인터넷/모바일 프로그래밍

Final Project

Rock Paper Scissors Game

Professor : 유시환 교수님

Student Name : 라마드한 | 탄 알로디아 클라우디 나탈리

Student ID : 32185224 | 32195204

Major : SW융합 모바일시스템공학과

Submitted on : June 19th, 2022

Table of Contents

1. Introduction

The final project of this class is to accomplish the purpose of this course, which is to create an android application program that can be used or have some significance to the hardware on the android board. The main idea of this project is to use Java and C/C++ to create an application, and the hardware usages can be approached using C language with JNI as the bridge to connect between both programming languages. For this project, we decided to create a simple game called “Rock Paper Scissors” game.

2. Important Concept

a. Android NDK

The Android NDK (Native Development Kit) is what makes almost all of this possible as most of the code is written in C that means that we need the NDK. An NDK would compile C and C++ code into a native library and package it into the APK using Gradle, Android Studios’ integrated build system. Thankfully Android Studio 22 (Chipmunk) comes standard with its own NDK(CMake) so it releases any further need to install and download our own NDK, and with the use of NDK we can now call the C code in the native library through the use of the JNI (Java Native Interface).

b. JNI in android

The JNI (Java Native Interface) is the integral part of this whole project as it allows us to connect C written code into our Java based main activity it does this with the use of the NDK as previously written above a standard JNI looks like this :

```
JNIEXPORT void JNICALL Java_com_example_<project_name>_<Native_code>
      (JNIEnv * env, jobject obj)
```

Where it would be automatically generated in the native library when native function is written to the main activity with the name of the projects.

c. Hardware on the board

The hardware itself is called the Hanback electronics SM5s4210, and on the right side there is a lot of hardware on the board that can be used for our application. However, after several thoughts, we decided to implement only some of the hardware that is useful to make our game more fun. Those hardware are piezo, text LCD, full color LED and dot matrix. The main purpose of these hardwares is to show the changing states of the game results which are win, lose and draw. More specific explanation is given below:

- Piezo

  It is used to produce sound when the program states the status of the game; For example, if the user wins, the piezo will produce a high note to show that the user wins the game. On the other hand, if the user loses the game, the piezo will produce a low note to show that the user loses. Also, if the result of the game is a draw, it will produce a middle note.

- Text LCD

  It is used to show some texts that correspond to the results of the game. For instance, if the user wins, the text LCD will show “CONGRATULATIONS! YOU WIN!”. If the user loses the game, the text LCD will show “OH NO! COMPUTER WIN!”, and if the result is a draw, the text LCD will show “DRAW!”

- Full Color LED

  It is used to show some colors that correspond to the results of the game. For example, green corresponds to win, blue corresponds to draw and red corresponds to lose. Everytime the result is shown, the full LED color will turn on depending on the result of the game.

- Dot Matrix

Dot matrix is used to display “ROCK PAPER SCISSORS GAME” on the entire game. It will display as soon as the application is executed.

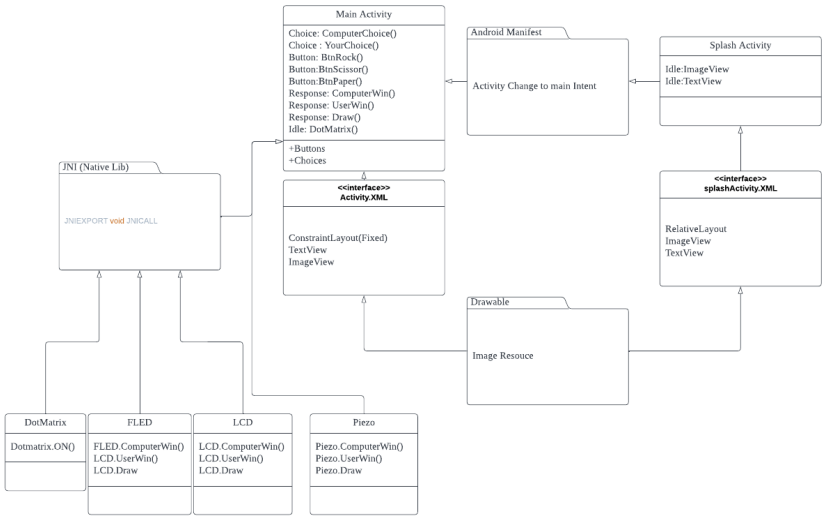
For this application, piezo, text LCD and full color LED are programmed to work at the same time to serve the purpose of showing the results of the game between the user and the computer. However, the dot matrix will always be on while the game is running.

d. Android Application Programming

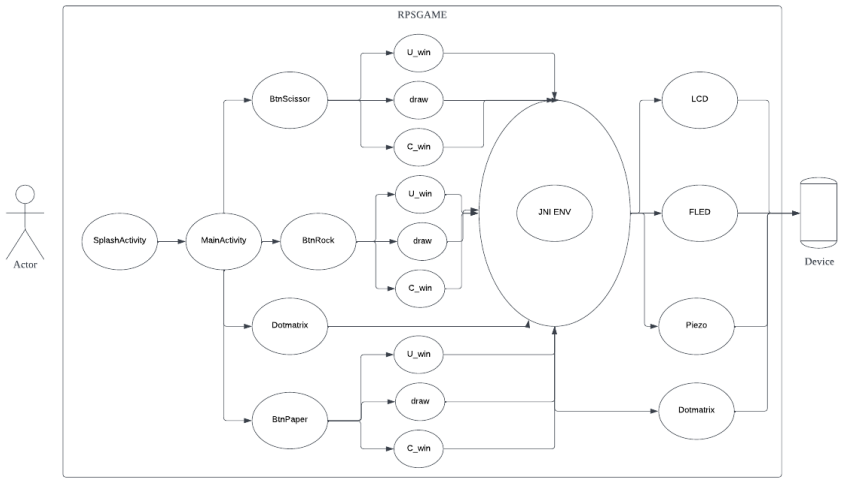
IDE that is used to create this project is Android Studio because we are essentially coding an android based application so the choice to use android's own IDE is a given one but some codes are translated into C for easier usage in CLion because we figured that some codes are better when translated

3. Software Design

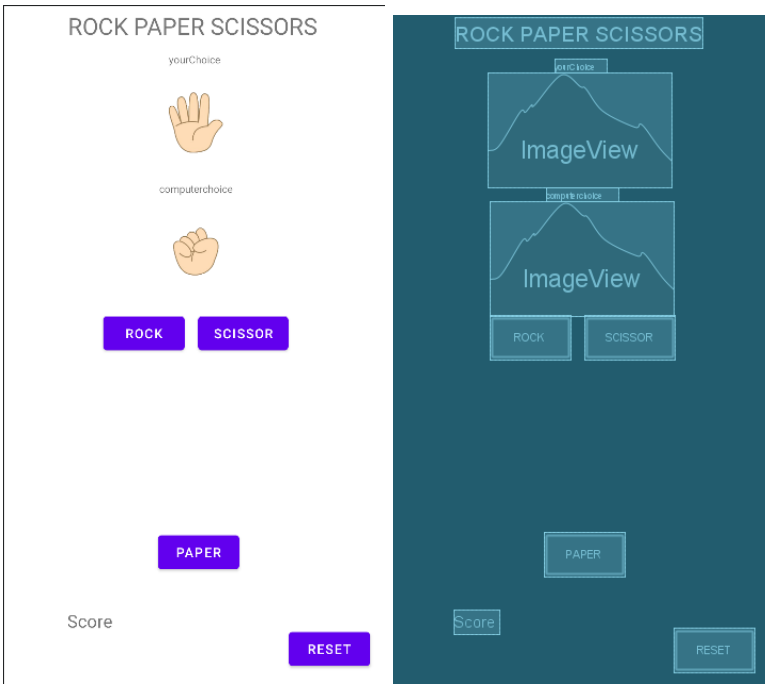
a. UML Diagram



b. Use Case Diagram



c. Screen Activity Design



4. Implementation

a. Details of implementation

Our implementation are focused on what would look enjoyable on the hardware as much if it made sense to use on the board; Of course focusing on the board itself as the android chip and version given on the board is an android 5.1

(Jelly bean) that means our IDE and our compiler has to comply with the version of the board's android. Therefore, we set up the board with a Jellybean compiler to reduce the chance of version error, and then specify the C++ toolbox on the android studio choices of activities, choosing this would then automatically create the Cmakefiles and the Native library that we will need for this project. To gain the code we ssh'd into the assam server so that we can use the C written code on our app. In the code work, we have five Java classes and one xml file which are:

i. MainActivity.java

This is where our main code is created, it consists of the code for our rock paper scissors game and inside the code, it calls 3 java classes that help to connect to the JNI (DotMatrix class, FullLED class, LEDManager class)

ii. DotMatrix.java

This is where we call the JNI function and we copy the code from the M3DotMatrix application and do some modification that suits our application.

iii. FullLED.java

In this class, we connect to the main library by using System.loadLibrary and call the native function that connects to the generated header at the native-lib.c where the implementation to the hardware takes place.

iv. PiezoManager.java

In this class, it contains java code that is translated from the c code that is provided in the M3Piezo application. Through this code, we can connect to the board without the use of JNI.

v. TextLCD.java

In this class, we do the same implementation that we did to FullLED.java and add some functions to make it easier to be used from the MainActivity.java.

vi. activity_main.xml.

This file contains the user interface design of the application.

b. Code Structure

First, we designed the app using android studio deciding on using the constraint layout

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context=".MainActivity">
```

Then we add the resources that will connect to the app

```
<TextView
   android:id="@+id/textView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_marginTop="4dp"
   android:layout_marginEnd="8dp"
   android:layout_marginRight="8dp"
   android:gravity="center"
   android:text="ROCK PAPER SCISSORS"
   android:textAlignment="center"
   android:textSize="25sp"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintHorizontal_bias="0.514"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintTop_toTopOf="parent" />

<TextView
   android:id="@+id/yourChoice"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
```

```
    android:layout_marginStart="56dp"
    android:layout_marginLeft="56dp"
    android:layout_marginTop="12dp"
    android:layout_marginEnd="56dp"
    android:layout_marginRight="56dp"
    android:gravity="center_horizontal"
    android:text="yourChoice"
    android:textSize="11sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />

<ImageView
    android:id="@+id/iv_yourChoice"
    android:layout_width="200dp"
    android:layout_height="124dp"
    app:layout_constraintBottom_toTopOf="@id/computerChoice"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.497"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/yourChoice"
    app:srcCompat="@drawable/paper" />

<TextView
    android:id="@+id/computerChoice"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="computerchoice"
    android:textSize="11sp"
    app:layout_constraintEnd_toEndOf="@+id/iv_yourChoice"
    app:layout_constraintHorizontal_bias="0.517"
    app:layout_constraintStart_toStartOf="@+id/iv_yourChoice"
    app:layout_constraintTop_toBottomOf="@+id/iv_yourChoice" />

<ImageView
```

```
   android:id="@+id/iv_computerChoice"
   android:layout_width="200dp"
   android:layout_height="124dp"
   app:layout_constraintBottom_toTopOf="@+id/buttonRock"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintHorizontal_bias="0.507"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintTop_toBottomOf="@+id/computerChoice"
   app:srcCompat="@drawable/rock" />

<TextView
   android:id="@+id/Score"
   android:layout_width="wrap_content"
   android:layout_height="27dp"
   android:layout_marginStart="68dp"
   android:layout_marginLeft="68dp"
   android:layout_marginBottom="56dp"
   android:text="Score"
   android:textSize="19sp"
   app:layout_constraintBottom_toBottomOf="parent"
   app:layout_constraintStart_toStartOf="parent" />'

<Button
   android:id="@+id/buttonRock"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_marginTop="48dp"
   android:text="Rock"
   app:layout_constraintBaseline_toBaselineOf="@+id/buttonScissor"
   app:layout_constraintBottom_toTopOf="@+id/buttonPaper"
   app:layout_constraintStart_toStartOf="@+id/iv_computerChoice"
   app:layout_constraintTop_toBottomOf="@+id/iv_computerChoice"
   app:layout_constraintVertical_bias="0.0" />

<Button
   android:id="@+id/buttonScissor"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="37dp"
    android:text="Scissor"
    app:layout_constraintBottom_toTopOf="@+id/buttonPaper"
    app:layout_constraintEnd_toEndOf="@+id/iv_computerChoice"
    app:layout_constraintTop_toBottomOf="@+id/iv_computerChoice"
    app:layout_constraintVertical_bias="0.0" />

<Button
    android:id="@+id/buttonPaper"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="56dp"
    android:text="Paper"
    app:layout_constraintBottom_toTopOf="@+id/btn_Reset"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.513"
    app:layout_constraintStart_toStartOf="parent" />

<Button
    android:id="@+id/btn_Reset"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="16dp"
    android:text="Reset"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent" />
```

Then we design the main activity by defining the main class,

```
public class MainActivity extends AppCompatActivity
```

adding resources and adding definitions,

```
Button buttonRock, buttonScissor, buttonPaper, btn_Reset;
TextView Score;
ImageView iv_yourChoice, iv_computerChoice;
int HumanScore, ComputerScore = 0;
```

connecting the resources to the activity XML,

```
buttonPaper = (Button) findViewById(R.id.buttonPaper);
buttonRock = (Button) findViewById(R.id.buttonRock);
buttonScissor = (Button) findViewById(R.id.buttonScissor);
btn_Reset = (Button) findViewById(R.id.btn_Reset);
iv_computerChoice = (ImageView) findViewById(R.id.iv_computerChoice);
iv_yourChoice = (ImageView) findViewById(R.id.iv_yourChoice);
Score = (TextView) findViewById(R.id.Score);
```

connecting the Drawable Resource to the main activity resource,

```
       buttonPaper.setOnClickListener(new View.OnClickListener() {
       @SuppressLint("SetTextI18n")
       @Override
       public void onClick(View view) {
        iv_yourChoice.setImageResource(R.drawable.paper);
        String message = play_turn("paper");
                           Toast.makeText(MainActivity.this,    message,
Toast.LENGTH_SHORT).show();
Score.setText("You  :  " + Integer.toString(HumanScore) + " Computer :
" + Integer.toString(ComputerScore));
       }
});

buttonRock.setOnClickListener(new View.OnClickListener() {
       @Override
       public void onClick(View view) {
        iv_yourChoice.setImageResource(R.drawable.rock);
        String message = play_turn("rock");
                           Toast.makeText(MainActivity.this,    message,
```

```
Toast.LENGTH_SHORT).show();
Score.setText("You : " + Integer.toString(HumanScore) + " Computer :
" + Integer.toString(ComputerScore));
   }
});

buttonScissor.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
       iv_yourChoice.setImageResource(R.drawable.scis);
       String message = play_turn("scissor");
                        Toast.makeText(MainActivity.this,    message,
Toast.LENGTH_SHORT).show();
          Score.setText("You : " + Integer.toString(HumanScore) + "
Computer : " + Integer.toString(ComputerScore));
   }
});
```

And since it is a user vs computer game the computer needs to be able to choose randomly and to show computer choice.

```
//set the computer image based on his choice
if (computer == "rock") {
   iv_computerChoice.setImageResource(R.drawable.rock);
} else if (computer == "scissor") {
   iv_computerChoice.setImageResource(R.drawable.scis);
} else if (computer == "paper") {
   iv_computerChoice.setImageResource(R.drawable.paper);
}
```

Then, we add the scoring method (example : Draw situation)

```
if (computer == yourChoice) {
return "Draw!";
}
```

To use the hardware of the board, we use JNI to connect the text LCD, full LED and dot matrix. In order to do that, we create new classes for text LCD, full LED and dot matrix.

```
public TextLCD textLCD = new TextLCD();
public FullLED fullLED = new FullLED();
public DotMatrix dotMatrix = new DotMatrix();
```

The detailed explanation of the implementation of those classes are given below:

a. TextLCD.java

Inside the class, we add the systemLoadLibrary to connect to the main activity library. Then, we create some function to call the native function that connects to the JNI function. Those function are:

- Void UserWin()
- Void ComputerWin()
- Void Draw()

These functions serve the same purpose which is to initialize and clear the text LCD in the board and then it will print out a specific message in the first and second line of the text LCD.

Below that code, we create the native function that enables us to use the JNI by connecting us to the JNI function that is created in the native-lib.c. Those native function are:

- public native void on()
- public native void off()
- public native void initialize()
- public native void clear()
- public native void print1Line()
- public native void print2Line()

In the native-lib.c, those JNI functions are auto generated, such as:

- JNIEXPORT void JNICALL Java_com_example_rpstest2_TextLCD_on (JNIEnv * env, jobject obj){}
- JNIEXPORT void JNICALL Java_com_example_rpstest2_TextLCD_off (JNIEnv * env, jobject obj){}
- JNIEXPORT void JNICALL Java_com_example_rpstest2_TextLCD_initialize (JNIEnv * env, jobject obj){}
- JNIEXPORT void JNICALL Java_com_example_rpstest2_TextLCD_clear (JNIEnv * env, jobject obj){}
- JNIEXPORT void JNICALL Java_com_example_rpstest2_TextLCD_print1Line (JNIEnv * env, jobject obj, jstring msg){}
- JNIEXPORT void JNICALL Java_com_example_rpstest2_TextLCD_print2Line (JNIEnv * env, jobject obj, jstring msg){}

Then we would take the code from the assam to use; For example, we take the LCD_On code from the TextLcd.c code in the assam, specifically from the example program that is already installed in the board, which is M3TextLCD.

```
JNIEXPORT void JNICALL Java_com_example_rpstest2_TextLCD_on
       (JNIEnv * env, jobject obj){
   if (fd == 0)
       fd = open("/dev/fpga_textlcd", O_WRONLY);
   assert(fd != 0);

   ioctl(fd, TEXTLCD_ON);
}
```

This is how the TextLCD.java would look like:

```
public class TextLCD {
   static {
       System.loadLibrary("rpstest2");
   }

   void UserWin(){
       initialize();
       clear();
       print1Line("CONGRATULATIONS!");
       print2Line("YOU WIN!");
   }

   void ComputerWin(){
       initialize();
       clear();
       print1Line("OH NO!");
       print2Line("COMPUTER WIN!");
   }

   void Draw(){
       initialize();
       clear();
       print1Line("DRAW!");
   }
   public native void on();
   public native void off();
   public native void initialize();
   public native void clear();

   public native void print1Line(String str);
   public native void print2Line(String str);

}
```

b. FullLED.java

Inside the class, we add the systemLoadLibrary to connect to the main activity library. Then, we create a function to call the native function that connects to the JNI function. That function is written as:

- public native int FLEDCONTROL(int ledNum, int red, int green, int blue);

Then, this function auto generates the JNI function in native-lib.c, such as:

- JNIEXPORT jint JNICALL Java_com_example_rpstest2_FullLED_FLEDCONTROL(JNIEnv *env, jobject thiz, jint led_num, jint red, jint green, jint blue) {}

Then, we use the code from M3Fullcolorled application for the implementation to the hardware with a little modification.

c. DotMatrix.java

Inside the class, we add the systemLoadLibrary to connect to the main activity library. Then, we create a function to call the native function that connects to the JNI function. That function is written as:

- public native void DotMatrixControl(String str);

Consequently, this native function auto generates the JNI function in the native-lib.c, such as:

JNIEXPORT void JNICALL Java_com_example_rpstest2_DotMatrix_DotMatrixControl(JNIEnv *env, jobject thiz, jstring str) {}

We use the code from the M3Dotmatrix application for this JNI function and for the hardware implementation in the DotMatrix.java. We did some modification and create some function to make it easier to be used from the main activity class, such as:

- void ON()

This function is used to turn on the dot matrix by setting the value for the setStart() function as true and executing the run() function.

- void OFF()

  This function is used to turn off the dot matrix by setting the value for the setStart() function as false.

After those classes are created, now we can call the functions to use the hardware from the MainActivity.java. For example:

```
textLCD.ComputerWin();
textLCD.UserWin();
textLCD.Draw();
dotMatrix.ON();
fullLED.FLEDCONTROL(9, 100, 0, 0);
```

Furthermore, for the piezo usage, we translated straight from the c to the java code as it would be faster and easier for the usage. The code is written in the PiezoManager Java class. This is a brief example of the code:

```
public class PiezoManager {
   private static final String TAG = "PiezoManager";
         private   static   final   File   PIEZO_DEVICE_FILE   =   new
File("/dev/fpga_piezo");
   private static char note = 0;
```

In the code, we create specific functions to make the function calling become easier and more streamlined. Those function are:

- static void UserWin()
- static void ComputerWin()
- static void Draw()

  In these functions, the note is set with a certain value and resets the note everytime the function is called. In result, it is easier for the main activity to call the function. For example:

```
PiezoManager.ComputerWin();
```

At last, we would run it on the hanback device by first connecting the board to the PC using the USB cable and wait until the installation is finished. After the installation is finished, the user can play the rock paper scissors game.

c. How to Play

The rock paper scissors game simply works by launching the application on the android hardware. The splash screen displays the title of the game and it continues to the main program which is the game. Users can simply touch the button on the screen according to their pick. At the same time, the computer chooses their pick and displays it on the screen. The hardwares such as piezo, text LCD and full color LED will show the status of the game (win, lose or draw); The scores are displayed on the screen. Also, there is a "reset" button to reset the game.

5. Documentation

a. Video link of our application

https://youtu.be/tyPQnq9dRRU

b. Link to the source code

https://github.com/Rizzyrizz/RPS_FINAL

6. Feedback

a. Trials and Errors

At first, there was a lot of trial and error that came with it as we were still new to JNI work and had to research it first. Because of the prolonged time of research, our project time was also pushed back by about 1 week which left us with little time to actually work on the physical project, a lot of the problems came from the JNI connection to the main Activity, as testing went on and on the JNI would sometimes not get the right call, which then we had to experiment more, such as

by adding the load library function, but most of these tests weren't run on the eventual app that we would use, it would instead be tested on dummy projects just so that we would have an understanding on how calling and the JNI works. After a week of trials, we spent around 3 days making the app which went through testing phases such as experimenting with different additions, but we decided on keeping it simple so that we can match time constraints and keep workflow working. After the app was tested and working perfectly, we started adding the additional JNI work, but of course only adding the ones that would make sense for a project like this. Therefore, we decided to only use 4 things those being the

- TextLCD to display text
- Full Color LED to display color based on text
- Piezo to make a noise every time a choice is made
- And the Dotmatrix to display the name of the game

After we made the choices and decided not to add anymore hardware implementation, final implementation was finally done and thankfully went smoothly without errors.

b. Personal Feelings

Farhan : Well, I would be lying if I said that this class wasn't difficult and that there was a lot of stress, but that didn't come from the class itself, it was also from the many project related classes that I enrolled in this semester. Overall, it was a very interesting and fun class, and I enjoyed it thoroughly to the end, yes, even with the many errors that me and Audi had to overcome to get this far. We thought that it was worth it to experience teamwork, although I could say that if for the next semester they could get hardware that is more up to date that would be nice, this is not to say that the hardware given was bad, it's just that working on something that is more up to date would lower the danger of version backlash but that's it for me.

Audi: To be honest, I have a lot of consideration before taking this course because I know that this class may be hard and it has a final project to be worked on.

Moreover, I also did not take the related class last semester which is operating systems, which means that I need to study more to attend this class. However, I decided to take this class because I believe that through this class, I will get more knowledge and practices that will be useful in the future. Thankfully, I get to work with my teammate, Farhan for the course's main project. The project itself is, ofcourse, difficult for us because we have a lack of knowledge in the beginning. But as the time flows, we are able to cover up and work on the project through many ups and downs and it results in a satisfying product for us. Personally, I am satisfied with this project even though many trials and errors need to be overcome first. Overall, this project and class was pretty fun as we got to work with the hardware directly, and we got to experience great teamwork and a fun project.

7. Conclusion

In conclusion, we agreed that it was a fun exercise to work on this even with all the stress that came with it, it was still pretty useful for us to study more java and C in this python, kotlin world. We ended up learning a lot more about hardware and how to be patient (especially during gradle builds and installations) and we are proud of what we have achieved. Although it may not be much, it took a lot of hard work and team meetings to achieve it. The reason for that is because our classes always clash, so it was hard for us to make time for this project, but we eventually finished it and we are proud of our work, and got a better understanding of how native libraries work and how android apps work which is always good for the future. We would like to thank Professor Yoo See Hwan for giving us this opportunity and our classmates for helping us during this time with explanations. Thank you for the class and the help, both are appreciated!

References

- https://sodocumentation.net/android/topic/8674/android-java-native-interface--jni-
- https://developer.android.com/training/articles/perf-jni
- https://developer.android.com/ndk/guides
- https://developer.android.com/ndk/guides/concepts
- https://issuu.com/xolaly/docs/2013_small_
- https://www.youtube.com/watch?v=pyXnX2SEaFc
- https://android.googlesource.com/platform/ndk/+/master/docs/BuildSystemMaintainers.md
- https://developer.android.com/studio/intro