Rapport Projet EB03

Réalisation d’une application oscilloscope sous Android

Mathieu DAUDIN

Sébastien PICOCHE

Objectif : Le but de ce projet était de pouvoir communiquer en bluetooth avec un oscilloscope et de réussir à recevoir et envoyer des informations à celui-ci.

1. Etat d’avancement

Au cours de ce projet nous avons pu aller jusqu’à la lecture et l’affichage des données reçues de l’oscilloscope sur un graphe sur l’application Android. Nous arrivons aussi à gérer la luminosité de la DEL avec un slider qui change la calibration.

Cependant nous n’avons pas pu gérer l’amplitude du signal et le nombre d’échantillons affichés sur le graphe. En effet lorsque nous recevons les trames bluetooth de l’oscilloscope et que nous les mettons en forme, le nombre de points varie continuellement et nous n’avons jamais 500 points.

Nous arrivons tout de même à reconnaître la forme d’un signal carré ou sinusoïdal mais avec un nombre de périodes différent à chaque lecture de données.

1. Fonctionnement espéré

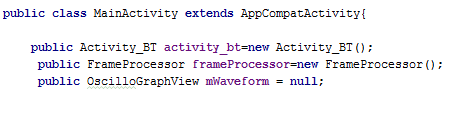
Dans un premier temps il faut cliquer sur le bouton bluetooth en haut à droite qui nous emmène sur une nouvelle activité. Dans cette activité il y a une liste des appareils déjà appairés et une liste des appareils bluetooth visibles. Il faut cliquer dans cette dernière liste et choisir l’oscilloscope. Il faut ensuite attendre que la DEL de connexion de l’oscilloscope cesse de clignoter et appuyer sur le bouton retour pour retourner sur l’activité principale.

Dans cette activité principale il faut appuyer sur le switch « ch1 » pour ouvrir l’envoi de données sur ce canal et sur l’oscilloscope nous pouvons alors observer le signal. En appuyant de nouveau sur le switch « ch1 » nous fermons la communication avec le canal 1.

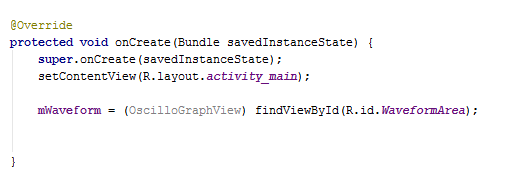
Le slider permet de faire varier l’intensité de la DEL.

1. Articulation des classes et fonctionnement de différentes classes
2. La classe MainActivity

La classe MainActivity étend de AppCompatActivity qui est la classe de base pour les activités. Nous créons ensuite 3 attributs :

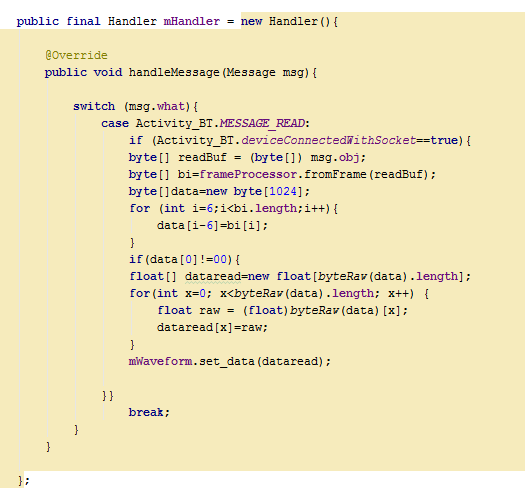
* Un qui instancie l’activité bluetooth
* Un qui instancie un objet de type frameProcessor pour la gestion des trames bluetooth
* Un pour le graphe de l’oscillateur

Nous sur chargeons la méthode onCreate qui va définir le contenu de l’activité lors de sa création :

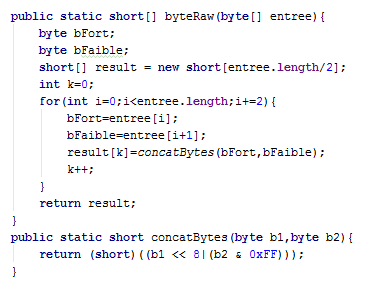


Nous créons ensuite un handler qui va permettre de mettre à jour l’UI (le graphe de l’oscilloscope) avec des données issues du thread « connectedThread » que nous verrons plus tard. La lecture des données ne peut s’effectuer seulement si un socket a été échangé entre la tablette et l’oscilloscope. Le connectedThread va lire en permanence les données issues de l’oscilloscope et va les mettre dans un message que nous récupérons à l’aide ce handler (msg.obj). Ce message est mis dans une variable readBuffer. Ensuite nous décodons ce buffer à l’aide la méthode fromFrame que nous verrons plus tard. La méthode fromFrame nous renvoie un buffer dont les valeurs de tensions commencent à partir du rend 7, les informations précédentes sont pour l’offset etc.. Nous remplissons donc un nouveau tableau « data » qui va inclure seulement les valeurs de tensions.

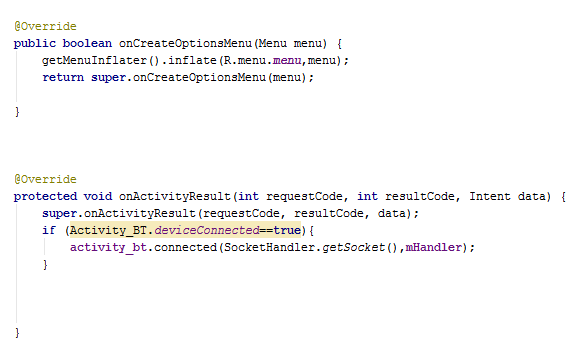
Ensuite nous utilisons la méthode byteRaw qui va permettre de réunir deux à deux ces valeurs pour obtenir les données finales (dataread). Nous actualisons ensuite le graphe avec ces données.



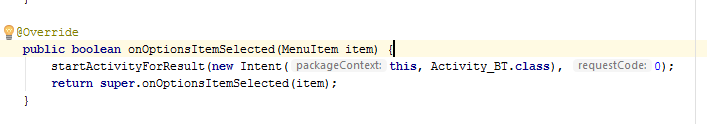
La méthode byteRaw prend en entrée un tableau de bytes qui vont être réunit deux à deux successivement pour obtenir un tableau de short à la fin du processus. Ce tableau de short représentera les valeurs de tensions. Ces valeurs de tension sont castées en float ci-dessus pour le traçage de la coube sur loscilloscope.



Nous créons ensuite le menu dans la barre de menu et lorsque nous nous sommes connectés dans l’activité bluetooth, en revenant sur l’activité principale, nous lançons le thread pour écrire et lire les données de l’oscilloscope qui prend en paramètres le socket et le handler.

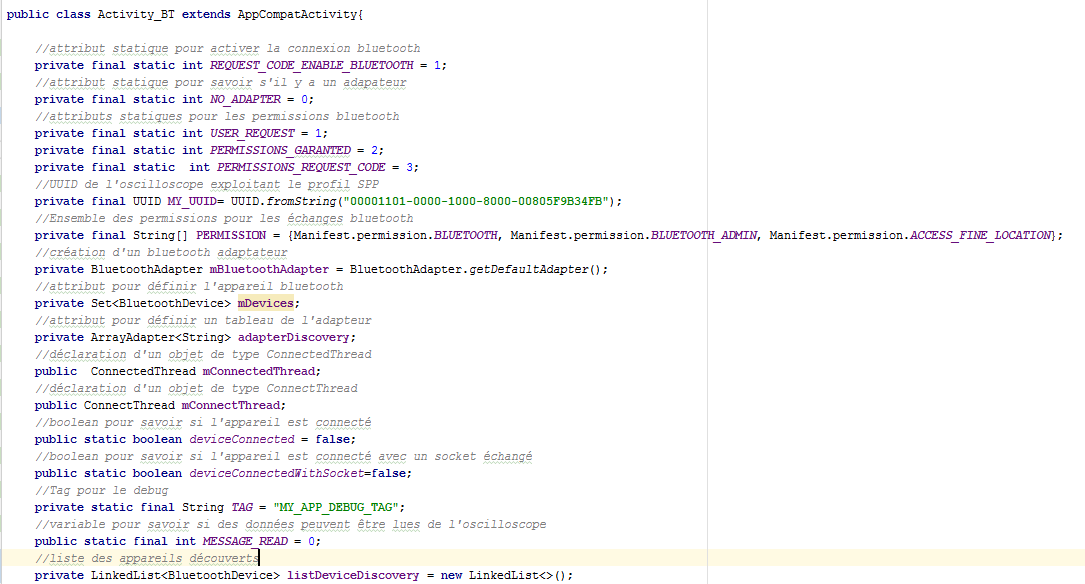


Ensuite lors de l’appui sur le symbole bluetooth, nous lançons l’activité « Activity\_BT » en attendant des résultats.



1. La classe Activity\_BT

Nous créons une classe pour lancer l’activité de connexion bluetooth. Cette classe étend de AppCompatActivity explicitée plus haut. Nous définissons les attributs suivants :



Ensuite nous implémentons la méthode onCreate qui s’exécute lors de la création de l’activité.

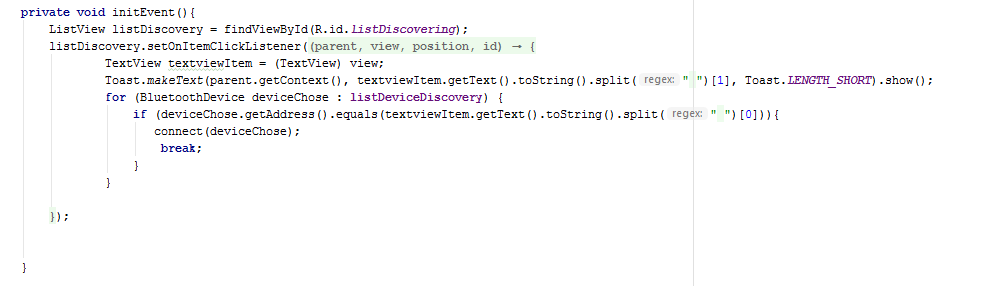
Dans un premier temps cette méthode va mettre en place le layout défini pour cette activité. Ensuite nous lançons une méthode BlutoothConfig que nous verrons plus tard. Ensuite nous initialisons les évènements que nous verrons plus tard.

Ensuite nous initialisons un nouveau ArrayAdapter qui va permettre de mettre en forme et de rendre dynamique les éléments de la liste des appareils découverts et appairés. Nous créons ensuite un filtre pour filtrer l’intention en fonction de si l’appareil bluetooth est trouvé. Nous démarrons ensuite l’intent découvrable pour l’appareil puisse être trouvé en bluetooth.

Si toute les permissions bluetooth sont acceptées, nous recevons l’intention découvrable et nous disons dans quelle liste les appareils découverts doivent apparaître. Nous lançons ensuite la découverte des appareils bluetooth.



La méthode initEvent permet de définir le clic sur les éléments de la liste des appareils découverts. Nous parcourons la liste des appareils découverts et si la chaine de caractère de l’élément sélectionné contient une adresse MAC présente dans la liste des appareils découverts alors nous lançons le thread de connexion à cet appareil.

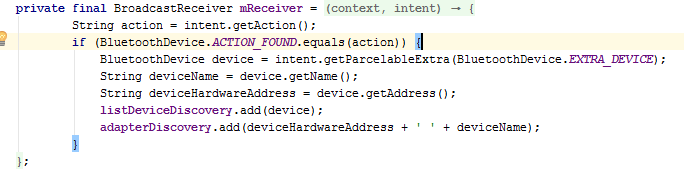


Nous définissons ensuite la méthode pour les permissions bluetooth. Si l’adaptateur bluetooth vaut null alors nous signifions qu’il n’y a pas d’adaptateur bluetooth. Puis nous définissons l’adaptateur bluetooth avec l’adaptateur par défaut.

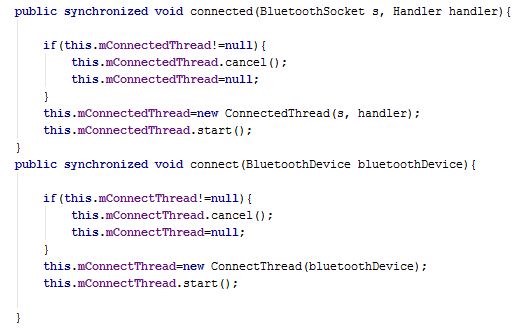
Si la version d’android est supérieur à une certaine valeur nous demandons des autorisations supplémentaires pour que le bluetooth fonctionne correctement. Si toutes les autorisations sont acceptées alors nous le signifions grâce à la variable PERMISSIONS\_GARANTED.



Nous définissons ensuite le broadcast receiver pour recevoir des informations de l’intent de découverte des appareils bluetooth. Si la découverte a trouvé un appareil bluetooth alors nous récupérons l’objet « device » de cette intention et nous en extrayons son nom et son adresse MAC. Nous ajoutons ensuite cet appareil à la liste des appareils découverts et nous ajoutons à l’adapter la chaîne de caractère avec l’adresse MAC et le nom de l’appareil séparés par un blanc.



Nous définissons ensuite les méthodes publiques synchronisées pour lancer les threads pour se connecter en bluetooth et pour la lecture et l’écriture de données en bluetooth une fois que l’on est connecté en bluetooth. Si les threads sont lancés nous les arrêtons et si les threads ne sont pas lancés, nous les démarrons. Ceci est fait pour éviter tout conflit.

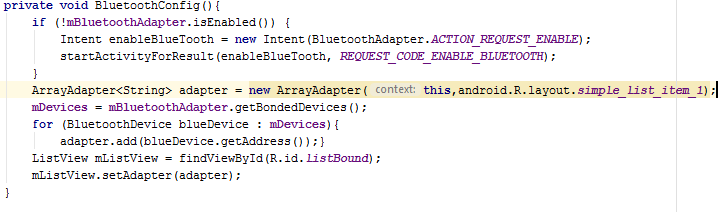


Nous définissons ensuite une méthode publique pour écrire des données sur l’appareil bluetooth.

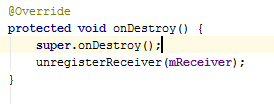


Nous avons ensuite la méthode BluetoothConfig. Si l’adapter bluetooth n’est pas autorisé, nous créons une nouvelle intention pour demander d’autoriser le bluetooth. Puis nous démarrons cette intention qui attend un résultat pour savoir si l’utilisateur a autorisé le bluetooth.

Nous remplissons aussi dans cette méthode la liste des appareils déjà appairés sur le même principe que pour la liste des appareils découverts.



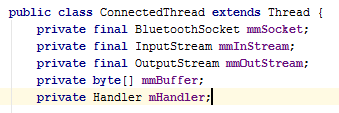
Nous avons ensuite une méthode pour supprimer le receveur de l’ACTION\_FOUND.



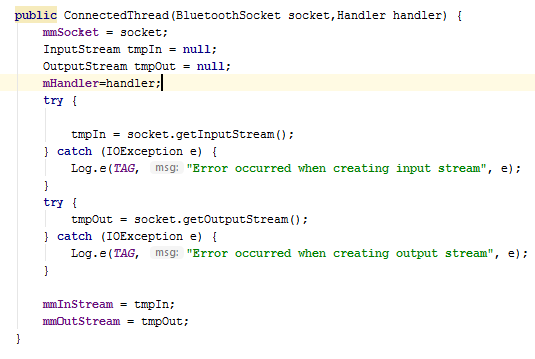
A l’intérieur de la classe Activity\_BT nous avons une classe interne connectedThread qui hérite de Thread qui est un thread pour gérer l’écriture et la lecture de données issue de l’appareil bluetooth connecté.

Cette classe interne possède 5 attributs :

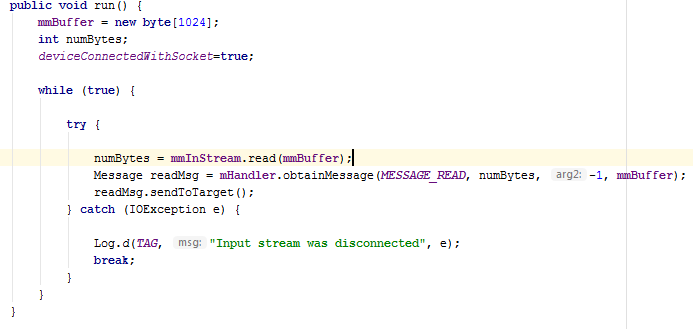
* Un socket
* Un inputStream pour les données entrantes
* Un outputStream pour les données sortantes
* Un buffer pour stocker les données envoyées par l’appareil bluetooth
* Un handler pour faire passer les données lues dans le thread dans l’UI et actualiser celle-ci.



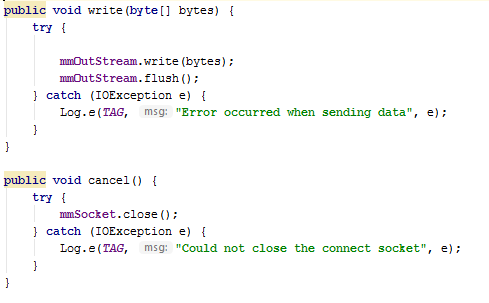
Cette classe possède le constructeur suivant avec comme paramètres un socket et un handler. Dans un premier nous essayons d’obtenir les inputStreams et les outputStreams pour pouvoir par la suite écrire et lire des données de l’appareil.



Nous définissons ensuite la méthode run de ce thread. Nous avons le buffer pour stocker les informations lues de l’oscilloscope, le nombre de bytes lus et nous passons la variable deviceConnectedWithSocket à vrai pour indiquer que la communication avec l’appareil est maintenant ouverte. Nous lisons ensuite les informations lues sur l’inputStream et nous envoyons le message obtenu à l’UI. Cette lecture est faite continuellement pour ne rater aucune donnée.



Nous avons ensuite une méthode write pour écrire des données sur l’appareil connecté qui prend en paramètre un tableau de bytes. Ces données sont écrites sur l’outputStream. Nous avons enfin la méthode cancel pur arrêter le thread.



Nous avons ensuite la définition d’une autre classe interne pour établir la connexion