# TP1

* Télécharger GitHub Desktop et Git pour Windows.
* GitHub Desktop -> File -> Clone repository -> coller le lien copié depuis le github <https://github.com/gxfab/LO52_A2019.git>
* $git config –global user.name “xxx”;

$git config –global user.email [xxx@example.com](mailto:xxx@example.com)

(Linux)

* GitHub desktop -> Branche -> Name : FGurlsDev -> Create branch based on master
* GitHub desktop -> Current branch -> FGurlsDev
* Créer directement un répertoire dans le répertoire cloné s’appelle LO52\_2019\_FGurlsDev, modifier le fichier README.txt.
* Télécharger et installer AndroidStudio.
  + Java (jdk, jre)
  + SDK
* Android Studio -> Start a new Android Studio project -> Configurer les informations du projet -> Empty Activity (MainActivity.java)

Rajouter un bouton dans le layout principal (activity\_main.xml).

Configurer l’action de bouton (MainActivity.java, setOnClickListener)

Créer le deuxième activité (SecondActivity.java)

Rajouter un label dans le layout secondaire (activity\_second.xml)

* + Remarques :
    - Pour utiliser l’activité existée pour activer la nouvelle activité, la nouvelle activité n’est pas encore créée, donc nous devons utiliser newActivity.class mais pas newActivity.this.
    - ‘This’ est une référence d’un membre de l’objet actuel à partir d’une méthode d’instance ou d’un constructeur.
  + Problème : Android Studio n’arrive pas à retrouver le divice HUAWEI
    - Raison et Solution : drive de huawei ne peut pas être utilisé par Android Studio, donc télécharger Google USB Driver dans SDK Manager – Google USB Driver.
  + Pour changer la nouvelle activité, utiliser la classe Intent.
* Copier le répertoire src dans le repertoire LO52\_A2019/LO52\_2019\_FGurlsDev/Part 1

(TP2 est supprimé)

# TP3

* Implémentation de la libusb :
  1. Fichiers sources : core.c, descriptor.c, io.c, sync.c, os/darwin\_usb.c, os/linux\_usbfs.c

Fichiers headers : libusb.h, libusbi.h, os/darwin\_usb.h, os/linux\_usbfs.h

* 1. Modifier le fichier Rsc-libusb/libusb-1.0.3/libusb/Android.mk (MakeFile pour le composant libusb) :

LOCAL\_PATH:=$(call my-dir)

include $(CLEAR\_VARS)

LOCAL\_SRC\_FILES:= \

core.c \

descriptor.c \

io.c \

sync.c \

os/darwin\_usb.c \

os/linux\_usbfs.c

LOCAL\_C\_INCLUDES+= \

$(LOCAL\_PATH) \

$(LOCAL\_PATH)/os

LOCAL\_MODULE:=libusb

include $(BUILD\_SHARED\_LIBRARY)

* 1. Solution pour l’erreur TIMESPEC\_TO\_TIMEVAL - Rajouter les codes suivant dans le fichier io.c :

#define TIMESPEC\_TO\_TIMEVAL(tv, ts) do { \

(tv)->tv\_sec = (ts)->tv\_sec; \

(tv)->tv\_usec = (ts)->tv\_nsec / 1000; \

} while (0)

* 1. Solution pour l’erreur library "libusb.so" not in prelink map – Rajouter la ligne suivante dans le fichier Android.mk : LOCAL\_PRELINK\_MODULE:=false
* Implémentation d’un nouveau produit Android :
  1. Renommer le fichier Rsc-libusb/device/TP3\_A\_VOUS\_JOUEZ.mk comme lo52\_FGurlsDev.mk
  2. Modifier ce fichier :
     + Configurer les informations simples de produit :

PRODUCT\_NAME:=lo52\_FGurlsDev

PRODUCT\_DEVICE:=lo52\_FGurlsDev

PRODUCT\_BRAND:=lo52\_FGurlsDev

PRODUCT\_MODEL:=lo52\_FGurlsDev

* + - Il héritera du produit hikey de Linaro :

$(call inherite-product,device/Linaro/hikey/hikey.mk)

* + - Personnaliser les propriétés :

PRODUCT\_PROPERTY\_OVERRIDES+= \

ro.hw=lo52 \

net.dns1=8.8.8.8 \

net.dns2=4.4.4.4

* Surcharger le fichier sym\_keyboard\_delete.png
  1. Chercher “aosp sym\_keyboard\_delete.png” sur Google;
  2. Cliquer sur premier lien, le fichier peut être trouvé dans le projet platform\_frameworks\_base;
  3. Créer un répertoire **overlay** dans le répertoire **Rsc-libusb/device**;
  4. Créer les répertoire avec la même arborescence mirror comme ce que l’on a trouver dans le projet :

overlay/platform\_frameworks\_base/core/res/res/drawable-en-mdpi

* 1. Rajouter cette ligne de code dans le fichier lo52\_FGurlsDev.mk :

DEVICE\_PACKAGE\_OVERLAYS:=overlay

* Ajouter la libusb aux packages du produit : PRODUCT\_PACKAGES+=libusb
* Commit les modifications sur la branche FGurlsDev