Projet Kart électrique

Revue Finale

Etudiant : PENNE Florent

La Salle Avignon
Frères des écoles chrétiennes

Table des matières

Revue Finale 1
Etudiant : PENNE Florent 1

Table des matières 2

Présentation du projet 3

Études et analyse du besoin 5
Objectifs 5
Cas d'utilisation 5
Synoptique du projet 6
Ressources logicielles 7
Ressources matérielles 7
Contraintes économiques et budgétaires 7
Répartition des tâches 8
Planification 9
Exigences 10
Déploiement 11

Communication 12
Protocole de transmission 13
Format 14
Configuration Xbee (Commande AT) 15
Installation 15
Configuration 16

Programmation 18
Classe IHMKart 19
Classe Communication 20
Code source 21

Interface Homme-Machine 23

Xml 25

CSV 26

Recette 27

Glossaire 28

Présentation du projet

Il s'agit de réaliser la conception complète de la chaîne de télémétrie des essais en d'un kart électrique sur circuit.

L'opérateur doit pouvoir **démarrer un essai** (une campagne d'acquisition de données télémétriques) d'un kart en précisant son **code opérateur** ainsi que l'**identifiant du kart**.

L'opérateur doit être en mesure de suivre, **en temps réel**, l'affichage en des **données télémétriques collectées** lors de l'essai. Lorsqu'il mettra fin à l'acquisition, il pourra alors exploiter les données sous forme de **graphiques** et les conserver dans le **format CSV(**Comma-separated values).
Ce format permettra une sauvegarde des données enregistrées qui par la suite pourront être importées dans un tableur.
L'essai doit être **horodaté** et une détection du kart à superviser doit être réalisée avant de démarrer la collecte des données télémétriques.

Études et analyse du besoin

Objectifs

Mon objectif est de concevoir l'**application logicielle** qui permettra, par le biai d'une **communication radio** entre le kart et le poste de l'opérateur, la réception des données de télémétrie enregistrées par les différents capteurs et ainsi la **supervision** d'un essais.

Cas d'utilisation

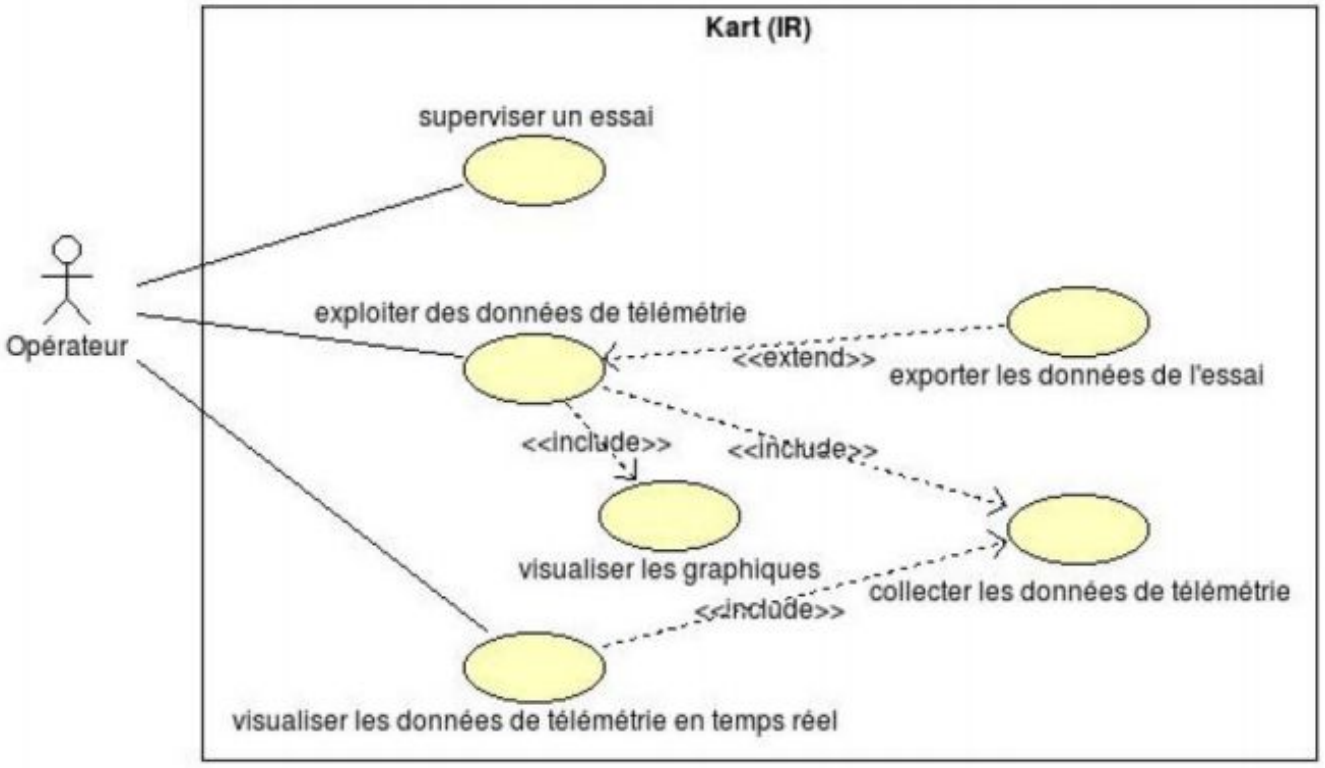


Diagramme de cas d'utilisation

Le projet est composé de 3 sous parties :

- **L'acquisition** des données de télémétrie
- La **transmission** des données
- **L'affichage** et la **sauvegarde** des données

Synoptique du projet

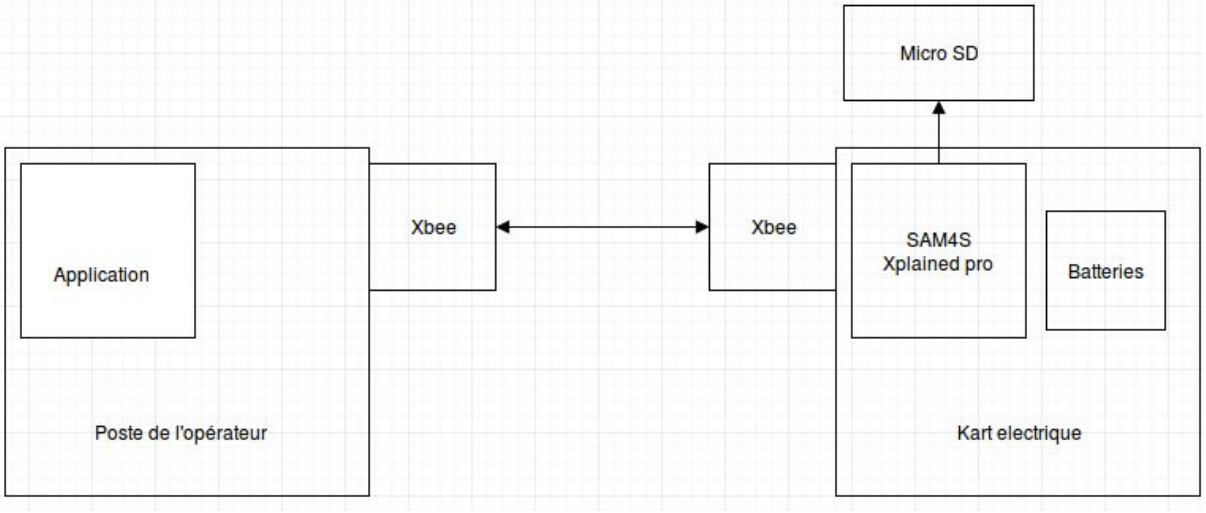


Schéma synoptique

Le kart est indépendant du système,il est en mesure de sauvegarder sur une mémoire locale (Carte micro SD) les données de télémétrie.
Il communique néanmoins avec le PC de l'opérateur grâce à une connection sans fils Xbee sur lequel l'application de supervision du kart sera exécutée, tandis que le système embarqué SAM4S enverra des données enregistrée par les différents capteurs.

Ressources logicielles

- QT creator 4.8.1
- Bouml 4.23
- Subversion 1.6.17
- Doxygen 1.7.6.1
- Gantt
- GNU Linux (Ubuntu 12.04)

Ressources matérielles

- Kart electrique
- Pack de 4 cellules de 12 V
- Chargeur de batterie
- Carte SAM4S Xplained pro
- Module Xbee S2C

Contraintes économiques et budgétaires

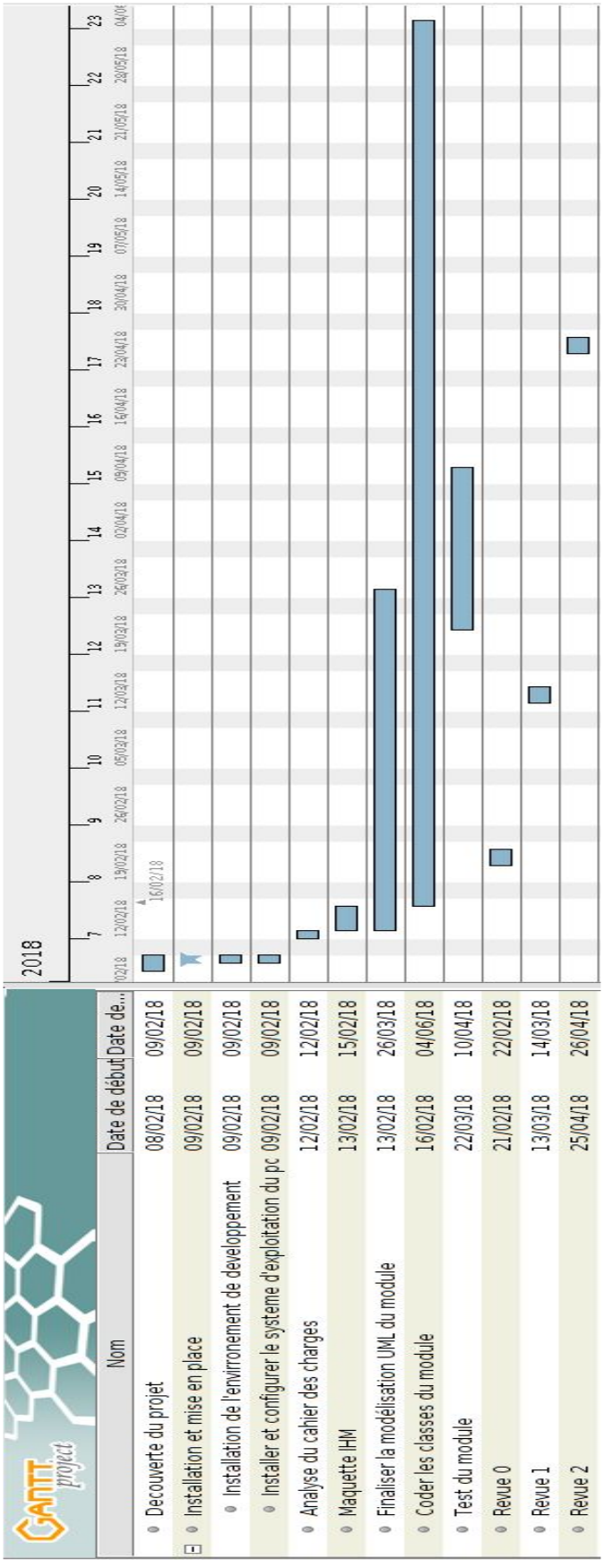
Le système sera **maquetté et conservé par l'établissement.**

Le prix des batteries du kart n'a pu être pris en charge car **trop coûteux** (>1000€), de plus, dûs à l'environnement du Campus Lasalle 84, aucune piste ou circuit ne nous permettrai de faire rouler le kart dans des conditions adéquates.

Répartition des tâches

- Etudiant 1 (EC) :
    - Mesurer le courant batterie
    - Mesurer la tension batterie
    - Mettre en forme les mesures
    - Mesurer la température moteur
    - Stocker les données sur support amovible
- Etudiant 2 (EC) :
    - Mesurer la vitesse de déplacement du kart
    - Mesure la distance parcourue
    - Transmettre ces mesures via une liaison sans fil
- **Etudiant 3 (IR) :**
    - **Exploiter des données de télémétrie**
    - **Collecter des données de l'essai**
    - **Visualiser les données de télémétrie en temps réel**
    - **Visualiser les graphiques**
    - **Exporter les données de l'essai en CSV**

Planification



Exigences

Les **données de télémétrie** sont :

- la tension (en Volts) de la batterie
- le courant (en Ampères) de la batterie
- le pourcentage de charge de la batterie
- la température moteur (en degrés Celsius)
- la vitesse (en km/h)

L'application logicielle devra assurer :

- la supervision de l'essai d'un kart
- d'acquérir les données de télémétrie par transmission sans fil
- d'afficher en « temps réel » les données reçues périodiquement
- d'exploiter les données de télémétrie sous forme de graphes
- d'exporter les données de télémétrie dans le format CSV

Les exigences qualité pour ma partie sont :

- Le développement se fera de manière itérative et incrémentale.
- La modélisation du système devra être réalisée avec le langage de modélisation UML 2.
- L'architecture du logiciel sera «orientée objet» et multitâche si besoin.
- Le langage de programmation à utiliser est le C++.
- L'IHM doit être développée avec l'API Qt de Qt Development Frameworks.
- L'implémentation des structures de données doit privilégier les structures de données de la STL ou équivalent dans Qt.
- Le codage doit respecter le standard de codage C++ en cours dans la section.
- La chaîne de production des exécutables doit être réalisée avec un gestionnaire de type make/qmake et le compilateur GNU g++.
- Le gestionnaire de gestion de versions utilisé sera subversion.
- La documentation du code sera générée à partir de doxygen.
- La suite bureautique libre LibreOffice ou OpenOffice sera utilisée pour tous les documents
- "papier" et les diaporamas ;
- Les données de l'application seront exclusivement gérées par des fichiers XML.

Déploiement

l'Application **IHMkart** s'exécutera sur un ordinateur de type PC disposant d'un système d'exploitation GNU/linux (Ubuntu 12.04). Elle utilisera la bibliothèque **qwt** qui permet de créer et paramétrer des graphiques pour L'**IHM** (Interface Homme-Machine).

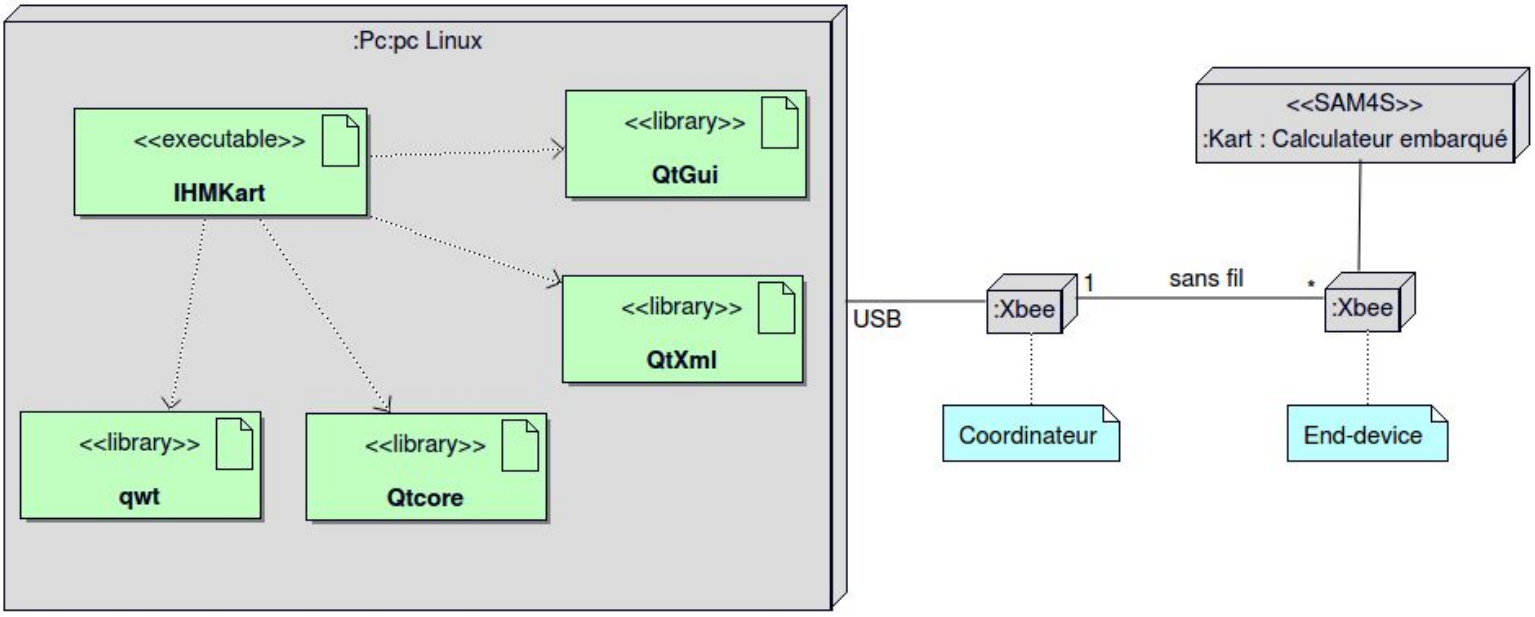


Diagramme de déploiement

Communication

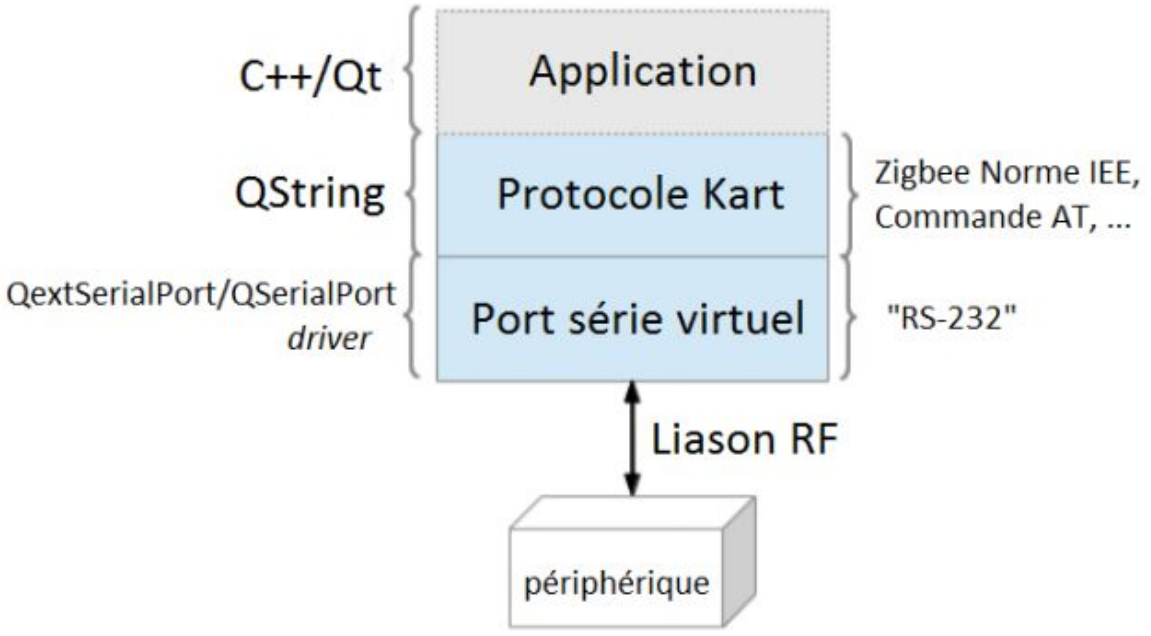
Les modules Xbee permettent des **communication à consommation réduite** pour les réseaux à dimension personnel comme des WPAN (*Wireless Personal Area Networks*).

Ces modules suivent un protocole Zigbee basé sur la norme IEEE 802.15.4 qui est destinée aux communication de faible puissance.

Le module Xbee placé sur le PC (GNU/linux) de l'opérateur est exploité comme un port série **virtuel** à cause de la raréfaction des ports séries physique sur les ordinateurs modernes.

Protocole de transmission

Le protocole du kart est sous forme de chaîne de caractères est conçu pour permettre l'échange de données entre le système embarqué et le PC de l'opérateur.



Le protocole est **indépendant** par rapport au protocole de transmission, ici Xbee.
Par exemple, le XBee utilise le protocole IEEE 802.15.4 pour la couche 2 du modèle OSI.
Le protocole est indépendant de la transmission utilisée (ici sans fil), nous pourrions l'utiliser avec une liaison série RS232 par exemple.

Format

J'ai choisi un format **fixe** dont les champs sont séparés par des ",".
La trame se presentera sous la forme suivante :

$ID_KART,TENSION,COURANT,CHARGE,TEMPERATURE_MOTEUR,VITESSE\r

- "$" correspond au délimiteur de début de trame.
- "ID_KART" correspond à l'identifiant du kart.
- "TEMPERATURE_MOTEUR correspond à la température moteur du kart (°C).
- "TENSION" correspond au voltage des batteries (Volt).
- "COURANT" correspond à l'intensité des batteries (Ampère).
- "VITESSE" correspond à la vitesse de déplacement du kart (Km/h).
- "CHARGE" correspond au pourcentage de batterie restant.
- "\r" délimite la fin de la trame.

Un champ de données peut être vide car les trames ne sont pas acquittées (ack) par le recepteur.Si le recepteur détecte une trame invalide, elle ne sera pas ni renvoyée ni prise en compte.

Configuration Xbee (Commande AT)

La configuration xbee dépend du système d'exploitation et du logiciels utilisé, pour ce projet sous la distribution Ubuntu, j'ai configuré les modules grâce à **cutecom** (Un émulateur de terminal série).

Cutecom fournit une interface graphique permettant la communication avec le matériel physique (xbee) par une liaison série.

Installation

Avant de configurer les modules xbee, il est nécessaire d'installer la bibliothèque lixbee que l'on aura téléchargé préalablement.
On installe la bibliothèque grâce aux commandes suivantes :

```
$ unzip libxbee-v3.libxbee-76d63381f85f.zip
$ cd libxbee-v3.libxbee-76d63381f85f/
$ make configure
$ make all
$ sudo make install
```

Pour éviter tout problèmes lors de l'installation on effectue une sauvegarde préalable de la configuration actuelle :

```
$ lsusb > lsusb-avant.txt
$ lsmod > lsmod-avant.txt
$ dmesg > dmesg-avant.txt
```

Ensuite on connecte le module xbee à configurer et on sauvegarde les modifications systemes :

```
$ lsusb > lsusb-apres.txt
$ lsmod > lsmod-apres.txt
$ dmesg > dmesg-apres.txt
```

Il faut ensuite donner les droits (lecture/écriture) à l'utilisateurs de communiquer sur USB0 :

```
$ sudo  chmod  666  /dev/ttyUSB0
```

Configuration

Par défaut, le module Xbee aura pour paramètres :

- 9600 bauds
- 8 bits
- 1 bit de stop
- Pas de parité

Il faudra avant d'exécuter les commandes suivante, activer le mode programmation grâce à la commande “+++” sans délimiteurs de fin (*No line end)*
Le mode commande se désactive au bout de 10 secondes sans entrée (*Timeout).*

Commande	Valeur	Explication
ATID	2525	Adresse du réseau
ATCH	C	Permet de changer le channel
ATMY	Simulateur :5000 PC (ihm) : 5001	Adresse du module dans le réseau
ATAP	0	Permet de changer le mode de fonctionnement
ATDH	0	pour utiliser des adresse sur 16 bits
ATDL	Simulateur : 5001 PC (ihm): 5000 ou Broadcast: FFFF	Adresse du destinataire dans le réseau
ATWR	-	Écrit la nouvelle configuration dans la mémoire flash du module
ATCN	-	Sort du mode configuration

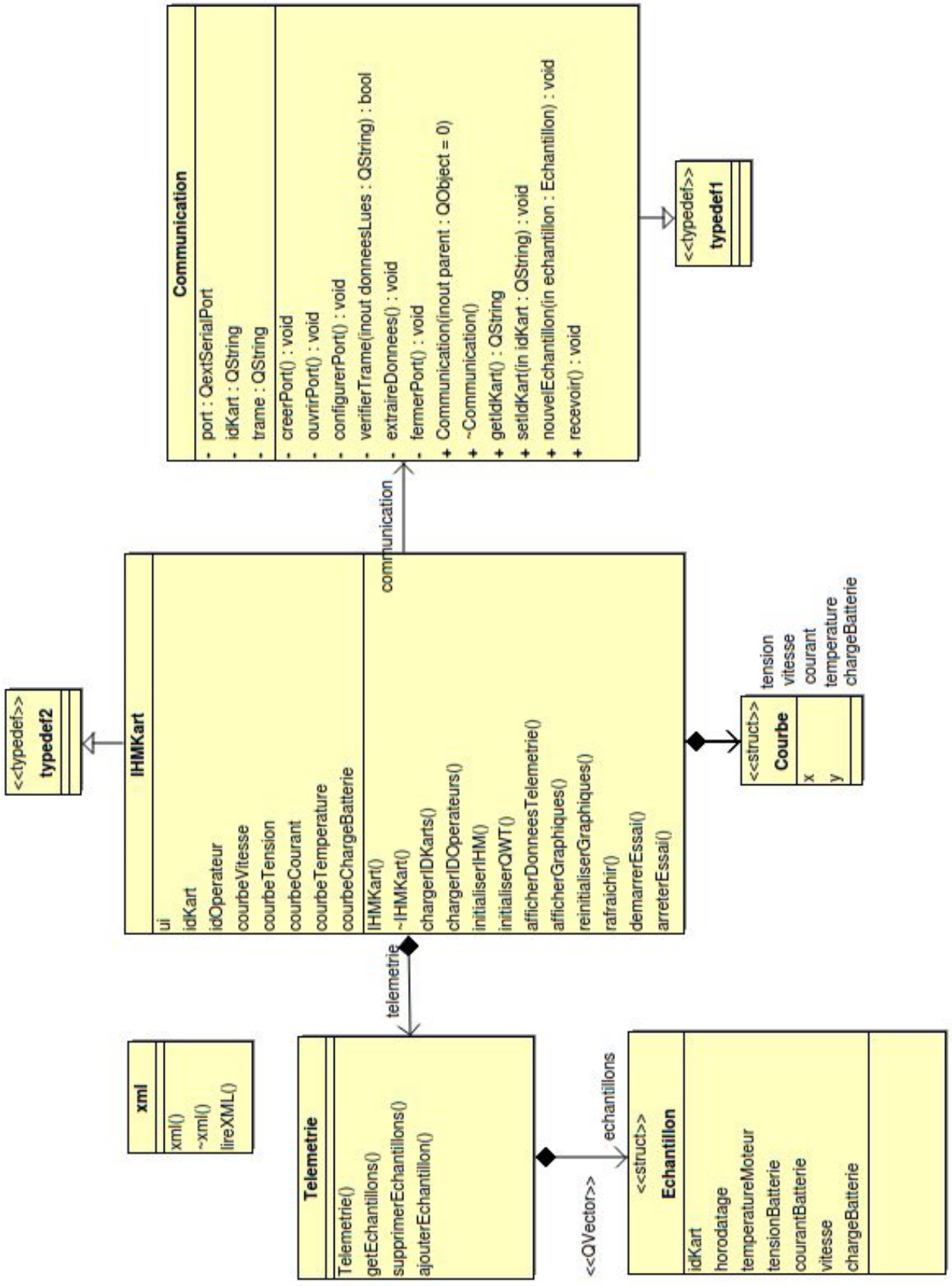
Pour que les 2 modules Xbee puisse communiquer, ils doivent disposer du même **PAN ID** (Commande “ATID”) et doivent être sur le même canal (Commande “ATCH”). Pour modifier une valeur, il suffit de préciser la commande AT avec “=” suivi de la valeur modifiée (Exemple : ATID=2526).

Pour notre projet, nous utiliserons le mode “**transparent**” du module Xbee,ce mode sélectionné par défaut (ATAP=0) agit comme une liaison série classique tout en conservant l’avantage de la technologie sans fils.

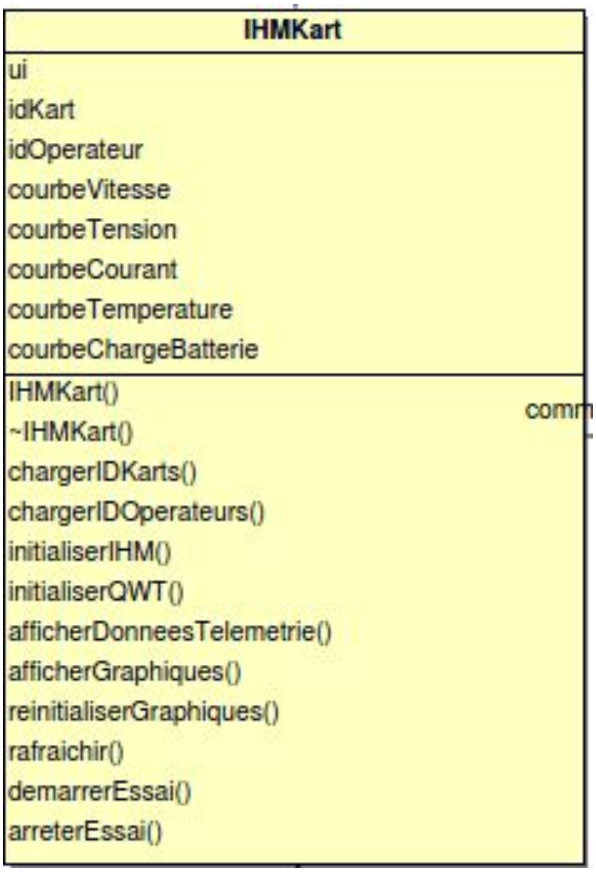
Les 2 modules Xbee sont ici en **simplex** (1 émetteur , 1 récepteur), dans un développement de plus grande envergure de ce projet il faudra envisager une communication en “étoile” où plusieurs kart communiqueront avec un opérateur.

Ici, le kart prend le rôle de “end-device” et le PC de l'opérateur quant à lui prend le rôle de coordonnateur capable dans un réseau en “étoile” de gérer plusieurs “end-device” et donc plusieurs kart.

Programmation

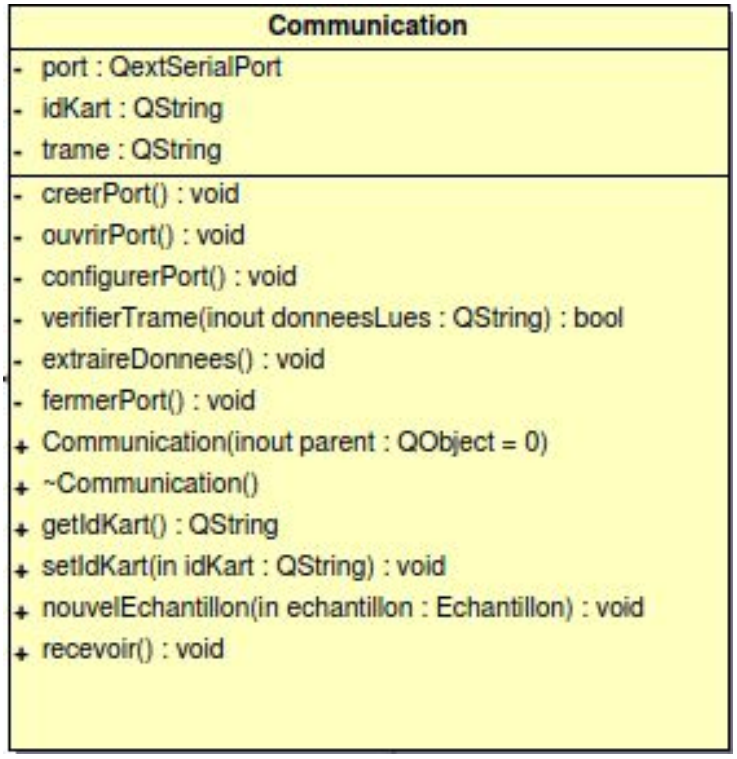


Classe IHMKart



La classe IHMKart est la classe principale de l'application, elle permet de démarrer/arrêter un essai, d'afficher les données de télémétrie sous formes de graphiques et de valeurs(en utilisant des **widgets)**.
Elle récupère aussi les identifiants kart et Opérateurs.

Classe Communication



La classe Communication est chargée de créer et de configurer le port ainsi que de lire les données reçu en mode “Event Driver” et vérifie l'intégrité de la trame en contrôlant le nombre de champs des trames reçu.

Code source

Scénario de réception d'une trame

```
void Communication::ouvrirPort()
{
    //Ouverture port droit ecriture lecture
    port->open(QIODevice::ReadWrite | QIODevice::Unbuffered);
    qDebug() << port->isOpen() << PORT;
}

/**
 * @brief Configure le port
 *
 * @fn Communication::configurerPort
 */
void Communication::configurerPort()
{
    if(port->isOpen())
    {
        port->setBaudRate(BAUD9600);
        port->setDataBits(DATA_8);
        port->setStopBits(STOP_1);

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

/**
 * @brief Verifie la conformité de la trame
 *
 * @fn Communication::verifierTrame
 * @param donneesLues
 * @return bool
 */
bool Communication::verifierTrame(QString &donneesLues)
{
    if(donneesLues.startsWith("$"))
    {
        if(donneesLues.endsWith("\r"))
        {
            return true;
        }
    }
    return false;
}
```

Méthode configurerPort()

ConfigurerPort() est une méthode indispensable du projet,elle détermine le débit (9600 bauds), le nombre de bits de données (8 bits) , et le bit de stop, si le port est ouvert .

Méthode verifierTrame()

La méthode verifierTrame() est chargée de verifier la presence d'un délimiteur de début et d'un délimiteur de fin de trame

```
/**
 * @brief Extrait les données de télémetrie et decoupe la trame
 *
 * @fn Communication::extraireDonnees
 */
void Communication::extraireDonnees()
{
    // 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);
    }
}
```

Méthode extraireDonees()

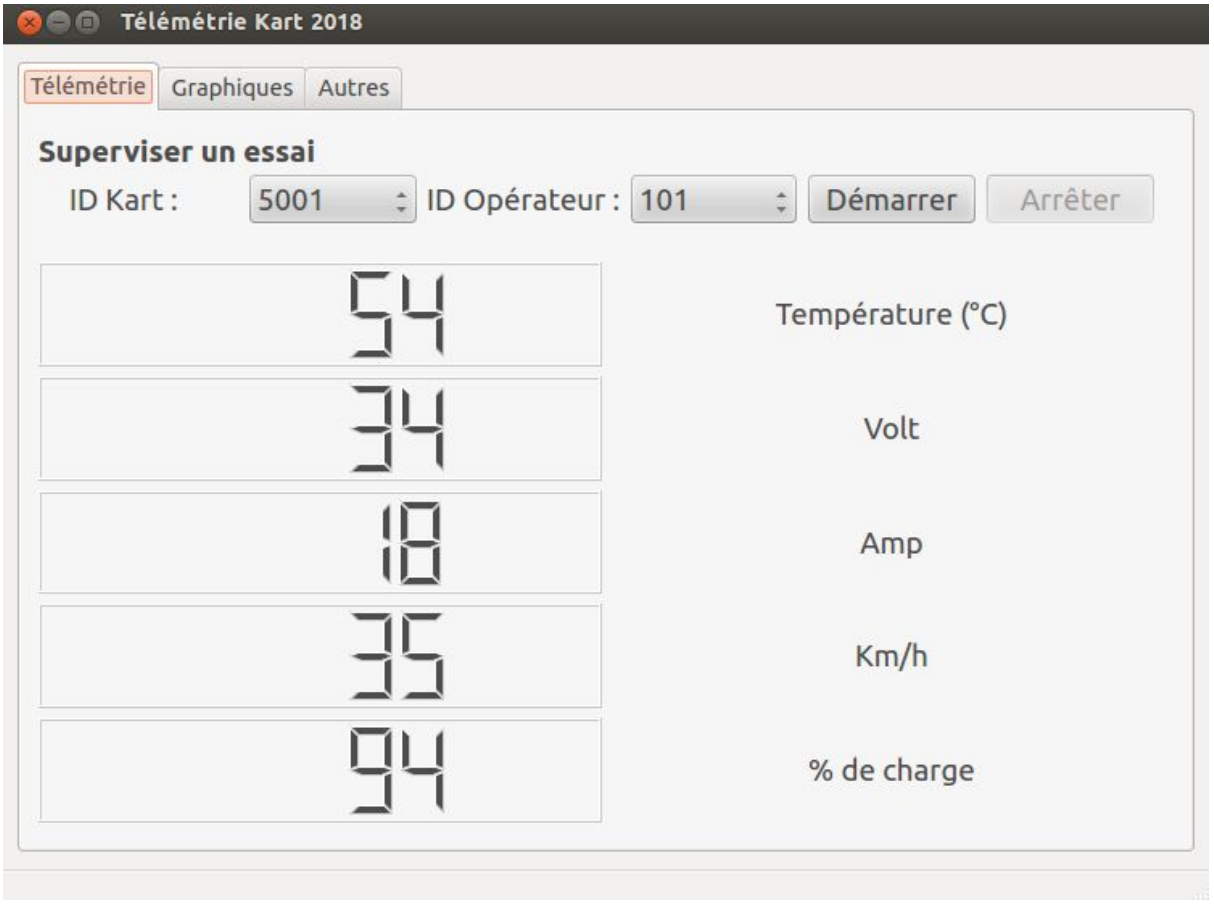
Cet méthode retire le délimiteur de début et le délimiteur de fin et découpe la trame en champs avant de les envoyer vers un conteneur.

```
/**
 * @brief Affichage des données reçues
 *
 * @fn IHMKart::afficherDonneesTelemetrie
 * @param echantillon
 */
void IHMKart::afficherDonneesTelemetrie(Echantillon echantillon)
{
    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");
}
```

Méthode afficherDonneesTelemetrie(Echantillon echantillon)

Cette méthode affiche sur l'interface les données collectées sous formes de valeurs dans un format LCD

Interface Homme-Machine



Mon interface graphique se compose de 3 onglets :

- Télémétrie

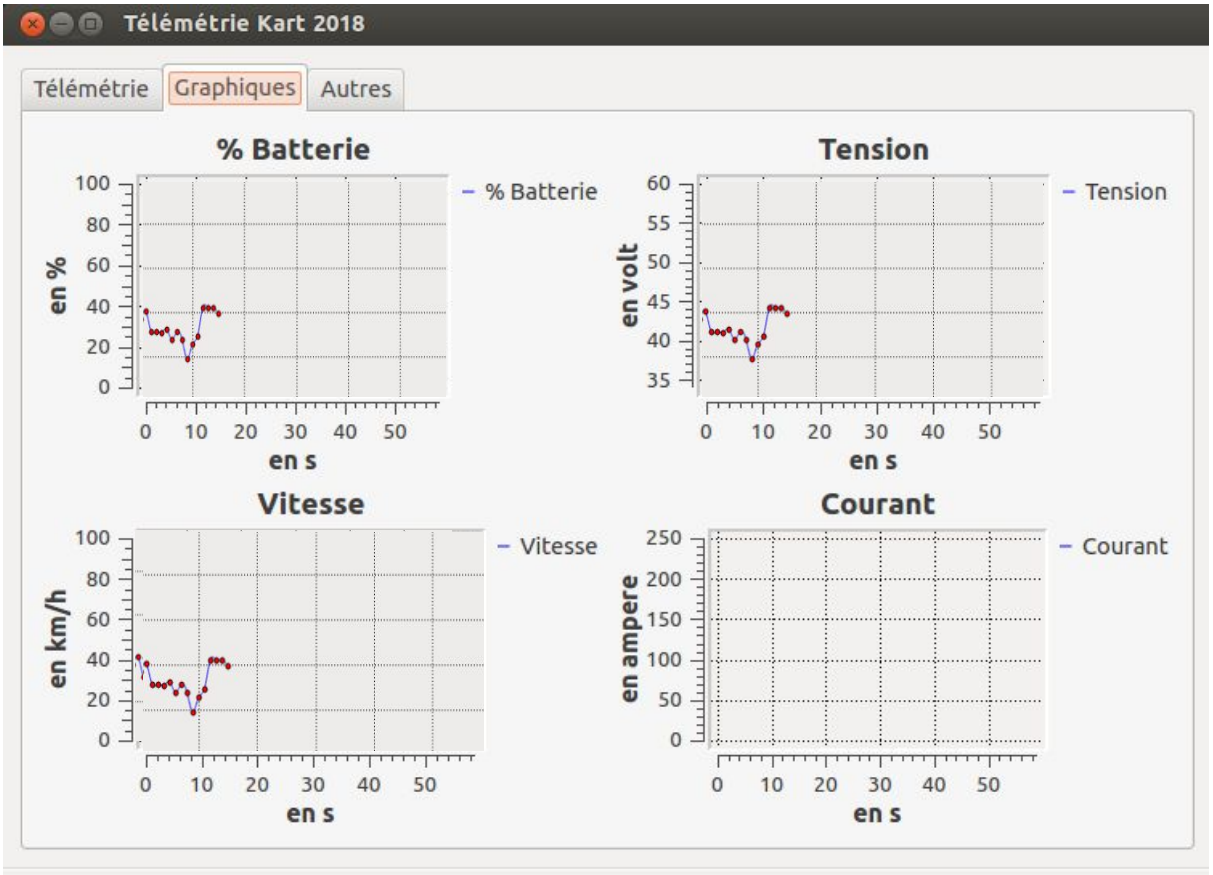
Dans cet onglets, l'opérateur a accès au mesure relevé par les capteurs en chiffre et en temps réels.C'est aussi dans cet onglets que l'on peut démarrer ou arrêter un essai.

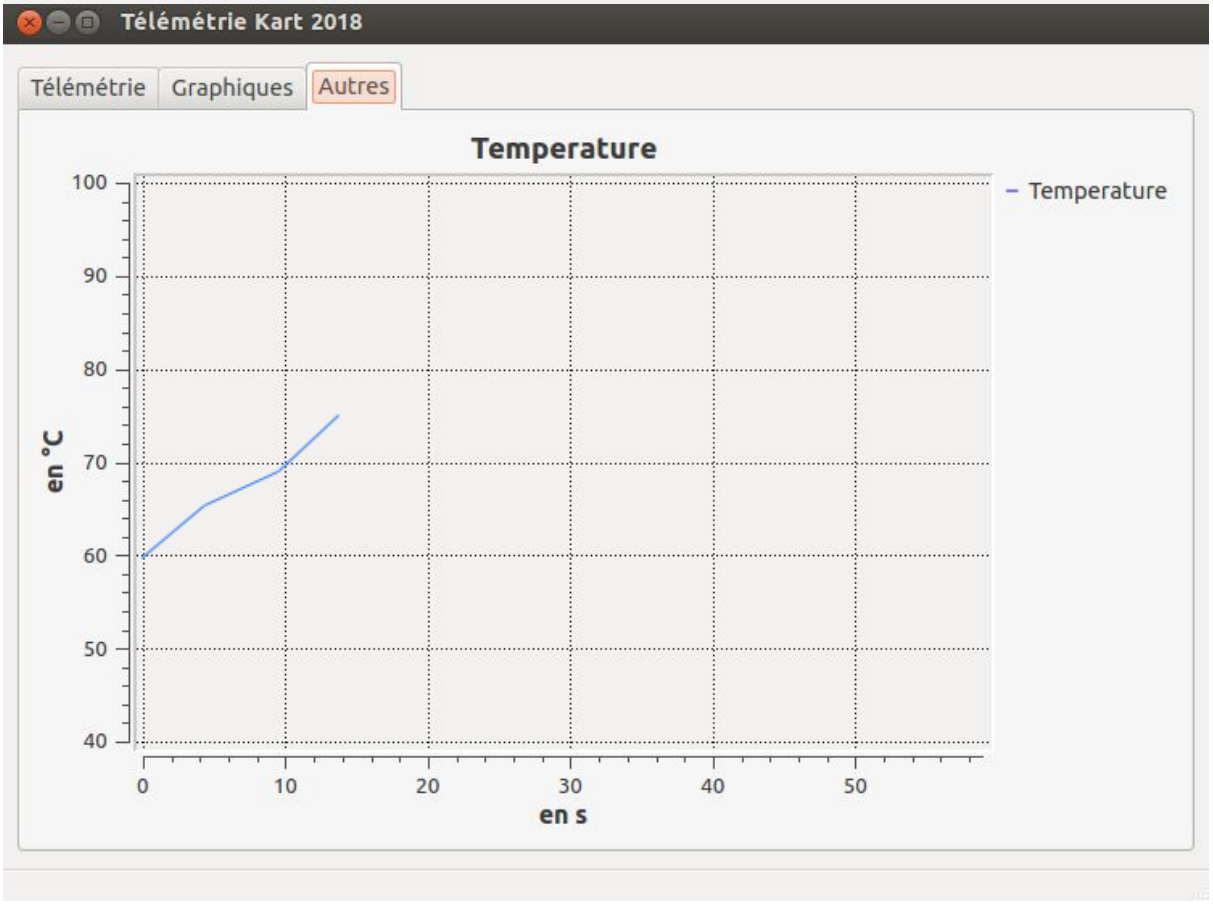
- Graphiques

On retrouve dans cet onglets les 4 graphiques les plus importants sur le même écran (% de batterie restant , tension de la batterie , intensité de la batterie , et vitesse du kart)

- Autres

On retrouve dans cet onglet le graphique de température moteur.





Xml

Les deux fichiers «kart.xml» et «operateur.xml» sont enregistrés dans un format standard qui permet à l'application d'échanger et de distribuer des données.

XML est un langage de balisage extensible, un document xml obéit aux règles syntaxiques et peut répondre à une DTD (Document Type Definition) ce qui n'est pas le cas ici.
De plus ce langage permet une universalité et une portabilité car en caractère ASCII et dispose d'un encodage UNICODE.
Dans ce projet il est utilisé comme un répertoire pour lister les karts et les opérateurs, qui utilisent ce dispositif.

Les documents Xml sont structurés en cinq parties:

- Le prologue : indique la version XML, le jeu de caractères et la présence ou pas d'une DTD (non présent dans ce projet).

- les instructions de traitement : instructions relatives à des applications qui traiteront le document comme une feuille de style (non présent dans ce projet).

- Commentaires : qui permet de commenter le document (exemple: «liste des numéro de karts).

- l'arbre des éléments : le contenu du document. Il considéré comme l'élément racine qui peut contenir d'autres éléments.

- Les éléments : qui sont aussi appelés des nœuds ou node sont définis par des balises :
    - <element> : balise d'ouverture◦
    - </element> : balise de fermeture
    - <element/> : balise vide

CSV

L'opérateur peut s'il le souhaite, exporter les données de l'essai dans un format CSV. Pour pouvoir sauvegarder toutes les informations d'un kart pendant un essai. Les données seront enregistrées sous forme de tableur.

Exemple de fichier CSV, à travers une application qui lit ces fichiers :

Tension (V)	Courant (A)	Charge (%)	Température (°C)	Vitesse (km/h)
12,8	63	97	20	8,7
14	79	97	22	12,2
16,2	82	96,7	23	15,4
18,5	87	96,7	23	18,6
19,6	92	96,5	25	21,9
24	101	96,5	26	27,3
26,5	113	96,3	28	33,4
29	123	96,3	31	38
32	127	96	37	44

Recette

Description	OUI	NON
La supervision d'un essaipermet de démarrer latélémétrie	✓	
La récupération des donnéestélémétriques est opérationnelle	✓	
L'affichage des donnéestélémétriques en temps réel estfonctionnel	✓	
L'affichage des donnéestélémétriques sous forme degraphiques est fonctionnel	✓	
L'exportation des donnéestélémétriques au format CSVest possible		✗

Glossaire

Données télémétriques : des informations qui sont envoyées à distance par des procédés optiques, acoustiques où radio.

CSV : (Comma Separated Values) est un fichier tableur,contenant des données sur chaque ligne séparée par un caractère de séparation («,» , «;» où une tabulation).

API : (Application Programing Interface) une interface de programmation applicative est un ensemble normalisé de classes, de méthodes ou de fonctions par lesquelles un logiciel (ici Qt creator) offre des services et une bibliothèque.

Subversion : (en abrégé svn) est un logiciel de gestion de versions. Le principe est d'avoir un dépôt centralisé et unique. Subversion fonctionne sur le mode client-serveur.

Doxygen : est un générateur de documentation sous licence libre capable de produire une documentation logicielle à partir du code source d'un programme. Pour cela il utilise une syntaxe de différents commentaires.

QWidget : QWidget est la classe de base de tous les objets d'interfaces utilisateurs. Un widget est l'atome de l'interface utilisateur : il peut recevoir un événement (souris, clavier où système...) et dispose d'un représentation graphique à l'écran.

QwtThermo : est un widget qui permet de faire des thermomètres.

QString : est un type de Qt , il représente le type string en C++

Dongle : est un appareil qui peut être branché directement sur un port USB de notre ordinateur, il fonctionne comme une passerelle entre l'ordinateur et le module Xbee

IEE 802.15.4 : est un protocole de communication défini par l'IEEE. Il est destiné aux réseaux sans fil de la famille des LR WPAN (Low Rate Wireless Personal Area Network) du fait de leur faible consommation et du faible débit des dispositifs utilise ce protocole. Comme le protocole de communication Zigbee.