**QT : LECTURE TRAMES GPS**

**Public concerné : STS SNIR 2e année**

**Compétences abordées :**

Développement en langage C++ (réalisation d’IHM, analyse de fichiers, extraction de données)

**Ressources :**

* **Environnement de développement :** IDEQt creator
* **Trames NMEA :** [*http://www.gpspassion.com/forumsen/topic.asp?TOPIC\_ID=17661*](http://www.gpspassion.com/forumsen/topic.asp?TOPIC_ID=17661)*, wiki NMEA 01803*
* **Langage C++ :** 
  + Aide en ligne de l’IDE (classes Qt)
  + [www.cplusplus.com](http://www.cplusplus.com)
  + La faq de développez.com : <http://cpp.developpez.com/faq/>

**Consignes :** Un compte –rendu sera expédié par mail à la fin de chaque séance de 3h.

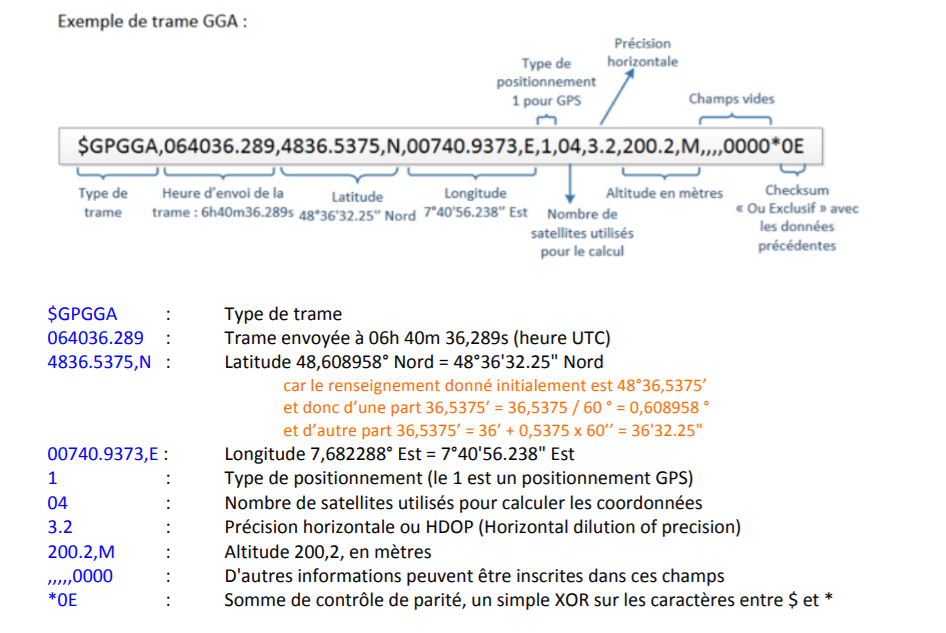
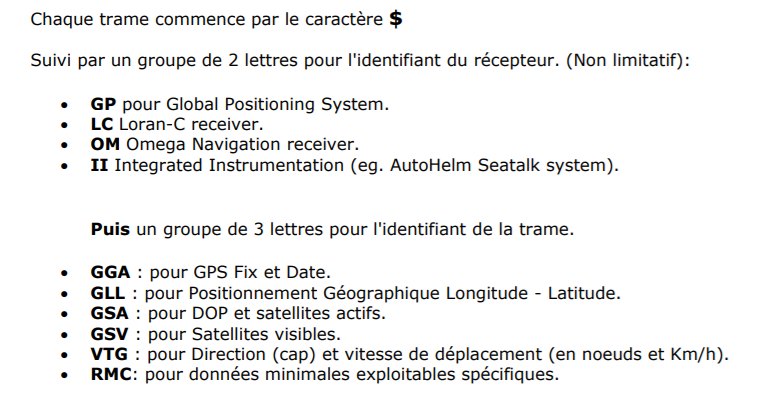
**NB :** Ce sujet s’étalera sur 3 séances de 3h.

## Objectif :

Vous devrez réaliser un programme permettant la lecture des trames GPS, l'interprétation de ces trames et l'affichage des données reçues ainsi que l'enregistrement des données dans un autre format.

Une trame GPS est une chaîne de caractères classique. Les trames sont contenues dans des fichiers **trame**\*.**txt** fournis par l'enseignant.

## ETAPE 1 – PRESENTATION DE LA TECHNOLOGIE GPS

En vous aidant des ressources proposées ci-dessus, rédigez une présentation Powerpoint (ou équivalent), expliquant la technologie GPS (trames NMEA, types de trames). Vous analyserez le cas particulier de la trame **GGA**, qui sera exploitée dans la suite de l’exercice. 

## Thème : Localisation, cartographie et mobilité - Les coordonnées géographiquesETAPE 2 – LECTURE D'UNE TRAME

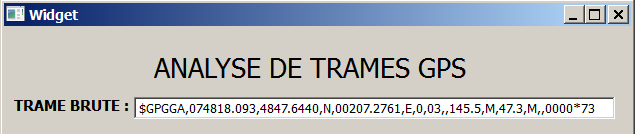
Au lancement de l'application, vous devez :

• Lire le contenu du fichier **trame1.txt** (une seule ligne) (cf lien hypertexte sur le drive gmail elevesnirjf reçu dans votre boîte mail)

• Ecrire cette trame dans un **lineEdit**

Vous utiliserez les classe suivantes : **QFile, QTextStream**

L'IHM, évolutive, devra ressembler à celle-ci :

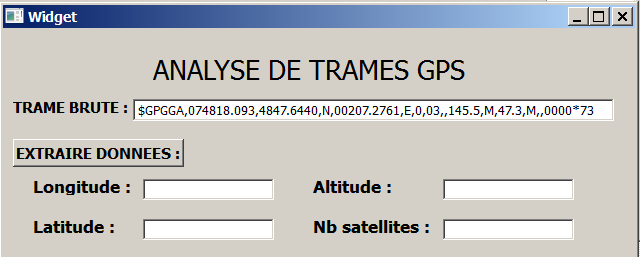


## ETAPE 3 – ETUDE DE LA TRAME – NORME NMEA

Vous devez chercher sur le net des informations sur la norme **NMEA** donnant des indications sur les champs d'une trame GPS.

Servez-vous de ces données pour extraire de la trame précédente, les infos de longitude, latitude, altitude et nombre de satellites.

L'IHM devra ressembler à celle-ci :



Vous utiliserez la méthode **split**(...) de **QString** afin de découper la trame en plusieurs sous-chaînes de caractères. Vous obtiendrez un **QStrinList** qui est une sorte de tableau (liste chainée en réalité) de QString.w

Dans la classe **QStringList** , vous exploiterez la méthode **size**(..) permettant de connaître le nombre d'éléments dans la liste et la surcharge de tableau **[…]** pour accéder plus facilement aux éléments de la liste.

A savoir : **QStrinList** hérite de QList qui est une classe permettant d'obtenir des listes de tout ce que l'on veut (**QString, QFile,** etc.....)

## ETAPE 4 – SAUVEGARDE DES DONNÉES

Vous ajouterez un bouton "**Sauvegarde**" en bas à droite de l'IHM. Ce bouton enregistrera directement (sans **QFileDialog**) dans un fichier texte : **DataTrame.csv**, les 4 infos extraites de la trame.

Le format d'enregistrement dans le fichier sera :

**longitude;latitude;altitude;NbSatellite; suivi d'un retour à la ligne.**

**Note :** Les **;** (points-virgule) serviront à séparer les données lors de l'importation du fichier dans un tableur.

**ATTENTION !** Lors de plusieurs clics successifs sur le bouton de sauvegarde, les données enregistrées

devront se suivrent et ne pas écraser les anciennes.

## ETAPE 5 – LECTURE DE PLUSIEURS TRAMES

Le fichier **trame2.txt** contient plusieurs trames (une par ligne) représentant les points de passage d'un parcours.

Vous ajouterez un bouton "**Ouvrir**" au dessus à gauche de "**Trame Brute**".

Ce bouton devra ouvrir une fenêtre de dialogue d'ouverture de fichier. Vous utiliserez pour cela la classe **QFileDialog**.

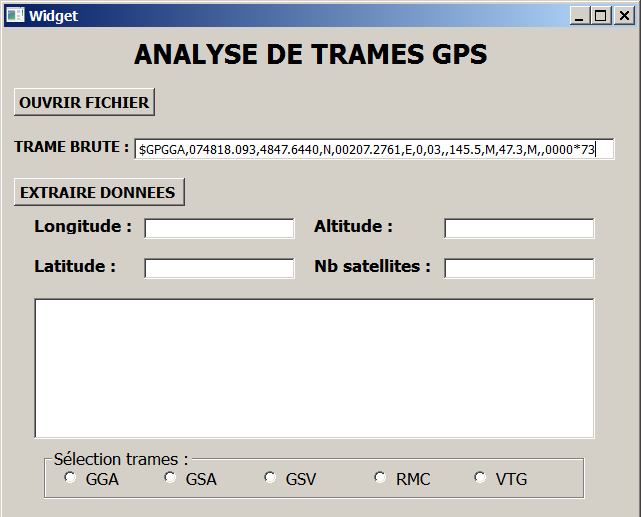
Une fois le fichier accessible, lisez-le ligne par ligne et insérez chaque trame dans un **textEdit** que vous ajouterez dans votre fenêtre.

**Indices :** Méthode **QString GetOpenFileName(…)** et méthodes **readAll()** de **QTextstream**.

Lire tout le fichier dans un Qstring et découper la chaîne obtenue à chaque retour chariot **’’\n’’.**

## ETAPE 6 – SÉLECTION DU TYPE DE TRAME

Ajouter 6 objets de type **radioButton** : un pour chaque type de trame contenue dans **trame2.txt (GGA,GSA,GSV,RMC,VTG**) et un pour l'ensemble des trames.



Le principe étant qu'un **clic** sur l'un des types, affiche seulement les trames du type sélectionné.

De là vous effectuerez le code nécessaire à l'affichage des trames du type sélectionné.

**ETAPE 7 –LECTURE DES TRAMES PROVENANT D’UN CAPTEUR GPS USB**

**NB : Il y a peu de capteurs GPS, donc il faudra partager le matériel.**

**Objectif : Compléter votre application graphique de façon à lire via un port USB le capteur GPSGlobalsat.**

* Installer le **driver windows** suivant mode d'emploi du dossier DriverGPS :

**https://drive.google.com/drive/folders/0ByFdUCMb1cofeHFsY0NkMnhBT0U?usp=sharing**

* Repérer le port USB (gestion des périphériques)

Utiliser le logiciel Haicom viewer pour visualiser les trames GPS **(configuration : 4800 bauds, 8 bits de données, pas de parité, 1 bit de stop).**

**NB** : Lors des tentatives d'accès au port série par programme, arrêter le programme Haicom.

* Lire la documentation Qt et visualiser le tutoriel suivant sur la classe QtSerialPort.

**https://doc.qt.io/qt-5/qtserialport-index.html**

**https://www.youtube.com/watch?v=UD78xyKbrfk**

**autre ex. :**

[**https://www.youtube.com/watch?v=AX-HhBXBzGg**](https://www.youtube.com/watch?v=AX-HhBXBzGg)

* Modifier l'IHM GPS du TP en cours de façon à sélectionner la source des trames GPS par **des boutons radio, soit** le fichier **trame2.txt**, **soit** le port série (capteur réel).

Le contenu sera affiché dans la zone d’édition (textEdit) existante.

