

Kart

1.1

Généré par Doxygen 1.7.6.1

Jeudi Juin 7 2018 20 :28 :16

Table des matières

1	Page principale du projet Kart	1
1.1	Introduction	1
1.2	Table des matières	1
2	Changelog	2
3	Configuration	3
4	Manuel d'installation	3
5	Recette IR	4
6	Fichiers XML	4
7	A propos	4
8	Licence GPL	4
9	Liste des choses à faire	5
10	Documentation des classes	5
10.1	Référence de la classe Communication	5
10.1.1	Description détaillée	6
10.1.2	Documentation des constructeurs et destructeur	6
10.1.3	Documentation des fonctions membres	6
10.1.4	Documentation des données membres	9
10.2	Référence de la structure Courbe	10
10.2.1	Documentation des données membres	10
10.3	Référence de la structure Echantillon	10
10.3.1	Documentation des données membres	10
10.4	Référence de la classe IHMKart	11
10.4.1	Description détaillée	13
10.4.2	Documentation des constructeurs et destructeur	14
10.4.3	Documentation des fonctions membres	14
10.4.4	Documentation des données membres	21
10.5	Référence de la classe Telemetry	22

10.5.1	Documentation des constructeurs et destructeur	22
10.5.2	Documentation des fonctions membres	23
10.5.3	Documentation des données membres	23
11	Documentation des fichiers	24
11.1	Référence du fichier Changelog.dox	24
11.2	Référence du fichier communication.cpp	24
11.2.1	Description détaillée	24
11.3	Référence du fichier communication.h	24
11.3.1	Description détaillée	24
11.3.2	Documentation des macros	24
11.4	Référence du fichier ihmkart.cpp	25
11.4.1	Description détaillée	25
11.5	Référence du fichier ihmkart.h	25
11.5.1	Description détaillée	26
11.6	Référence du fichier main.cpp	26
11.6.1	Description détaillée	26
11.6.2	Documentation des fonctions	26
11.7	Référence du fichier README.dox	26
11.8	Référence du fichier telemetrie.cpp	26
11.8.1	Description détaillée	27
11.9	Référence du fichier telemetrie.h	27
11.9.1	Description détaillée	27

1 Page principale du projet Kart

1.1 Introduction

Conception d'une chaine de télémétrie des essais en course d'un kart électrique.

1.2 Table des matières

- [Configuration](#)
- [Manuel d'installation](#)
- [Changelog](#)
- [Recette IR](#)
- [Fichiers XML](#)

- [A propos](#)
- [Licence GPL](#)

Dépôt SVN : <https://svn.riouxsvn.com/kart-2018>

2 Changelog

r15 | fpenne | 2018-06-07 18 :09 :00 +0200 (jeu. 07 juin 2018) | 1 ligne
version 1.1

r14 | fpenne | 2018-05-25 16 :59 :55 +0200 (ven. 25 mai 2018) | 1 ligne
Tag de la version 1.0

r13 | fpenne | 2018-04-18 15 :10 :12 +0200 (mer. 18 avril 2018) | 1 ligne
Ajout classe XML

r12 | tvaira | 2018-04-16 11 :31 :31 +0200 (lun. 16 avril 2018) | 1 ligne
Verification des TODO

r11 | fpenne | 2018-04-06 11 :54 :45 +0200 (ven. 06 avril 2018) | 1 ligne
Ajout de courbe en temps réel et modification du layout

r10 | fpenne | 2018-04-04 17 :55 :40 +0200 (mer. 04 avril 2018) | 1 ligne
Ajout courbe vitesse tension et courant en temps réel

r9 | tvaira | 2018-04-01 12 :57 :56 +0200 (dim. 01 avril 2018) | 1 ligne
Retour sur la revue 2

r8 | tvaira | 2018-03-24 09 :48 :16 +0100 (sam. 24 mars 2018) | 1 ligne
Ajout parametrage Doxygen

r7 | tvaira | 2018-03-24 08 :31 :35 +0100 (sam. 24 mars 2018) | 1 ligne
Mise à jour du simulateur (cf. protocole)

r6 | fpenne | 2018-03-21 16 :57 :39 +0100 (mer. 21 mars 2018) | 1 ligne

Ajout des signaux , edition de l'affichage

r5 | fpenne | 2018-03-21 15 :51 :36 +0100 (mer. 21 mars 2018) | 1 ligne

Ajout du projet Karting : réception, extraction et affichage de 2 champs de la trame

r4 | fpenne | 2018-03-21 12 :12 :49 +0100 (mer. 21 mars 2018) | 1 ligne

correction du commit precedent

r3 | fpenne | 2018-03-21 12 :06 :52 +0100 (mer. 21 mars 2018) | 1 ligne

Obtention trame , decoupage de celle ci , affichage sur IHM , port configuré

r2 | tvaira | 2018-02-03 11 :11 :52 +0100 (sam. 03 févr. 2018) | 1 ligne

Ajout initial (tv)

r1 | www-data | 2018-02-03 11 :02 :45 +0100 (sam. 03 févr. 2018) | 1 ligne

Creating initial repository structure

3 Configuration

Poste de développement :

- Distribution : Ubuntu 12.04.5 LTS
- OS : GNU/Linux
- Noyau : Linux
- Version : 3.8.0-44-generic
- Machine : x86_64
- Processeur : Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
- Mémoire RAM : 8129984 kB

Liste des paquets Qt nécessaires :

- libqwt6 libqt4-xml libqtgui4 libqtcore4 libqt4-svg

Modules Xbee : voir dossier

4 Manuel d'installation

Fabrication de l'exécutable :

- `qmake`
- `make`

5 Recette IR

Étudiant 3 : Penne Florent

- La supervision d'un essai permet de démarrer la télémétrie
- La récupération des données télémétriques est opérationnelle
- L'affichage des données télémétriques en temps réel est fonctionnel
- L'affichage des données télémétriques sous forme de graphiques est fonctionnel
- L'exportation des données télémétriques au format CSV est possible

6 Fichiers XML

A faire définir la structure des fichiers XML

7 A propos

Auteur

Penne Florent <*florent.penne@orange.fr*>

Version

0.9

Date

2018

8 Licence GPL

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

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

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

9 Liste des choses à faire

Page [Fichiers XML](#)

définir la structure des fichiers XML

Membre [IHMKart](#) : [:afficherGraphiques](#) ([Echantillon](#) [echantillon](#))

réaliser l'affichage des graphiques pour la charge, tension et courant

Membre [IHMKart](#) : [:arreterEssai](#) ()

exporter les données télémétriques au format CSV

Membre [IHMKart](#) : [:chargerIDKarts](#) ()

lire les ID kart dans le fichier xml et charger la liste

Membre [IHMKart](#) : [:chargerIDOperateurs](#) ()

lire les ID opérateur dans le fichier xml et charger la liste

Membre [IHMKart](#) : [:initialiserQWT](#) ()

initialiser les autres graphiques (charge, tension et courant)

10 Documentation des classes

10.1 Référence de la classe [Communication](#)

Permet la réception des trames envoyées par le kart.

```
#include <communication.h>
```

Connecteurs publics

- void [recevoir](#) ()

Signaux

- void [nouvelEchantillon](#) ([Echantillon](#) [echantillon](#))

Fonctions membres publiques

- [Communication](#) (QObject *parent=0)
Constructeur par défaut.
- [~Communication](#) ()
Destructeur.
- QString [getIdKart](#) ()
- void [setIdKart](#) (QString [idKart](#))

Fonctions membres privées

- void [creerPort](#) ()
- void [ouvrirPort](#) ()
- void [configurerPort](#) ()
- bool [verifierTrame](#) (QString &donneesLues)
- void [extraireDonnees](#) ()
- void [fermerPort](#) ()

Attributs privés

- QextSerialPort * [port](#)
- QString [idKart](#)
- QString [trame](#)

10.1.1 Description détaillée

Auteur

Penne Florent

Version

0.9

10.1.2 Documentation des constructeurs et destructeur

10.1.2.1 Communication : :Communication (QObject * *parent* = 0)

Paramètres

<i>parent</i>	QObject l'adresse de l'objet Qt parent
---------------	--

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

```
        : QObject (parent), port (NULL),  
        idKart ("")  
{  
    creerPort ();  
    ouvrirPort ();  
    configurerPort ();  
}
```

10.1.2.2 Communication : :~Communication ()

Références [fermerPort\(\)](#).

```
{  
    fermerPort ();  
}
```

10.1.3 Documentation des fonctions membres

10.1.3.1 Communication : :configurerPort () [private]

Références [port](#), et [recevoir\(\)](#).Référéncé par [Communication\(\)](#).

```
{  
    if (port->isOpen ())  
    {  
        port->setBaudRate (BAUD9600);  
    }
```



```

        port->setDataBits(DATA_8);
        port->setStopBits(STOP_1);

        connect(port, SIGNAL(readyRead()), this, SLOT(recevoir()));
    }
}

```

10.1.3.2 Communication : :creerPort() [private]

Références [PORT](#), et [port](#).

Référencé par [Communication\(\)](#).

```

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

```

10.1.3.3 Communication : :extraireDonnees() [private]

Références [CHAMP_CHARGE](#), [CHAMP_COURANT](#), [CHAMP_IDKART](#), [CHAMP_TEMPERATURE](#), [CHAMP_TENSION](#), [CHAMP_VITESSE](#), [Echantillon : :chargeBatterie](#), [Echantillon : :courantBatterie](#), [Echantillon : :horodatage](#), [Echantillon : :idKart](#), [idKart](#), [nouvelEchantillon\(\)](#), [Echantillon : :temperatureMoteur](#), [Echantillon : :tensionBatterie](#), [trame](#), et [Echantillon : :vitesse](#).

Référencé par [recevoir\(\)](#).

```

{
    // Retire le \r
    trame.chop(1);

    // Retire le $
    trame.remove(0, 1);
    QStringList champs = trame.split(",");
    qDebug() << Q_FUNC_INFO << champs;

    Echantillon echantillon;

    echantillon.idKart = champs.at(CHAMP_IDKART);
    echantillon.horodatage = QDateTime::currentDateTime();
    echantillon.temperatureMoteur = champs.at(CHAMP_TEMPERATURE).toDouble();
    echantillon.tensionBatterie = champs.at(CHAMP_TENSION).toDouble();
    echantillon.courantBatterie = champs.at(CHAMP_COURANT).toDouble();
    echantillon.vitesse = champs.at(CHAMP_VITESSE).toDouble();
    echantillon.chargeBatterie = champs.at(CHAMP_CHARGE).toInt();

    if(echantillon.idKart == idKart)
    {
        // Envoie le signal qui contient l'échantillon extrait
        emit nouvelEchantillon(echantillon);
    }
}

```

10.1.3.4 Communication : :fermerPort() [private]

Références [port](#).

Référencé par [~Communication\(\)](#).

```

{
    port->close();
}

```

10.1.3.5 QString Communication : :getIldKart ()

Renvoie

QString

Références [idKart](#).

```
{  
    return idKart;  
}
```

10.1.3.6 Communication : :nouvelEchantillon (Echantillon *echantillon*) [signal]

Paramètres

<i>echantillon</i>	
--------------------	--

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

10.1.3.7 Communication : :ouvrirPort () [private]

Références [port](#).

Référencé par [Communication\(\)](#).

```
{  
    //Ouverture port droit ecriture lecture  
    port->open(QIODevice::ReadWrite | QIODevice::Unbuffered);  
}
```

10.1.3.8 Communication : :recevoir () [slot]

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

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

```
{  
    QByteArray donnees;  
  
    while(port->bytesAvailable())  
    {  
        donnees += port->readAll();  
        usleep(50000);  
    }  
  
    //qDebug() << Q_FUNC_INFO << donneesLues;  
    QString donneesLues(donnees);  
    qDebug() << Q_FUNC_INFO << donneesLues;  
  
    if(verifierTrame(donneesLues))  
    {  
        trame = donneesLues;  
        extraireDonnees();  
    }  
}
```

10.1.3.9 void Communication : :setIdKart (QString idKart)

Paramètres

<i>idKart</i>	
---------------	--

Références [idKart](#).

Référéncé par [IHMkart : :arreterEssai\(\)](#), et [IHMkart : :demarrerEssai\(\)](#).

```
{
    this->idKart = idKart;
}
```

10.1.3.10 Communication : :verifierTrame (QString & donneesLues) [private]

Paramètres

<i>donnees- Lues</i>	
--------------------------	--

Renvoie

bool

Référéncé par [recevoir\(\)](#).

```
{
    if(donneesLues.startsWith("$"))
    {
        if(donneesLues.endsWith("\r"))
        {
            return true;
        }
    }
    return false;
}
```

10.1.4 Documentation des données membres

10.1.4.1 QString Communication : :idKart [private]

Référéncé par [extraireDonnees\(\)](#), [getIdKart\(\)](#), et [setIdKart\(\)](#).

10.1.4.2 QTextSerialPort* Communication : :port [private]

Référéncé par [configurerPort\(\)](#), [creerPort\(\)](#), [fermerPort\(\)](#), [ouvrirPort\(\)](#), et [recevoir\(\)](#).

10.1.4.3 QString Communication : :trame [private]

Référéncé par [extraireDonnees\(\)](#), et [recevoir\(\)](#).

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

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

10.2 Référence de la structure Courbe

La structure de données pour une courbe.

```
#include <ihmkart.h>
```

Attributs publics

- `QVector< double > x`
- `QVector< double > y`

10.2.1 Documentation des données membres

10.2.1.1 `QVector<double> Courbe : :x`

les valeurs (abscisse)

Référéncé par `IHMkart : :afficherGraphiques()`, et `IHMkart : :reinitialiserGraphiques()`.

10.2.1.2 `QVector<double> Courbe : :y`

les valeurs (ordonnée)

Référéncé par `IHMkart : :afficherGraphiques()`, et `IHMkart : :reinitialiserGraphiques()`.

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

- [ihmkart.h](#)

10.3 Référence de la structure Echantillon

La structure de données pour un échantillon.

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

Attributs publics

- `QString idKart`
- `QDateTime horodatage`
- `double temperatureMoteur`
- `double tensionBatterie`
- `double courantBatterie`
- `int vitesse`
- `int chargeBatterie`

10.3.1 Documentation des données membres

10.3.1.1 `int Echantillon : :chargeBatterie`

Référéncé par `IHMkart : :afficherDonneesTelemetrie()`, `IHMkart : :afficherGraphiques()`, et `Communication : :extraireDonnees()`.

10.3.1.2 double Echantillon : :courantBatterie

Référencé par [IHMKart : :afficherDonneesTelemetrie\(\)](#), [IHMKart : :afficherGraphiques\(\)](#), et [Communication : :extraireDonnees\(\)](#).

10.3.1.3 QDateTime Echantillon : :horodatage

Référencé par [IHMKart : :afficherDonneesTelemetrie\(\)](#), et [Communication : :extraireDonnees\(\)](#).

10.3.1.4 QString Echantillon : :idKart

Référencé par [Communication : :extraireDonnees\(\)](#).

10.3.1.5 double Echantillon : :temperatureMoteur

Référencé par [IHMKart : :afficherDonneesTelemetrie\(\)](#), [IHMKart : :afficherGraphiques\(\)](#), et [Communication : :extraireDonnees\(\)](#).

10.3.1.6 double Echantillon : :tensionBatterie

Référencé par [IHMKart : :afficherDonneesTelemetrie\(\)](#), [IHMKart : :afficherGraphiques\(\)](#), et [Communication : :extraireDonnees\(\)](#).

10.3.1.7 int Echantillon : :vitesse

Référencé par [IHMKart : :afficherDonneesTelemetrie\(\)](#), [IHMKart : :afficherGraphiques\(\)](#), et [Communication : :extraireDonnees\(\)](#).

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

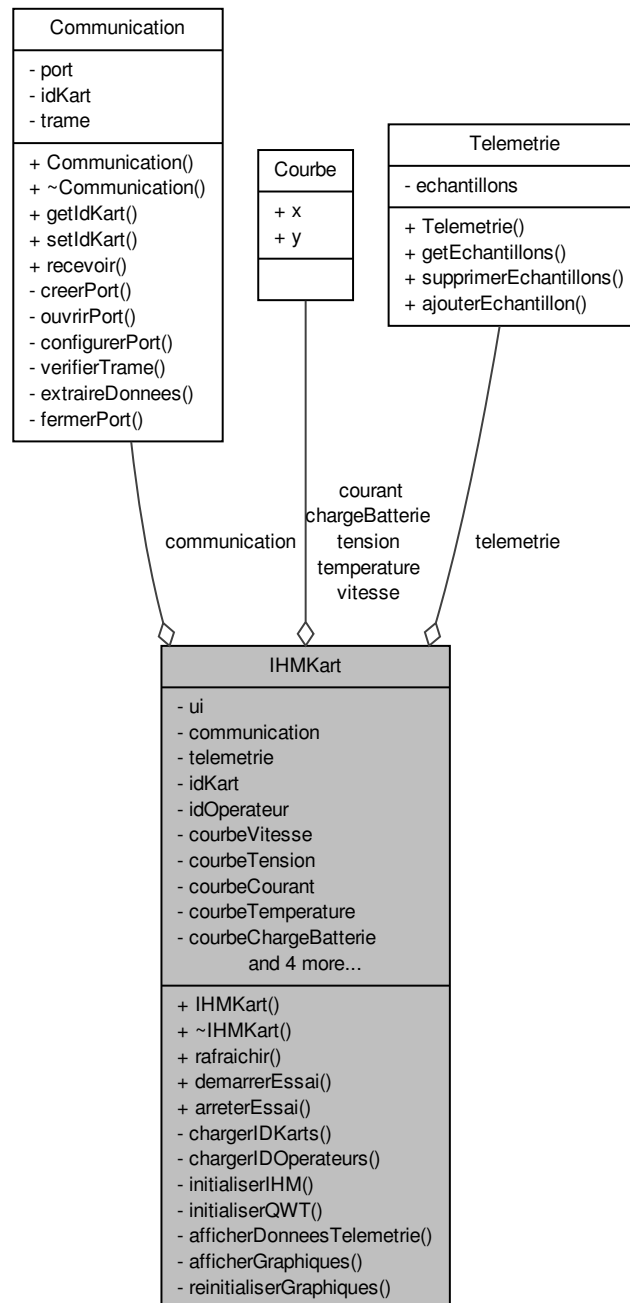
– [telemetrie.h](#)

10.4 Référence de la classe IHMKart

Fenêtre principale de l'application.

```
#include <ihmkart.h>
```

Graphe de collaboration de IHMKart :



Connecteurs publics

- void [rafraichir](#) ([Echantillon](#) echantillon)
Slot gérant l'affichage des données reçues.
- void [demarrerEssai](#) ()
Permet à un opérarteur de démarrer un essai pour un kart.
- void [arreterEssai](#) ()

Fonctions membres publiques

- [IHMKart](#) (QWidget *parent=0)
Constructeur de la classe fenêtre principale.
- [~IHMKart](#) ()
Destructeur.

Fonctions membres privées

- void [chargerIDKarts](#) ()
- void [chargerIDOperateurs](#) ()
- void [initialiserIHM](#) ()
- void [initialiserQWT](#) ()
- void [afficherDonneesTelemetrie](#) ([Echantillon](#) echantillon)
Affichage des données reçues.
- void [afficherGraphiques](#) ([Echantillon](#) echantillon)
Affichage des graphiques.
- void [reinitialiserGraphiques](#) ()

Attributs privés

- Ui : :IHMKart * [ui](#)
- [Communication](#) * [communication](#)
- [Telemetrie](#) [telemetrie](#)
- QString [idKart](#)
- QString [idOperateur](#)
- QwtPlotCurve * [courbeVitesse](#)
- QwtPlotCurve * [courbeTension](#)
- QwtPlotCurve * [courbeCourant](#)
- QwtPlotCurve * [courbeTemperature](#)
- QwtPlotCurve * [courbeChargeBatterie](#)
- [Courbe](#) [vitesse](#)
- [Courbe](#) [tension](#)
- [Courbe](#) [courant](#)
- [Courbe](#) [temperature](#)
- [Courbe](#) [chargeBatterie](#)

10.4.1 Description détaillée

Auteur

Penne Florent

Version

0.9

10.4.2 Documentation des constructeurs et destructeur

10.4.2.1 IHMKart : :IHMKart (QWidget * *parent* = 0) [explicit]

Paramètres

<i>parent</i>	QObject l'adresse de l'objet Qt parent (par défaut 0 pour la fenêtre principale)
---------------	--

Références [arreterEssai\(\)](#), [chargerIDKarts\(\)](#), [chargerIDOperateurs\(\)](#), [communication](#), [demarrerEssai\(\)](#), [initialiserIHM\(\)](#), [initialiserQWT\(\)](#), [rafraichir\(\)](#), et [ui](#).

```

    QMainWindow(parent),
    ui(new Ui::IHMKart)
{
    initialiserIHM();
    initialiserQWT();

    chargerIDKarts();

    chargerIDOperateurs();

    communication = new Communication(this);

    connect(ui->bDemarrageEssai, SIGNAL(clicked()), this, SLOT(demarrerEssai()));
    connect(ui->bArretEssai, SIGNAL(clicked()), this, SLOT(arreterEssai()));
    connect(communication, SIGNAL(nouvelEchantillon(Echantillon)), this, SLOT(
        rafraichir(Echantillon)));
}

```

10.4.2.2 IHMKart : :~IHMKart ()

Références [ui](#).

```

{
    delete ui;
}

```

10.4.3 Documentation des fonctions membres

10.4.3.1 void IHMKart : :afficherDonneesTelemetrie (Echantillon *echantillon*)
[private]

Paramètres

<i>echantillon</i>	
--------------------	--

Références [Echantillon : :chargeBatterie](#), [Echantillon : :courantBatterie](#), [Echantillon : :horodatage](#), [Echantillon : :temperatureMoteur](#), [Echantillon : :tensionBatterie](#), [ui](#), et [Echantillon : :vitesse](#).

Référencé par [rafraichir\(\)](#).

```

{
    ui->lcdTension->display(echantillon.tensionBatterie);
    ui->lcdCourant->display(echantillon.courantBatterie);
}

```



```

    ui->lcdCharge->display(echantillon.chargeBatterie);
    ui->lcdVitesse->display(echantillon.vitesse);
    ui->lcdTemperature->display(echantillon.temperatureMoteur);
    qDebug() << Q_FUNC_INFO << echantillon.horodatage.toString("dd/MM/yyyy
        hh:mm");
}

```

10.4.3.2 void IHMKart : :afficherGraphiques (Echantillon echantillon) [private]

Paramètres

<i>echantillon</i>	
--------------------	--

A faire réaliser l'affichage des graphiques pour la charge, tension et courant

Références [Echantillon](#) : [:chargeBatterie](#), [chargeBatterie](#), [courant](#), [Echantillon:::courantBatterie](#), [courbeChargeBatterie](#), [courbeCourant](#), [courbeTemperature](#), [courbeTension](#), [courbeVitesse](#), [temperature](#), [Echantillon : :temperatureMoteur](#), [tension](#), - [Echantillon : :tensionBatterie](#), [ui](#), [Echantillon : :vitesse](#), [vitesse](#), [Courbe : :x](#), et [Courbe : :y](#).

Référencé par [rafraichir\(\)](#).

```

{
    QTime maintenant = QTime::currentTime();
    int seconde = maintenant.second();

    if(seconde == 0)
    {
        // on efface les tracés précédents
        vitesse.x.clear();
        vitesse.y.clear();
        courbeVitesse->setSamples(vitesse.x, vitesse.y);
        ui->qwtGraphiqueVitesse->replot();

        tension.x.clear();
        tension.y.clear();
        courbeTension->setSamples(tension.x, tension.y);
        ui->qwtGraphiqueTension->replot();

        courant.x.clear();
        courant.y.clear();
        courbeCourant->setSamples(courant.x, courant.y);
        ui->qwtGraphiqueCourant->replot();

        temperature.x.clear();
        temperature.y.clear();
        courbeTemperature->setSamples(temperature.x, temperature.y);
        ui->qwtGraphiqueTemperature->replot();

        chargeBatterie.x.clear();
        chargeBatterie.y.clear();
        courbeChargeBatterie->setSamples(chargeBatterie.x, chargeBatterie.y);
        ui->qwtGraphiqueChargeBatterie->replot();
    }

    // on ajoute les mesures
    vitesse.x.push_back((double)seconde);
    vitesse.y.push_back((double)echantillon.vitesse);

    tension.x.push_back((double)seconde);
    tension.y.push_back((double)echantillon.tensionBatterie);

    courant.x.push_back((double)seconde);
    courant.y.push_back((double)echantillon.courantBatterie);
}

```

```

temperature.x.push_back((double)seconde);
temperature.y.push_back((double)echantillon.temperatureMoteur);

chargeBatterie.x.push_back((double)seconde);
chargeBatterie.y.push_back((double)echantillon.chargeBatterie);

// on trace les points x,y
courbeVitesse->setSamples(vitesse.x, vitesse.y);
ui->qwtGraphiqueVitesse->replot();

courbeTension->setSamples(tension.x, tension.y);
ui->qwtGraphiqueTension->replot();

courbeCourant->setSamples(courant.x, courant.y);
ui->qwtGraphiqueCourant->replot();

courbeTemperature->setSamples(temperature.x, temperature.y);
ui->qwtGraphiqueTemperature->replot();

courbeChargeBatterie->setSamples(chargeBatterie.x, chargeBatterie.y);
ui->qwtGraphiqueChargeBatterie->replot();

}

```

10.4.3.3 IHMKart : :arreterEssai() [slot]

A faire exporter les données télémétriques au format CSV

Références [communication](#), [idKart](#), [Communication : :setIdKart\(\)](#), et [ui](#).

Référencé par [IHMKart\(\)](#).

```

{
    // Démarre la collecte des données pour l'id kart
    communication->setIdKart(QString(""));
    ui->statusBar->showMessage(QString::fromUtf8("Arrêt de l'essai pour le kart
        %1").arg(idKart));

    ui->listeIDKarts->setEnabled(true);
    ui->listeIDOperateurs->setEnabled(true);
    ui->bDemarrageEssai->setEnabled(true);
    ui->bArretEssai->setEnabled(false);
}

```

10.4.3.4 void IHMKart : :chargerIDKarts() [private]

A faire lire les ID kart dans le fichier xml et charger la liste

Références [ui](#).

Référencé par [IHMKart\(\)](#).

```

{
    for(int i = 5001; i < 5004; i++)
    {

```

```

        ui->listeIDKarts->addItem(QString("%1").arg(i));
    }
}

```

10.4.3.5 void IHMKart : :chargerIDOperateurs () [private]

A faire lire les ID opérateur dans le fichier xml et charger la liste

Références [ui](#).

Référéncé par [IHMKart\(\)](#).

```

{
    for(int i = 101; i < 103; i++)
    {
        ui->listeIDOperateurs->addItem(QString("%1").arg(i));
    }
}

```

10.4.3.6 IHMKart : :demarrerEssai () [slot]

Références [communication](#), [idKart](#), [reinitialiserGraphiques\(\)](#), [Communication : :setIdKart\(\)](#), [Telemetrie : :supprimerEchantillons\(\)](#), [telemetrie](#), et [ui](#).

Référéncé par [IHMKart\(\)](#).

```

{
    ui->listeIDKarts->setEnabled(false);
    ui->listeIDOperateurs->setEnabled(false);
    ui->bDemarrageEssai->setEnabled(false);
    ui->bArretEssai->setEnabled(true);

    // Démarre la collecte des données pour l'id kart
    idKart = ui->listeIDKarts->currentText();
    telemetrie.supprimerEchantillons();
    communication->setIdKart(idKart);
    ui->statusBar->showMessage(QString::fromUtf8("Démarrage de l'essai pour le
        kart %1").arg(idKart));

    ui->lcdTension->display(0);
    ui->lcdCourant->display(0);
    ui->lcdCharge->display(0);
    ui->lcdVitesse->display(0);
    ui->lcdTemperature->display(0);
    reinitialiserGraphiques();
}

```

10.4.3.7 void IHMKart : :initialiserIHM () [private]

Références [ui](#).

Référéncé par [IHMKart\(\)](#).

```

{
    ui->setupUi(this);
    ui->bDemarrageEssai->setEnabled(true);
    ui->bArretEssai->setEnabled(false);

    // Affichage de la page d'accueil
    ui->tabWidget->setCurrentIndex(0);
}

```

10.4.3.8 void IHMKart : :initialiserQWT () [private]

A faire initialiser les autres graphiques (charge, tension et courant)

Références [courbeChargeBatterie](#), [courbeCourant](#), [courbeTemperature](#), [courbeTension](#), [courbeVitesse](#), et [ui](#).

Référencé par [IHMKart\(\)](#).

```
{
    QwtPlotGrid *grid;

    // Un titre
    ui->qwtGraphiqueVitesse->setTitle(QString::fromUtf8("Vitesse"));
    ui->qwtGraphiqueTension->setTitle(QString::fromUtf8("Tension"));
    ui->qwtGraphiqueCourant->setTitle(QString::fromUtf8("Courant"));
    ui->qwtGraphiqueTemperature->setTitle(QString::fromUtf8("Temperature"));
    ui->qwtGraphiqueChargeBatterie->setTitle(QString::fromUtf8("% Batterie"));

    // une légende à droite
    ui->qwtGraphiqueVitesse->insertLegend(new QwtLegend(), QwtPlot::RightLegend
    );
    ui->qwtGraphiqueTension->insertLegend(new QwtLegend(), QwtPlot::RightLegend
    );
    ui->qwtGraphiqueCourant->insertLegend(new QwtLegend(), QwtPlot::RightLegend
    );
    ui->qwtGraphiqueTemperature->insertLegend(new QwtLegend(),
    QwtPlot::RightLegend);
    ui->qwtGraphiqueChargeBatterie->insertLegend(new QwtLegend(),
    QwtPlot::RightLegend);

    // cadrillage
    grid = new QwtPlotGrid;
    grid->setMajPen(QPen(Qt::black, 0, Qt::DotLine));
    grid->attach(ui->qwtGraphiqueVitesse);

    grid = new QwtPlotGrid;
    grid->setMajPen(QPen(Qt::black, 0, Qt::DotLine));
    grid->attach(ui->qwtGraphiqueTension);

    grid = new QwtPlotGrid;
    grid->setMajPen(QPen(Qt::black, 0, Qt::DotLine));
    grid->attach(ui->qwtGraphiqueCourant);

    grid = new QwtPlotGrid;
    grid->setMajPen(QPen(Qt::black, 0, Qt::DotLine));
    grid->attach(ui->qwtGraphiqueTemperature);

    grid = new QwtPlotGrid;
    grid->setMajPen(QPen(Qt::black, 0, Qt::DotLine));
    grid->attach(ui->qwtGraphiqueChargeBatterie);

    // configuration des axes
    ui->qwtGraphiqueVitesse->setAxisTitle(ui->qwtGraphiqueVitesse->xBottom,
    QString::fromUtf8("en s"));
    ui->qwtGraphiqueVitesse->setAxisScale(QwtPlot::xBottom, 0.0, 59.0);
    ui->qwtGraphiqueVitesse->setAxisTitle(ui->qwtGraphiqueVitesse->yLeft,
    QString::fromUtf8("en km/h"));
    ui->qwtGraphiqueVitesse->setAxisScale(QwtPlot::yLeft, 0.0, 100.0);

    ui->qwtGraphiqueTension->setAxisTitle(ui->qwtGraphiqueTension->xBottom,
    QString::fromUtf8("en s"));
    ui->qwtGraphiqueTension->setAxisScale(QwtPlot::xBottom, 0.0, 59.0);
    ui->qwtGraphiqueTension->setAxisTitle(ui->qwtGraphiqueTension->yLeft,
    QString::fromUtf8("en volt"));
    ui->qwtGraphiqueTension->setAxisScale(QwtPlot::yLeft, 34.0, 60.0);

    ui->qwtGraphiqueCourant->setAxisTitle(ui->qwtGraphiqueCourant->xBottom,
```

```

        QString::fromUtf8("en s"));
    ui->qwtGraphiqueCourant->setAxisScale(QwtPlot::xBottom, 0.0, 59.0);
    ui->qwtGraphiqueCourant->setAxisTitle(ui->qwtGraphiqueCourant->yLeft,
        QString::fromUtf8("en ampere"));
    ui->qwtGraphiqueCourant->setAxisScale(QwtPlot::yLeft, 0.0, 250.0);

    ui->qwtGraphiqueTemperature->setAxisTitle(ui->qwtGraphiqueTemperature->
        xBottom, QString::fromUtf8("en s"));
    ui->qwtGraphiqueTemperature->setAxisScale(QwtPlot::xBottom, 0.0, 59.0);
    ui->qwtGraphiqueTemperature->setAxisTitle(ui->qwtGraphiqueTemperature->
        yLeft, QString::fromUtf8("en °C"));
    ui->qwtGraphiqueTemperature->setAxisScale(QwtPlot::yLeft, 40.0, 100.0);

    ui->qwtGraphiqueChargeBatterie->setAxisTitle(ui->qwtGraphiqueChargeBatterie
        ->xBottom, QString::fromUtf8("en s"));
    ui->qwtGraphiqueChargeBatterie->setAxisScale(QwtPlot::xBottom, 0.0, 59.0);
    ui->qwtGraphiqueChargeBatterie->setAxisTitle(ui->qwtGraphiqueChargeBatterie
        ->yLeft, QString::fromUtf8("en %"));
    ui->qwtGraphiqueChargeBatterie->setAxisScale(QwtPlot::yLeft, 0.0, 100.0);

    //Instancier les courbes
    courbeVitesse = new QwtPlotCurve("Vitesse");
    courbeTension = new QwtPlotCurve("Tension");
    courbeCourant = new QwtPlotCurve("Courant");
    courbeTemperature = new QwtPlotCurve("Temperature");
    courbeChargeBatterie = new QwtPlotCurve("% Batterie");

    //ajout à la légende
    courbeVitesse->setLegendAttribute(QwtPlotCurve::LegendShowLine, true);
    courbeTension->setLegendAttribute(QwtPlotCurve::LegendShowLine, true);
    courbeCourant->setLegendAttribute(QwtPlotCurve::LegendShowLine, true);
    courbeTemperature->setLegendAttribute(QwtPlotCurve::LegendShowLine, true);
    courbeChargeBatterie->setLegendAttribute(QwtPlotCurve::LegendShowLine, true
    );

    // tracé en bleu
    courbeVitesse->setPen(QPen(Qt::blue));
    courbeTension->setPen(QPen(Qt::blue));
    courbeCourant->setPen(QPen(Qt::blue));
    courbeTemperature->setPen(QPen(Qt::blue));
    courbeChargeBatterie->setPen(QPen(Qt::blue));

    // avec des points rouge
    courbeVitesse->setSymbol(new QwtSymbol(QwtSymbol::Ellipse, Qt::red, QPen(
        Qt::black), QSize(5,5)));
    courbeTension->setSymbol(new QwtSymbol(QwtSymbol::Ellipse, Qt::red, QPen(
        Qt::black), QSize(5,5)));
    courbeCourant->setSymbol(new QwtSymbol(QwtSymbol::Ellipse, Qt::red, QPen(
        Qt::black), QSize(5,5)));
    courbeTemperature->setSymbol(new QwtSymbol(QwtSymbol::Ellipse, Qt::red,
        QPen(Qt::black), QSize(5,5)));
    courbeChargeBatterie->setSymbol(new QwtSymbol(QwtSymbol::Ellipse, Qt::red,
        QPen(Qt::black), QSize(5,5)));

    // style de tracé
    courbeVitesse->setStyle(QwtPlotCurve::Lines); // ligne
    courbeVitesse->setCurveAttribute(QwtPlotCurve::Fitted);

    courbeTension->setStyle(QwtPlotCurve::Lines);
    courbeTension->setCurveAttribute(QwtPlotCurve::Fitted);

    courbeCourant->setStyle(QwtPlotCurve::Lines);
    courbeCourant->setCurveAttribute(QwtPlotCurve::Fitted);

    courbeTemperature->setStyle(QwtPlotCurve::Lines);
    courbeTemperature->setCurveAttribute(QwtPlotCurve::Fitted);

    courbeChargeBatterie->setStyle(QwtPlotCurve::Lines);

```

```

    courbeChargeBatterie->setCurveAttribute(QwtPlotCurve::Fitted);

    // la courbe
    courbeVitesse->setRenderHint(QwtPlotItem::RenderAntialiased);
    courbeTension->setRenderHint(QwtPlotItem::RenderAntialiased);
    courbeCourant->setRenderHint(QwtPlotItem::RenderAntialiased);
    courbeTemperature->setRenderHint(QwtPlotItem::RenderAntialiased);
    courbeChargeBatterie->setRenderHint(QwtPlotItem::RenderAntialiased);

    // la courbe sur le plot
    courbeVitesse->attach(ui->qwtGraphiqueVitesse);
    courbeTension->attach(ui->qwtGraphiqueTension);
    courbeCourant->attach(ui->qwtGraphiqueCourant);
    courbeTemperature->attach(ui->qwtGraphiqueTemperature);
    courbeChargeBatterie->attach(ui->qwtGraphiqueChargeBatterie);

}

```

10.4.3.9 IHMKart : :rafraichir (Echantillon *echantillon*) [slot]

Paramètres

<i>echantillon</i>	
--------------------	--

Références [afficherDonneesTelemetrie\(\)](#), [afficherGraphiques\(\)](#), [Telemetrie : :ajouterEchantillon\(\)](#), et [telemetrie](#).

Référencé par [IHMKart\(\)](#).

```

{
    afficherDonneesTelemetrie(echantillon);
    telemetrie.ajouterEchantillon(echantillon);

    afficherGraphiques(echantillon);
}

```

10.4.3.10 void IHMKart : :reinitialiserGraphiques () [private]

Références [chargeBatterie](#), [courant](#), [courbeChargeBatterie](#), [courbeCourant](#), [courbeTemperature](#), [courbeTension](#), [courbeVitesse](#), [temperature](#), [tension](#), [ui](#), [vitesse](#), [Courbe : :x](#), et [Courbe : :y](#).

Référencé par [demarrerEssai\(\)](#).

```

{
    // on efface les tracés précédents
    vitesse.x.clear();
    vitesse.y.clear();
    courbeVitesse->setSamples(vitesse.x, vitesse.y);
    ui->qwtGraphiqueVitesse->replot();

    tension.x.clear();
    tension.y.clear();
    courbeTension->setSamples(tension.x, tension.y);
    ui->qwtGraphiqueTension->replot();

    courant.x.clear();
    courant.y.clear();
    courbeCourant->setSamples(courant.x, courant.y);
    ui->qwtGraphiqueCourant->replot();

    temperature.x.clear();
}

```

```
temperature.y.clear();
courbeTemperature->setSamples(temperature.x, temperature.y);
ui->qwtGraphiqueTemperature->replot();

chargeBatterie.x.clear();
chargeBatterie.y.clear();
courbeChargeBatterie->setSamples(chargeBatterie.x, chargeBatterie.y);
ui->qwtGraphiqueChargeBatterie->replot();
}
```

10.4.4 Documentation des données membres

10.4.4.1 Courbe IHMKart : :chargeBatterie [private]

Référencé par [afficherGraphiques\(\)](#), et [reinitialiserGraphiques\(\)](#).

10.4.4.2 Communication* IHMKart : :communication [private]

Association vers un objet [Communication](#)

Référencé par [arreterEssai\(\)](#), [demarrerEssai\(\)](#), et [IHMKart\(\)](#).

10.4.4.3 Courbe IHMKart : :courant [private]

Référencé par [afficherGraphiques\(\)](#), et [reinitialiserGraphiques\(\)](#).

10.4.4.4 QwtPlotCurve* IHMKart : :courbeChargeBatterie [private]

Référencé par [afficherGraphiques\(\)](#), [initialiserQWT\(\)](#), et [reinitialiserGraphiques\(\)](#).

10.4.4.5 QwtPlotCurve* IHMKart : :courbeCourant [private]

Référencé par [afficherGraphiques\(\)](#), [initialiserQWT\(\)](#), et [reinitialiserGraphiques\(\)](#).

10.4.4.6 QwtPlotCurve* IHMKart : :courbeTemperature [private]

Référencé par [afficherGraphiques\(\)](#), [initialiserQWT\(\)](#), et [reinitialiserGraphiques\(\)](#).

10.4.4.7 QwtPlotCurve* IHMKart : :courbeTension [private]

Référencé par [afficherGraphiques\(\)](#), [initialiserQWT\(\)](#), et [reinitialiserGraphiques\(\)](#).

10.4.4.8 QwtPlotCurve* IHMKart : :courbeVitesse [private]

le graphique vitesse

Référencé par [afficherGraphiques\(\)](#), [initialiserQWT\(\)](#), et [reinitialiserGraphiques\(\)](#).

10.4.4.9 QString IHMKart : :idKart [private]

Référencé par [arreterEssai\(\)](#), et [demarrerEssai\(\)](#).

10.4.4.10 QString IHMKart : :idOperateur [private]

10.4.4.11 Telemetrie IHMKart : :telemetrie [private]

Composition d'un objet [Telemetrie](#)

Référencé par [demarrerEssai\(\)](#), et [rafraichir\(\)](#).

10.4.4.12 Courbe IHMKart : :temperature [private]

Référencé par [afficherGraphiques\(\)](#), et [reinitialiserGraphiques\(\)](#).

10.4.4.13 Courbe IHMKart : :tension [private]

Référencé par [afficherGraphiques\(\)](#), et [reinitialiserGraphiques\(\)](#).

10.4.4.14 Ui : :IHMKart* IHMKart : :ui [private]

Référencé par [afficherDonneesTelemetrie\(\)](#), [afficherGraphiques\(\)](#), [arreterEssai\(\)](#), [chargerIDKarts\(\)](#), [chargerIDOperateurs\(\)](#), [demarrerEssai\(\)](#), [IHMKart\(\)](#), [initialiserIHM\(\)](#), [initialiserQWT\(\)](#), [reinitialiserGraphiques\(\)](#), et [~IHMKart\(\)](#).

10.4.4.15 Courbe IHMKart : :vitesse [private]

les points de la courbe vitesse

Référencé par [afficherGraphiques\(\)](#), et [reinitialiserGraphiques\(\)](#).

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

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

10.5 Référence de la classe Telemetrie

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

Fonctions membres publiques

- [Telemetrie](#) ()
Constructeur par défaut.
- QVector< [Echantillon](#) > [getEchantillons](#) ()
- void [supprimerEchantillons](#) ()
Vide les échantillons stockés pour un essai.
- void [ajouterEchantillon](#) ([Echantillon](#) echantillon)

Attributs privés

- QVector< [Echantillon](#) > [echantillons](#)

10.5.1 Documentation des constructeurs et destructeur

10.5.1.1 Telemetrie : :Telemetrie ()

les données télémétriques pour un kart

```
{  
}
```

10.5.2 Documentation des fonctions membres

10.5.2.1 Telemetrie : :ajouterEchantillon (Echantillon *echantillon*)

Paramètres

<i>echantillon</i>	
--------------------	--

Références [echantillons](#).

Référéncé par [IHMkart : :rafraichir\(\)](#).

```
{  
    echantillons.push_back(echantillon);  
    qDebug() << Q_FUNC_INFO << echantillons.size();  
}
```

10.5.2.2 Telemetrie : :getEchantillons ()

Renvoie

QVector<Echantillon> les données télémétriques pour un kart

Références [echantillons](#).

```
{  
    return echantillons;  
}
```

10.5.2.3 Telemetrie : :supprimerEchantillons ()

Références [echantillons](#).

Référéncé par [IHMkart : :demarrerEssai\(\)](#).

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

10.5.3 Documentation des données membres

10.5.3.1 QVector<Echantillon> Telemetrie : :echantillons [private]

Référéncé par [ajouterEchantillon\(\)](#), [getEchantillons\(\)](#), et [supprimerEchantillons\(\)](#).

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

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

11 Documentation des fichiers

11.1 Référence du fichier Changelog.dox

11.2 Référence du fichier communication.cpp

Définition de la classe [Communication](#).

```
#include "communication.h" #include <unistd.h> #include <QDebug>
```

11.2.1 Description détaillée

Auteur

Penne Florent

Version

0.9

11.3 Référence du fichier communication.h

Déclaration de la classe [Communication](#).

```
#include <QObject> #include <QStringList> #include "qextserialport.-  
h" #include "telemetrie.h"
```

Classes

- class [Communication](#)
Permet la réception des trames envoyées par le kart.

Macros

- #define [PORT](#) "/dev/ttyUSB0"
- #define [CHAMP_IDKART](#) 0
- #define [CHAMP_TEMPERATURE](#) 1
- #define [CHAMP_TENSION](#) 2
- #define [CHAMP_COURANT](#) 3
- #define [CHAMP_VITESSE](#) 4
- #define [CHAMP_CHARGE](#) 5

11.3.1 Description détaillée

11.3.2 Documentation des macros

11.3.2.1 #define [CHAMP_CHARGE](#) 5

Référencé par [Communication](#) : `:extraireDonnees()`.

11.3.2.2 #define CHAMP_COURANT 3

Référencé par [Communication : :extraireDonnees\(\)](#).

11.3.2.3 #define CHAMP_IDKART 0

Le champ IDKART de la trame

Référencé par [Communication : :extraireDonnees\(\)](#).

11.3.2.4 #define CHAMP_TEMPERATURE 1

Référencé par [Communication : :extraireDonnees\(\)](#).

11.3.2.5 #define CHAMP_TENSION 2

Référencé par [Communication : :extraireDonnees\(\)](#).

11.3.2.6 #define CHAMP_VITESSE 4

Référencé par [Communication : :extraireDonnees\(\)](#).

11.3.2.7 #define PORT "/dev/ttyUSB0"

Fichier périphérique XBee

Référencé par [Communication : :creerPort\(\)](#).

11.4 Référence du fichier ihmkart.cpp

Définition de la classe [IHMkart](#).

```
#include "ihmkart.h"    #include "ui_ihmkart.h"    #include  
"communication.h" #include <qwt_legend.h> #include <qwt_  
_plot_grid.h> #include <qwt_plot_curve.h> #include <qwt_  
symbol.h> #include <QDebug>
```

11.4.1 Description détaillée

Auteur

Penne Florent

Version

0.9

11.5 Référence du fichier ihmkart.h

Déclaration de la classe [IHMkart](#).

```
#include <QMainWindow>          #include <qwt_plot_curve.h> x
#include "telemetrie.h"
```

Classes

- struct [Courbe](#)
La structure de données pour une courbe.
- class [IHMKart](#)
Fenêtre principale de l'application.

11.5.1 Description détaillée

11.6 Référence du fichier main.cpp

Programme principal.

```
#include <QtGui/QApplication> #include "ihmkart.h"
```

Fonctions

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

11.6.1 Description détaillée

11.6.2 Documentation des fonctions

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

Paramètres

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

Renvoie

int

```
{
    QApplication a(argc, argv);
    IHMKart w;
    w.show();

    return a.exec();
}
```

11.7 Référence du fichier README.dox

11.8 Référence du fichier telemetrie.cpp

Définition de la classe [Telemetrie](#).

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

11.8.1 Description détaillée

Auteur

Penne Florent

Version

0.9

11.9 Référence du fichier telemetrie.h

Déclaration de la classe [IHMKart](#).

```
#include <QVector> #include <QDateTime>
```

Classes

- struct [Echantillon](#)
La structure de données pour un échantillon.
- class [Telemetrie](#)

11.9.1 Description détaillée

Index

- ~Communication
 - Communication, 6
- ~IHMkart
 - IHMkart, 14
- CHAMP_CHARGE
 - communication.h, 24
- CHAMP_COURANT
 - communication.h, 24
- CHAMP_IDKART
 - communication.h, 25
- CHAMP_TEMPERATURE
 - communication.h, 25
- CHAMP_TENSION
 - communication.h, 25
- CHAMP_VITESSE
 - communication.h, 25
- Changelog.dox, 24
- Communication, 4
 - ~Communication, 6
 - Communication, 5
 - Communication, 5
 - configurerPort, 6
 - creerPort, 6
 - extraireDonnees, 6
 - fermerPort, 7
 - getIdKart, 7
 - idKart, 9
 - nouvelEchantillon, 7
 - ouvrirPort, 7
 - port, 9
 - recevoir, 7
 - setIdKart, 8
 - trame, 9
 - verifierTrame, 8
- Courbe, 9
 - x, 9
 - y, 9
- Echantillon, 10
 - chargeBatterie, 10
 - courantBatterie, 10
 - horodatage, 10
 - idKart, 10
 - temperatureMoteur, 10
 - tensionBatterie, 10
 - vitesse, 10
- IHMkart, 11
 - ~IHMkart, 14
 - IHMkart, 14
 - afficherDonneesTelemetrie, 14
 - afficherGraphiques, 15
 - arreterEssai, 16
 - chargeBatterie, 21
 - chargerIDKarts, 16
 - chargerIDOperateurs, 17
 - communication, 21
 - courant, 21
 - courbeChargeBatterie, 21
 - courbeCourant, 21
 - courbeTemperature, 21
 - courbeTension, 21
 - courbeVitesse, 21
 - demarrerEssai, 17
 - idKart, 21
 - idOperateur, 21
 - IHMkart, 14
 - initialiserIHM, 17
 - initialiserQWT, 17
 - rafraichir, 20
 - reinitialiserGraphiques, 20
 - telemetrie, 21
 - temperature, 22
 - tension, 22
 - ui, 22
 - vitesse, 22
- PORT
 - communication.h, 25
- README.dox, 26
- Telemetrie, 22
 - Telemetrie, 22
 - ajouterEchantillon, 23
 - echantillons, 23
 - getEchantillons, 23
 - supprimerEchantillons, 23
 - Telemetrie, 22
- afficherDonneesTelemetrie
 - IHMkart, 14
- afficherGraphiques
 - IHMkart, 15
- ajouterEchantillon
 - Telemetrie, 23
- arreterEssai
 - IHMkart, 16

- chargeBatterie
 - Echantillon, [10](#)
 - IHMkart, [21](#)
- chargerIDKarts
 - IHMkart, [16](#)
- chargerIDOperateurs
 - IHMkart, [17](#)
- communication
 - IHMkart, [21](#)
- communication.cpp, [24](#)
- communication.h, [24](#)
 - CHAMP_CHARGE, [24](#)
 - CHAMP_COURANT, [24](#)
 - CHAMP_IDKART, [25](#)
 - CHAMP_TEMPERATURE, [25](#)
 - CHAMP_TENSION, [25](#)
 - CHAMP_VITESSE, [25](#)
 - PORT, [25](#)
- configurerPort
 - Communication, [6](#)
- courant
 - IHMkart, [21](#)
- courantBatterie
 - Echantillon, [10](#)
- courbeChargeBatterie
 - IHMkart, [21](#)
- courbeCourant
 - IHMkart, [21](#)
- courbeTemperature
 - IHMkart, [21](#)
- courbeTension
 - IHMkart, [21](#)
- courbeVitesse
 - IHMkart, [21](#)
- creerPort
 - Communication, [6](#)
- demarrerEssai
 - IHMkart, [17](#)
- echantillons
 - Telemetrie, [23](#)
- extraireDonnees
 - Communication, [6](#)
- fermerPort
 - Communication, [7](#)
- getEchantillons
 - Telemetrie, [23](#)
- getIdKart
 - Communication, [7](#)
- horodatage
 - Echantillon, [10](#)
- idKart
 - Communication, [9](#)
 - Echantillon, [10](#)
 - IHMkart, [21](#)
- idOperateur
 - IHMkart, [21](#)
- ihmkart.cpp, [25](#)
- ihmkart.h, [25](#)
- initialiserIHM
 - IHMkart, [17](#)
- initialiserQWT
 - IHMkart, [17](#)
- main
 - main.cpp, [26](#)
- main.cpp, [26](#)
 - main, [26](#)
- nouvelEchantillon
 - Communication, [7](#)
- ouvrirPort
 - Communication, [7](#)
- port
 - Communication, [9](#)
- rafraichir
 - IHMkart, [20](#)
- recevoir
 - Communication, [7](#)
- reinitialiserGraphiques
 - IHMkart, [20](#)
- setIdKart
 - Communication, [8](#)
- supprimerEchantillons
 - Telemetrie, [23](#)
- telemetrie
 - IHMkart, [21](#)
- telemetrie.cpp, [26](#)
- telemetrie.h, [27](#)
- temperature
 - IHMkart, [22](#)

- temperatureMoteur
 - Echantillon, [10](#)
- tension
 - IHMkart, [22](#)
- tensionBatterie
 - Echantillon, [10](#)
- trame
 - Communication, [9](#)
- ui
 - IHMkart, [22](#)
- verifierTrame
 - Communication, [8](#)
- vitesse
 - Echantillon, [10](#)
 - IHMkart, [22](#)
- x
 - Courbe, [9](#)
- y
 - Courbe, [9](#)