



UTBM – *Université de Technologie de Belfort-Montbéliard*

A2020

LO52 - Rapport Travaux Pratiques

TP1 : Mise en place de l'environnement de développement.....	3
1. Environnement de base	3
2. Découverte d'Android Studio et développement d'une application HelloWorld.....	4
TP2 : Manipulation d'un Kernel Linux.....	6
1. Environnement de configuration du Kernel.....	6
2. Nouvelle configuration de Kernel.....	6
3. Différences entre les configurations	7
4. Bonus : Création d'images.....	7

*Aymeric Robitaille,
Anaïs JARNO*

*UTBM – A2020
LO52 – Rapport de TP*

Auteurs

Anaïs JARNO
Aymeric ROBITAILLE

Encadrants

Fabien BRISSET
Pierre ROMET

TP1 : Mise en place de l'environnement de développement

1. ENVIRONNEMENT DE BASE

L'objectif, ici, est de mettre en place un environnement propice au développement d'applications Android.

La première étape était de se procurer un dépôt Git pour la gestion des sources.

Pour cela, il faut bien évidemment commencer par installer Git si ce n'est pas fait. Dans notre cas, Git était déjà installé sur nos machines. Aussi, il nous restait juste à configurer Git pour nous identifier lors des commits. Ainsi, nous avons exécuter les commandes suivantes pour la configuration de notre nom et adresse mail :

```
git config --global user.name "xxx"  
git config --global user.email xxxx@xxx.com
```

Ensuite, le dépôt Git avait déjà été créé pour tous les élèves de l'UV LO52. Il nous fallait donc le récupérer en local. Nous avons ainsi cloné le dépôt à l'aide de la commande suivante :

```
git clone https://github.com/gxfab/LO52\_A2020.
```

Cela a créé un dossier `LO52_A2020`, lié à nos sources Git, dans lequel nous nous sommes positionnés pour la suite du travail.

Pour différencier notre travail de celui des autres groupes, nous avons créé une branche portant le nom des différents membres de notre groupe de travail, et nous nous sommes directement positionnés dessus pour être prêts à débiter le travail :

```
git checkout -b RobitailleAymeric_JarnoAnais
```

Enfin, après avoir reporté dans le fichier texte TP1.txt, les commandes réalisées jusque-là, nous avons mis à jour le dépôt distant en uploadant nos changements (branche et modifications) :

```
git add . && git commit -m "TP1 Création de branche »  
git push -u origin RobitailleAymeric_JarnoAnais
```

La seconde étape consistait à installer/configurer l'environnement de développement Android Studio.

À ce niveau-là, nous avons eu un problème général concernant le proxy UTBM et l'installation du SDK. Bien que ces soucis aient été résolus par la suite, nous avons eu le temps d'installer Android Studio sur nos machines personnelles et avons donc poursuivi le TP dessus. La configuration fut rapide, elle est simple et très bien guidée par l'assistant d'installation de l'IDE.

2. DÉCOUVERTE D'ANDROID STUDIO ET DÉVELOPPEMENT D'UNE APPLICATION HelloWorld

L'objectif de cette deuxième partie de TP était de prendre en main l'IDE, comprendre le fonctionnement et la liaison entre les fichiers d'une application réalisée avec Android Studio, via le développement d'une application HelloWorld.

Cette application devait se constituer de deux principales vues, ici on nommera ces vues des Activités. Elle doit comprendre une première Activité ne possédant qu'un simple bouton. Et sur clic de ce bouton, on est redirigé vers une seconde Activité comprenant le texte « Hello World ».

Pour cela, nous avons tout d'abord créé un projet que nous avons placé dans le dossier sources de TP1. L'assistant de création de projet, nous a permis de créer directement un projet comprenant une activité vide – *Empty Activity*. Nous avons choisi de travailler avec le langage Kotlin ce qui nous permet de suivre les dernières recommandations de Google et donc les tendances de programmation actuelles.

Nous disposons ainsi déjà d'une activité MainActivity. Celle-ci comprend deux composants : MainActivity.kt (fichier kotlin présent dans le dossier de sources java) et activity_main.xml (dans les ressources, plus précisément dans le dossier de layout).

Le fichier activity_main.xml détaille les composants de la vue. Nous lui avons donc simplement ajouté un Button depuis l'assistant graphique et design. Dans le fichier de ressource string.xml, nous avons ajouté une nouvelle entrée, navigate_hello_activity correspondant au texte « Naviguer vers une nouvelle activité ». Nous avons ensuite assigné ce string à la propriété « text » du bouton de MainActivity. Finalement, nous avons contraint ce bouton de manière à ce qu'il soit au milieu de l'écran.

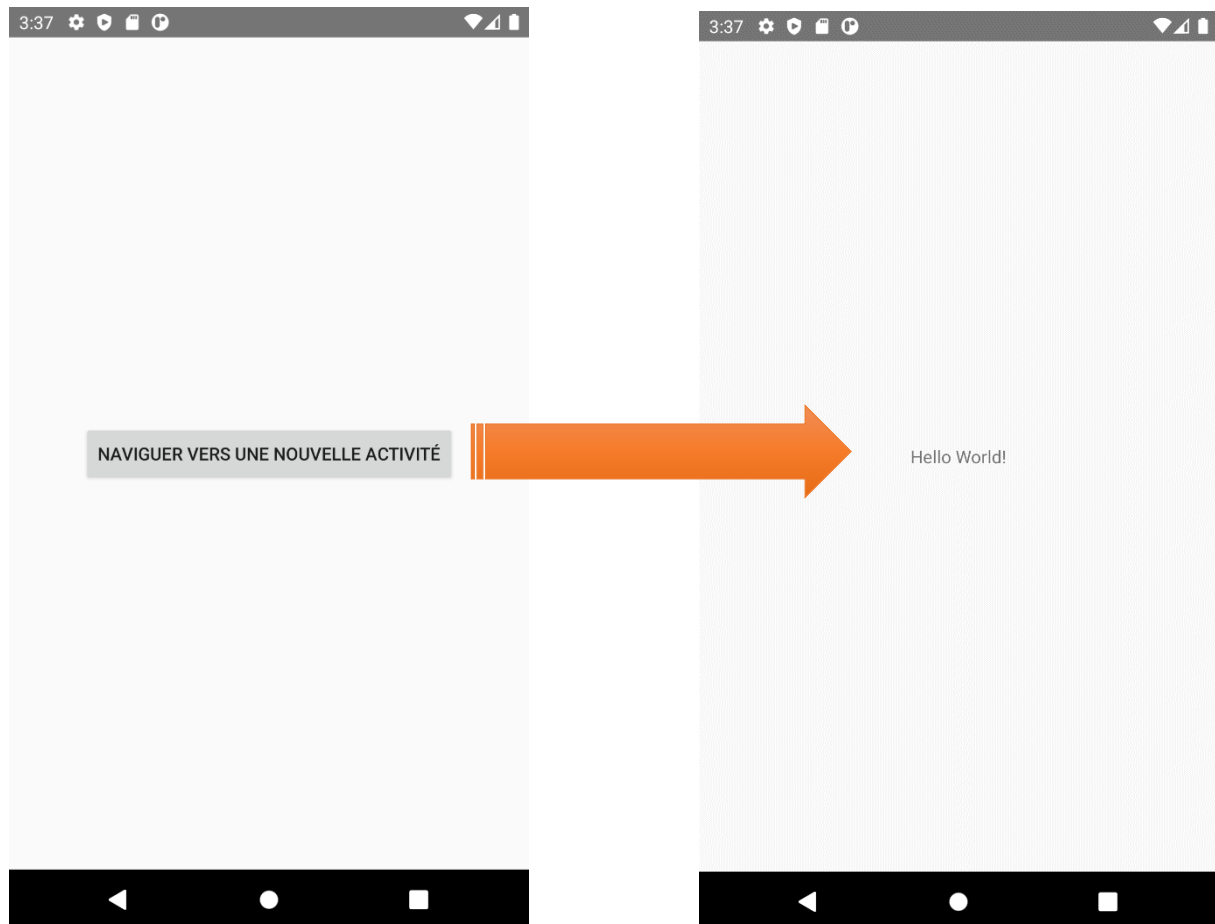
Ensuite, nous avons créé une autre activité dans laquelle nous avons ajouté un TextView avec le texte « Hello World » et les mêmes contraintes, pour qu'il soit au milieu.

Nos deux activités étant à présent créées, nous avons ouvert le code de MainActivity.kt pour y ajouter la méthode « navigateToHello ». Nous avons ensuite lié cette méthode au layout du MainActivity en tant que propriété « onClick » du bouton. De cette manière, le code sera exécuté lorsque le bouton est appuyé. Le code de cette méthode est simple et se résume à un changement d'Activité en appelant la méthode « startActivity » de la manière suivante :

```
fun navigateToHello(view: View) {  
    val intent = Intent(this, HelloActivity::class.java)  
    startActivity(intent)  
}
```

Nous avons pu tester l'application aussi bien sur un téléphone Android que sur un émulateur directement lié à l'IDE. Après avoir lancé l'application, nous avons pu constater que l'activité avec le texte était bien lancée lors de l'appui sur le bouton.

Ainsi, on obtient le résultat suivant :



TP2 : Manipulation d'un Kernel Linux

1. ENVIRONNEMENT DE CONFIGURATION DU KERNEL

L'objectif, ici, est de mettre en place un environnement propice au développement d'applications Android.

```
sudo apt-get install repo
mkdir android-kernel && cd android-kernel
repo init -u https://android.googlesource.com/kernel/manifest -b hikey-linaro-android-4.19
repo sync

cp arch/mips/configs/generic/board-ranchu.config .
make board-ranchu.config

repo status
repo diff
```

Ainsi, nous avons regroupé ces diverses commandes dans un script auquel on a ajouté la commande `set -e` pour interrompre le script en cas d'erreur.

2. NOUVELLE CONFIGURATION DE KERNEL

Par la suite, nous avons à modifier la configuration précédemment obtenue afin d'assurer certaines fonctionnalités. Pour cela, nous avons donc dû :

- Assurer la compatibilité pour la carte ARMv8 Versatile, Qualcomm et Realtek
- Activer le NFC et le protocole NFC HCI
- Activer l'option Frequency Scaling de la CPU
- Activer le support de l'HDML CEC et activer le support LED

Par la suite, il nous a été demandé d'optimiser la configuration en désactivant toutes les autres options superflues. Au premier abord, il nous a été très complexe de savoir si une option pouvait, ou non, être considérée comme « superflue ». Finalement, voici les différentes optimisations qui ont été mises en œuvre :

- Suppression des plateformes non demandées
- Désactivation des options qui suivent :
 - PCI, on suppose ici qu'aucun slot PCI n'est disponible sur la carte
 - Support de l'UEFI
 - Bluetooth
 - Plan 9 Ressource sharing
 - Carte son
 - Support de carte SD/MMC/SDIO

- DRI, étant donné que nous n'avons aucune information particulière sur le module de la carte graphique



Nous avons également supposé que ça ne poserait pas de problème au niveau de Dalvik si nous supprimions en plus la virtualisation au niveau du kernel.

3. DIFFÉRENCES ENTRE LES CONFIGURATIONS

Nous avons enregistré les différentes configurations dans un dossier nommé configs/ du TP2. On retrouve ainsi les fichiers de configuration sous les noms suivants :

- *hikey*, la configuration de base
- *ranchu*, la configuration de base merge avec board-ranchu.config
- *custom*, la configuration de ranchu avec les activations demandés (armv8, hdmi, nfc, led, ...)
- *custom_opti*, la configuration custom suite à la suppression d'options superflues

Quant aux différences remarquées entre les différentes configurations, on peut les retrouver dans le tableau suivant :

Configuration	Hikey	Ranchu	Custom	Custom_opti
Hikey				
Ranchu	Ajout de la configuration pour une carte ranchu			
Custom		Activation des options telles que : ARMv8, HDMI, NFC, LED...		
Custom_opti			Suppression des options superflues	

4. BONUS : CRÉATION D'IMAGES

Les images compressées ont été enregistrées dans le dossier boot/ du TP2.