# Programmation Mobile – Android

TP Capteurs et caméra
Johann Bourcier d'après le TP d'Arnaud Blouin (INSA)

## 1. Objectif

Le but de ce TP de 4h est de comprendre comment utiliser les capteurs et les caméras embarqués sur les appareils mobiles et de savoir les utiliser dans le cas de la plate-forme Android.

L'application Android à développer est un outil permet de recenser et d'utiliser chaque capteur et caméra fournis par l'appareil. La figure suivante montre l'écran principal de l'application recensant dans une liste les capteurs fournis par l'appareil.


*Figure 1: activité principale de l'application Android à développer*

Il est alors possible de cliquer sur chaque capteur (chaque élément de la liste) pour afficher une activité dédiée au capteur sélectionné. Par exemple la figure suivante montre l'activité liée à l'accéléromètre de l'appareil. Cette activité affiche les informations standards fournis par les capteurs Android ainsi que les informations spécifiques au capteur en question (ici les forces d'accélération).

Pour chaque capteur, vous devrez également trouver une utilisation simple à implémenter dans cette activité. Par exemple, pour l'accéléromètre, vous devrez faire en sorte qu'en cas de forte accélération un bruit de sabre laser (que l'on vous fournit) soit lu. Pour le capteur de proximité, vous pourrez continuer sur cette voie de padawan pour lire un son du contact de deux sabres laser lorsque deux appareils sont très proches.

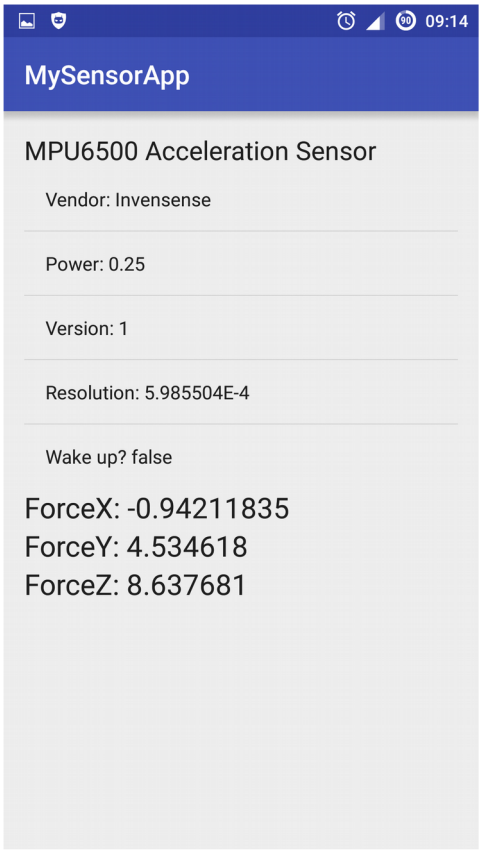


*Figure 2: Activité pour l'accéléromètre*

## 2. Activité principale

**Q1.** Créer un nouveau projet Android dont l'activité principale (*SensorsActivity*) sera celle illustrée par la Figure 1. Vous utiliserez une liste (*ListView*, http://www.vogella.com/tutorials/AndroidListView/article.html) pour afficher les capteurs de l'appareil (la liste affichera le nom du capteur). En ce qui concerne *ListView*, il faudra utiliser un adaptateur (*ArrayAdapter*) pour ajouter des éléments dans un *ListView*. Aidez-vous de la documentation officielle : https://developer.android.com/guide/topics/sensors/sensors_overview.html. Vous aurez notamment besoin d'utiliser les classes *SensorManager* et *Sensor*.

Pour réaliser cette partie du TP, vous aurez besoin des acquis des TP précédents, en particulier :

- Les activités Android
- La description de l'interface graphique dans un document XML
- Le *findViewById*

La hauteur (*height*) de votre liste de capteurs ne devra pas prendre toute la place afin de ne pas masquer d'éventuel autres éléments qui suivraient. Pour cela, référez vous à : https://stackoverflow.com/questions/15479915/height-of-listview-fills-the-whole-screen-although-set-as-wrap-content

## 3. Activité spécifique aux capteurs : l'accéléromètre

**Q2.** Créer une seconde activité *SensorActivity*. La spécificité de cette activité est qu'elle sera complétée programmatiquement par un Fragment spécifique au capteur choisi (on verra cela plus tard).

**Q3.** Faites en sorte qu'un clic sur un élément de la liste de l'activité principale lance une activité *SensorActivity*. Cette activité (cf. figure 2) affichera (pour l'instant) les informations standards à tous les capteurs Android (toujours à l'aide d'un ListView).

Vous aurez besoin :

- de placer le titre de l'activité et la liste dans un layout du type *LinearLayout* dont l'*id* sera *layoutDetails*. Cette étape va simplifier la question 4. Les attributs de ce layout seront :

```
android:orientation="vertical"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
```

Les éléments de ce layout (le titre et la liste) devront avoir les attributs suivants :

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

Le but de ces attributs est de minimiser la taille des widgets (*wrap_content*) dans le but de faciliter la question 4.

- d'utiliser le listener (écouteur d'évènements) setOnItemClickListener pour réaliser une action lors d'une pression sur un élément de la liste.
- De lancer une activité, lors d'une pression sur un élément de la liste, en utilisant les concepts *Intent*, *startActivity*. https://developer.android.com/training/basics/firstapp/starting-activity.html
- De fournir en paramètre de l'activité *SensorActivity* le nom du capteur à utiliser (cf. point numéro 5 du lien précédent).

**Q4.** Créer un fragment *AccelerometerFragment*. Ce fragment affichera les données spécifiques à l'accéléromètre à l'aide de widgets *TextView* (cf. bas de la figure 2).

Vous aurez notamment besoin :

- d'ajouter programmatiquement le fragment à l'activité *SensorActivity* lors de la création de cette dernière : grâce au nom du capteur passé en paramètre, vous récupérez le capteur en question et en fonction de son type vous ajoutez tel ou tel fragment (pour l'instant le fragment *AccelerometerFragment* pour le type *Sensor.TYPE_ACCELEROMETER*). Il vous faudra alors instancier le fragment en question pour l'ajouter au layout *layoutDetails* de la question 3. Pour cela vous utiliserez le concept de *FragmentTransaction* de la manière suivante :

```
FragmentTransaction fragTransaction = getFragmentManager().beginTransaction();
fragTransaction.add(idDuLayout, fragAAjouter);
fragTransaction.commit();
```

**Q5.** Affichez en temps réel les informations de l'accéléromètre (cf. bas de la figure 2). Pour cela, votre fragment devra écouter le capteur (implémentez l'interface *SensorEventListener* et enregistrez le capteur comme *listener* du capteur, *registerListener*). N'oubliez pas de dé-enregistrer le fragment du capteur (*unregisterListener*) à la fin de l'activité (dans le fragment surchargez l'opération *onDetach*).

**Q6.** Faites en sorte que le son du sabre laser soit lu lors d'une forte accélération. Voici une bonne partie du calcul :

```
long curTime = System.currentTimeMillis();
if ((curTime - lastUpdate) > 100) {
    long diffTime = (curTime - lastUpdate);
    lastUpdate = curTime;
     float speed = Math.abs(x + y + z - last_x - last_y - last_z)/ diffTime *
10000;
    if (speed > SHAKE_THRESHOLD) {
...
    }
    last_x = x;
    last_y = y;
    last_z = z;
}
```

Vous aurez besoin de lire un fichier audio avec *MediaPlayer* (https://developer.android.com/guide/topics/media/mediaplayer.html). N'oubliez également pas de fermer le *MediaPlayer* à la fin de l'activité (*mediaPlayer.release()*).

Vous placerez le fichier audio dans un dossier *raw* que vous créerez dans le dossier *res*.

## 4. Activité spécifique aux capteurs : le capteur de proximité

**Q7.** Faites de même que pour l'activité précédente mais pour le capteur de proximité. En cas de forte proximité avec un objet, il faudra lire le fichier audio correspondant au contact de deux sabres laser.

## 5. Gestion des caméras

**Q8.** Ajoutez à l'activité principale la gestion des caméras comme le montre la figure de droite ci-dessous. Vous pouvez récupérer de la même manière que pour les capteurs la liste des caméras avec `(CameraManager) getSystemService(Context.CAMERA_SERVICE)`.

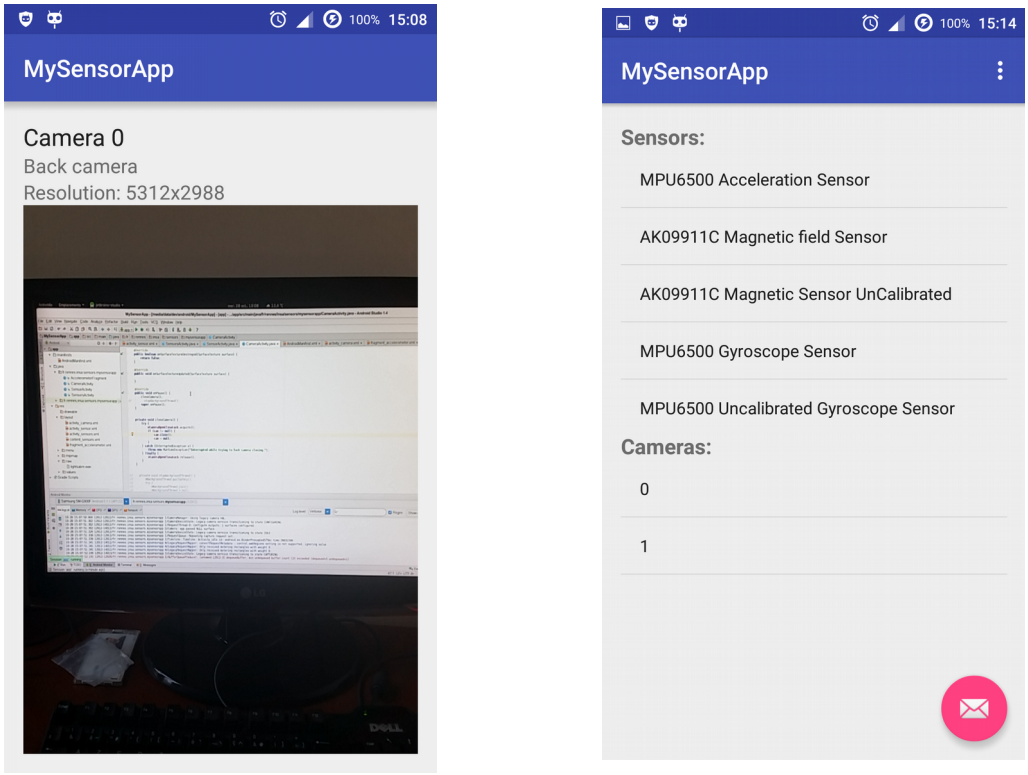


*Figure 3: Activité dédiée à l'utilisation d'une caméra et nouvelle activité principale*

**Q9.** Créez une nouvelle activité affichant des caractéristiques et le flux vidéo de la caméra en direct. Cette activité se lancera lors d'une pression sur le numéro de la camera dans l'activité principale (comme pour les capteurs). Inspirez-vous du code suivant pour afficher la vidéo : *https://github.com/googlesamples/android-Camera2Video/blob/master/Application/src/main/java/com/example/android/camera2video/Camera2VideoFragment.java*.

Pour vous aider :

- la caméra sera affichée dans un widget du type *TextureView* ;
- à la création de cette nouvelle activité, votre TextureView ne sera pas forcément initialiser (cela peut prendre un peu de temps). Implémentez donc l'interface TextureView.SurfaceTextureListener associée avec `textureView.setSurfaceTextureListener(this);`
- if vous faudra vérifier les droits d'accès à la caméra :
  `if(getPackageManager().checkPermission(Manifest.permission.CAMERA, getApplicationContext().getPackageName())==PackageManager.PERMISSION_GRANTED) {…}`
- N'oubliez pas de fermer la caméra à la fin de l'activité (dans *onPause*).