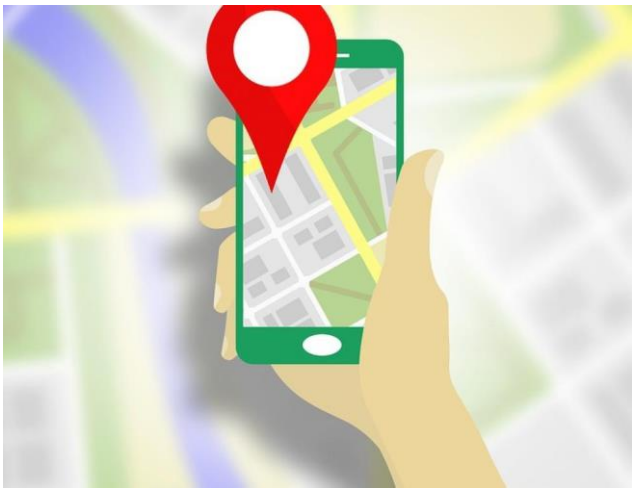
Compte rendu du projet d'Internet des Objets

FindPeople&Things : un objet connecté géolocalisable



Réalisé par :

Sanvee Myriam
Benabed Djahida
Kessaï Aghilas

Étudiants en L3 informatique

Encadré par :

M. Massinissa Hamidi
M. Aomar Osmani

Sommaire :

I-Introduction

1) La problématique :

Dans le cadre du cours d'Internet des Objets, nous avons été menés à élaborer un objet connecté. Notre objectif était de développer un objet simple qui sera utile à la société.

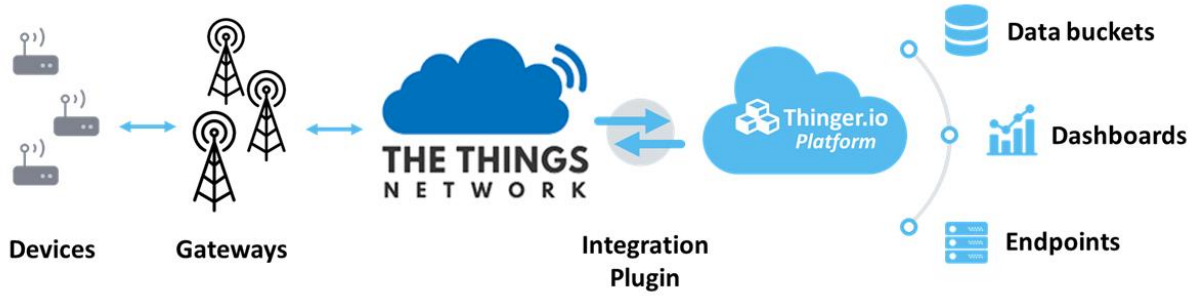
Après quelques jours de discussion, nous avons décidé de répondre à la problématique: comment peut-on localiser une personne ou un objet sans limite de distance et à n'importe quel moment de la journée ?

Étant donné la croissance rapide de la population et l'évolution du mode de vie actuel, nous avons de nos jours moins de temps, plus d'enfants et plus de biens matériels. Un problème fréquent que rencontrent les parents est celui de savoir ou est leur enfant. En effet, les parents sont souvent inquiétés quant à la localisation de leur enfant (nous avons tous rencontré un enfant qui a perdu ses parents dans un grand centre commercial par exemple). Parallèlement, il y a le problème de localisation de biens matériels importants tel qu'une voiture, une moto ou un vélo. Lors d'un vol de voiture, il est peu fréquent que la police retrouve le véhicule volé car le vol se déroule en général la nuit et les malfaiteurs ont généralement le temps de disparaître dans une autre ville. Comment est-il possible de localiser le véhicule si celui ci n'est pas équipé d'un capteur GPS ?

Pour répondre à cette problématique nous avons décidés de développer un objet connecté FindPeople&Things. Cet objet connecté sera géolocalisable à travers une application qu'on aura aussi développée. Ainsi, si on veut géolocaliser notre véhicule, nous avons qu'à dissimuler cet objet dans le véhicule et, voir la localisation sur notre téléphone à travers l'application ! L'objectif est donc de réaliser un objet de petite taille dont on pourra géolocaliser à travers une application simple.

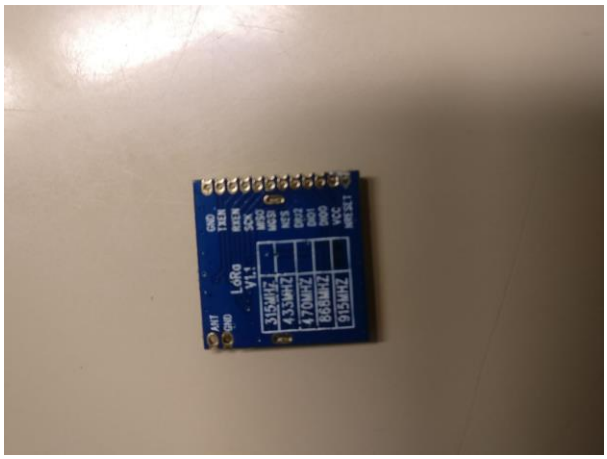
2) Le problème du module

Notre principal problème était de trouver un module GPS qui pourra communiquer avec notre application et envoyer des informations de localisation. Au début nous avons pensé à utiliser la technologie LoRaWAN (un protocole de communication permettant la communication à bas débit). On devait donc utiliser 'The Things Network' qui est un réseau LoRaWAN pour l'internet des objets.



Source image : https://docs.thinger.io/plugins/the-things-network

On peut sur l'image ci-dessus comment notre objet (device) pourrait nous envoyer des informations à travers la plateforme Thinger.io. Compte tenu de la contrainte du temps, nous avons décidés de ne pas commander un module LORA par internet de peur qu'il soit livré trop en retard. Nous avons donc acheté un module LoRa dans le magasin HackSpark à Paris. Hélas nous avons rencontré des problèmes lors du branchement du module avec l'ESP32 car le module LoRa a été fourni sans pin, ce qui a rendu difficile le branchement avec soudage.



Le module LoRa

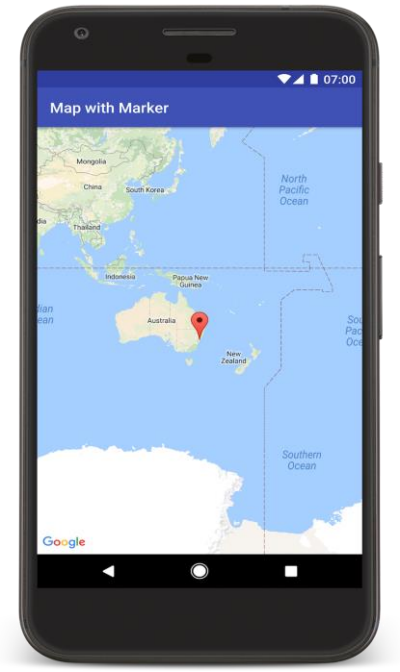
Nous avons donc laissé tomber l'idée du module LoRa. Après plusieurs recherches sur internet, nous sommes allés au magasin Letmeknow à Paris ou nous nous sommes procuré le module GSM SIM800L, ce module étant *'l'un des plus petits modules GSM du monde avec une taille de 2.2cm x 1.8 cm' (source : www.letmeknow.fr).*

Notre problème a été réglé : nous avons donc trouvé un module GSM qui peut envoyer des SMS et donc envoyer un SMS contenant la localisation comment nous l'avons imaginé.

3) Description du projet

Après avoir réglé le problème du module GSM, on pouvait donc avancer dans le projet. Parallèlement, nous avons commencé à développer l'application.

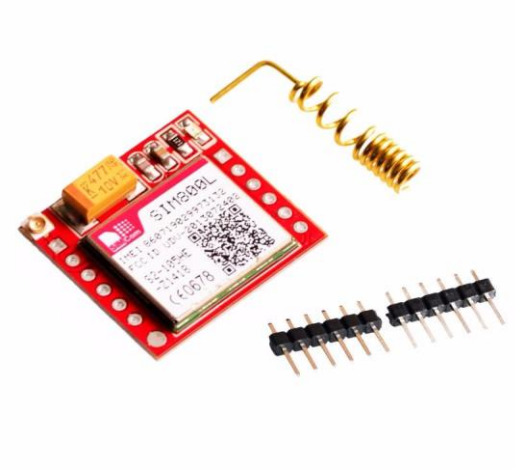
Notre but était donc de réaliser une application qui communique avec le module SIM800L. Le but est simple : l'application reçoit un SMS contenant les informations de localisation du module SIM800L et traduit ces informations avec une localisation sur la carte. L'application devra donc comporter une carte et un 'marker' indiquant la position de notre objet FindP&T.



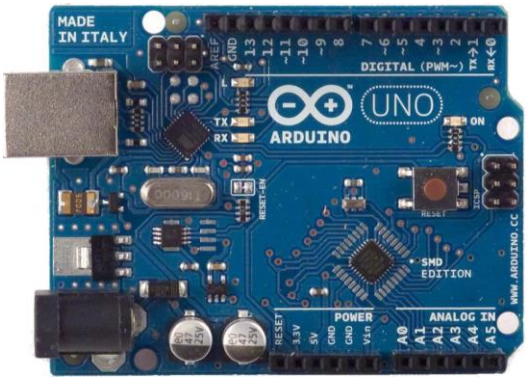
Carte GoogleMaps avec un marker indiquant une position

II- L'objet connecté

Comme dit précédemment, nous avons opté pour le module GSM SIM800L. Ce module est branché avec l'Arduino UNO fournit par l'Université.



Module SIM800L



Arduino

Le module GSM SIM800L est l'un des plus petits modules GSM du monde avec une taille de 2.2 cm x 1.8 cm. C'est un module puissant qui démarre automatiquement et recherche automatiquement le réseau. Il inclut notamment le Bluetooth 3.0+EDR et la radio FM (récepteur uniquement). Il vous permettra d'échanger des SMS et de passer des appels.

Ce module nécessite une alimentation entre 3,4V et 4,4V. L'alimentation 5V de l'Arduino ne lui convient donc pas. Pour contrer ce

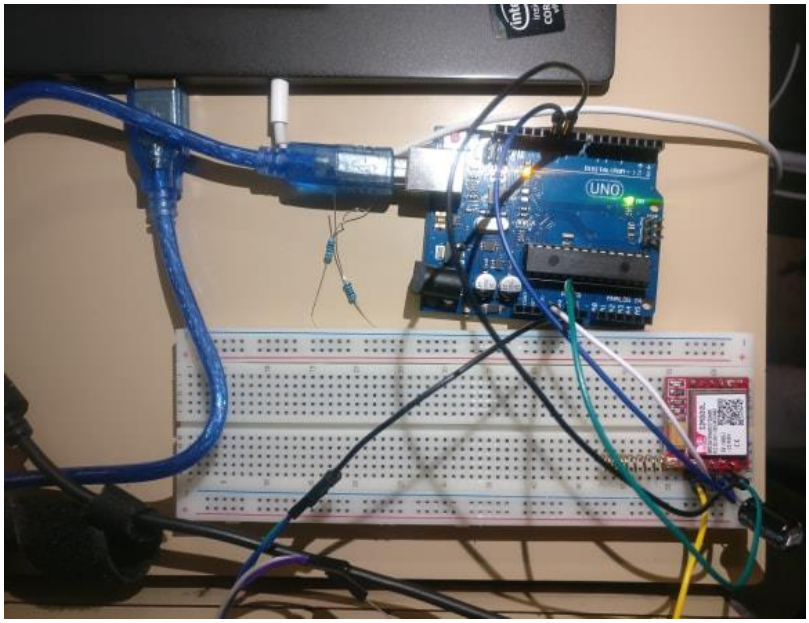
problème d'alimentation, on ajoute une diode 1N4007 entre le 5V de l'Arduino et le pin VCC du SIM800L.

On utilise aussi : la diode 1n4007 et le condensateur 100nF50v

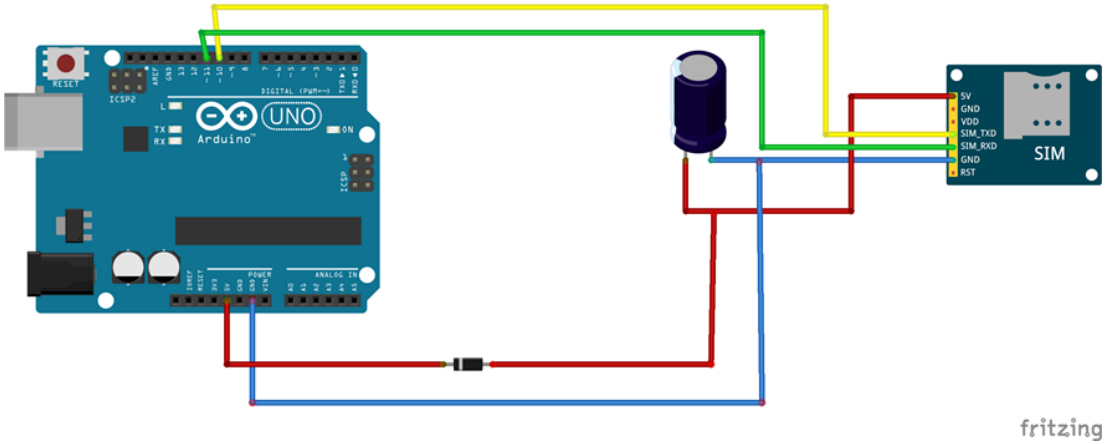
La diode 1n4007

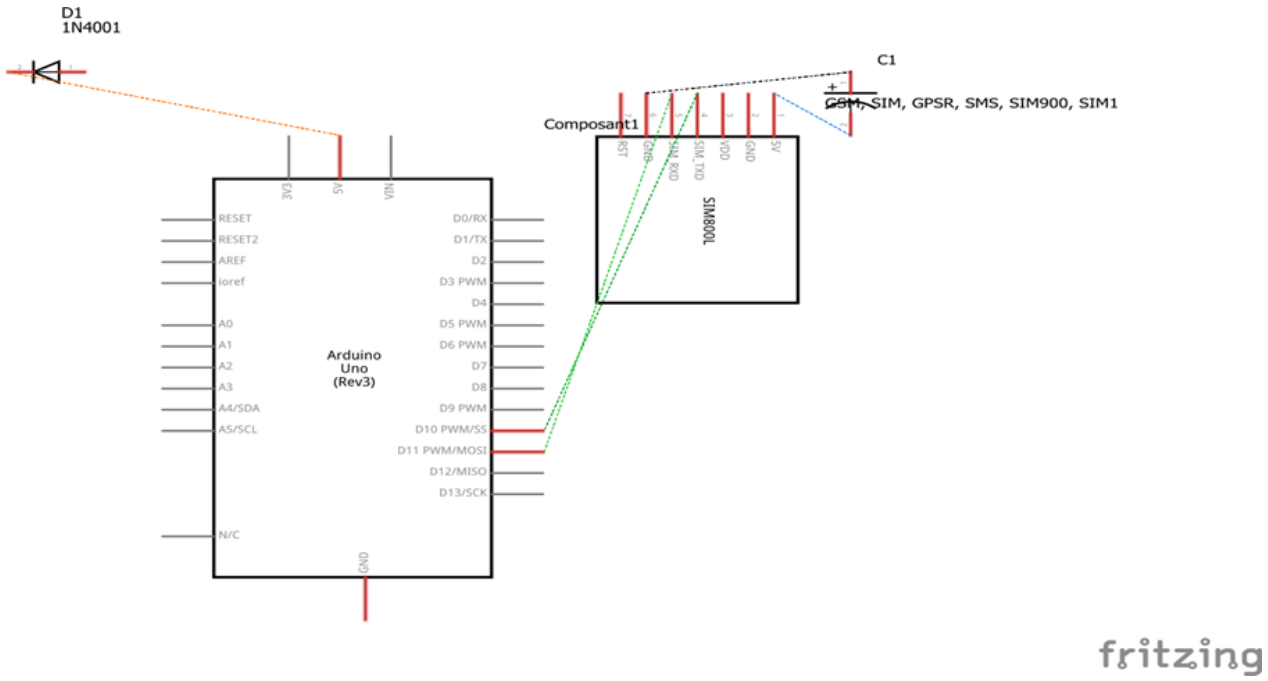
Le condensateur 100nF50v

Le branchement :



Branchement avec Fritzing :





Matériel utilisé :

- ➔ Module SIM800L
- ➔ Arduino Uno
- ➔ Une carte SIM
- ➔ 400 hole board (breadboard)
- ➔ Condensateur
- ➔ Des fils
- ➔ Diode

Le fonctionnement du module :

Il faut insérer la carte SIM dans le compartiment prévu à cet effet sur le module, comme montré dans la photo ci-dessous. Le sens

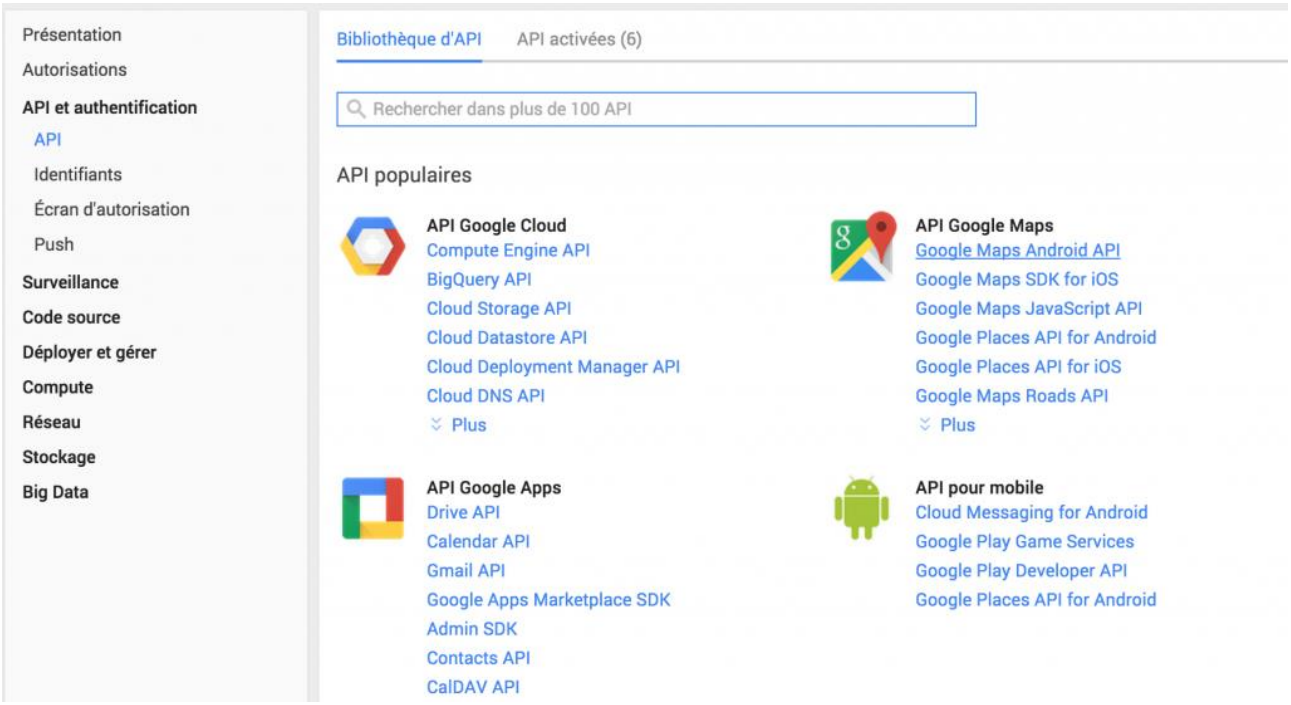
d’insertion de la carte SIM dans le module à son importance. La carte SIM doit être une micro SIM.

Insertion de la SIM dans le module SIM800L

On lance le programme où on a inclus la bibliothèque < SoftwareSerial.h> pour utiliser toutes les commandes AT pour envoyer les SMS, faire des appels et aussi pour récupérer la localisation du module GSM pour l’envoyer à l’application. A la réception d’un message spécial (‘GPS ON’) le module envoie sa localisation au numéro de téléphone qui lui a envoyé ce message spécial. Voir fichiers .ino
(NB. Si votre carte SIM contient un code PIN vous avez le choix : soit vous le désactivez sinon il y a une partie dans la fonction setup() pour déverrouiller votre carte SIM).

III- Développement de l’application

Compte tenu du fait que l’application que nous devons développer est une application de géolocalisation, il est plus que nécessaire d’afficher une carte. Pour cela, nous utilisons la Google Maps API sur le site https://developers.google.com/android:



Après obtention d'une clé unique qui identifie notre projet, on l'inclut dans un fichier important qui est le fichier manifest:

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="@string/google_maps_key" />
```

Aussi, il est important de gérer les permissions. En effet, si on essaye d'accéder à la localisation sans avoir eu la permission de la part de l'utilisateur, alors l'application s'arrêtera systématiquement.

Il est important d'indiquer dans le fichier AndroidManifest.xml les permissions qu'on va demander à l'utilisateur:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```

En effet le

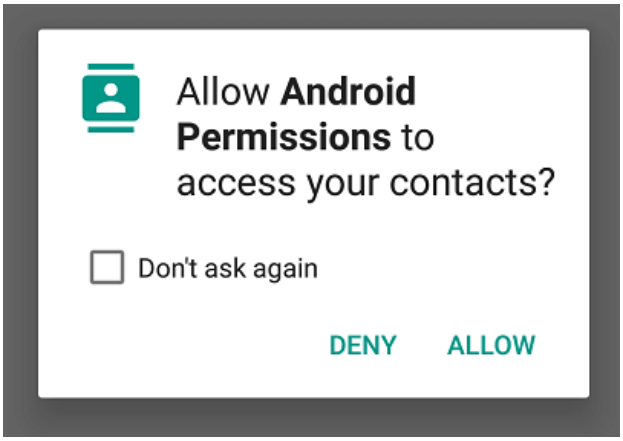
fichier AndroidManifest.xml est un fichier qui contient les informations essentielles concernant notre application.

Avant d'accéder à la localisation par exemple, on doit vérifier si l'application à le droit d'y accéder. Nous avons une méthode qui permet de vérifier cela : ContextCompat.checkSelfPermission.

```
if (ContextCompat.checkSelfPermission(thisActivity, Manifest.permission.WRITE_CALENDAR)
        != PackageManager.PERMISSION_GRANTED) {
    // Permission is not granted
}
```

Source : https://developer.android.com/training/permissions/requesting#java

Si nous avons pas la permission, on doit demander la permission à travers la méthode : requestPermissions();
Suite à l'appel de cette méthode une pop-up s'affiche sur le télephone:



Ce pop-up s'affiche suite à la demande de permission d'accès aux contacts par exemple.

Lorsque l'utilisateur reçoit une demande de permission, alors le système invoque la méthode onRequestPermissionResult().
C'est dans cette méthode qu'on peut savoir si l'utilisateur à autoriser l'application ou pas (mais il n'est pas obligatoire de l'utiliser). En somme, on ne peut pas essayer d'accéder à la localisation si on n'a pas la permission nécessaire.

Il est important de savoir qu'une application Android est composée d'activités : une 'activity' = un 'écran'. On doit donc pouvoir passer d'un écran à un autre (par exemple accéder à la carte ou au tutoriel à partir du menu). On peut faire cela à travers 'intent' : un intent est une description abstraite d'une opération qui doit être exécutée.
(https://developer.android.com/reference/android/content/Intent).
Supposons que nous sommes dans l'Activity et nous voulons accéder à l'Activity2:

```
Intent intent = new Intent(this, Activ2.class);
startActivity(intent);
```

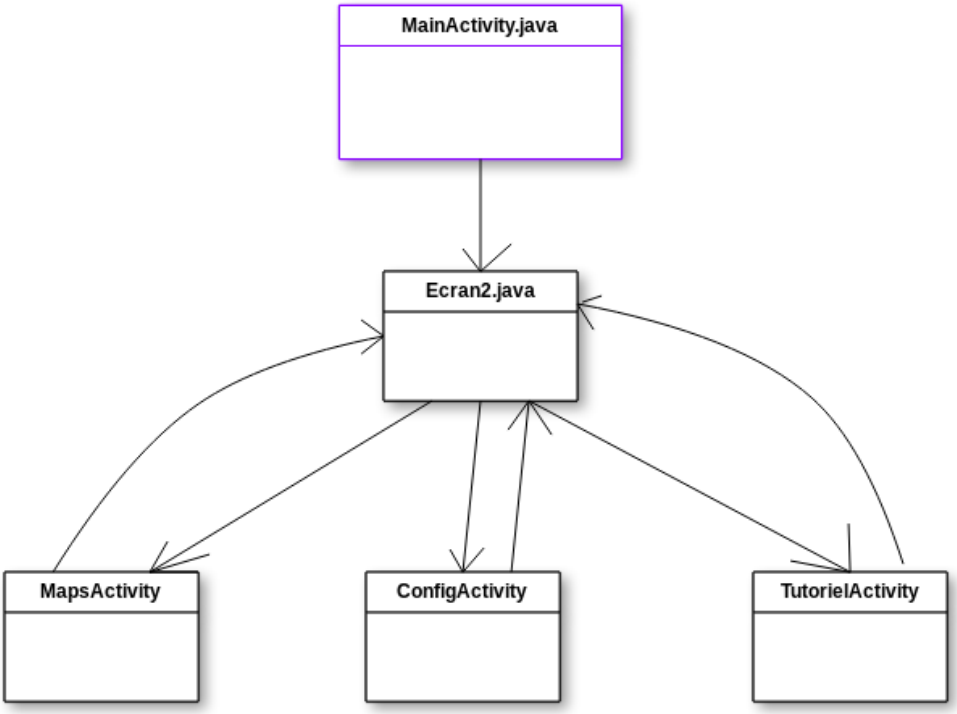
source : https://perso.univ-rennes1.fr/pierre.nerzic/Android/poly.pdf

IV- Les 2 applications Android

Compte tenu des circonstances (grèves et vacances) nous n'avons pas pu se réunir régulièrement afin de faire le développement sur la même application. C'est pour cela que nous avons développés 2 applications différentes mais ayant les mêmes fonctionnalités.

L'application1 :

Voici un schéma décrivant le fonctionnement de l'application :



MainActivity.java :

Ce fichier décrit l'Activity de l'écran d'accueil. En effet, c'est un 'Splash Screen', c'est à dire un simple affichage de quelques secondes qui finit par disparaître, il est en général utilisé comme une page de garde (elle permet de faire patienter l'utilisateur en attendant le chargement complet de l'application) :

Par suite de cet affichage on se retrouve sur l'Activity Ecran2.java :

Ici, on a le choix d'accéder : au tutoriel, à la carte (bouton localiser mon objet) ou de gérer ma localisation.

Si on appuye sur 'localiser mon objet' alors on se retrouve sur la MapsActivity (activity ou on affiche la carte) :

Un message (Toast) est affiché pour indiquer à l'utilisateur qu'il a bien autorisé l'application à accéder à la localisation du téléphone (image 1). Imaginons que l'utilisateur n'a pas préalablement autorisé l'application à accéder à la localisation, alors nous avons un pop-up qui s'affiche et qui demande l'autorisation d'accéder à la localisation (image2).

Image 1

Image2 : La carte ne s'ouvre pas si l'utilisateur n'a pas autorisé.

Tant que l'utilisateur appuye sur 'refuser' alors la pop-up ne cesse de s'afficher. Ceci s'explique par cette partie du code :

```
if (checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
        checkSelfPermission(Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions( activity: this,new String[]{
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_COARSE_LOCATION},PERMS_CALL_ID
    );
    return;
}
```

En fait, si on a pas accès a FINE_LOCATION et COARSE_LOCATION (coarse est utilisée par le réseau et fine est utilise par le GPS et le réseau)

alors on demande la permission :
Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION

Et ensuite on quitte la fonction (return;) .

```
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if(requestCode == PERMS_CALL_ID){
        checkPermissions();
    }
    else if(requestCode==PERMS_RECEIVE){
        checkPermissions();
    }
}
```

Suite au RequestPermission, on se retrouve dans la méthode onRequestPermissionResult, mais dans n'importe quel cas, on revient dans la méthode checkPermissions() qui, elle va demander l'autorisation si elle n'a pas été acquise.

Dans le cas où la permission est accordée alors on affiche la carte avec notre propre localisation et si cela a été configuré, la localisation de l'objet connecté FINDP&T :

Si on appuye sur le marker rouge, alors on voit la ville ainsi que le code postal. Attention, le marker rouge indique notre propre localisation, c'est-à-dire la localisation de l'utilisateur et non de l'objet FindP&T. Cela sera utile afin de donner à l'utilisateur une idée de la distance qui le sépare de son objet connecté.

Pour afficher la localisation de l'objet connecté, on doit 'revenir en arrière' a l'activité précédente (Activity Ecran2).

On revient donc à l’activité précédente :

On appuye sur ‘Gerer ma localisation’ (a gauche) et on se retrouve dans l’activité ConfigActivity(a droite).

Deux options s’offrent à nous. Si on veut localiser notre objet connecté alors on appuye sur le bouton ‘Localiser mon objet’. Suite au clic sur ce bouton, l’application envoie un message ‘GPS ON’ au numéro entré par l’utilisateur (ce numéro doit correspondre à la SIM entrée dans le module SIM800l de l’objet connecté). Suite a la réception du message ‘GPS ON’, l’objet envoie systématiquement les données sur sa latitude et longitude sous la forme d’un message de ce format : latitudeXlongitude. Par exemple, l’objet peut envoyer ce message : 45.22656X2.2556254
Lors de la réception de ce message, l’application récupère le message sous forme de string et le sépare en 2 avec comme condition de séparer en 2 la ou il y a un ‘X’.
On se retrouve donc avec 2 string contenant pour une la latitude et l’autre la longitude. On récupère ces valeurs dans des variable de type double et on affiche ensuite sur la carte un marker bleu qui indique la position de l’objet.

```
String latitudeLongitude=null;
if(checkSelfPermission(Manifest.permission.RECEIVE_SMS) == PackageManager.PERMISSION_GRANTED) {
    latitudeLongitude = ReceiveSms.getMessage();
   //Toast.makeText(this,"latitudeLongitude not NULL",Toast.LENGTH_SHORT).show();
}
```

latitudeLongitude vaut le message reçu pendant que l'application a été lancée, latitudeLongitude vaudra donc null si on ne reçoit pas de message, étant donné que getMessage() renverra null. Il est donc important de gérer cette erreur et de n'afficher aucun marker bleu sur la carte.

```
String[] separation = latitudeLongitude.split( regex: "X");
String latitude1 = separation[0];
String longitude2 = separation[1];
final double  lat1=Double.parseDouble(latitude1);
final double long2=Double.parseDouble(longitude2);
```

On fait le traitement correspondant pour récupérer la latitude et la longitude de notre objet.

On voit ici le marker bleu avec comme titre 'FINDP&T'qui indique la position de notre objet connecté.

La 2eme option est d’appuyer sur le bouton ‘Envoyer ma localisation’ : cette option permet d’envoyer sa localisation a un proche et lui permettre de le localiser (il doit posséder l’application FINDPT aussi).

Enfin, la dernière Activity qu’on peut accéder est celle du tutoriel :

Une classe importe qui ne représente pas une Activity, est la classe ReceiveSms qui permet de recevoir un SMS et donc recevoir les informations de localisation.

L’application2 :

L’application 1 et 2 ont le même objectif qui est celui de localiser une personne ou un objet.
Seulement les méthodes diffèrent, en effet dans l’application 2 nous avons quand même essayé de partir sur l’idée de base qui était d’utiliser une plateforme qui nous sert de point relais entre l'objet et

l'applications. Après avoir réfléchi nous avons opté pour Firebase qui est plus facile d'accès via Android Studio.

L'idée est de concentrer une base de données sur cette plateforme afin que tout objets connectés avec cette base de données aient accès aux données des autres utilisateurs. Pour gérer tout cela des points indispensables ont été pris en compte.

La premier point est d'ajouter un utilisateur, en effet l'ajout d'un utilisateur est non négligeable . Elle nous permet d'agrandir à n'importe quelle moment la base de données selon le nombre d'utilisateur.

Le deuxième point est la mise à jour des données, sans cela l'application n'a aucune utilité. Cependant c'est à l'utilisateurs de remettre à jour ses données en inscrivant tout simplement son nom et son numéro. Le numéro est aussi essentiel à l'identification de la personne, lorsque la personne s'identifie on lui attribue un identifiant et grâce au numéro nous pouvons distinguer les utilisateurs sans accéder directement à leur identifiant. Cela est dû au faîte que nous n'avons pas utilisé la méthode d'authentification. Si nous avions opté pour l'authentification par mail par exemple l'identifiant peut être récupérer et ensuite utilisé pour être mis à jour automatiquement.

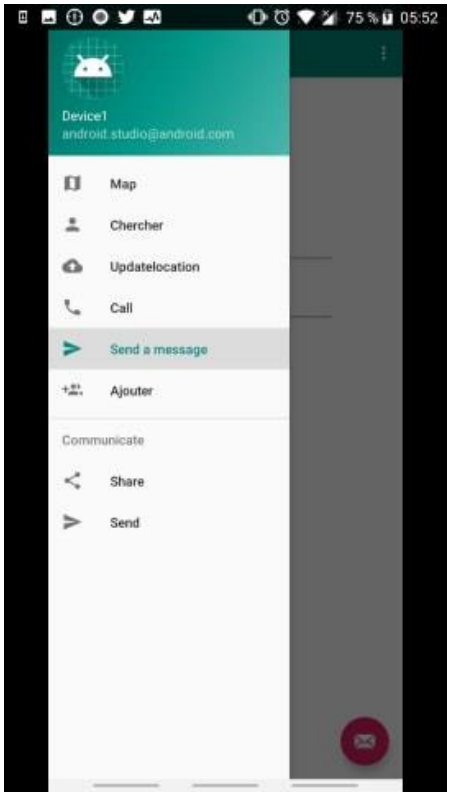
Le troisième point est d'identifier la personne que l'on veut afin de pouvoir éviter l'amas de marker sur la carte. Pour ce cas nous avions utilisé exactement la même méthode que pour mettre à jour les données d'une personne. C'est aussi plus simple pour l'utilisateur de mémoriser un numéro et le nom pour rechercher une personne plutôt que l'identifiant de la personne ou l'objet de à localiser qui est une chaîne de caractère assez complexe.

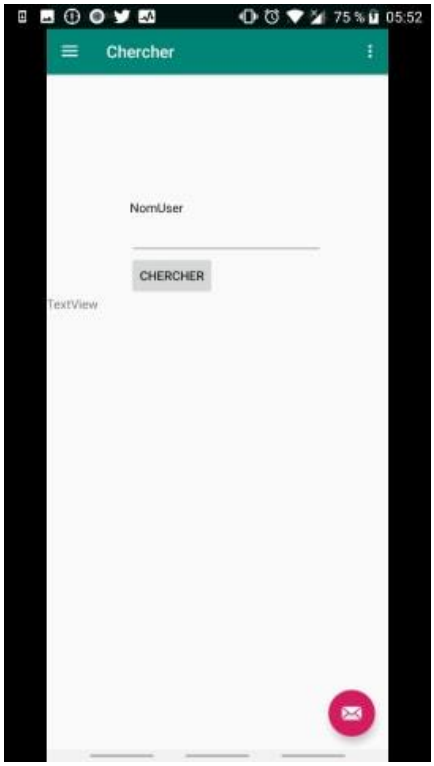
L'application à deux fonctionnalités en plus de celles qu'on a citées précédemment. En effet la première est de pouvoir appeler ses proches en cas de problème ou tout simplement de les avertir de votre situation. La deuxième option est de pouvoir envoyer des messages pour les mêmes raisons que l'option d'appel.

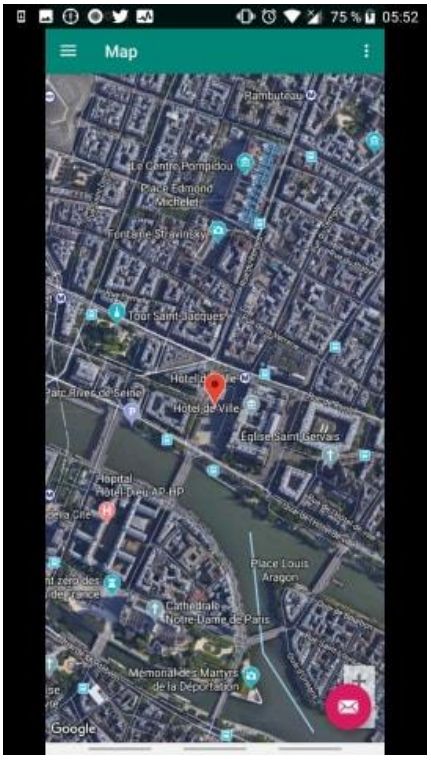
L'AGENCEMENT DE L'APPLICATION

Nous avons choisi pour la deuxième application une vue compacte qui permet l'utilisateur de visualiser d'entrer les options qui lui sont proposé. Ainsi la prise en main de l'application sera plus simple et plus rapide ce qui est avantageux pour les personne qui ne sont pas très familier avec la technologie.

Les Activity de l'application :







Les activités des deux applications sont accompagnées de fichiers XML qui dictent la mise en page de celles-ci : affichage de texte, les boutons,etc. Il fallait aussi se familiariser avec Firebase ce qui était difficile pour nous (d'ailleurs il y a des fonctionnalités non complètes dû à cela : c'est le cas pour l'activité recherche et de mise à jour(application 2).

V-Conclusion

Pour finir, l'UE d'Internet des Objets nous a permis de découvrir le monde des objets connectés. Nous avons découvert comment développer une application Android dans l'IDE Android Studio. Au début, c'était difficile de se familiariser avec Android Studio. On devait comprendre comment le front end se faisait avec les fichiers XML tout en programmant en JAVA pour le backend.