

# Rapport d'analyse

**Easy GNSS : Développement d'un logiciel pour GNSS low-cost**



ÉCOLE NATIONALE  
DES SCIENCES  
GÉOGRAPHIQUES

Nassim CHEBBAH (Chef de projet)

Arthur FAVREAU

Edgar LENHOF

Sanam MONANY



Février – Mai 2019

## Table des matières

---

Introduction.....	2
I. Contexte du projet.....	2
1. Le GNSS.....	2
2. Contexte général.....	4
3. Etat actuel du projet.....	4
4. Commanditaires.....	7
5. Livrables.....	8
6. Utilisateurs finaux.....	8
7. Problématique.....	8
8. Enjeux.....	8
9. Aspects financiers.....	8
10. Aspects sociaux.....	9
II. Objectifs et Besoins.....	10
1. Objectifs.....	10
2. Contraintes.....	10
4. Diagrammes de cas d'utilisation.....	11
III. Analyse fonctionnelle.....	12
1. Fonctionnalités.....	12
2. Diagrammes de classes.....	13
3. Diagramme d'activité.....	14
IV. Etude technique.....	15
1. Diagramme de déploiement/composants.....	15
2. Diagramme d'états-transitions.....	16
V. Réalisation et suivi de projet.....	17
1 Risques.....	17
2. Diagramme de GANTT.....	17
3. Diagramme de PERT.....	18
4. Nom & Logo.....	19
Glossaire.....	20
Liens utiles.....	21

---

## Introduction

---

Ce rapport d'analyse a vocation à décrire le projet de manière succincte.

Nous tenons à remercier nos commanditaires, ainsi que toute l'équipe pédagogique pour leur accompagnement.

### I. Contexte du projet

---

#### 1. Le GNSS

Un GNSS (Système de positionnement par satellites), est un ensemble de composants reposant sur une constellation de satellites artificiels permettant de fournir à un utilisateur sa position 3D, sa vitesse et l'heure par l'intermédiaire d'un récepteur portable de petite taille. Il se caractérise par une précision décimétrique et sa couverture mondiale.

Le premier système de positionnement par satellites est le Global Positioning System (GPS), qui fixe les principes de fonctionnement repris par les systèmes de navigation par satellites développés par d'autres pays. Le système GPS repose sur une constellation d'une trentaine de satellites qui permet à un utilisateur, situé sur n'importe quel point du globe, d'avoir toujours au minimum quatre satellites à portée. Le terminal de l'utilisateur calcule sa position grâce au signal émis par chacun des satellites.

L'URSS a de son côté développé GLONASS, entré en fonction en 1996. L'Union Européenne avec le système Galileo et la Chine avec le système Beidou-2 (COMPASS) développent leur propre système qui devrait être complètement opérationnel en 2020. Le Japon (QZSS) et l'Inde avec l'IRNSS développent de leur côté un système assurant une couverture uniquement régionale dont la Chine dispose également avec Beidou-1.

L'utilisation de récepteurs s'est généralisée pour répondre aux besoins des professionnels et grands publics. Les récepteurs permettent souvent d'exploiter les signaux de plusieurs systèmes notamment GLONASS et GPS.

Un système de positionnement par satellites fournit les coordonnées géographiques en trois dimensions (longitude, latitude, hauteur ellipsoïdale) du récepteur, la vitesse de déplacement, la date et l'heure à son utilisateur. Cette information est obtenue en mesurant la distance à un instant donné entre le récepteur de l'utilisateur et un satellite artificiel dont la position dans l'espace est connue avec précision. En combinant la mesure simultanée de la distance d'au moins quatre satellites, le récepteur est capable de fournir la position et l'altitude avec une précision de l'ordre d'une dizaine de mètres et la vitesse avec une précision de quelques cm/s. Le récepteur peut être au sol ou embarqué dans un véhicule en déplacement.

Pour mesurer la distance entre le récepteur et le satellite, la trajectoire précise de ce dernier doit être connue. Celle-ci est reconstituée à partir de deux types de messages envoyés par le satellite au récepteur :

- Les données d'almanach sont transmises en permanence et fournissent la position approximative des satellites de navigation dans le ciel. Elles permettent au récepteur de repérer rapidement les satellites visibles depuis la position de son utilisateur.
- Les données d'éphémérides fournissent des données de position beaucoup plus précises qui sont actualisées sur des périodes de quelques heures afin de tenir compte des plus petits changements affectant l'orbite des satellites. Ce sont ces données qui sont utilisées pour le calcul de la position.

Connaissant la trajectoire que suit le satellite, le récepteur, pour calculer la position, doit théoriquement utiliser la même heure que le satellite. En effet, compte tenu de la vitesse à laquelle circule le signal (300 000 km/s), une désynchronisation de 10 millisecondes entre l'horloge du satellite et celle du récepteur engendre une erreur de calcul de la position de 3 000 km. La précision et la stabilité de l'heure du satellite sont garanties par l'emport de plusieurs horloges atomiques qui fournissent une heure qui ne dérive que de quelques nanosecondes par jour. Le récepteur, par contre, ne peut être équipé d'une horloge aussi précise pour des raisons de coût et d'encombrement. L'heure est fournie par un oscillateur à quartz dont la dérive journalière moyenne est de 10 millisecondes.

Pour déterminer sa position, sa vitesse et l'heure, le récepteur calcule la distance à laquelle se trouve le satellite à partir des données de l'éphéméride et en se basant sur son horloge interne. Mais ce calcul est entaché d'erreurs, du fait de la désynchronisation des horloges mais également parce que différents phénomènes viennent perturber la propagation du signal :

- Le signal est ralenti durant sa traversée de l'atmosphère (ionosphère et troposphère) de manière variable
- Le signal peut être réfléchi par le sol avant d'atteindre le récepteur
- Le signal peut être bloqué par les constructions, les montagnes, ou une forêt dense.

La méthode de trilateration permet théoriquement de calculer position, vitesse et temps en utilisant le signal de trois satellites : la distance à laquelle se situe un satellite positionne l'utilisateur à la surface d'une sphère dont le centre est le satellite. L'intersection de 3 sphères permet d'identifier un point unique dans l'espace. Un quatrième satellite est néanmoins requis pour permettre de déterminer le décalage des horloges et réduire les incertitudes liées aux autres sources de perturbation du signal.

Le récepteur est souvent couplé à un calculateur qui détermine le cap à suivre pour rejoindre un point de coordonnées connues ou qui affiche une carte numérique sur un écran. Le récepteur peut également être interfacé à un ordinateur portable muni d'un logiciel cartographique, ou à une centrale de navigation intégrant également tous les autres senseurs de bord (compas, tachymètre, autres systèmes de radionavigation...). Le récepteur peut aussi être couplé à un téléphone cellulaire ou satellitaire qui retransmet automatiquement la position du mobile à un central. Ce central peut alors contrôler, gérer ou surveiller le déplacement des mobiles.

## **2. Contexte général**

Le GNSS (Système de positionnement par satellites), de par sa grande précision et sa couverture mondiale, a montré son utilité dans de nombreux domaines comme la navigation, la topographie ou encore la géophysique.

La communauté intéressée par les GNSS se fait de plus en plus vaste. Cependant, de par leur coût de plusieurs dizaines de milliers d'euros, les GNSS sont loin d'être à la portée de toutes les bourses.

Aujourd'hui, avec l'apparition du micro-ordinateur Raspberry Pi, dont le coût unitaire ne dépasse pas les 50 euros, l'idée de fabrication de GNSS low-cost est apparue. La communauté s'intéresse de plus en plus à ce projet, notamment le grand public, ainsi que les pays émergents, disposant d'un budget limité.

Aussi, avec l'avènement du « Do It Yourself » ("Fais le toi-même" en anglais), la communauté s'est lancé le défi de créer de A à Z un GNSS low-cost, en utilisant également des matériaux de récupération ou peu onéreux. L'utilisation de données Open Source est aussi envisagée pour plus de transparence. A terme, la création d'un GNSS de l'ordre de quelques centaines d'euros s'avère possible.

## **3. Etat actuel du projet**

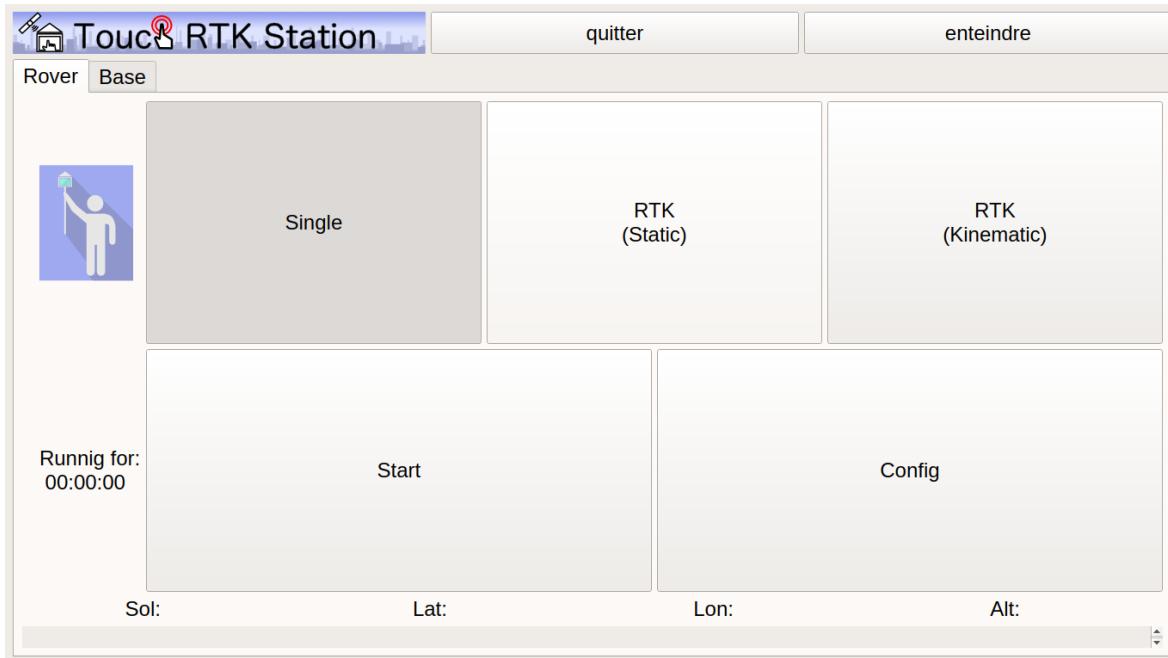
L'ENSG collabore depuis 3 ans avec nos commanditaires sur le projet. Cette collaboration a mené au développement de l'application RTKBase, qui permet de faire du GNSS de manière low-cost. Elle a été développée en C et en C++ par des élèves de l'ENSG lors des projets développement de la filière PPMD. RTKBase s'appuie sur l'application RTKLIB qu'il compile directement dans le code afin de réaliser les traitements GNSS nécessaires. Au fil des années et des différents projets scolaires, le code est devenu de plus en plus complet et par la même occasion de plus en plus complexe. Ainsi, il devient aujourd'hui plus difficile de comprendre et de reprendre le code pour y ajouter de nouvelles fonctionnalités.

De plus, les commanditaires souhaitent que la refonte du code soit faite en Python plutôt qu'en C, afin de permettre à une communauté de développeurs plus large de pouvoir le reprendre à leur tour. Nos commanditaires ont réalisé la partie hardware avec du matériel low-cost qui nous a été confié pour la durée du projet.

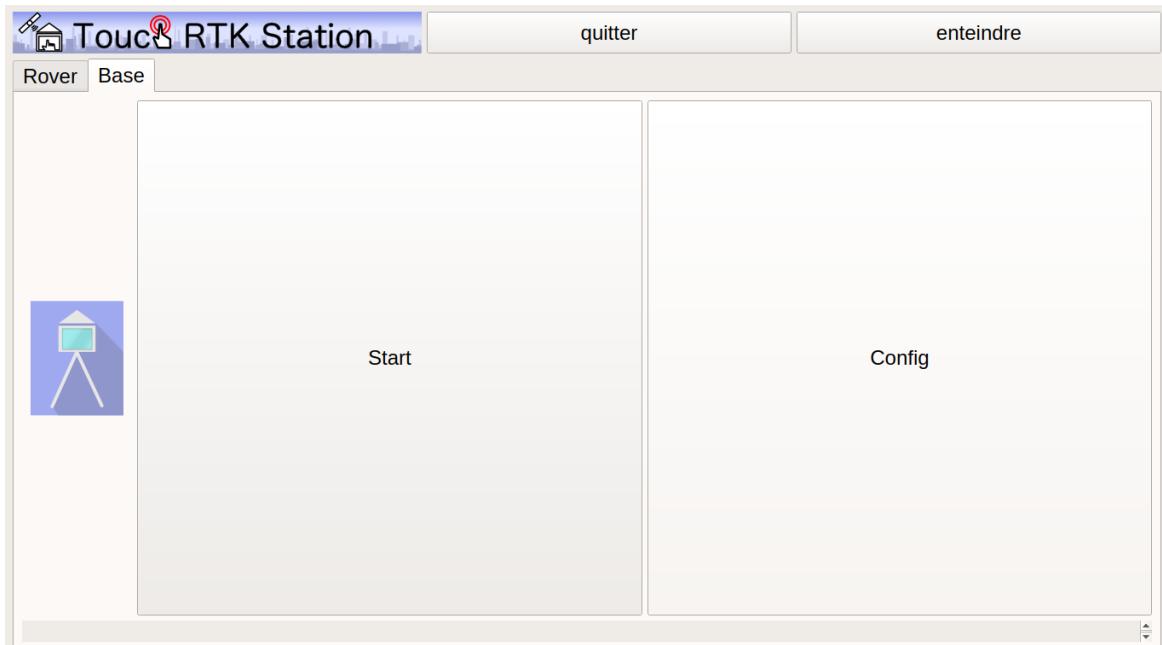
En parallèle, une autre application, TouckRTKStation, a été développée par un universitaire japonais, M. Taro Suzuki. Le code, écrit en Python, est plus abordable que celui de RTKBase, mais ne possède pas autant de fonctionnalités. Contrairement à RTKBase, TouchRTKStation compile l'application RTKLIB en dehors du code.

Qui plus est, bien que le code de M. Suzuki fonctionne, de nombreuses choses sont à reprendre au niveau de l'architecture et des conventions informatiques. M. Suzuki a également réalisé la partie hardware de TouchRTKStation, que nous n'avons cependant pas à disposition. En revanche, nous disposons de hardwares low-cost similaires, réalisés par d'anciens IT1 lors d'un projet à Forcalquier.

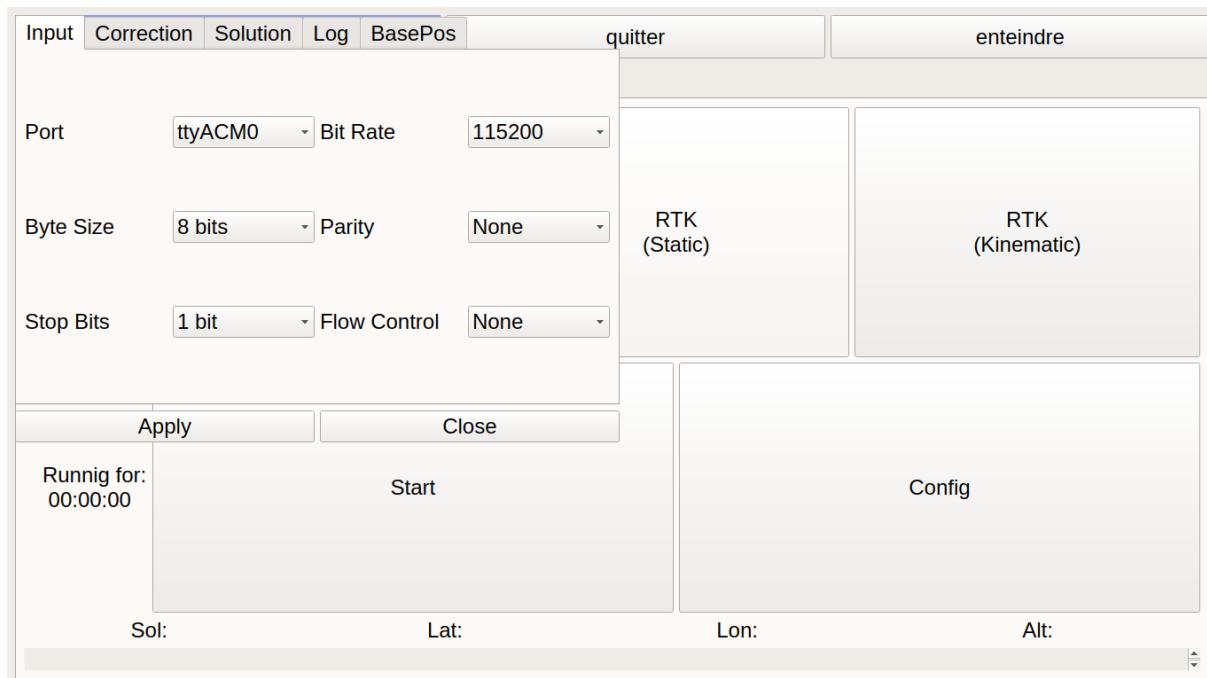
Le but de notre projet est donc de développer un équivalent de RTKBase en Python, l'application Easy GNSS. Pour cela, nous allons repartir du code de TouchRTKStation, le remodeler, et y rajouter les fonctionnalités de RTKBase. Notre projet sera amené à être testé cette année par un groupe d'IT1, dans le cadre du projet de fin de stage à Forcalquier.



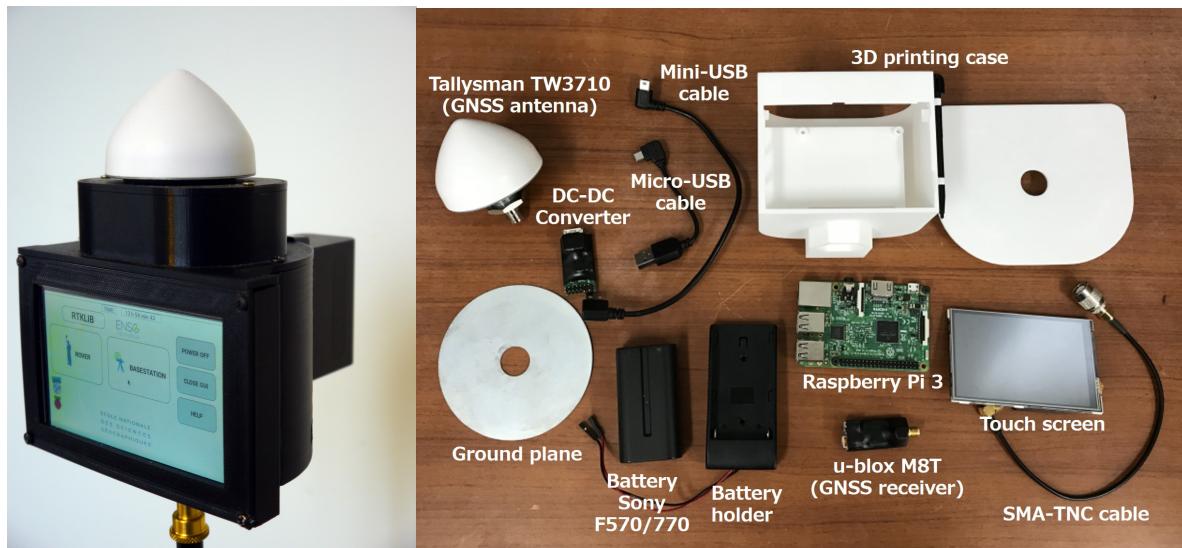
Screenshot de l'interface de TouchRTKStation: Mode Rover



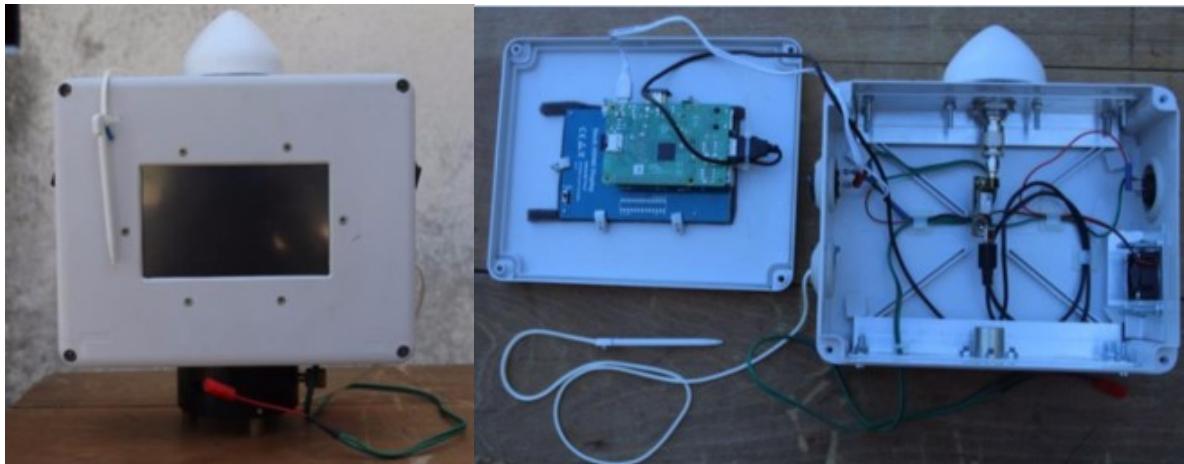
Screenshot de l'interface de TouchRTKStation : Mode Base



Screenshot de l'interface de TouchRTKStation : Fenêtre de configuration



Hardware de RTKBase



Hardware de l'ENSG



Hardware de M. Taro Suzuki

### 4. Commanditaires

Deux commanditaires extérieurs nous supervisent pendant ce projet :

- Jean-Yves Perrin, employé à la Gendarmerie Nationale
- Francklin N'Guyen Van, intermittent à France Télévisions

Nos deux commanditaires sont à l'initiative du projet. Ils se sont rencontrés dans le cadre d'un évènement du Club de Drones de France. Tous deux passionnés par les drones, ils souhaiteraient à terme pouvoir faire du positionnement par drones à partir de leur GNSS low-cost. Cela pourrait s'avérer utile pour réaliser des projets en géomatique, par exemple en photogrammétrie.

Jacques Beilin, enseignant-chercheur au DPTS, suit également de près ce projet. Il souhaiterait que notre application Easy GNSS soit testée par des IT1 à Forcalquier. Le fait que le code soit en Python facilitera leur travail, car ils n'ont pour la plupart jamais codé en C ou en C++.

## 5. Livrables

Nous devons rendre plusieurs livrables :

- Aux commanditaires : Code commenté et documentation utilisateur
- A l'équipe pédagogique : Rapport d'analyse et présentations (COPIL et soutenance final)

## 6. Utilisateurs finaux

Le produit fini est destiné à des utilisateurs ayant des moyens limités, et ayant besoin d'une précision la plus proche possible de celle des appareils professionnels. Ces personnes ne sont pas nécessairement familiarisées avec la géomatique et/ou la programmation.

Le faible coût du GNSS low-cost pourrait permettre une meilleure accessibilité au positionnement. Par exemple, il pourrait s'exporter à l'international, notamment dans les pôles de géomatique des pays en voie de développement, qui n'ont pas les moyens d'acheter un GNSS professionnel.

## 7. Problématique

A terme, les commanditaires souhaitent offrir un outil prêt à l'emploi à la communauté. Cet outil doit être le plus facile d'utilisation possible, avec une bonne ergonomie. Les logiciels utilisés doivent être Open Source. En outre, une bonne documentation et un code bien commenté sont nécessaires, au cas où le code devrait être amélioré ultérieurement. Le GNSS, tout en restant low-cost, doit permettre un positionnement suffisamment précis, sans quoi il perdrait tout son intérêt.

## 8. Enjeux

Le principal enjeu est de faire vivre le projet, où se sont investis des collaborateurs du monde entier. En particulier, l'ENSG est investie dans ce projet depuis 3 ans, et beaucoup d'élèves de promotions variées ont déjà travaillé dessus, ou seront amenés à le faire, car le projet est loin d'être terminé. Il est donc essentiel de mettre à profit les travaux précédemment réalisés. Enfin, il est essentiel de maintenir un partenariat durable avec les commanditaires.

## 9. Aspects financiers

Nous estimons le coût en temps pour l'équipe à 30 heures pour la partie analyse et 100 heures pour la partie développement.

---

Nous disposons de 1 300 euros de matériel prêté par les commanditaires : 4 stations équipées de TouchRTKStation et RTKBase, 5 batteries Sony et Trimble, une antenne et son récepteur radio, ainsi que divers chargeurs, câbles, housses et étuis.

Un GNSS low-cost aurait au final un coût avoisinant les 300 euros. A ce coût, il faut rajouter d'éventuels surcoûts pour l'accès à un serveur NTrip, et, selon les pays, l'accès au réseau GNSS permanent.

### 10. Aspects sociaux

Les commanditaires sont passionnés de drones, et ont de bonnes connaissances en GNSS grâce à la pratique. Francklin N'Guyen Van a également de bonnes connaissances en programmation.

Notre professeur référent, Jacques Beilin est expert en GNSS et en programmation. Il peut répondre sans problème à nos questions.

Dans l'équipe, tout le monde a de bonnes connaissances en GNSS, et une aisance en Python. Nassim et Arthur ont des connaissances en C acquises en classe préparatoire. Edgar a acquis des connaissances en C++ suite à son séjour Erasmus. Enfin, Sanam a déjà travaillé sur le projet lors du stage à Forcalquier de l'an dernier.

Après vote à l'unanimité, Nassim a été désigné chef de projet pour ses qualités de meneur d'équipe.

## II. Objectifs et Besoins

---

### 1. Objectifs

Nous avons 4 principaux objectifs, que nous devons respecter :

- Réécrire le code en C et en C++ de RTKBase en Python
- Ajouter des fonctionnalités à TouchRTKStation, à commencer par celles présentes sur RTKBase. Faire un fork du code de TouchRTKStation sur GitHub.
- Rendre le code facilement compréhensible et modifiable par un utilisateur ultérieur, en ajoutant de la documentation, en renseignant l'architecture logicielle, et en commentant le code.
- Ajouter des fonctions de post-processing

Si le temps nous le permet, nous pourrions ajouter des fonctionnalités supplémentaires.

### 2. Contraintes

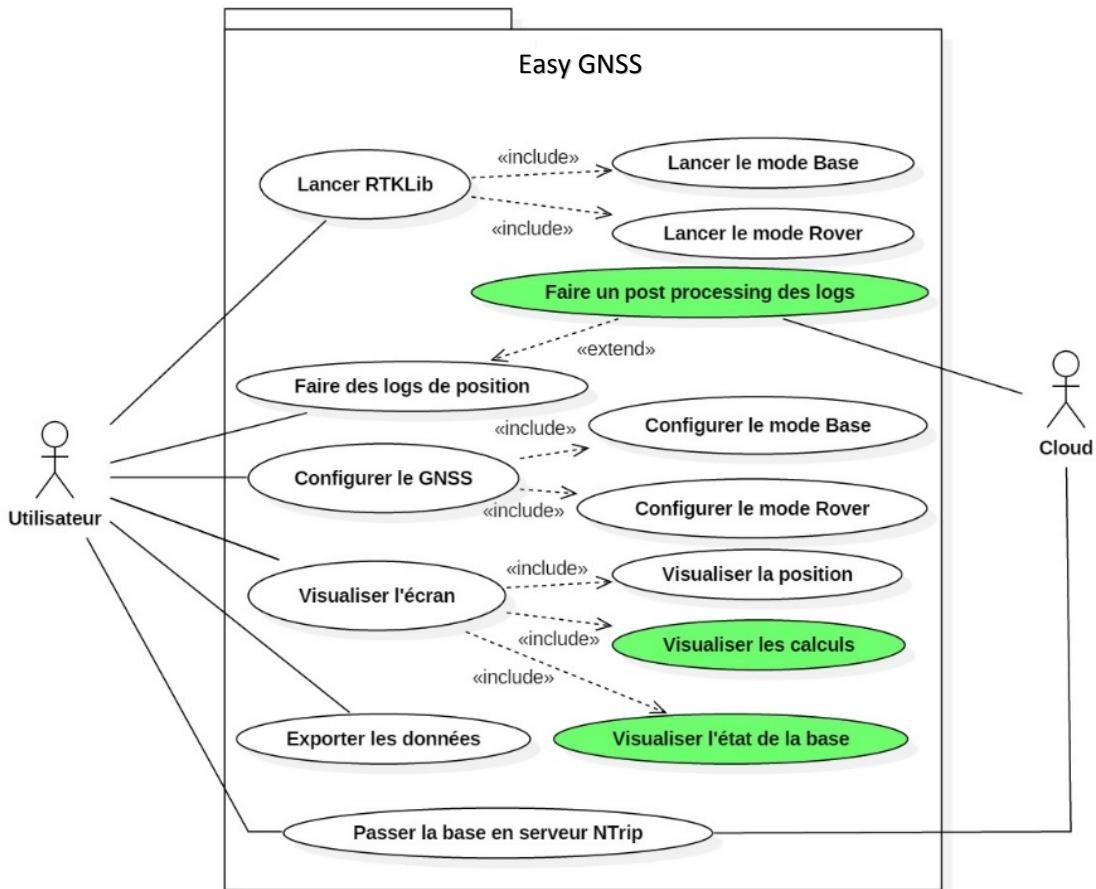
Nous devons d'abord gérer la compatibilité du code avec les hardwares, contrainte sans laquelle le GNSS ne pourrait pas fonctionner. Le code doit également être simple à comprendre et à reprendre, et nous devons donc respecter un certain nombre de contraintes :

- Le code doit être en Python car c'est un langage simple à comprendre et à utiliser.
- La documentation doit être rédigée en anglais, tout comme les commentaires et variables du code. Cette documentation doit être la plus complète possible, avec notamment une notice d'utilisation du GNSS et un compte-rendu des modifications apportées.
- Dans une optique low-cost et de transparence, nous devons utiliser des logiciels Open Source.
- Nous devons redéfinir l'architecture logicielle qui reste encore fragile. Nous allons donc devoir utiliser des patrons de conception dans le but de respecter les principes SOLID de programmation.

### 3. Recueil du besoin – Acteurs

Les utilisateurs finaux de l'application sont essentiellement des particuliers. On peut y trouver des dronistes qui veulent positionner leur drone pour faire de la photogrammétrie et de la cartographie, des agriculteurs qui veulent positionner leurs tracteurs pour automatiser leurs traitements et plus généralement n'importe quelle personne qui souhaite faire du positionnement à moindre coût.

#### 4. Diagrammes de cas d'utilisation



*Diagramme de cas d'utilisation de Easy GNSS*

### III. Analyse fonctionnelle

---

#### 1. Fonctionnalités

Nous devons implémenter toutes les fonctionnalités de RTKBase dans Easy GNSS, auxquelles nous ajoutons quelques fonctionnalités supplémentaires :

- Ajouter des fonctions de post-processing (**Priorité 0 : Obligatoire**)
- Ajouter tous les modes de calcul de RTKBase (Single, SBAS, DGPS, PPP-kinematic/static, RTK-kinematic/static) (**Priorité 0 : Obligatoire**)
- Rendre le code compatible avec l'écran 5 pouces de RTKBase et le 4 pouces de TouchRTKStation (**Priorité 1 : Recommandé**)
- Sur le Rover : Configurer automatiquement la position de la base (**Priorité 1 : Recommandé**)
- Ajouter l'affichage des satellites (**Priorité 1 : Recommandé**)
- Rendre l'affichage de l'interface plus ergonomique (**Priorité 2 : Secondaire**)

## 2. Diagrammes de classes

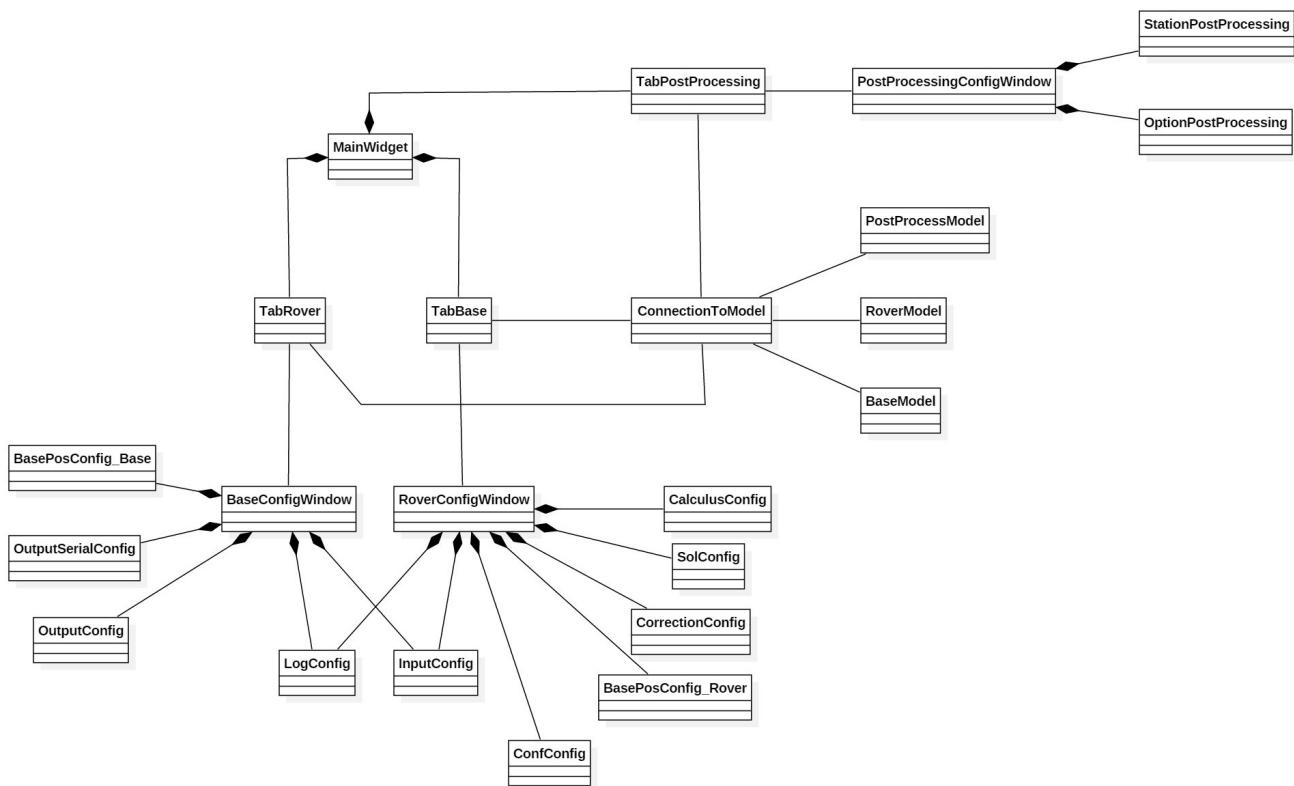


Diagramme de classes d'Easy GNSS (Disponible sur le drive en annexe)

Après analyse, nous avons conclu qu'il était plus intéressant de partir de l'application TouchRTKStation qui constitue une base de travail opérationnelle et moins complexe à aborder que RTKBase. Une refonte en profondeur de l'architecture est cependant impérative afin de clarifier son fonctionnement.

### 3. Diagramme d'activité

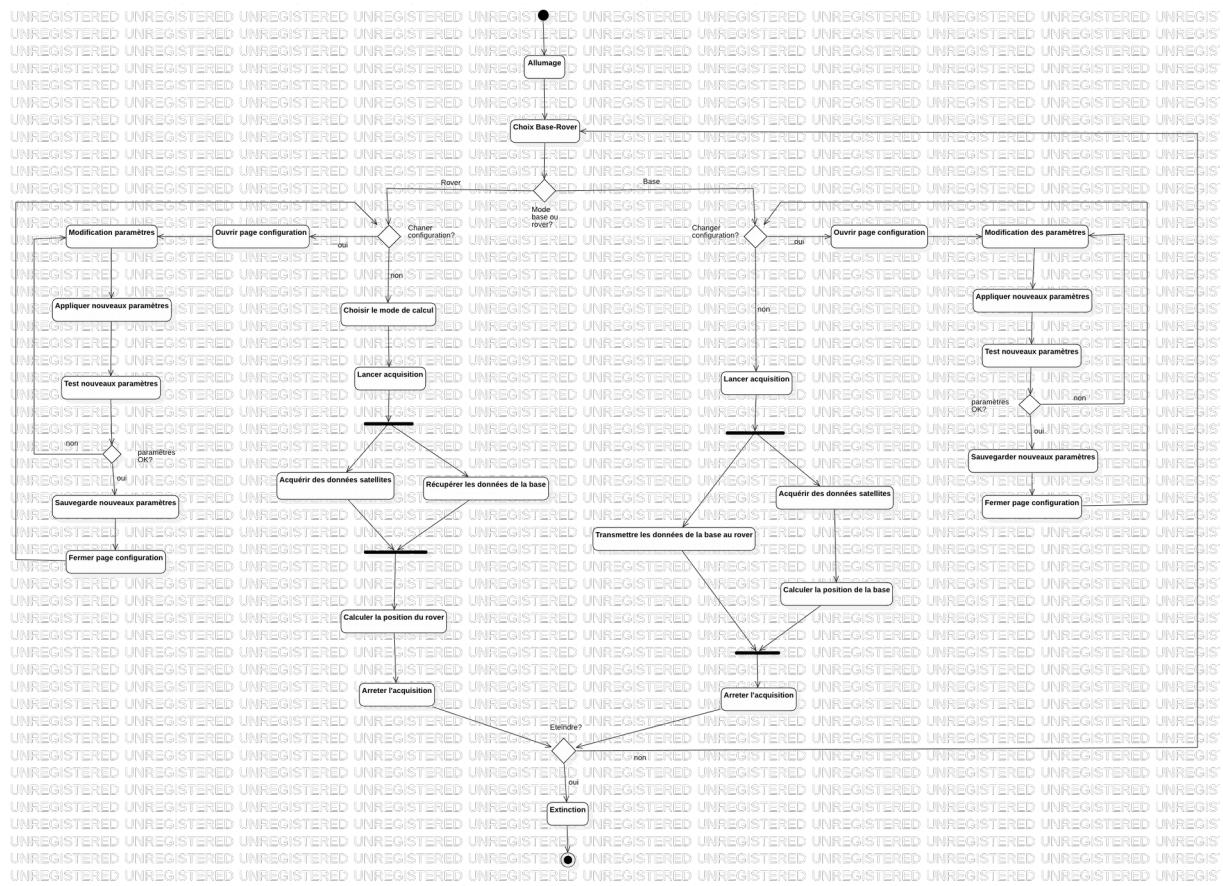


Diagramme d'activité d'Easy GNSS (Également disponible sur le drive)

Ce diagramme d'activité montre l'utilisation typique de l'application où l'utilisateur choisit entre un mode base ou rover et peut modifier différents paramètres avant de lancer l'acquisition.

## IV. Etude technique

### 1. Diagramme de déploiement/composants

Au niveau du fonctionnement global de l'application, le code Python assure principalement l'interface graphique avec l'utilisateur qui indique les options et configurations avant de lancer l'acquisition. Le code traite ensuite ces informations pour écrire et les transmettre au bon format aux différents modules extérieurs (antenne, RTKLIB, radio, NTRIP).

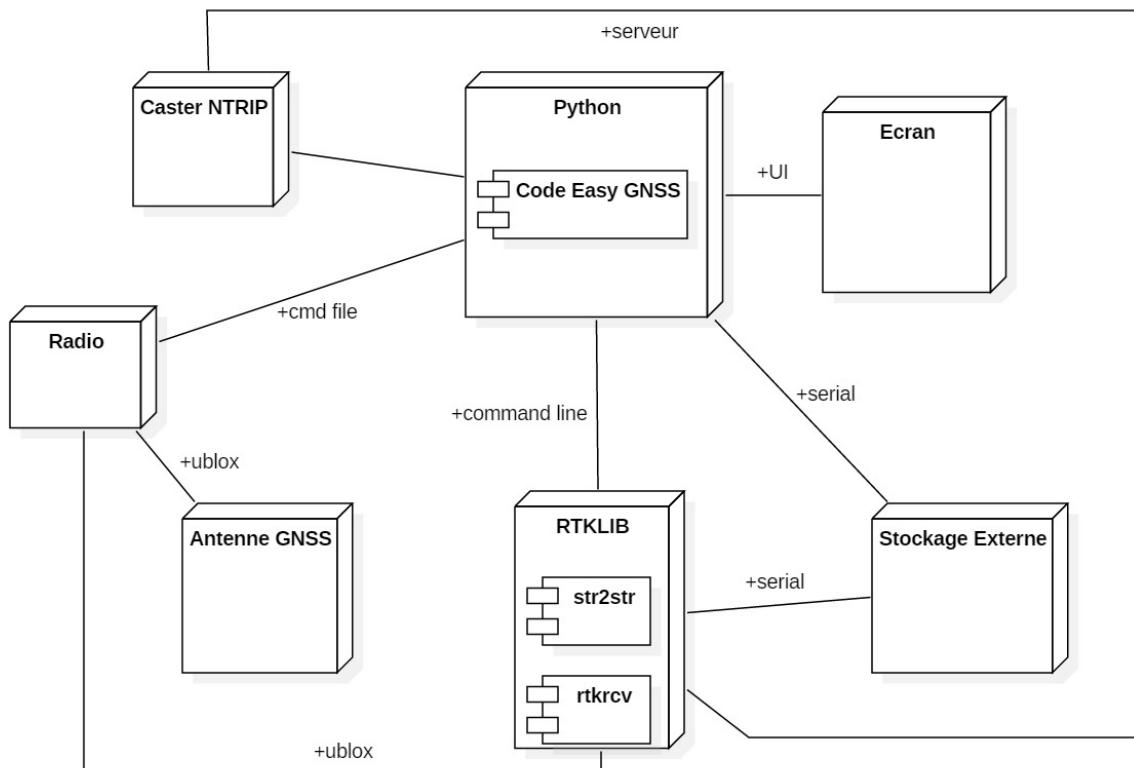


Diagramme de déploiement/composants d'Easy GNSS

## 2. Diagramme d'états-transitions

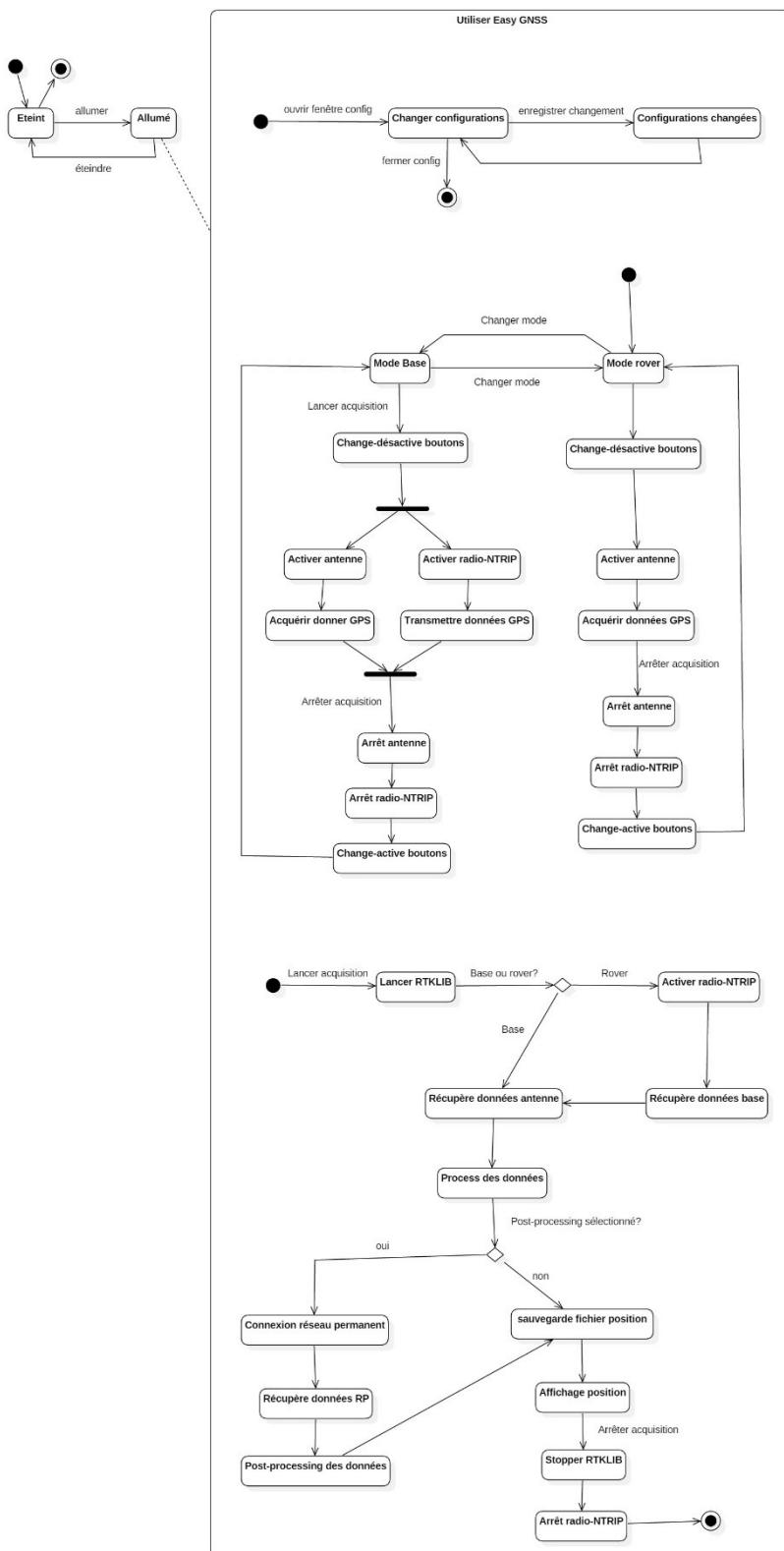


Diagramme d'états-transitions  
d'Easy GNSS (Également  
disponible sur le drive)

## V. Réalisation et suivi de projet

### 1 Risques

Nous avons identifié plusieurs risques, que nous avons classés dans ce tableau :

Numéro	Risque	Gravité	Occurrence	Impacts	Action préventive	Action corrective
1	Délai insuffisant	Majeure	Assez faible	Projet de Forcalquier 2019 compromis	Priorisation des tâches Répartition efficace des tâches	Documentation suffisamment complète pour continuer
2	Perte/Dégradation du matériel	Majeure	Faible	Image des membres de l'équipe et de l'école	Précautions	Rachat du matériel
3	Perte de données	Modérée	Faible	Ecart avec le planning prévisionnel	Outils de gestion de versions	Reprise des versions précédentes
4	Conflits avec les commanditaires	Modérée	Faible	Ecart avec le planning prévisionnel Surcharge de travail	Communication régulière Compréhension du besoin	Recadrement des besoins des commanditaires
5	Problèmes de communication avec les commanditaires	Mineure	Moyenne	Ecart avec le planning prévisionnel	Entretiens réguliers en prévision Diversification des moyens de communication	Contact avec le professeur référent
6	Conflits dans l'équipe	Mineure	Assez faible	Ecart avec le planning prévisionnel	Communication efficace	Intervention du chef de projet

### 2. Diagramme de GANTT

Pour visualiser le déroulement du projet (Parties analyse et développement), nous avons réalisé le diagramme de GANTT suivant :

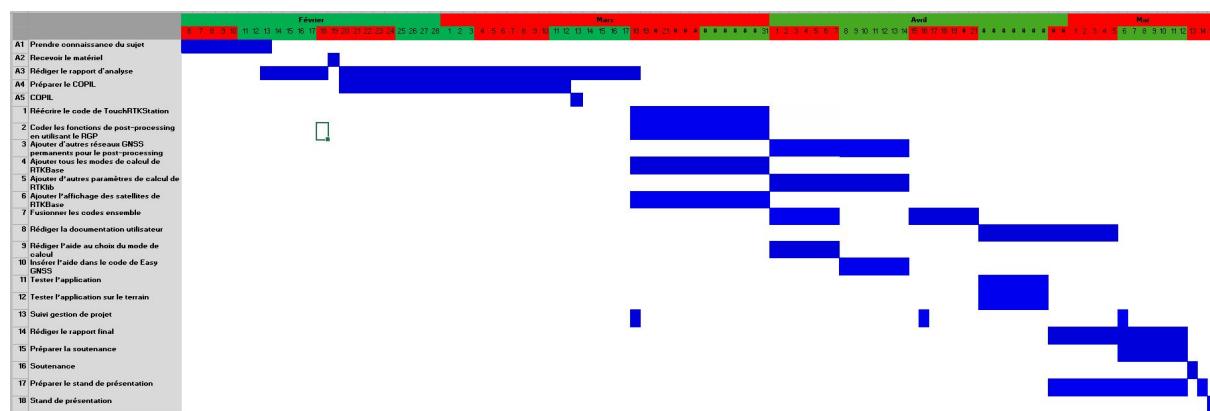


Diagramme de GANTT initial (Également disponible sur le drive)

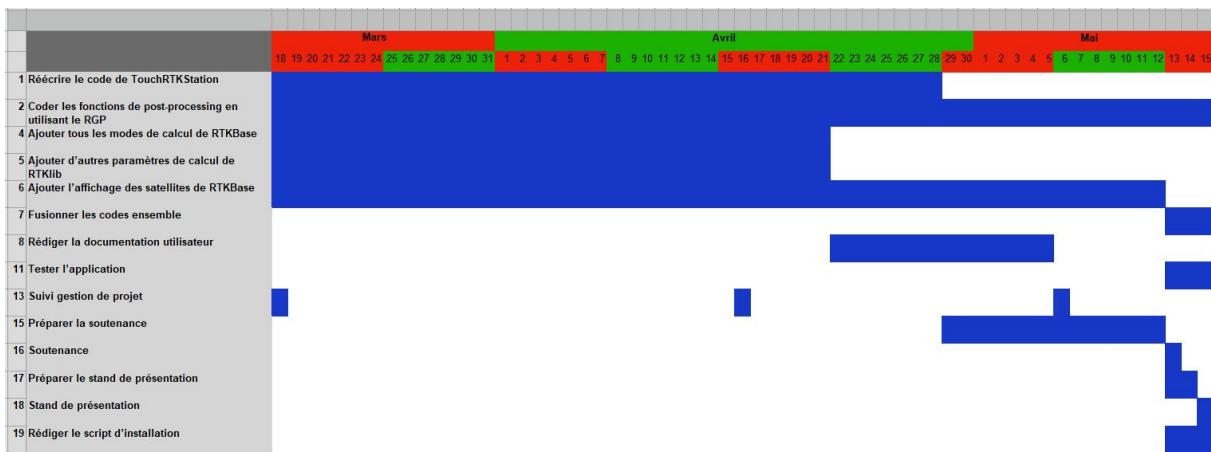


Diagramme de GANTT final (Également disponible sur le drive)

### 3. Diagramme de PERT

Pour organiser la répartition des tâches, nous avons fait un diagramme de PERT de la partie développement. Pour la partie analyse, ce n'était pas nécessaire, car toute l'équipe a travaillé sur le rapport et le diaporama. Les numéros correspondent aux numéros des tâches du GANTT. Nous avons suivi le code couleur suivant :

**Nassim : Jaune**

**Arthur : Vert**

**Edgar : Orange**

**Sanam : Violet**

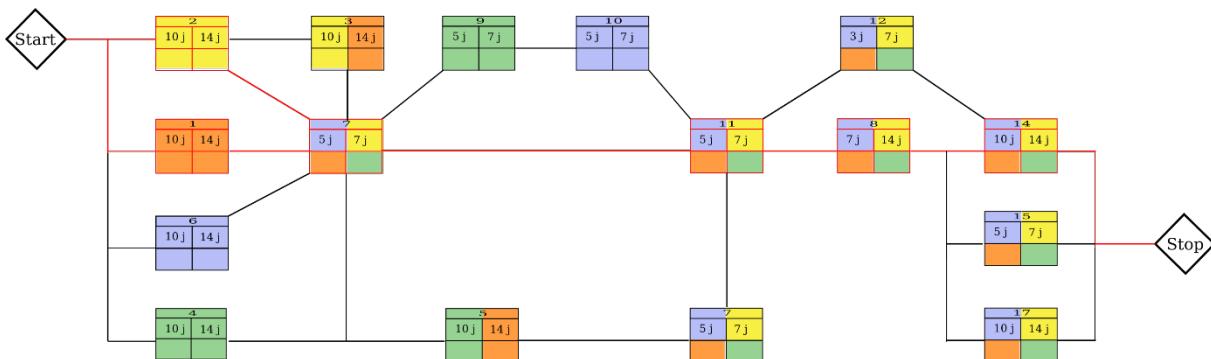


Diagramme de PERT initial (Également disponible sur le drive)

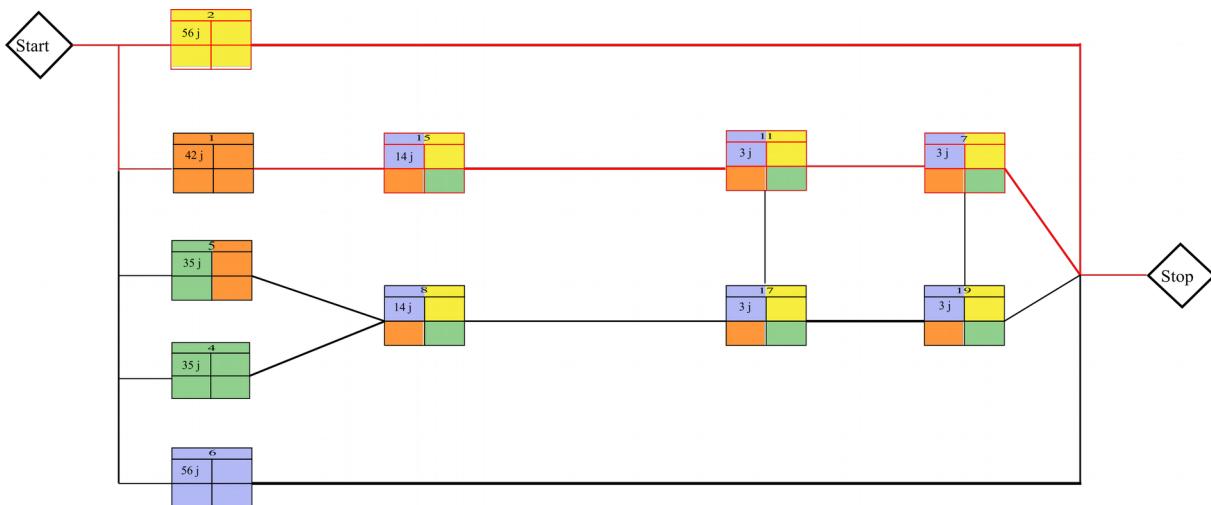


Diagramme de PERT final (Également disponible sur le drive)

#### 4. Nom & Logo

Le projet a pour nom « Easy GNSS », en référence au GNSS (thème du projet) et à son aspect accessible pour tous (facile d'accès et d'utilisation).

Quant au logo, il représente un récepteur GNSS fabriqué par l'ENSG. A l'intérieur du GNSS, nous avons intégré le logo de Python, le langage utilisé pour le projet.

Nous avons également intégré le nom du projet en haut, et le nom de l'école en bas du logo. La spirale et la couleur jaune sont des choix purement esthétiques.

En référence à l'école, nous avons utilisé la couleur verte, l'aiguille de la boussole, ainsi que la police d'écriture du logo ENSG. Ainsi, nous montrons la contribution forte de l'école dans le projet.



Logo du projet Easy GNSS

## Annexes

---

### Glossaire

**Hardware** : Matériel physique, composants.

**Software** : Logiciel ou application qui est présente sur un ordinateur.

**Raspberry Pi 3** : Micro-ordinateur apprécié pour son prix et ses performances.

**Caster** : Serveur de flux de données.

**Open Source** : Logiciel permettant la libre redistribution, l'accès au code source et la création de travaux dérivés.

**RTK (Real Time Kinematic)** : Technique de positionnement par satellite, basée sur la mesure de phase des ondes porteuses des signaux émis. La précision de cette technique est de l'ordre de 10 centimètres.

**Base** : Récepteur fixe dont la position est connue précisément qui compare sa position réelle à celle calculée à partir du signal GPS puis émet les corrections aux mobiles.

**Mobile/Rover** : Récepteur mobile qui calcule sa position par rapport à la base en s'appuyant sur les corrections apportées.

**RTKbase** : Logiciel codé en C et en C++ par des élèves de l'ENSG permettant d'utiliser la technique de positionnement RTK en tant que base ou mobile.

**TouchRTKStation** : Logiciel écrit en Python et permettant de faire du GNSS low-cost, codé par Taro Suzuki de la Waseda University.

**NTRIP (Networked Transport of RTCM via Internet Protocol)** : Méthode standardisée de transmission au rover des données de correction de la base.

**RTKLib** : Librairie open-source proposant un ensemble de programmes pour le positionnement standard et précis par GNSS.

**rtkrcv** : Module de RTKLib, permet d'effectuer les traitements de RTKLib en ligne de commande.

**str2str** : Module de RTKlib, assure la transmission des données de la base vers le rover.

**Post-processing** : Comparaison des points mesurés par les récepteurs GNSS avec des points connus issus d'un réseau permanent pour améliorer la précision des récepteurs.

**Single** : Mode de positionnement direct, les données sont directement traitées par RTKLib sans apport de correction.

**SBAS (Satellite-Based Augmentation System)** : Mode de positionnement qui incorpore les corrections radio-diffusées par satellites d'un réseau régional (WAAS, EGNOS...)

**PPP (Precise Point Positioning)** : Mode de positionnement direct qui associe des horloges et des orbites précises permettant d'obtenir une position précise avec un seul récepteur.

**RTK-static** : Mode de positionnement composé d'une station de base et de stations mobiles. Les stations reçoivent des données d'un premier satellite et on effectue une double différence entre ces données et celles issues d'un second satellite.

**RTK-kinematic** : Mode de positionnement similaire à RTK-static mais adapté pour le déplacement des récepteurs dans l'espace.

## Liens utiles

Code de TouchRTKstation : <https://github.com/taroz/TouchRTKStation>

Code de RTKbase : [https://github.com/Francklin2/RTKLIB\\_Touchscreen\\_GUI](https://github.com/Francklin2/RTKLIB_Touchscreen_GUI)

Librairie RTKlib : <http://www.rtklib.com/>

Drive contenant les diagrammes complets : <https://drive.google.com/drive/folders/1-D3477r2nxRdvNKAkVLE8U3c9ANWT879>