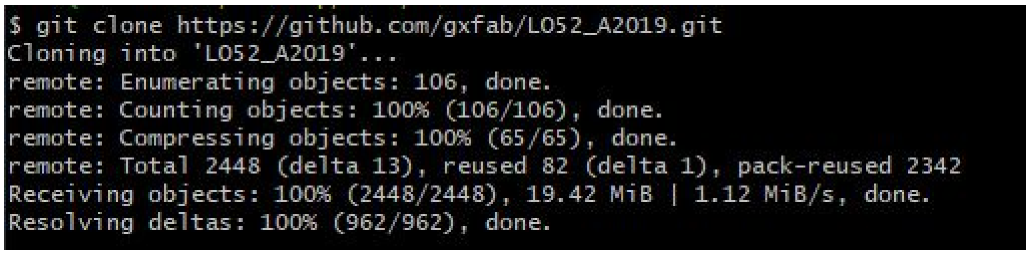
TP1

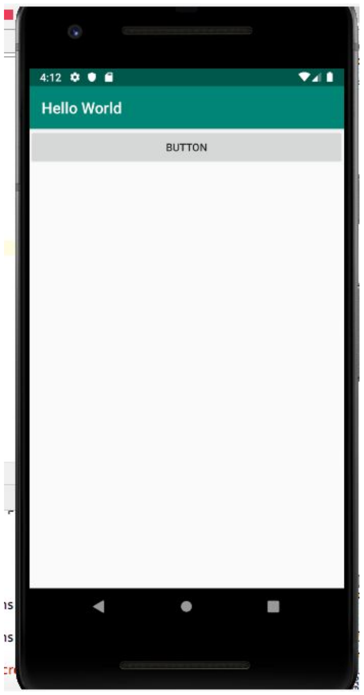
- Télécharger et installer git https://desktop.github.com/
- Clonez le dépôt

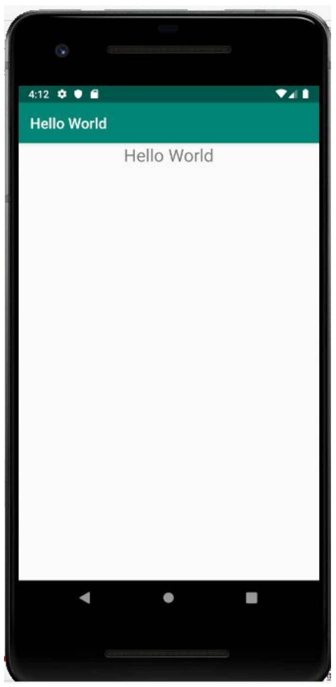


```
$ git clone https://github.com/gxfab/LO52_A2019.git
Cloning into 'LO52_A2019'...
remote: Enumerating objects: 106, done.
remote: Counting objects: 100% (106/106), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 2448 (delta 13), reused 82 (delta 1), pack-reused 2342
Receiving objects: 100% (2448/2448), 19.42 MiB | 1.12 MiB/s, done.
Resolving deltas: 100% (962/962), done.
```

- Configure pour le non et email :
  $ git config --global user.name "xxx"
  $ git config --global user.email xxxx@xxx.com
- Créer branch:
  $ git checkout -B keepit
- Installer Android Studio https://developer.android.com/studio
- App HelloWorld:
  - MainActivity afficher un button sur la première page
  - Utiliser Intent pour lancer la nouvelle activity

```
public void onClick(View v) {
   Intent intent=new Intent(MainActivity.this,HelloActivity.class);
   startActivity(intent);
}
```

  - 

→


TP3

- core.c, descriptor.c, io.c, sync.c, os/darwin_usb.c, os/linux_usbfs.c

```
#define TIMESPEC_TO_TIMEVAL(tv, ts)                              \
      do {                                                 \
            (tv)->tv_sec = (ts)->tv_sec;                        \
            (tv)->tv_usec = (ts)->tv_nsec / 1000;                  \
      } while (0)
```

  - Erreur library "libusb.so" not in prelink map:

    for error: build/tools/apriori/prelinkmap.c(137): library ‘libusb.so’ not in prelink map. This is because Android keeps track of all the shared libraries of the root file system (RFS). Therefore we need to add the library to the prelinked map found in ./build/core/prelink-linux-arm.map.

    ```
    libusb.so           0xA8000000
    ```

    mais le build/envsetup.sh n'est utilise maintenant, on ne sais pas comment le compiler

1

```
PRODUCT_NAME := lo52_KeepIt
```

2

```
$(call inherit-product, Linaro/product/hikey.mk)
```

3

```
PRODUCT_PROPERTY_OVERRIDES := \
   ro.hw=lo_52 \
   net.dns1=8.8.8.8 \
   net.dns2=4.4.4.4
```

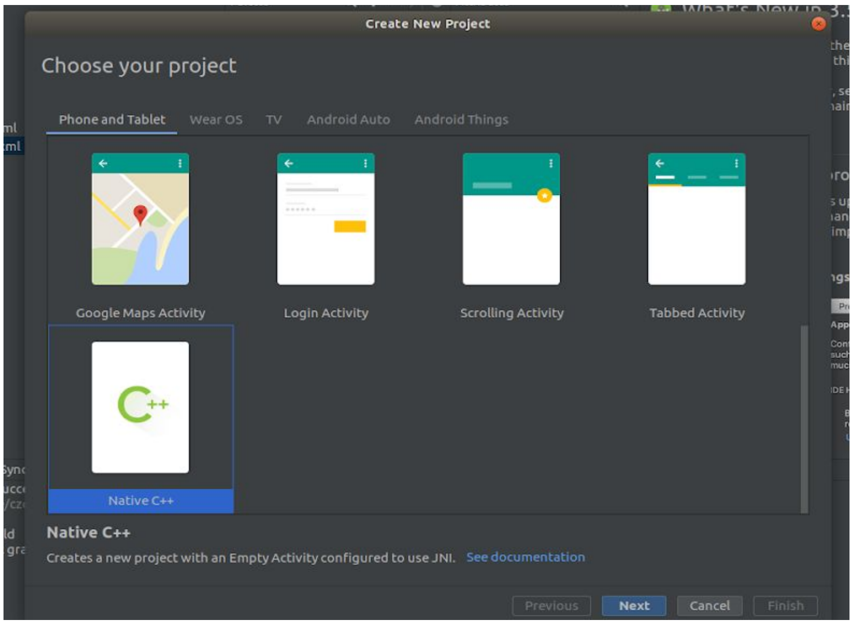
4

5

```
PRODUCT_PACKAGES := libusb
```

TP4

- Etape 1 : Create application with native c++

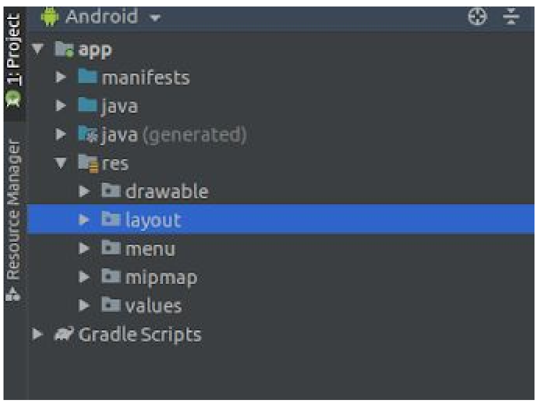

Create New Project

Choose your project

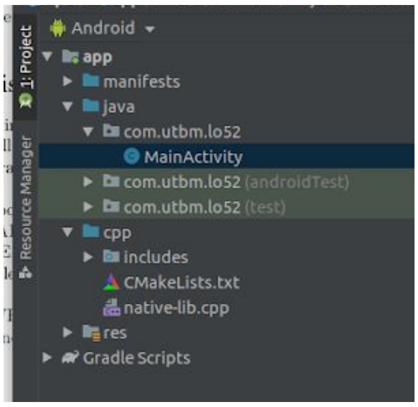
Native C++

Creates a new project with an Empty Activity configured to use JNI. See documentation

- Cela il creer repository */cpp/..* dans le projet
- Stucture application normal et Stucture application native c++ :



→



- Dans /cpp, il y a 2 fihciers importante :
    - CMakeLists.txt :

```
cmake_minimum_required(VERSION 3.4.1)   # version du cmake

add_library( # Sets the name of the library.
        native-lib     # nom du fichier .so
        # Sets the library as a shared library.
        SHARED   # Définissez le type de library, l'un est un fichier statique STATIC  .a et
l'autre est un fichier dynamique SHARED so.
        # Provides a relative path to your source file(s).
        native-lib.cpp )  # fichier a compiler

find_library( # Sets the name of the path variable.
        log-lib
        # Specifies the name of the NDK library that
        # you want CMake to locate.
        log)

target_link_libraries( # Specifies the target library.
        native-lib    # indiquer library target
        # Links the target library to the log library
        # included in the NDK.
        ${log-lib})
```

- native-lib.cpp :
  Code sources du c++

-Etape 2 : declaration des fonctions dans activity

java.com.utbm.lo52.MainActivity.java :

```java
public native String stringFromJNI();
public native String left();
public native String right();
public native String down();
public native String up();
public native int write(int number);
public native int read(int number);

//  importer c++ lib quand lancer application
static {
  System.loadLibrary("native-lib");
}
```

-Etape 3 : definition des fonctions dans cpp

- template pour ecrire les functions

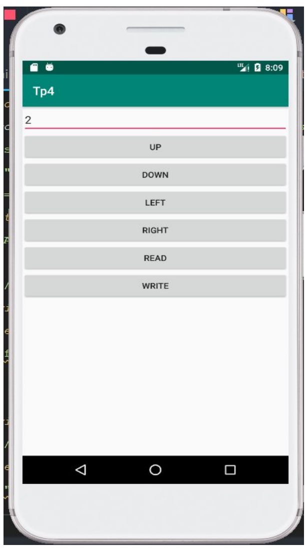
```
extern "C" JNIEXPORT jstring JNICALL
Java_[nom du package]_[activity]_[nom du fonction](
      JNIEnv *env,  jobject /* this */,  [ autre parametre en type JNI ]) {
  [ les contenue ];
  return [ return ];
}
```

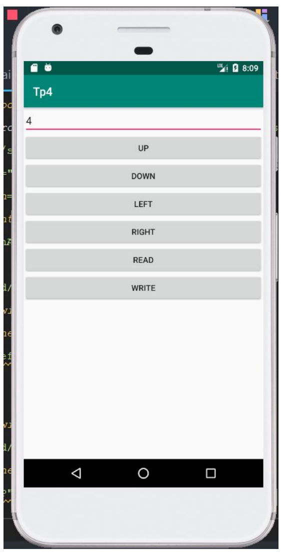
Exemple :

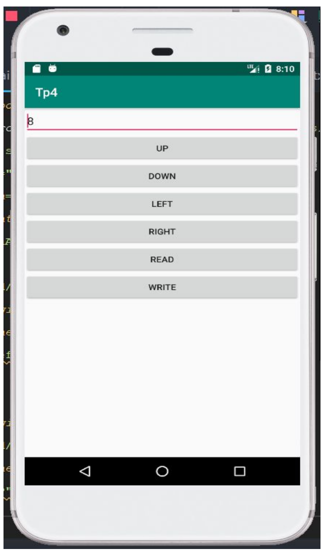
```cpp
#include <jni.h>
```

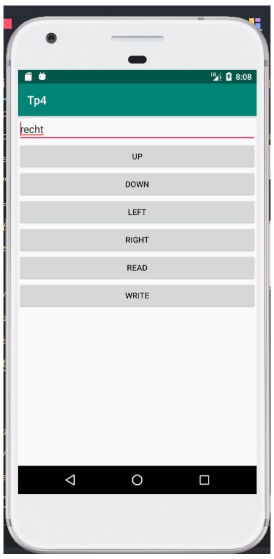
```
#include <string>

extern "C" JNIEXPORT jstring JNICALL
Java_com_utbm_lo52_MainActivity_stringFromJNI(
        JNIEnv *env,
        jobject /* this */) {
    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}

extern "C" JNIEXPORT jint JNICALL
Java_com_utbm_lo52_MainActivity_write(
        JNIEnv *env,
        jobject /* this */,
        jint number) {
    return number*number*number;
}

extern "C" JNIEXPORT jstring JNICALL
Java_com_utbm_lo52_MainActivity_left(
        JNIEnv *env,
        jobject /* this */) {
    std::string hello = "links";
    return env->NewStringUTF(hello.c_str());
}
```

-Resultat:


8:09
Tp4
2
UP
DOWN
LEFT
RIGHT
READ
WRITE


8:09
Tp4
4
UP
DOWN
LEFT
RIGHT
READ
WRITE


8:10
Tp4
8
UP
DOWN
LEFT
RIGHT
READ
WRITE


8:08
Tp4
recht
UP
DOWN
LEFT
RIGHT
READ
WRITE