**Cuadros Alexandre**
**Taffin Valentin**
**IMA 5**

# *PRE-SOUTENANCE PFE*

*«Réaliser deux trackers GPS permettant de suivre à distance le trajet d'un coureur »*

*Années 2016-2017*

# Contexte

Lors des marathons ou des courses natures (trails), les supporters aimeraient connaître la position de leur coureur favori. Pour cela, deux solutions existent : l'utilisation de programmes directement sur téléphone ou l'utilisation de matériel spécifique. Le projet consiste à concevoir deux trackers, un pour téléphone et un autonome.

Concernant le tracker du téléphone, nous pouvons utiliser le capteur GPS interne de l'appareil en ajoutant une carte d'extension afin d'y ajouter les capteurs souhaités :

- Bouton poussoir
- Accéléromètre
- etc..

Pour la version autonome, il faudra envisager les deux solutions que sont un module 2G et un module LORA tout en portant une attention toute particulière sur la consommation énergétique et sur les poids des dispositifs.

# SOMMAIRE:


## I : Cahier des Charges

-le module LoRa
-le module 2G
-Présentation du Tracker Smartphone
-Présentation du Tracker Autonome

## II : Travail effectué

-Partie serveur et Web
-Partie Hardware pour la partie Tracker Smartphone
-Partie Hardware pour la partie Tracker autonome

## III : Travail restant

# I : Cahier Des Charges

## le module LORA :

L'Alliance LoRa est un réseau étendu de faible puissance (LPWAN) exploitable avec une batterie sans fil adapté pour le réseau régional, national mais aussi mondial.

Le Réseau LoRa se base sur le protocole LoRaWAN (Long Range Wide-area ou réseau étendu de longue portée) qui est peu énergivore.

LoRaWan vise les exigences clés de l'internet des objets IOT tels que les services de communication bi-directionnelle, la mobilité et de localisation sécurisée.

L'architecture du réseau LoRaWan est utilisée sous forme de réseau hiérarchique où chaque passerelle peut transmettre les messages entre les appareils terminaux et un serveur de réseau central en arrière-plan.

Les passerelles sont connectées au serveur de réseau via une IP standard tandis que les appareils terminaux utilisent la communication sans fil à une ou plusieurs passerelles.

La communication entre les appareils terminaux et les passerelles est répartie sur les différents canaux de fréquence et des débits de données (en France 863 à 870MHz MHz, au US 902 à 928 MHz et Chine 779 à 787 MHz ).

Grâce à la technologie à étalement de spectre, les communications avec les différents débits de données ne se gênent pas et créent un ensemble de canaux «virtuels» de plus en plus performant.

Le réseau LoRa a un débit compris entre 0,3 à 50 kbps. Le débit (et la puissance d'émission) s'adapte automatiquement selon les besoins des objets, afin de limiter la bande passante et donc la consommation d'énergie. La portée du réseau est satisfaisant puisqu'elle est d'environ 20km en zone rurale et à 2km en zone urbaine.

Pour maximiser la durée de vie de la batterie des appareils terminaux et de la capacité globale du réseau, le serveur de réseau lorawan gère le débit de données et de sortie RF pour chaque dispositif individuellement au moyen d'un système de débit de données adaptatif (ADR).

## Le module 2G :

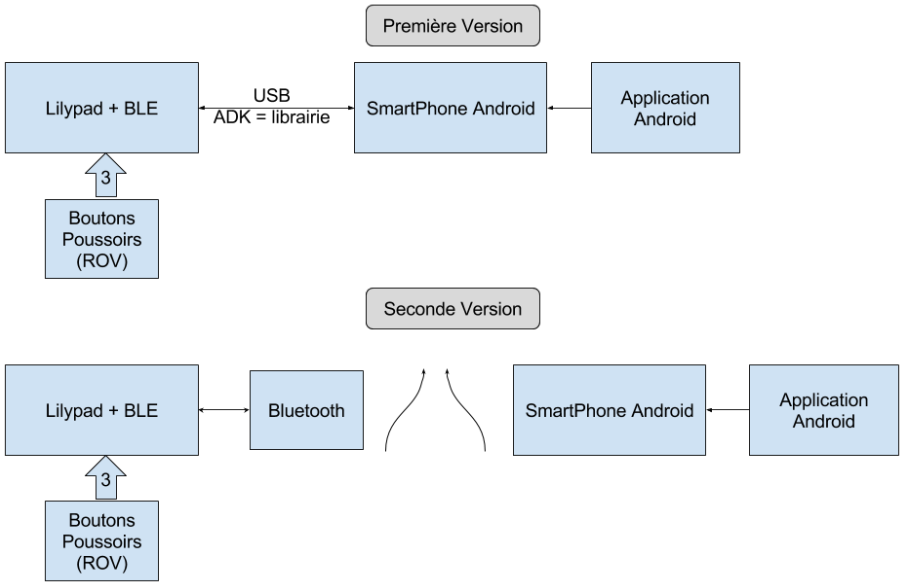
Le module 2G ou GSM (Global System for Mobile Communications) est une norme numérique de seconde génération pour la téléphonie mobile s'appuyant sur les transmissions numériques permettant une sécurisation des données. Le module 2g est un réseau longue portée (de quelques kilomètres en ville à 30 km en zone rurale) et consommateurs d'énergie. Cette technologie était principalement utilisé pour passer des appels téléphoniques. Néanmoins, L'utilisation de ce module est intéressant puisqu'elle apporte une meilleure qualité ainsi qu'une plus grande capacité à moindre coût pour l'utilisateur.

## Présentation du Tracker Smartphone :

L'Arduino et l'Android sont deux technologies très puissantes et ouvertes. Il peut être très intéressant de les connecter afin de concevoir le tracker du téléphone.
Pour cela, nous pouvons ajouter des capteurs sur une carte Arduino. Ensuite, nous pouvons programmer le microcontrôleur pour que les capteurs puissent envoyer un signal au téléphone. Le résultat pourra être récupérer à distance à l'aide un site web en utilisant le GPS interne du téléphone.

Description Fonctionnelle :

- Utilisation de la lilypad pour pouvoir envoyer les informations souhaitées par Bluetooth Low Energy
- Utilisation d'un Smartphone pour :
    - Récupérer les informations de la lilypad
    - Récupérer les données GPS du Smartphone
    - Envoyer les données GPS par Wifi
    - Récupération des données par un serveur et les traiter




## Présentation du Tracker autonome :

La partie tracker autonome consiste à réaliser un tracker gps de type "www.capturs.com" et d'une application web afin de visualiser en direct le suivie d'un coureur ou tout simplement accéder à l'historique de la course (temps, altitude ...).

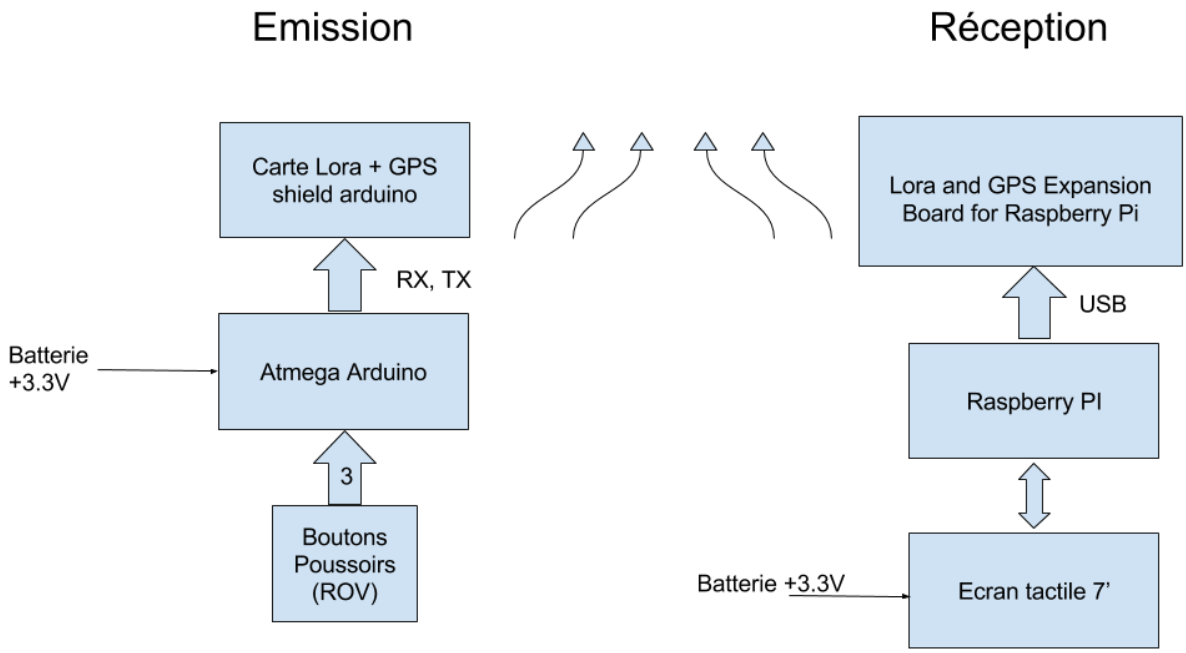
De plus le coureur a la possibilité d'envoyer des messages "pré-enregistrés" pendant sa course avec comme pour la partie précédente un bouton poussoir ( ou interface graphique ). Il nous faut :

- un moyen d'envoyer l'information que le coureur a appuyé sur le bouton poussoir au microcontrôleur (au moyen d'un fil ou d'un module Wifi/Bluetooth)
- bouton poussoir ou autre...

La course est un sport extérieur ce qui signifie que nous devons étudier et vérifier que le matériel utilisé est capable de résister à la température (>-0°), humidité et choc.

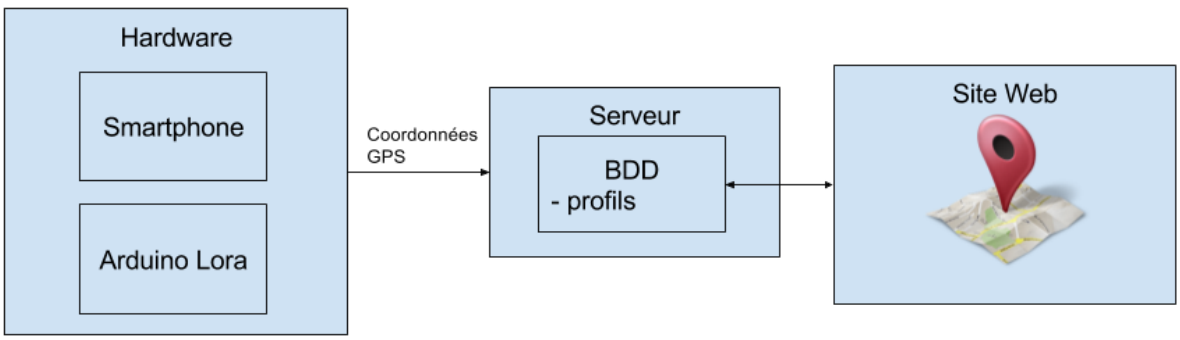
Description Fonctionnelle :

- Utilisation d'un module GPS A2200-A
- Utilisation de deux modules LoRa pour l'envoi des données
- Une batterie pour alimenter la carte électronique
- Utilisation de carte Arduino pour collecter les données dans un premier temps
- Récupération des données par un serveur et les traiter




# II : Travail effectué

Nous nous sommes principalement occupé de la partie “Application” du Tracker du Téléphone, mais qui pourra également être re-utilisé pour la partie autonome:




Celle-ci se décompose en plusieurs parties :

- Le côté hardware : notre smartphone ou notre carte arduino LoRa, ils transmettront leurs positions à intervalle régulier.
- Le côté serveur : reçoit, traite et stock les paquets envoyés par la partie hardware, puis les stocks à l'intérieur de notre base de donnée.
- Le site web : permet d'accéder aux différents profils des utilisateurs ,afin qu'ils puissent visualiser leur parcours, connaître leurs temps...

# Partie Serveur et Web :

## 1) Base de Donnée:

La base de donnée est réalisé avec mySQL, Le schéma ci dessous n'est qu'une ébauche de notre base de donnée et sera mis à jour suivant l'avancement de notre application. Celle-ci est découpé en plusieurs partie :

- "profil" qui contiendra toutes les informations du compte client.
- "point" qui correspond au coordonnée de géolocalisation.
- "itinéraire" composé d'une ensemble de points afin d'enregistrer des parcours prédéfinis.
- “profil_itineraire” qui permet de faire le lien entre un profil et des itinéraires.
- "positiongps", il sera renommé mais il sert à stocker la coordonnée GPS de notre client en temps réel.


## 2) Site Web:

Le site web est réalisé en PHP afin de récupérer les données sur la base de donnée, les technologies telles que le Javascript, AJAX, Json, Google API.

Page pour se connecter afin de récupérer les informations du client sur la base de donnée.


Dans la fenêtre Googlemap l'affichage des itinéraires en lien avec le profil client et une gestion de sa coordonnée gps en temps réel (on peut faire varier le temps de rafraîchissement). Un bouton déconnexion pour retourner à la page d'accueil.

Bienvenue dans votre espace utilisateur Valentin Taffin!

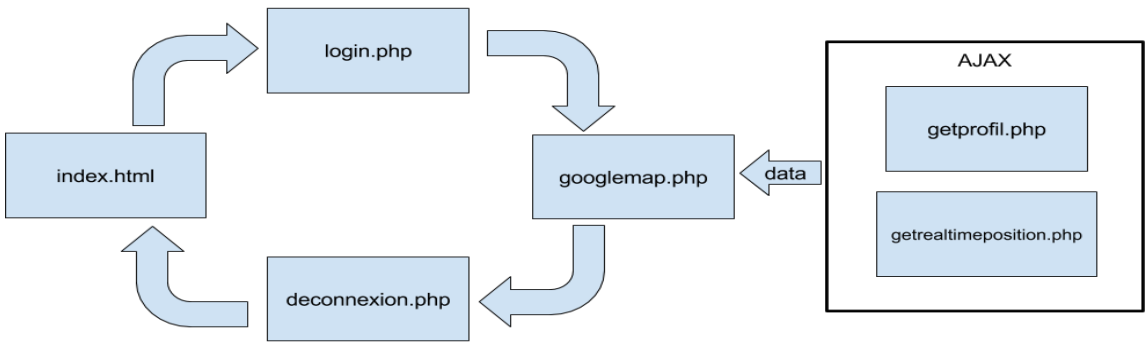
deconnexion

Bonjour Voici vos actualités récentes.

Parcours réalisé :


Affichage en temps réel de la position :


Voici le statemachine de notre application :




## 3) Serveur :

Le serveur est réalisé en nodeJS. Sont but est de se connecter à notre base de donnée et suivant la validation des paquets UDP qu'il reçoit il met à jour notre base de donnée.
Afin d'effectuer des opérations sur le serveur le contenu du paquet doit-être structuré de cette façon :

| Commande | Firstname | Lastname | Latitude | Longitude | arguments... |
|---|---|---|---|---|---|

Voici la commande utilisé pour tester l'envoie de paquets à notre serveur en local:

*echo -n "Valentin Taffin 50.630054 3.035401 null" >/dev/udp/127.0.0.1/8000*

Commande pour ajouter un point à un itinéraire :

*echo -n "AddPointToProfil Valentin Taffin itineraireID 50.630054 3.035401 null address" >/dev/udp/127.0.0.1/8000*

Après un envoi sur le serveur voici ce que l'on reçoit :

```
MacBook-Pro:Server Ovnistudio$ node server.js
server listening 127.0.0.1:8000
server got: Valentin Taffin from 127.0.0.1:61459
server got: Valentin Taffin from 127.0.0.1:51854
server got: Valentin Taffin 50.630054 from 127.0.0.1:53386
server got: Valentin Taffin 50.630054 3.035401 from 127.0.0.1:56693
server got: Valentin Taffin 50.630054 3.035401 from 127.0.0.1:51328
server got: Valentin Taffin 50.630054 3.035401 from 127.0.0.1:55717
```

## Partie Hardware pour la partie Tracker Smartphone :

La partie tracker Smartphone est intéressante puisqu'elle permet d'utiliser plusieurs technologies à la fois, c'est-à-dire la récupération de données GPS, le BluetoothLE, la wifi ainsi que la programmation sur Arduino IDE et Android Studio.

### 1) Partie mobile:

L'application mobile est faite à l'aide de l'environnement de développement "Android Studio". Elle est composée :

a) D'une classe permettant la récupération des données GPS

b) D'une classe regroupant la partie BluetoothLE

c) D'une classe permettant d'envoyer les données GPS au Serveur

Pour utiliser convenablement le logiciel, nous avons téléchargé le JDK au lien suivant : http://www.oracle.com/technetwork/java/javase/downloads/index.html

Le téléphone utilisé possède la version 5.1 d'Android ( Lolipop), le projet a donc été configuré en fonction de ce SDK.

Configuration du téléphone :

Pour se configurer en mode développeur, nous devons aller dans:

- « Paramètre » du téléphone,
- aller dans sécurité => activer « source inconnue ».
- Dans « à propos de l'appareil », nous avons besoin de cliquer 7 fois sur le numéro de version du téléphone pour pouvoir passer en mode « développeur » .
- Pour finir, nous allons dans « option de développement » pour activer le débogage USB.

a) La classe Gps:

Nous avons utilisé plusieurs méthodes pour s'abonner et se désabonner du GPS pour que le programme ne puisse rencontrer aucun problème si celui-ci n'est pas disponible.

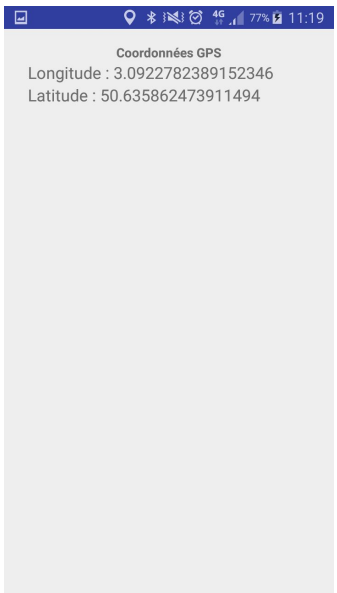
AbonnementGPS() - Méthode pour s'abonner à la localisation par gps

DesabonnementGPS() - méthode permettant de se désabonner de la localisation par GPS
OnProviderEnabled() - si le gps est activé , on s'abonne

OnProviderDisabled() - si le gps est désactivé, on se désabonne

OnLocationChanged() - on affiche dans un message la nouvelle localisation ( longitude et latitude)

Le programme demande au démarrage si le GPS est disponible, ensuite on s'y abonne et nous pouvons récupérer les données.




b) La classe BluetoothLE:

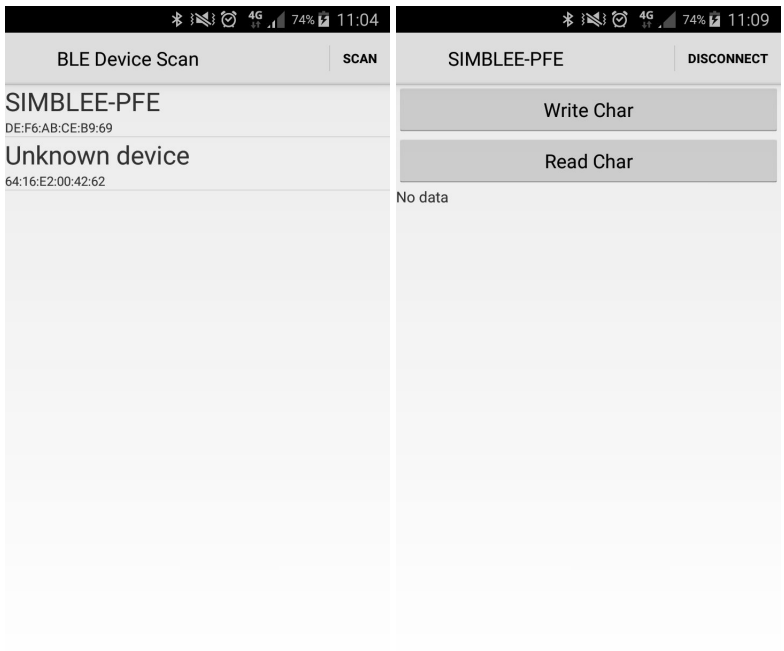
Pour concevoir cette partie, nous nous sommes tout d'abord intéressé à la partie Bluetooth classique, nous avons fait un programme permettant d'activer le Bluetooth, de repérer les dispositifs environnant disponible afin d'avoir la possibilité de se connecter.

La lilypad Simblee est utilisé en Bluetooth Low Energy ( Bluetooth 4.0) qui utilise très peu d'énergie pour fonctionner. Nous avons donc fait un programme permettant de :

- Scanner les dispositifs environnant en vérifiant que ceux-ci supportent le BluetoothLE.
- Une fois la lilypad repérer, nous nous connectons à celle-ci.
- Le programme a la possibilité de lire et écrire des données, pour cela nous construisons un GATT qui permet à deux dispositifs BLE de transférer des données en appelant des services et des caractéristiques en utilisant des ID de 16 bits ( Universal Unique Identifier).

Par exemple, toutes les caractéristiques BLE s'écrivent ainsi :
`0000XXXX-0000-1000-8000-00805f9b34fb`

Les quatres x représentent un champ dans lequel on insère nos propres ID 16 bits pour nos services et caractéristiques personnalisées et les utiliser comme un UUID prédéfini. .




Nous pouvons voir ci-dessus que notre Simblee a bien été trouvée, une fois connecté, nous pouvons utiliser les fonctions “envoyer” et “recevoir”.

c) La classe ClientUDP:

Nous avons combiné la partie récupération des données GPS avec l’envoi des données au serveur. Pour cela nous créons un paquet contenant :

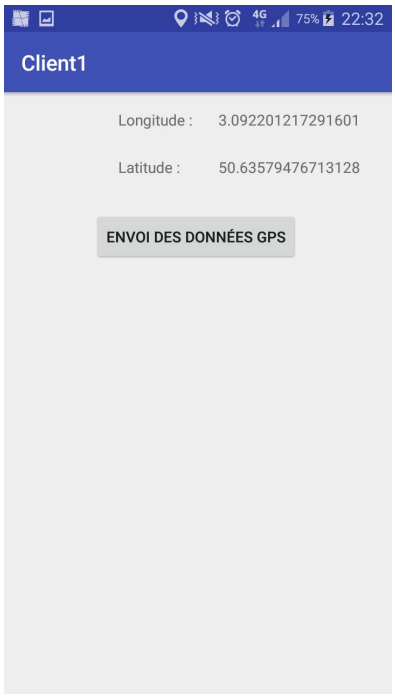
-le message que le serveur souhaite recevoir

-la taille du message

-l’adresse

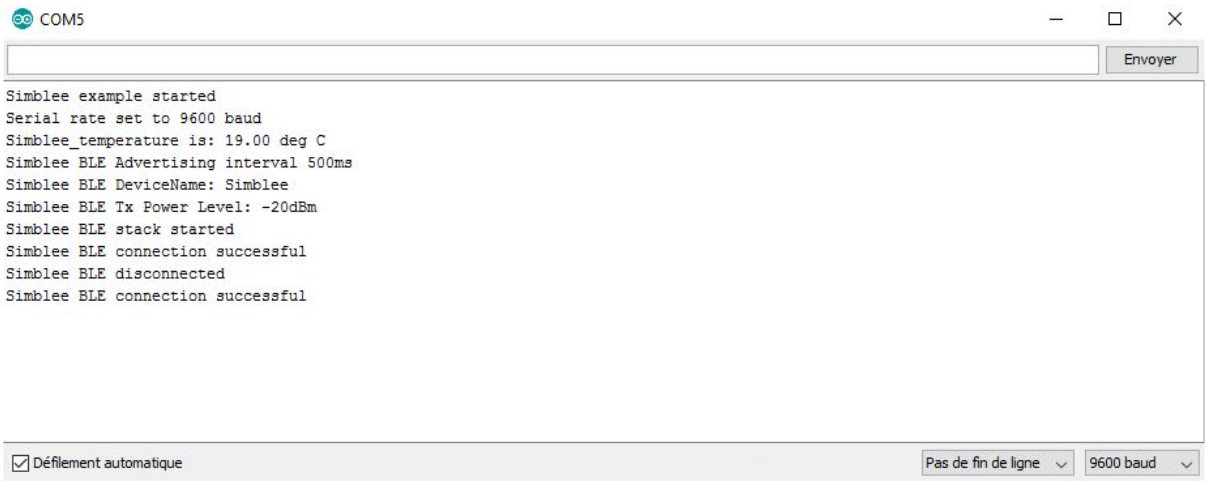
-le port

Celui-ci sera ensuite envoyé au serveur. A chaque appui du bouton, on actualise puis on envoie les données GPS au serveur.




2) Partie Lilypad :

Cette partie permet d'utiliser la Lilypad Simblee Ble pour envoyer les données que l'on souhaite au téléphone. Pour cela, nous utilisons le logiciel Arduino IDE pour pouvoir récupérer le signal du bouton poussoir et l'envoyer par bluetooth avec la fonction "Simbleeble.send". Une fois que le bouton poussoir est enclenché, le programme envoi la donnée au mobile. Nous utilisons ici la librairie Simbleeble.h. Ci-dessous nous pouvons vérifier si la connexion entre le téléphone et la lilypad c'est bien effectuée en écrivant dans le terminal de l'Arduino quand une connexion et une déconnexion est faite.




## Partie Hardware pour la partie Tracker autonome :

1 ) Test du module gps avec l'Arduino Uno

Une fois les composants récupérés, nous avons décidés d'essayer le module GPS en utilisant les pins principales puis en les câblants sur l'arduino directement. Pour vérifier son fonctionnement, nous avons

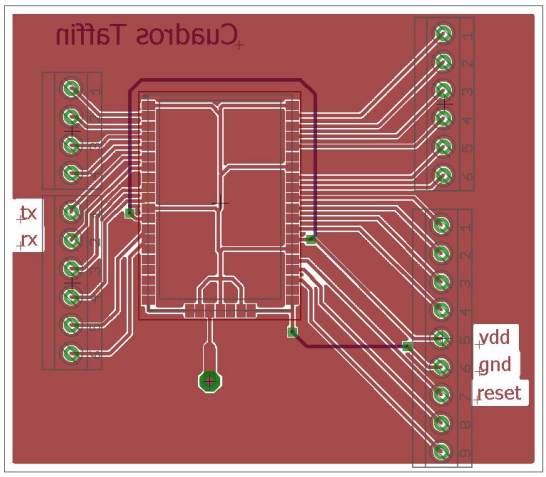
utilisé un programme permettant de récupérer les données GPS. Malheureusement, cette utilisation était trop tôt pour réellement vérifier comment fonctionne le module, nous avons donc décidé par la suite de créer des cartes électroniques de test pour chacun d'eux pour que leur utilisation soit plus simplifier.

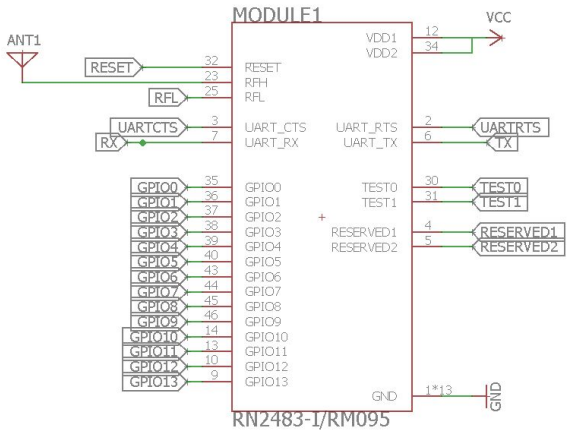
2 ) L'élaboration des cartes électroniques de test

Pour débuter, nous allons concevoir 2 cartes électroniques simplifiées afin de tester chaque module. Nous allons mettre en avant les Pins supposés importantes, mais aussi celles qui ne le sont pas ( afin de pouvoir les utiliser si besoin). Nous câblerons ensuite chaque carte électronique avec une carte Arduino Uno afin de récupérer les informations et vérifier son bon fonctionnement. Les pins que nous désirons utiliser dans un premier temps pour le module LoRa et le module GPS sont : Le VCC ,le GND,le Tx,le Rx ainsi que le Reset.

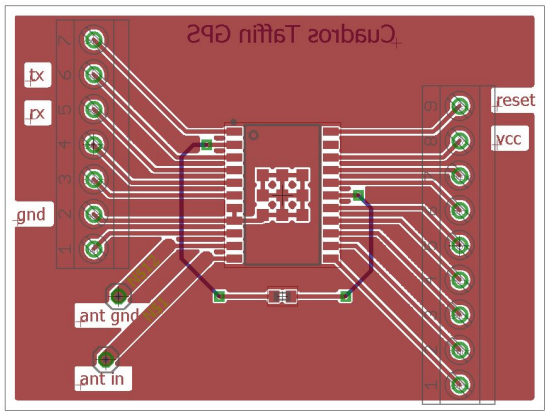
La transmission de donnée de l'Arduino vers les modules est de 5v. Les modules peuvent supporter 3.3v, nous implémenterons donc un transistor LM1117 pour réduire cette tension.

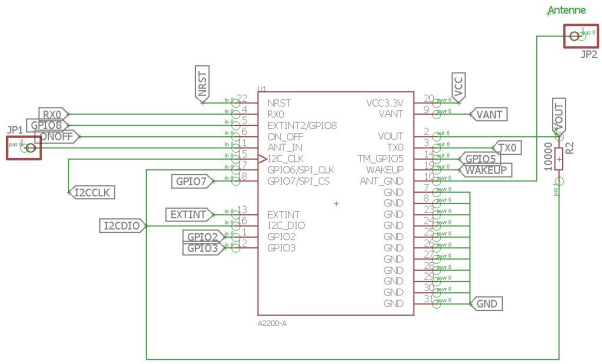
Première carte électronique : Le module LoRa RN2483.




Deuxième carte électronique : le module GPS A2200-A.



# III : Travail restant

**Pour la partie Autonome:**

La plus grande partie du travail restant concerne la partie autonome:

- Utiliser les cartes d'essai du GPS et du LoRa avec une arduino et vérifier la récupération des données.
- Si les tests sont concluant, refaire une nouvelle carte électronique permettant de rendre son utilisation autonome

**Pour la partie Téléphone :**

- Utiliser les bons UUID du Bluetooth Gatt pour pouvoir recevoir et envoyer des données.
- Assembler les programmes test.
- Rendre son utilisation simple en ajoutant une batterie à la lilypad.

**Pour la partie Web/Serveur:**

- Améliorer l'interface graphique
- Gestion des profiles
- Revoir l'architecture de la base de donnée
- Générer des itinéraires, événements...
- Liste d'amis
- Sécurité