] )
```

Paramètres

argc	
argv[]	

Renvoie

int

Référencé par Meteo : :replyFinished().

```
00023 {
00024     QApplication a(argc, argv);
00025     Ihm w;
00026     w.show();
00027
00028     return a.exec();
00029 }
```

9.9 Référence du fichier meteo.cpp

classe qui gere les requetes avec l'api

```
#include "meteo.h"
```

9.9.1 Description détaillée

Auteur

Bounoir Fabien
Villesseche Ethan

Version

4.1

9.10 Référence du fichier meteo.h

Declaration de la classe Meteo.

```
#include <QObject>
#include <QNetworkRequest>
```

```
#include <QNetworkReply>
#include <QNetworkAccessManager>
#include <QJsonDocument>
#include <QJsonObject>
#include <QJsonArray>
#include <QDebug>
#include <QGeoPositionInfoSource>
#include <QGeoCoordinate>
```

Classes

— class Meteo
Declaration de la classe Meteo.

Macros

— #define API_KEY "a2157cdc4a03c47c79e8414161c59762"
clé de l'API d'Openweather

9.10.1 Description détaillée

Version

4.1

Auteur

Bounoir Fabien
Villesseche Ethan

9.10.2 Documentation des macros

9.10.2.1 API_KEY

```
#define API_KEY "a2157cdc4a03c47c79e8414161c59762"
```

Référencé par Meteo : :creerUrlCoordonnerGps(), et Meteo : :creerUrlVille().

9.11 Référence du fichier README.md

9.12 Référence du fichier sonde.cpp

classe qui contient les valeurs des differents capteurs

```
#include "sonde.h"
```

9.12.1 Description détaillée

Auteur

Bounoir Fabien
Villesseche Ethan

Version

4.1

9.13 Référence du fichier sonde.h

Declaration de la classe Sonde.

```
#include <QObject>
#include <QString>
#include <QtDebug>
```

Classes

— class Sonde
Declaration de la classe Sonde.

9.13.1 Description détaillée

Version

4.1

Auteur

Bounoir Fabien
Villesseche Ethan

9.14 Référence du fichier transmission.cpp

classe qui s'occupe de la parti trame (configuration, reception / envoie de trame)

```
#include "transmission.h"
```

9.14.1 Description détaillée

Auteur

Bounoir Fabien
Villesseche Ethan

Version

4.1

9.15 Référence du fichier transmission.h

Declaration de la classe Transmission.

```
#include <QObject>
#include <QString>
#include <QSerialPort>
#include <unistd.h>
#include <qbluetoothaddress.h>
#include <qbluetoothservicediscoveryagent.h>
#include <QLowEnergyController>
#include <QBluetoothLocalDevice>
#include <QBluetoothSocket>
#include "sonde.h"
#include "gps.h"
```

Classes

— class Transmission
Declaration de la classe Transmission (Liaison série et Bluetooth)

9.15.1 Description détaillée

Version

4.1

Auteur

Bounoir Fabien
Villesseche Ethan

Index