Le bluetooth sous android

Nicolas Fleurot

19 mai 2013

Chapitre 1

Description

Chapitre 2

Mise en place

2.1 récupérer l'adaptateur

Pour pouvoir utilié une connexion bluetooth, il faut déja commencér par récupérer le prériphérique bluetooth de l'appareil. Il représente le socket de connexion, c'est par lui que passeron toutes les données.

```
1 // on recupere le l'adaptateur.
2 mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
3
4 // si l'adaptateur est null, le peripherique ne supporte pas le bluetooth
5 if (mBluetoothAdapter == null) {
6     // on affiche un message sur l'ecran de l'utilisateur.
7     Toast.makeText(this, "Bluetooth indisponnible", Toast.LENGTH_LONG).show();
8 return;
9 }
```

2.2 Activer l'adaptateur

Une fois l'adaptateur récupéré, il faut vérifier qu'il soit utilisable, et donc activé.

```
1 // si le bluetooth n'est pas activer
2 if (!mBluetoothAdapter.isEnabled()) {
3 // on affiche une fenetre pre-cree permettant sont activation
4     Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
5     startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
6 }
```

2.3 Liste des péripherique

Il faudrai au préalable crée un widget permettant l'affichage de la liste des périphérique, mais ce serai trop s'éloigner du sujet, ce ne sera donc pas vu dans ce cours.

2.3.1 Récupérer la liste des périphérique connu

Récupérer la liste des périphérique connu se fait de manière tres simple puisqu'il existe déjà une méthode l faisant à notre plaçe :

```
1 // on d\'eclare une collection qui stockera la liste
2 Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
3 // si on a detecte quelque chose
4 if(pairedDevices > 0) {
5     for(BluetoothDevice device : pairedDevices) {
6         // on ajoute les peripherique a la liste / au widget / etc...
7         mArrayAdapter.add(device.getName() + "\n" + device.getAdress());
8     }
9 }
```

2.3.2 Découvrire des périphériques

La découverte de périphérique est assé complexe [a finir]

```
1  private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
2      public void onReceive(Contect context, Intent intent) {
3          String action = intent.getAction();
4          if(BluetoothDevice.ACTION_FOUND.equals(action)){
5              BluetoothDevice device =
6              intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
7              mArrayAdapter.add(device.getName() + "\n" + device.getAdress());
8          }
9      }
10 };
11 IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
12 registerReceiver(mReceiver, filter);
```

Chapitre 3

Utilisation

Pour utilisé un périphérique bluetooth, il faut re-implanté la classe Thread :

```
1 private class ConnectedThread extends Thread {
2     ...
3 }
```

Il faut aussi lui mettre des attribut, corespondant au socket bluetooth et au flux entrant/sortant :

```
1 private final BluetoothSocket mmSocket;
2 private final InputStream mmInStream;
3 private final OutputStream mmOutStream;
```

Toute class a un constructeur, celle la prend en paramètre un socket bluetooth et initialise les flux avec objet temporaire (encapsulation, toussa toussa) :

```
1  public ConnectedThread(BluetoothSocket socket) {
2      mmSocket = socket;
3
4      // on cree les flux temporaire
5      InputStream tmpIn = null;
6      OutputStream tmpOut = null;
7
8      // on tente de recuperer les flux d'entree sortie du socket bluetooth
9      try {
10         tmpIn = Socket.getInputStream();
11         tmpOut = Socket.getOutputStream();
12     } catch(IOException e)
13     {
14         ...
15     }
16
17     // on initialise les flux de la class avec les flux temporaire
18     mmInStream = tmpIn;
19     mmOutStream = tmpOut;
20 }
```

3.1 Écriture

Écrire et donc envoyer des donné par l'intermédiaire du bluetooth est extremement simple, il suffit d'implementer une méthode qui prend en paramètre un tableau d'octet et qui se chargera de l'envoyer au flux de sortie :

```
1 public void write(byte[] bytes){
2     // on tente d'ecrire les donnees
3     try{
4         mmOutStream.write(bytes);
5     } catch(IOException e){
6         ...
7     }
8 }
```

3.2 Lecture

La lecture de données est beaucoup plus complexe, la thread tourne en continue à l'ecoute de données (c'est d'ailleur l'utilité de la thread, l'écoute et l'écriture étant bloquante, elles doivent s'éxecuter séparement), il va donc

falloire une variable de la taille d'un byte, permettant de recuperer une a une toute les données, un buffer pour cette même récuperation, et executer tout sa en boucle jusqu'à ce que le programme s'arrête ou qu'il y ai une erreur :

```
1  public void run(){
2      // notre buffer
3      byte[] buffer = new byte[1024];
4      int bytes;
5
6      while(true) {
7          try {
8              // on lit le flux
9              bytes = mmInStream.read(buffer);
10             /* on envoie les donnees la ou elle doivent aller */
11         } catch(IOException e) {
12             break;
13         }
14     }
15 }
```

3.3 Néttoyage

Il faut toujour tout nettoyer avant de quitter une application, le garbage collector fait une grosse partie du travaille, mais il ne peu s'attaquer à de la mémoire encore réfférencé, il faut donc le faire avant de quitter :

```
1  public void cancel() {
2      try{
3          mmSocket.close();
4      } catch(IOException e) {
5          // on peu afficher un message d'erreur ou ecrire dans d'eventuel log
6      }
7  }
```