TP03 (2h30) *Connexion à un Arduino*

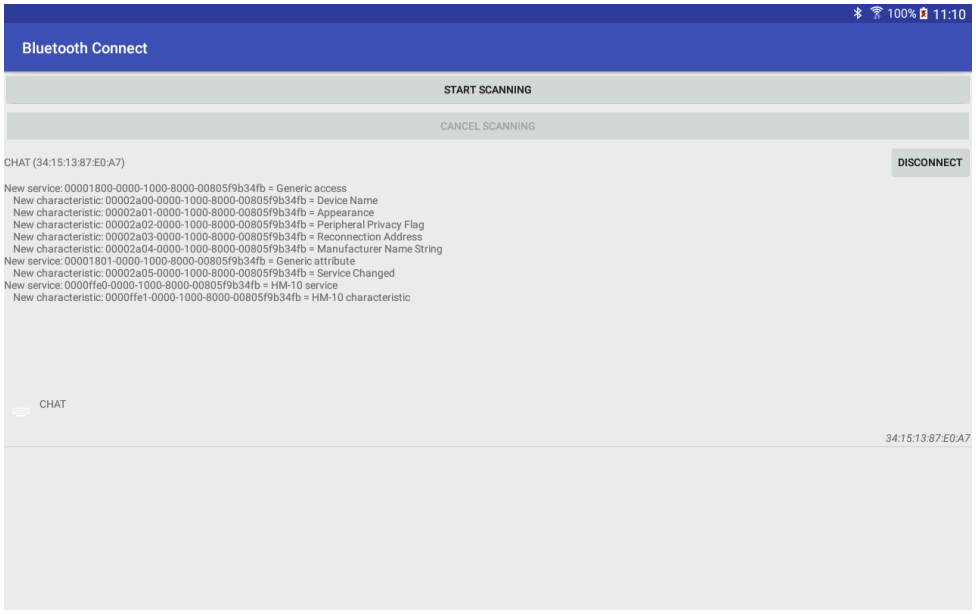
Ce TP consiste en la réalisation d'une application `Android` capable de se connecter à un micro-contrôleur Arduino en Bluetooth. À l'issue de ce TP, vous devez avoir compris et acquis les compétences suivantes :

— Connexion à un serveur GATT : `BluetoothGattCallback`

— Lecture des données émises pas l'Arduino : *service* et *characteristic*

I. Création de l'interface

Pour concevoir l'interface, il suffit de reprendre celle du TP précédent en y ajoutant une zone (au dessus de la liste des appareils découverts) permettant d'afficher les informations de la connexion courante. Cette zone est composé d'un `TextView` affichant l'appareil (un Arduino) sur lequel la tablette est connectée, un bouton pour se déconnecter et une zone de texte (un autre `TextView`) pour afficher les données lues depuis l'Arduino. Attention, la zone de texte doit être défilante, pour cela il est nécessaire de la mettre dans un composant `ScollView`. Voici une copie d'écran de ce que vous devez obtenir :



Lorsque l'utilisateur appuiera sur un élément de la liste, l'application tentera de se connecter à celui-ci.

> Pour tester que votre `TextView` fonctionne, n'hésitez pas à ajouter un long texte et voir comment il réagit. Le composant `TextView` d'`Android` possède des attributs lui permettant de bien s'intégrer dans un `ScrollView`.

II. Connexion à l'Arduino

Connexion à un serveur GATT

Dans le TP précédent, nous avons vu comment chercher les *devices Bluetooth Low Energy* (les appareils à portée). Chaque appareil BluetoothLE serveur héberge un serveur GATT mettant à disposition des clients des services et propriétés (*service* et *characteristic*). Le composant `HM-10` branché sur notre Arduino héberge un

serveur GATT. On peut donc s'y connecter. Pour lancer la connexion à un appareil, l'utilisateur aura juste à appuyer sur la ligne correspondante dans la liste des appareils trouvés lors de la détection.
La connexion s'effectue de manière asynchrone. `Android` fournit toutes les classes et méthodes nécessaires à l'établissement de cette connexion. La méthode `connectGatt()` d'un objet `BluetoothDevice` tente d'établir une connexion au serveur GATT hébergé sur le device. Cette méthode possède trois paramètres :

- Une référence sur un contexte (ou activité)
- Un booléen permettant de gérer la reconnexion (pour l'instant, on mettre la valeur `false`)
- Un objet utilisé pour répondre à chaque évènement levé pendant la connexion au serveur GATT

Le paramètre sur lequel nous allons nous concentrer est le dernier. Pour instancier un tel objet, il va falloir créer une classe qui va hériter de la classe `BluetoothGattCallback`. Cette classe va définir tous nos *callback*, c'est à dire nos réactions face à un nouvel évènement survenu lors de la connexion au serveur GATT. La première méthode à redéfinir est : `onConnectionStateChange()`. Cette méthode est appelée à chaque changement d'état de la connexion. Il vous ait demandé d'afficher un message dans les logs lorsque la connexion s'est bien établie. Pour cela, il faut vérifier que l'état de connexion est égale à la valeur `BluetoothProfile.STATE_CONNECTED`. Vous pouvez ensuite afficher dans les logs des messages lorsqu'une erreur est captée et indiquer systématiquement l'adresse `MAC` de l'appareil bluetooth concerné.
Une fois la connexion établie, il faudra mettre à jour l'interface pour correspondre avec la copie d'écran précédente. Hors, tous les appels aux méthodes de rappel de la classe `BluetoothGattCallback` sont appelées dans un thread séparé, il n'est donc pas possible de mettre à jour l'interface graphique directement depuis ces fonctions. Pour cela, il faut créer un `Handler`. Un `Handler`, peut-être considéré comme une sorte de boîte aux lettres réceptionnant et analysant ses messages. Plus précisément, lorsqu'un objet `Runnable` est envoyé à un `Handler`, la méthode `run()` de l'objet est appelée. Une fois que votre objet `Handler` est défini comme attribut de votre activité, vous pouvez lui envoyer des messages (des objets). Vous pourrez ainsi envoyer des objets `Runnable` depuis le thread des callbacks vers le thread principal de l'interface pour faire les mises à jour.

Découverte des services proposés par l'Arduino

Nous avons vu dans la partie cours qu'un serveur GATT proposait des services et pour chacun de ces services, un ensemble de propriétés. Nous allons maintenant afficher dans la zone de texte prévue à cet effet (le `TextView` prévu pour afficher les données lues) la liste des services et propriétés proposées par notre Arduino. Vous pouvez voir sur la copie d'écran précédente un exemple de ce que vous devez obtenir.
La découverte des services se fait simplement en appelant la méthode `discoverServices()` de votre objet `BluetoothGatt`. Cet appel devra être effectué quand la connexion au serveur GATT est établie avec succès. Comme pour la connexion, une méthode de l'objet `BluetoothGattCallback` est appelée lorsque les services ont été détectés. Il vous restera ensuite à parcourir la liste des services découverts et de les affichés. N'oubliez pas de parcourir la liste des propriétés de chaque service !
Les services et propriétés sont identifiés par des UUID. Pour faire la conversion d'un UUID vers un texte plus explicite, vous pouvez utiliser la classe `Sample_gatt_attributes.java` (fournie avec le sujet) qui contient une description des services et propriétés les plus couramment rencontrés. Elle contient également un descriptif pour les services et propriétés du composant `HM-10` que vous utilisez afin de vous faciliter la tâche.

Lecture des données de l'Arduino

Une fois la connexion établie, il faut lire les données provenant de l'Arduino. Pour ce TP, l'Arduino enverra, à chaque seconde, le temps écoulé depuis son allumage. Voici un extrait du code exécuté par l'Arduino :

```
time = millis(); // lire le temps en ms écoulé
if ( (time %1000) == 0) // chaque seconde
{
  seconde = time / 1000;
  BT.println(seconde); // le transmettre en Bluetooth avec \r\n
  ...
}
```

Le code précédent mettra à jour toutes les secondes la propriété nommée "HM-10 characteristic" dans le fichier `Sample_gatt_attributes.java`. Il vous faudra donc être informé à chaque mise à jour de la propriété.
Pour activer les notifications sur une propriété, il faut appeler la méthode `setCharacteristicNotification()` sur votre objet `BluetoothGatt` en lui passant en argument la propriété que vous souhaitez observer et la valeur `true`. Ainsi, la méthode `onCharacteristicChanged()` de votre objet `BluetoothGattCallback` sera exécutée à chaque changement de valeur de la propriété. Attention, le message reçu est un tableau d'octets qu'il vous faudra convertir en chaîne de caractères afin d'avoir un affichage correct.
Pour afficher la valeur numérique envoyée par l'Arduino, vous utiliserez la zone texte qui a servi à afficher la liste des services et propriétés.