Rapport de projet

GNSS lowcost

Valentin HUE

Samira KARIMOU

Paul MAZERAN

Sanam MONANY

Fiona PRUD'HOMME

ENSG Géomatique

ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

IGN

INSTITUT NATIONAL DE L'INFORMATION GÉOGRAPHIQUE ET FORESTIÈRE

Forcalquier

Juillet – Août 2018

Table des matières

Introduction ---------- 2

I. Liste de matériel ---------- 3

II. Préparation des composants ---------- 4

1) Raspberry Pi ---------- 4

2) Ecran tactile ---------- 5

3) Antenne et GNSS ---------- 5

4) Ntrip ---------- 8

III. Fabrication des GNSS ---------- 9

1) Programmation ---------- 9

2) Adaptation de l'alimentation ---------- 10

3) Assemblage des composants ---------- 12

IV. Essais des récepteurs GNSS ---------- 15

1) Acquisition sur le terrain ---------- 15

2) Traitement des données ---------- 17

4) Pistes d'amélioration ---------- 19

Conclusion ---------- 19

Annexe ---------- 20

Introduction

Le projet de fabrication d'une paire base-mobile de GNSS s'inscrit dans un contexte de développement de puces GNSS moins précises mais plus abordables en terme de coût que les puces de précision centimétrique.

Il a pour objectif d'étudier la possibilité de fabriquer des GNSS à moindre coût mais un peu moins précis que les appareils sophistiqués : c'est une précision de 10cm qui est demandée. Il s'agit donc de fabriquer une paire base-mobile de récepteurs GNSS L1 (mono-fréquence sensibles à la fréquence L1 = 1 575,42 MHz) à partir des composants fournis (à tester préalablement) et de vérifier sa fiabilité en effectuant des tests sur le terrain.

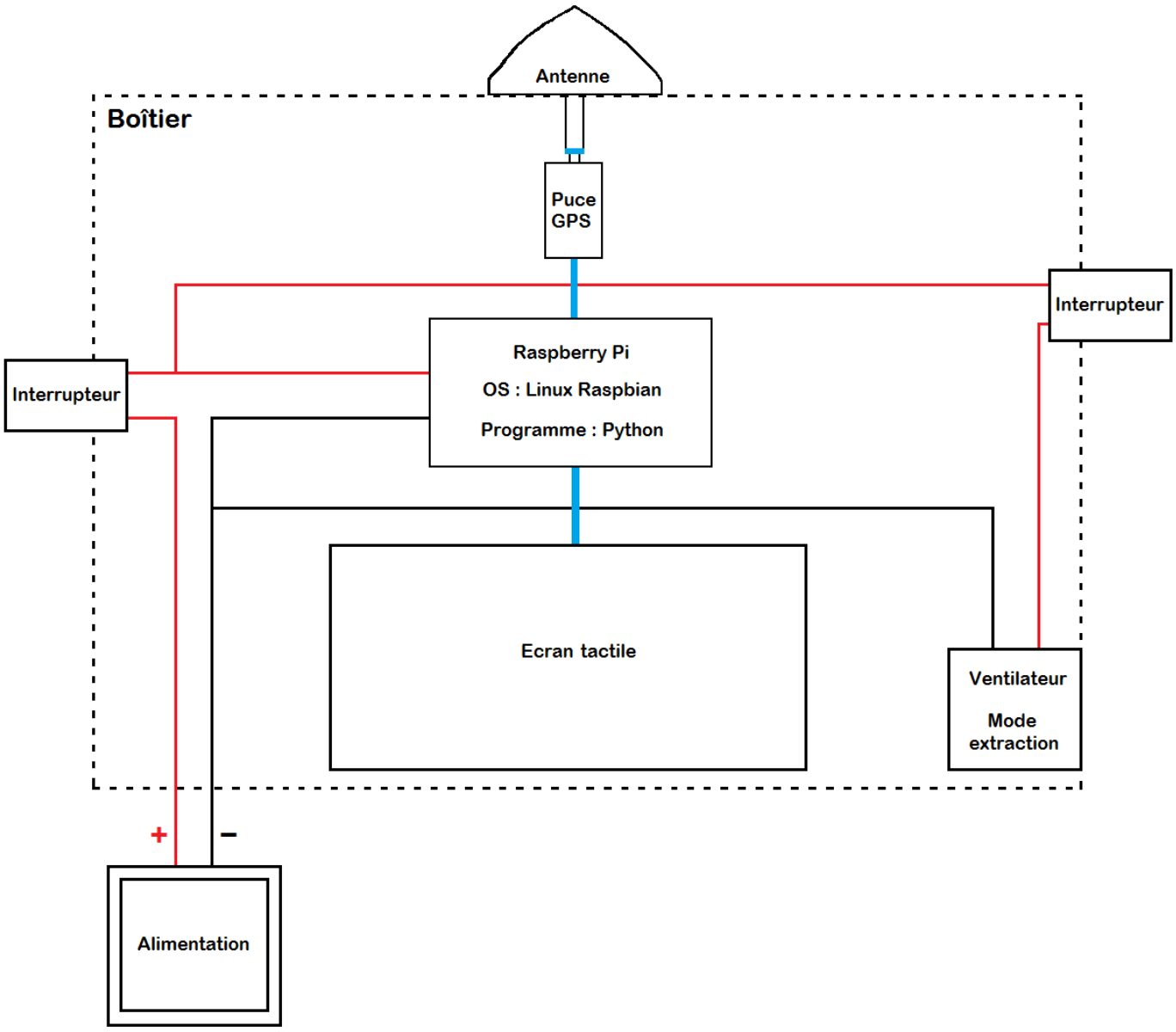


Schéma de l'agencement des composants d'un récepteur GNSS

I. Liste de matériel

➢ Deux Raspberry Pi3+  ---------- 60€ pièce (ebay Hongkong)

La Raspberry Pi est un nano-ordinateur peu puissant par rapport à un ordinateur standard mais suffisant pour un récepteur GNSS et facile à incorporer dans une boîte. La tension de son alimentation doit valoir près de 5V.

Nous avons acheté les Raspberry avec leurs starters kits respectifs mais une Raspberry seule coûte 35€ seulement.

➢ Deux puces GNSS L1 uBlox M8T  ---------- 65€ pièce (Digikey)

Cette puce GNSS ne reçoit que les ondes de fréquence L1 grâce à la communication sans fil gérée par l'entreprise Ublox.

C'est un modèle M8T, destiné à la mesure de positionnement précis.

Son prix abordable pour son utilité en fait un modèle pratique pour le récepteur GNSS.

➢ Deux antennes GNSS Tallysman  ------ 100€ pièce (ebay Hongkong)

C'est une antenne sensible aux ondes de la plage de fréquence 1,574 GHz ~ 1,606 GHz (qui inclut la fréquence L1), en particulier celles issues de GPS et de GLONASS.

➢ Deux écrans tactiles 5''  ---------- 35€ pièce (ebay Hongkong)

Cet écran est compatible avec la Raspberry, peu coûteuse par rapport à la majorité des écrans similaires et son aspect tactile est ergonomique.

➢ Deux boitiers  ---------- 10€ pièce (ebay Hongkong)

Cette boîte en plastique offre une bonne isolation thermique et est assez grande pour contenir tous les composants agencés correctement.

➢ Deux cartes microSD 16Go 

---------- 5€ pièce (ebay Hongkong)

La taille convient à la Raspberry et la capacité de stockage suffit.

➢ Deux batteries de moto  ---------- 10€ pièce (ebay Hongkong)
Ce sont des batteries de 12V communes, bon marché et faciles à transporter.

➢ Deux Ubec 12V/5V  ---------- 3€ pièce (ebay Hongkong)
L'Ubec est un transformateur et un stabilisateur de tension. Pour pouvoir alimenter la Raspberry avec la batterie, on doit intercaler cet Ubec entre les deux. Celui-ci se connecte au moyen de ports JST (voir **Annexe 1**).

➢ Deux ventilateurs  ---------- 5 € pièce (magasin de bricolage)
Un ventilateur est nécessaire pour éviter une surchauffe de la Raspberry à cause du climat.

➢ Autres composants ---------- 20€ (récupération ou magasin de bricolage)
Dans cette catégorie figure une embase récupérée au centre IGN mais qui s'avère être la seule concession onéreuse pour une une personne extérieure à l'IGN.

II. Préparation des composants

1) Raspberry Pi

La Raspberry est un petit ordinateur moins puissant que les tours d'ordinateur mais qui suffit pour notre projet. Pour la faire fonctionner il est nécessaire de lui fournir une carte micro-SD qui contient un OS (comme Windows ou Ubuntu pour les ordinateurs classiques). Notre Raspberry étant une Raspberry Pi3, elle venait avec Bluetooth et Wifi intégrés out-of-the-box. Ce modèle est donc beaucoup plus simple à mettre à jour et à utiliser avec un caster NTRIP. L'OS choisi a été le grand classique Raspbian une version de Debian optimisée pour la Raspberry.

Pour la première Raspberry, nous avons installé le système d'exploitation Raspbian, les pilotes de l'écran tactile et Qt5 (et donc PyQt5 pour python), qui est une bibliothèque graphique, séparément, ceux-ci n'étant pas fournies dans le même package. Par contre, pour la deuxième Raspberry nous avons téléchargé un seul package avec tous les pilotes, il suffisait d'exécuter les commandes nécessaire pour l'installation de chaque pilote.

2) Ecrans tactiles

L'un des écrans tactiles avait déjà été configuré pour une Raspberry. Il restait donc à configurer le second écran pour l'autre Raspberry.

Pour installer l'écran tactile nous avons exécuté les commandes ci-dessous fournies par le constructeur.

```
git clone https://github.com/goodtft/LCD-show.git
chmod -R 755 LCD-show
cd LCD-show/
sudo ./LCD5-show
```

On procède à la configuration des paramètres des Raspberry à l'aide de Raspi-config. Les Raspberry redémarrent alors automatiquement et leurs écrans tactiles respectifs sont configurés.

3) Antennes et GNSS

Le récepteur GNSS étant séparé des autres composants nous avons décidé de le tester séparément des autres composants pour vérifier son bon fonctionnement. Le récepteur GNSS peut être connecté à un ordinateur grâce à un câble ordinaire universel de téléphone portable ainsi qu'a une antenne. Une fois connecter à l'ordinateur le logiciel U-center développé par U-blox va nous permettre de le détecter, de le paramétrer et d'acquérir des données selon le mode qui nous convient.

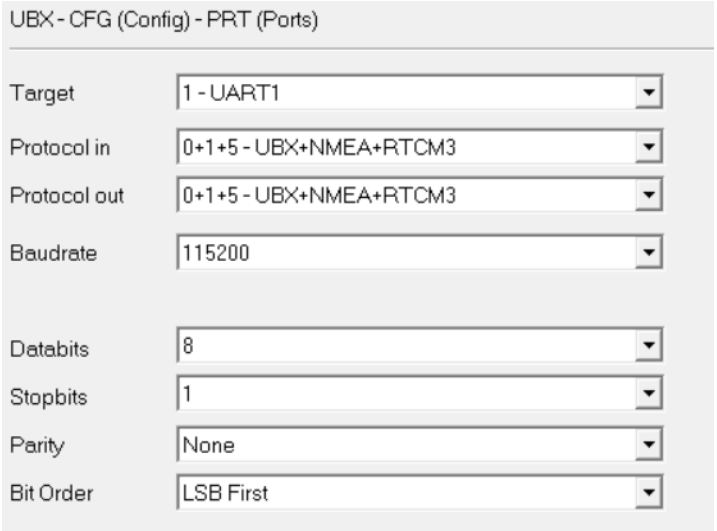
Pour paramétrer le récepteur GNSS il faut :

- Ouvrir le logiciel U-center téléchargeable ici : *https://www.u-blox.com/en/product/u-center-windows* ;

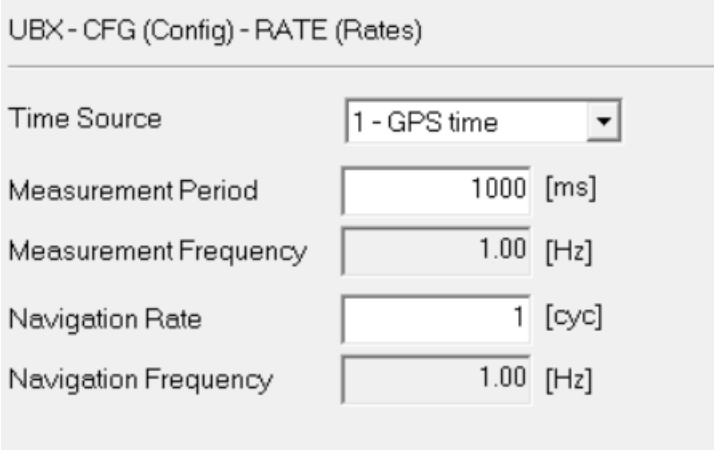
- Se connecter au port qui correspond à votre récepteur GNSS sélectionner : 'Receiver' -> 'Port' Si vous êtes connecter au bon port les fenêtres à droite de votre écran afficherons les GNSS vu par votre antennes ainsi que d'autre informations ;

- Pour afficher la fenêtre de paramétrage du récepteur GNSS appuyer sur les touches 'ctrl' + 'F9' ;

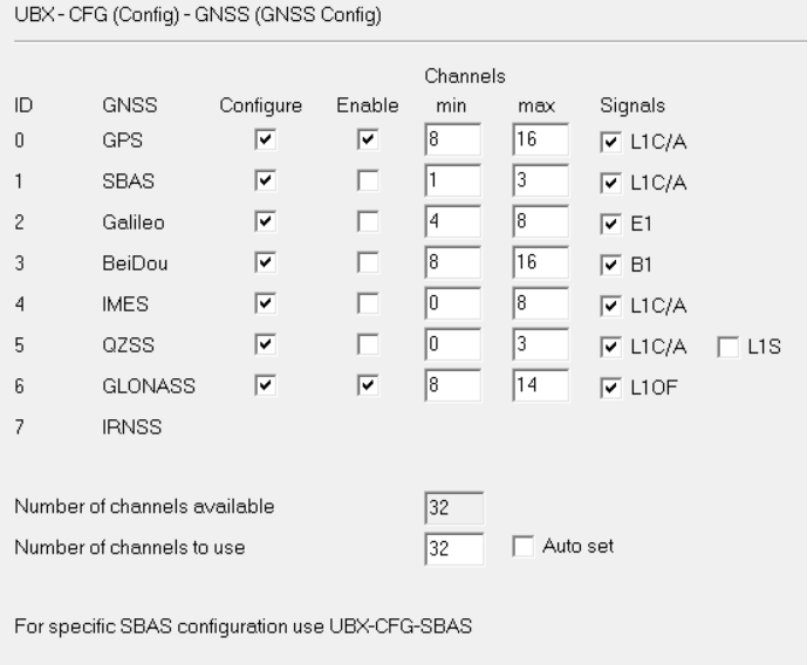
- Sélectionner l'onglet PRT (Port)



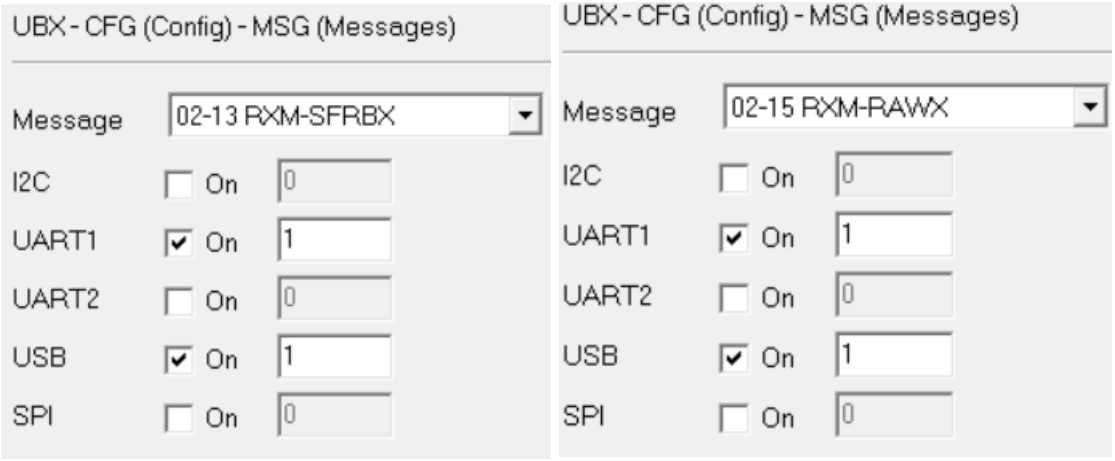
- Sélectionner l'onglet RATE (Rates)



- Sélectionner l'onglet GNSS (GNSS Config)



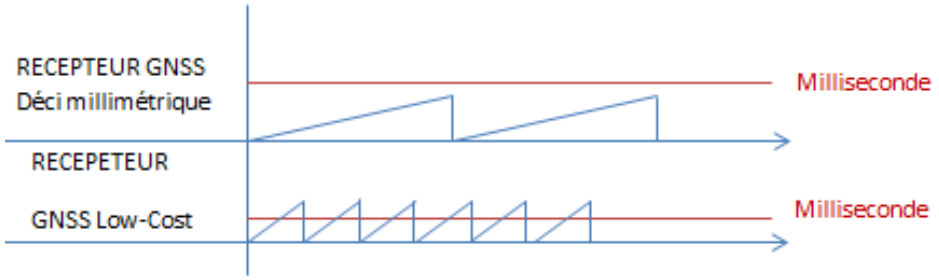
- Sélectionner l'onglet MSG (Messages)



A chaque onglet ne pas oublier de cliqué sur 'send' afin de d'enregistrer les paramètres choisis puis aller sur l'onglet CFG(Configuration) et appuyer une dernière fois sur 'send'. Pour plus d'informations : *https://github.com/Francklin2/RTKLIB_Touchscreen_GUI/wiki/6-Setting-the-Ublox-M8T-for-RAW*

Une fois le paramétrage fini nous pouvons passer aux tests. Deux tests ont permis de vérifier que l'antenne et le récepteur GNSS marchaient.

Le premier test fut de brancher le récepteur GNSS à l'antenne du centre IGN, pour que les données soient reçues par le récepteur de l'école ainsi que notre récepteur GNSS nous avons utilisé un spliter (voir annexe). Après traitement des données (voir **IV.2)**) nous avons vérifié que notre paramétrage était correct et que le récepteur GNSS fonctionnait bien. Grâce à ce test nous avons pu remarquer que les récepteurs GNSS low-cost avaient un asservissement supérieur à la milliseconde et une dérive plus forte que les GNSS ayant une précision au déci-millimètre :



Comparaison de la précision entre le récepteur low cost et un appareil professionnel

Nous n'avons pas pu résoudre ce problème mais il semble qu'il n'ait pas un impact assez grand pour biaiser nos données. Nous obtenons des données dont les coordonnées sembles précisent à quelques centimètres près (maximum 5).

Une fois que nous avions paramétré le récepteur GNSS et vérifié la qualité des observations avec une antenne de haute précision nous avons pu passer au test avec notre antenne low-cost. Nous avons stationné un point connu du centre IGN avec l'antenne low-cost et le récepteur GNSS dans une boite en bois et l'ordinateur placé en contrebas de l'installation. Ce test nous permis de vérifier que l'antenne et le récepteur GNSS marchaient et nous donnait des données précises au centimètre près.

4) NTRIP

Le positionnement en temps réel statique nécessite un échange de données entre la base et le mobile. La majorité des cas, cette liaison est assurée par onde radio.

N'ayant pas reçu le matériel radio avant le début du projet, nous avons étudié différents moyens de transmission (filaire, Bluetooth) avant de choisir le NTRIP.

Le NTRIP est basé sur une transmission de donnée par internet. Il nous affranchit donc des problèmes de distance.

Dans le procédé du NTRIP :

- L'appareil qui acquiert les données est appelé le **serveur**
- L'appareil qui reçoit les données est appelé **client**
- L'appareil qui permet la transmission de ces flux de données est appelé le **caster**

Remarque : on note une incohérence dans les notations du NTRIP puisque matériellement, la caster est un serveur.

Pour le positionnement en temps réel, la base joue le rôle du serveur NTRIP et le mobile celui du client NTRIP.

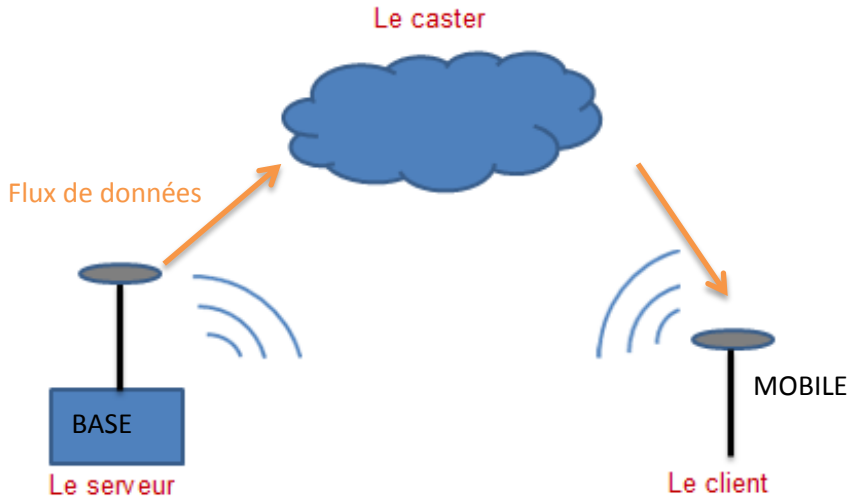


Schéma du positionnement en temps réel par NTRIP

La connexion à un caster nécessite un identifiant, un mot de passe, l'IP du caster, son port et un point de montage.

Pour nos tests, nous avons utilisé le caster NTRIP du RGP de Saint-Mandé qui nous a ouvert sur demande 3 points de montages FORC0, FORC1 et FORC2. Cette solution privilégiée nous a été très utile pour les tests mais doit être remplacée pour l'usage quotidien des GNSS. Une des possibilités peut être l'utilisation de Caster NTRIP gratuit tel que SNIP (*www.use-snip.com*).

III. Fabrication des GNSS

1) Programmation

Nous voulions tout d'abord utiliser un programme réalisé par Franklin N'Guyen et des élèves du Master PPMD de l'ENSG (lien vers le Github qui répertorie ce programme : *https://github.com/Francklin2/RTKLIB_Touchscreen_GUI*), mais face aux bugs et à la difficulté que cela représentait de les corriger, nous avons choisis de partir sur une solution en python.

Le programme utilisé, TouchRTKStation (lien vers le Github qui répertorie ce programme : *https://github.com/taroz/TouchRTKStation*), a été inspiré par le programme de Mr N'Guyen. Il diffère de ce dernier par le fait qu'il est codé en Python et qu'il utilise la version compilée de RTKlib. Python est un langage beaucoup plus clair et facile à modifier que le C++ (que nous ne connaissons pas) pour nous, nous avons donc pu corriger quelques bugs. Nous avons également apporté quelques modifications ergonomiques.

Un premier souci que nous avons rencontré était la difficulté à lire caractères trop petits sur l'écran de 4 pouces sur lequel se reflétait le soleil. Nous avons également ajouté un clavier virtuel codé en python avec PyQt5 pour écrire dans les zones de texte avec une solution en Python. Enfin, le logiciel ne disposait pas de fichier de configuration, les paramètres étaient écrits en dur dans le programme. Pour éviter d'avoir à retaper à chaque test les données à indiquer en entrée, nous avons ajouté un fichier lu au démarrage et éditable contenant ces informations.

Le programme de Taro Suzuki nous a permis de tester notre montage et de faire de nombreux tests. Lors de ces tests nous avons rencontré des problèmes, l'absence de clavier virtuel et l'obligation de retaper à chaque lancement tous les paramètres entre autres. Nous avons donc pu dresser une liste des fonctionnalités attendues classées d'après leur importance pour un éventuel projet reprenant notre montage pour recoder l'interface et le fonctionnement interne.

Ci-dessous en vert figurent les fonctionnalités présentes dans le code original.

Fonctionnalités primaires :

- Posséder les modes temps-réel statique, temps-réel cinématique, statique post-traitement

- Respecter une ergonomie de terrain

- Échanger des données entre la base et le rover grâce au NTRIP

- Être configurable (et que les options soient sauvegardables)

- Affichage des informations sur l'acquisition en cours

Fonctionnalités secondaires :

- Contrôler la Wifi (depuis le programme)

- Extinction de la Raspberry

- Export du fichier par mail ou sur drive

Nous avons fait une première implémentation des fonctionnalités les plus importantes pour l'utilisation du GNSS (clavier virtuel, changement de la taille de la police, bouton pour éteindre) mais ce n'est qu'une implémentation partielle et il manque complètement certaines des plus importantes. Nous ne disposions pas du temps pour refaire le programme de zéro et le rendre conforme aux attentes d'une personne qui souhaiterait l'utiliser pour faire de l'acquisition de points.

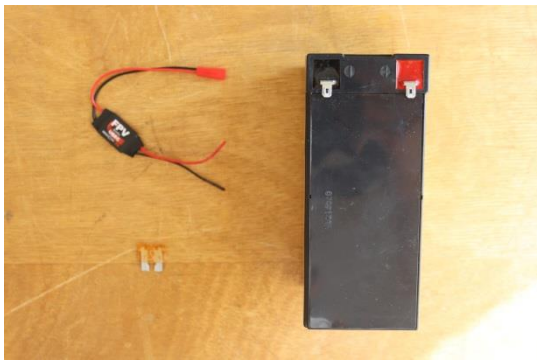
2) Adaptation de l'alimentation

Alimentation par batterie

L'objectif est de passer d'une tension de 12V délivré par la batterie à une tension 5V adapté à l'alimentation de la Raspberry. Nous utilisons donc un Ubec car il sert à la fois de transformateur et de stabilisateur de tension.

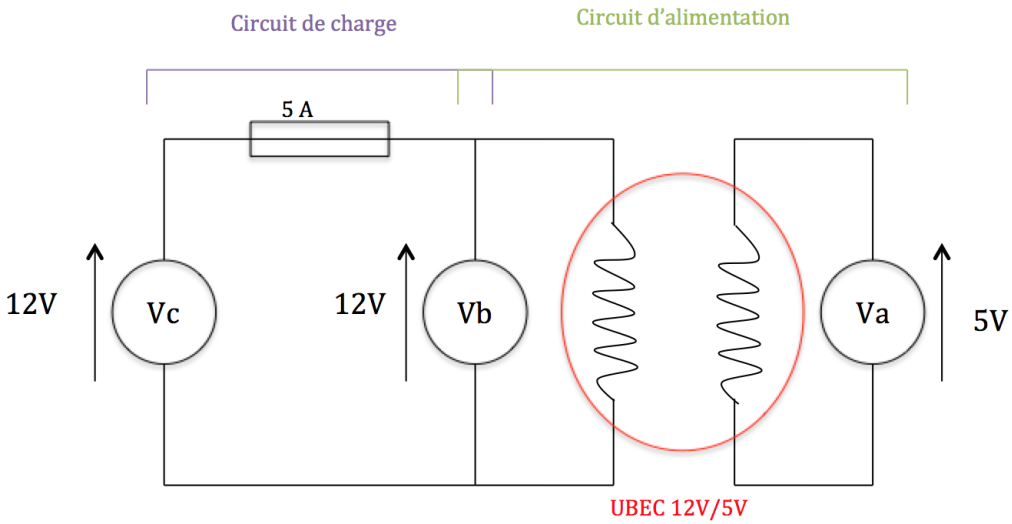
Liste du matériel :

- Une batterie 12V (récupérable sur moto, tondeuse, …)
- Un fusible 5A
- Un UBEC
- Fils, cosses



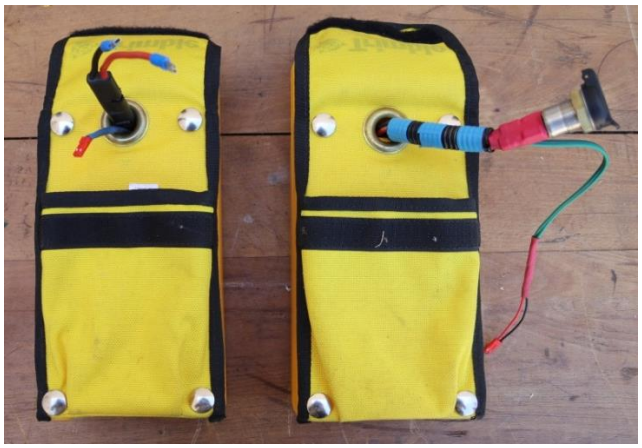
Ubec (en haut à gauche), fusible (en bas à gauche) et batterie (à droite)

On réalise le montage suivant :



Va : tension d'alimentation
Vc : tension de charge
Vb : tension de la batterie

Schéma du circuit électrique au sein de la boîte jaune



Alimentations par batterie

Remarque : le montage permet de recharger facilement la batterie avec un chargeur de batterie 12V.

Alimentation sur secteur

Afin de pouvoir alimenter les récepteurs sur secteur sans avoir à les démonter, nous avons également remplacé les ports micro-USB mâles des chargeurs des Raspberry (déjà en 5V) par des ports JST mâles identiques à celles des batteries.

Chargeur de Raspberry modifié

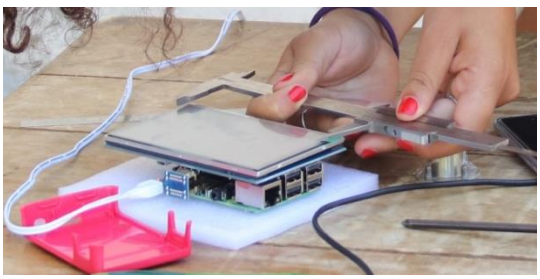
Remarque : Afin de ne pas griller la Raspberry, il est impératif de tester la tension en sortie de chaque moyen d'alimentation avec un multimètre.

Le boitier

Lors de la construction du boitier, nous avons ajouté des interrupteurs sur les circuits d'alimentation de la Raspberry et d'alimentation du ventilateur (5V également) dans le but d'économiser la batterie.

3) Assemblage des composants

On peut commencer par fixer l'écran tactile au couvercle de la boîte. Pour cela, il est nécessaire de mesurer les dimensions de l'écran et du couvercle avec précision. Un pied à coulisse ou un réglet est donc recommandé.



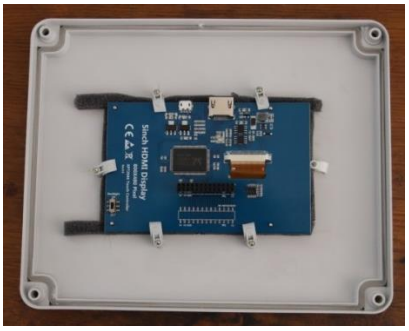
Mesure des dimensions de l'écran avec un pied à coulisse

On trace ensuite sur le couvercle la zone à découper.

Pour le découpage, il convient de faire quelques trous à la perceuse puis d'utiliser une scie pour découper à l'intérieur des traits.

Attention : Lors de la découpe, On ne doit pas déborder du rectangle dessiné. Le mieux est de vérifier régulièrement si le trou est adapté à l'écran et de le corriger à la lime si besoin.

On peut à présent fixer l'écran. Afin de préserver au mieux l'étanchéité de la boîte, il conviendrait d'entourer le trou de joint en mousse avant de fixer l'écran avec des attache-fils.



Couvercle de la boîte après fixation de l'écran

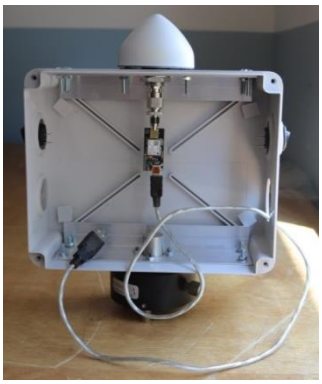
Le couvercle est prêt, nous travaillerons désormais exclusivement sur le fond de la boîte.

Pour commencer, il faut visser une barre de métal à l'intérieur du boîtier sur la partie inférieure et en percer le centre pour y visser le pas de vis de l'embase. L'intérêt est de stabiliser le pas de vis et d'apporter de la rigidité à la boîte.

Trous pour la ventilation (à gauche) et pas de vis de l'embase (à doite)

Dans l'optique de faire passer des fils et d'intégrer un système de ventilation, nous avons percé quelques trous dans la plaque de métal de part et d'autre du pas de vis.

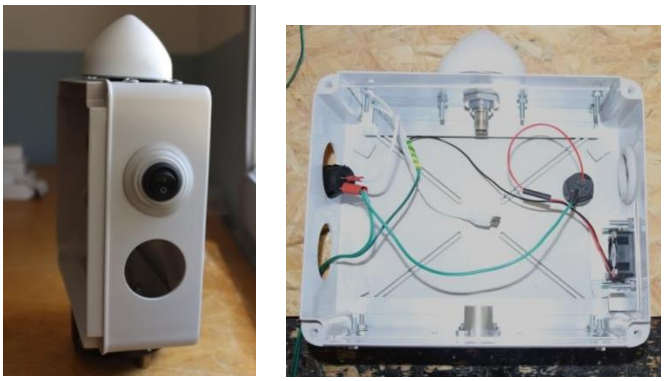
A présent, il s'agit de fixer l'antenne. Il suffit de placer une barre de métal à l'intérieur du boîtier sur la partie supérieure et d'y faire un trou au centre, toujours pour apporter stabilité et rigidité à l'ensemble. L'antenne peut y être vissée.



Aperçu de l'agencement de l'antenne et de la puce GNSS dans le boîtier

La dernière étape de la fabrication concerne le câblage : nous pouvons déjà placer deux interrupteurs sur le boîtier. L'un servira à allumer la Raspberry et l'écran, l'autre contrôlera le ventilateur. Nous avons placé un interrupteur sur chaque côté de la boîte, plus proches de l'antenne que de l'embase. Cette disposition permettra d'optimiser l'agencement des composants, en plaçant le ventilateur sous son interrupteur et les câbles reliés à la Raspberry du côté de son interrupteur. On peut également regrouper et fixer au boîtier les câbles d'un même interrupteur avec des fixe-câbles auto-adhésifs.

A propos des câbles reliés à la Raspberry, rappelons que le câblage de l'alimentation doit être réalisé suivant le schéma présenté dans la partie **I.2)**.

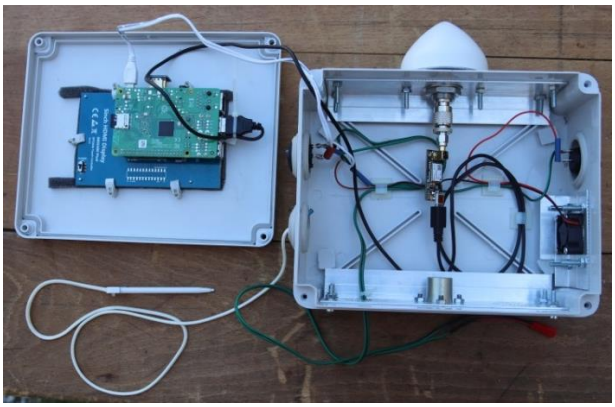


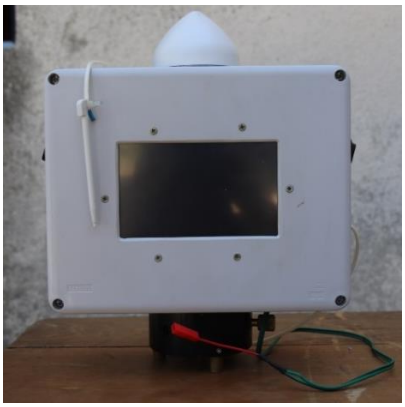
Placement de l'interrupteur (à gauche) et câblage interne (à droite)

Le ventilateur devrait être fixé sur une plaque en métal car il est plus petit que les ouvertures de la boîte, et branché à la Raspberry pour être alimenté. Avec les trous au fond la boîte, il servira à générer un flux d'air au sein du boîtier pour éviter la surchauffe de la Raspberry et les dépôts de poussière.

Attention : Le ventilateur doit être branché en mode extraction pour faire pénétrer moins de poussière dans le boîtier.

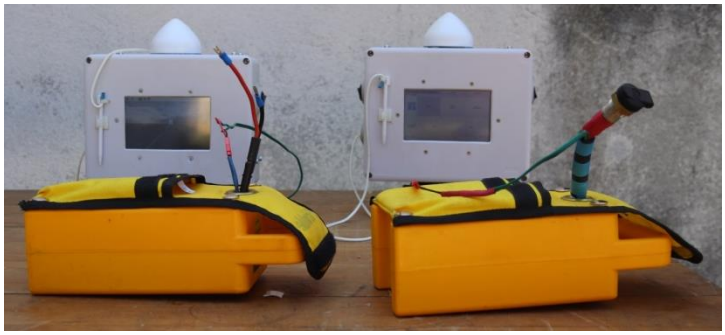
Il ne reste plus qu'à relier les différents composants, fermer le boîtier et vérifier que le récepteur fonctionne.





Récepteur GNSS complet (ouvert à gauche, fermé à droite)

On peut éventuellement ajouter un stylet.



Récepteurs GNSS prêts pour les tests finaux

VI. Essais des récepteurs GNSS

1) Acquisition sur le terrain

Lorsque la construction du boitier et les tests entre l'antenne low-cost et la puce GNSS fut finie nous avons pu passer aux tests avec le boitier complet ainsi que la batterie portable. TouchRTKStation nous permet d'utiliser différents modes d'acquisitions :

- Le mode single, qui permet d'obtenir des coordonnées en temps réel précises à une dizaine de mètres près (utilisé dans les téléphones portables). Pour notre projet ce mode n'est pas intéressant ;

- Le mode statique, qui nous permet ici d'obtenir des coordonnées plus précises que le mode single (ici infra-décimétrique) d'un point en le stationnant un certain temps.

- Le mode temps réel statique, qui permet d'obtenir des coordonnées d'un point stationné en temps réel aussi précises que celle obtenues en mode statique (grâce à l'utilisation du Ntrip ici).

- Le mode temps réel cinématique, qui permet d'obtenir les différentes coordonnées d'une trajectoire, grâce au mode Ntrip ici. Ce mode peut être utilisé en voiture, vélo, bateau, …

- Le mode base, qui permet au GNSS d'être une base, pour le Ntrip dans notre cas.

Nous avons pu tester uniquement les modes statique et temps réel statique par manque de temps (Voir **Annexe 3**).

Test mode statique :

Le premier test fut réalisé avec les deux récepteurs, un sur chaque pilier du centre IGN de Forcalquier. Nous avons obtenu des données incohérentes et précises à plusieurs mètres près. Pour déterminer l'origine de cet inconvénient, nous avons fait une série de différents tests.

Premier test

Pour savoir si les données reçues par l'antenne étaient perturbées par des interférences, nous avons fait trois tests. Pour ces trois tests nous avons branché l'antenne low-cost au récepteur GNSS lui-même branché à l'ordinateur. Nous avons utilisé le logiciel U-Center. Pour le premier test nous avons utilisé uniquement la boite en plastique, pour le deuxième nous avons allumé la rasberry et le dernier nous avons allumé la rasberry et l'écran tactile. De cette manière nous pouvions vérifier quel composant perturbait l'acquisition. D'après ces tests l'écran serait un des facteurs qui perturberait l'acquisition. La rasberry perturberait la composante altimétrique. Nous obtenons des résultats précis à 60 cm près.

Un deuxième problème est survenu aussi au cours du premier test, il concernait le script Python et était dû à l'absence de paramétrage de la puce GNSS. Comme vu dans la partie **II.3)** un paramétrage rigoureux est nécessaire. Pour régler ce problème nous avons créé un fichier texte comprenant les paramètres les plus importants. Puis nous l'avons implémenté dans le script python.

Grâce à toutes ces modifications, nous avons constaté qu'en stationnant au moins 2h sur le point nous obtenons des coordonnées planimétriques précises au centimètre près mais des coordonnées altimétriques précises à 40 cm près.

Test mode temps réel statique :

Après avoir fini les tests pour le mode statique nous avons pu essayer le mode temps réel statique en utilisant le Ntrip. Nous n'avons pas réussi avec ToucheRTKStation à nous connecter au caster. La cause serait principale serait la mauvaise connexion internet mais nous n'avons pas pu le temps de vérifier ou trouver les autres problèmes.

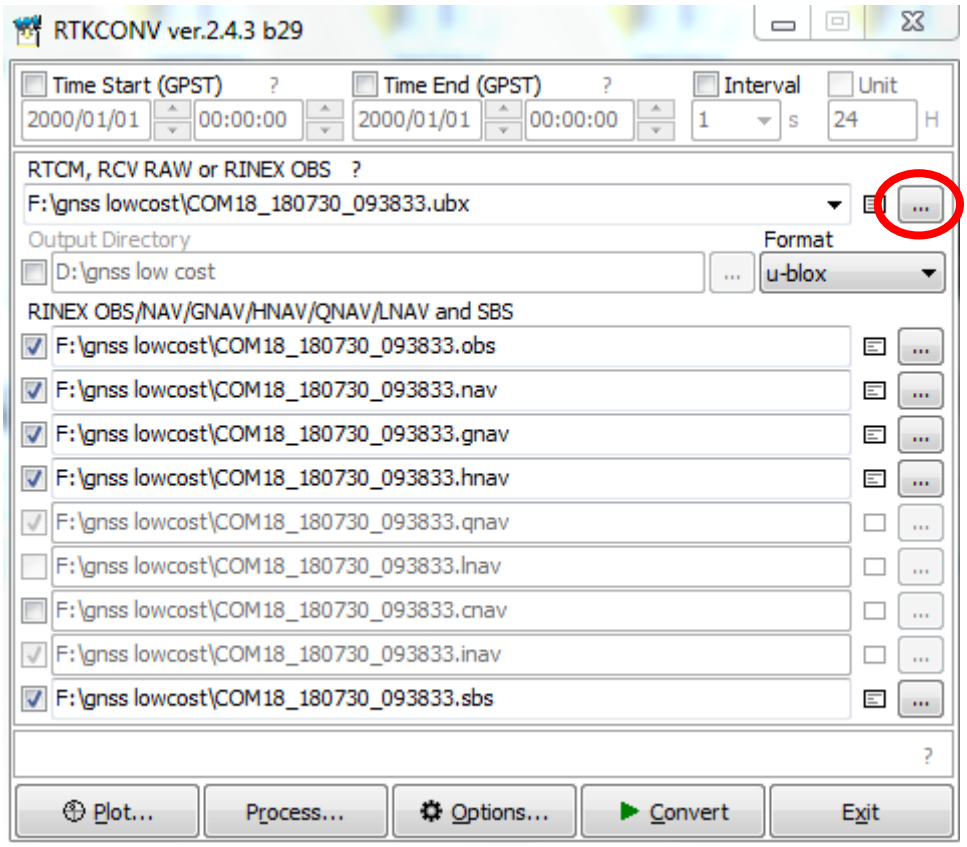
2) Traitement des données

Mode Static :

Après l'acquisition des données sur le point stationné par le GPS, nous obtenons un fichier .ubx (format binaire propriétaire d'Ublox). On convertit tout d'abord le fichier .ubx en fichier Rinex afin qu'il soit pris en charge par les logiciels de traitement de données GPS/GNSS notamment par le calcul GNSS en ligne que nous allons exécuter par la suite sur le site :

http://tychobrahe2018.ensg.eu/

Nous procédons à la conversion à l'aide de l'onglet RTKCONV dans le logiciel RTKLIB. Le fichier .ubx doit être importé avec le bouton entouré en rouge sur l'image ci-dessous, en modifiant le format en u-blox.



On lance la conversion, ce qui nous génère un fichier .obs en Rinex avec toutes les observations. Le fichier en sortie est trop volumineux pour être traité par le calcul en ligne, le logiciel teqc est alors l'outil le plus adéquat pour résoudre de nombreux problèmes de prétraitement avec les données GPS.

Pour ce faire, il faut ouvrir une fenêtre de commande depuis le dossier contenant le fichier Rinex et taper la ligne de commande suivante :

teqc −O.obs C1L1S1D1 -S -E −O.dec 15 fichierenentree.obs > fichierensortie.obs

Cette ligne de commande permet de supprimer les données non nécessaires, à savoir GPS, BLEIDOU et GALILEO ('-S –E –O').

Comme l'acquisition est faite chaque seconde, le fichier en sortie ne contiendra que les enregistrements effectués à 15 secondes d'écart.

Après ce traitement, le nouveau fichier .obs est moins volumineux, il peut donc être traité sur le calcul en ligne, qui nous renvoie par mail les coordonnées du point sur lequel a stationné le GPS en Lambert93 et en données cartésiennes.

Pour vérifier la cohérence des données obtenues, nous pouvons stationner un point connu du centre IGN puis comparer les coordonnées reçues aux vraies coordonnées du point sur le site suivant :

ftp://tychobrahe2016.ensg.eu/

Mode RTK :

Pour ce mode, les fichiers sont traités automatiquement. Il suffit d'ouvrir le fichier .pos et de vérifier les coordonnées.

3) Pistes d'amélioration

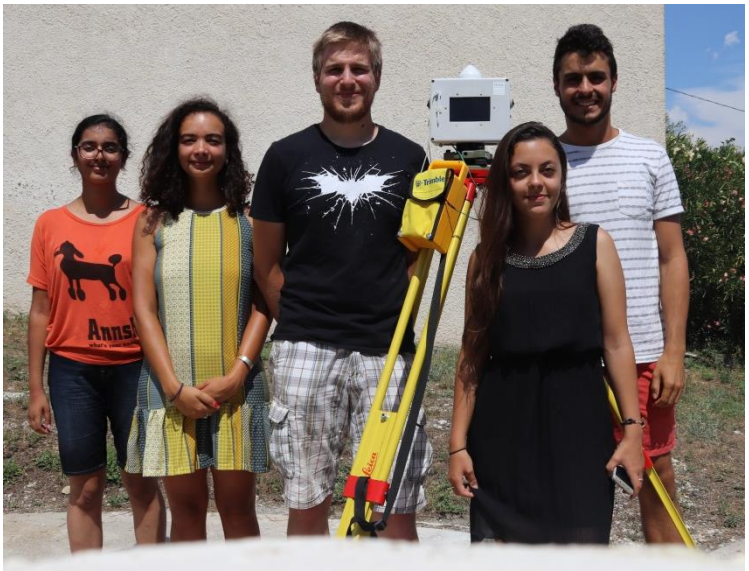
Les tests obtenus ont montrés que certains des composants utilisés pourraient ne pas convenir pour la fabrication d'un récepteur GNSS :

- L'écran tactile impacte fortement les données reçues par l'antenne, il serait nécessaire de le remplacer ou de trouver le moyen de supprimer cet impact.
- Il est possible que l'antenne et/ou la puce GPS soi(en)t trop sensible(s) aux interférences.
- De manière générale, tout composant électrique peut causer des écarts de mesures : le ventilateur, une mauvaise soudure,…Un travail plus professionnel que le nôtre pourrait au moins réduire les erreurs de mesure.
- On peut ajouter un filtre sur les trous de la ventilation pour réduire encore la quantité de poussière entrant dans la boîte.

En termes d'ergonomie, il faut être en mesure de modifier la taille de police d'écriture, la luminosité et la résolution de l'écran qui est difficile à lire en extérieur.

Conclusion

Avec du matériel de récupération ou bon marché, le prix d'un seul de nos récepteurs GNSS est cinquante fois inférieur à celui d'un appareil professionnel et ce pour une précision adéquate, sachant que des alternatives moins onéreuses de certains matériaux sont envisageables.



Annexe

Annexe 1

Ports JST mâle (à gauche) et femelle (à droite)

Annexe 2

Clavier Numérique en PyQt5 :

Notre GNSS low cost a pour interface un écran tactile. Il en découle des difficultés pour écrire les mots passes adresse et autres actions nécessitant un clavier. En effet, la principale (et seule solution) consistait à utiliser un programme linux émulant le comportement d'un clavier, Matchbox-Keyboard. Cette solution apportait autant de solution que de problèmes, le lien entre le code python et le clavier aurait été fait par l'appel de commande mais nous ne disposions pas de contrôle sur son emplacement.
Nous avons donc choisi de codé un clavier en python en utilisant la bibliothèque graphique PyQt5. Pour rappel PyQt5 est une bibliothèque python ayant pour but de faire l'interface entre python et une bibliothèque C++, Qt, qui est une des bibliothèques les plus importantes dans le secteur (utilisée par les Smart TV notamment).
La première solution était de créer un clavier avec des touches mis en brute dans le code. Elle convenait dans un premier temps au vu des faibles demandes du projet. Nous avons décidé de le pousser plus loin sous forme d'un plug-in pour PyQt5 qui fournirait un système implémentant le clavier à l'aide de fichier .xml contenant toute les informations nécessaires pour générer avec le même code des pavés numérique, des claviers qwerty et azerty ou des claviers personnalisés. Tout cela avec en tête l'adaptabilité et la possibilité de le réutiliser dans de nombreux projets. Les fichiers nécessaires pour générer les différents claviers sont du même format que les claviers de Matchbox-Keyboard.

Lien vers le Github recensant le code du clavier :
https://github.com/Syn-crow/PyQt5_Keyboard/blob/master/README.md

Annexe 3

	Type boitier	Type Antenne	Type Acquisition	précision
Ordinateur (U-center)	Boitier en bois	Antenne Low-cost	Statique	1 à 5 cm
	Boitier raspberry sans raspberry	Antenne Low-cost	Statique	1 à 5 cm
	Boitier raspberry avec raspberry sans écran	Antenne Low-cost	Statique	Coordonnées planes :1 à 5cm Composante verticale : 30 cm
	Boitier raspberry avec raspberry avec écran	Antenne Low-cost	Statique	40 à 60 cm
	Aucun	Antenne Forcalquier	Statique	1 à 5 cm

	Type boitier	Type Antenne	Type Acquisition	précision
Raspberry (TouchRTKStation)	Aucune	Antenne Forcalquier	Statique	1 à 5 cm
	Boitier raspberry	Antenne Low-cost	Statique	Coordonnées planes :1 à 5cm Composante verticale : 30 cm
	Boitier raspberry	Antenne Low-cost	Ntrip statique	Pas testé
	Boitier raspberry	Antenne Low-cost	Ntrip cinématique	Pas testé
	Boitier raspberry	Antenne Low-cost	Base Ntrip	Pas testé

Annexe 4

Lien vers le GitLab du projet : https://gitlab.com/jbeilin/TouchRTKStation