TIROUMOUROUGANE Vishnouraj
LAW-SOUNE Rubens
MAITREJEAN Joris

24/03/2017
M4101

Projet tuteuré : Pilotage de drone

DUT R&T
Session 2016/2017 – an 2

IUT Villetaneuse
Université Paris 13

Table des matières

Remerciements

Nous tenions tout d'abord à remercier madame Coti, professeur à l'IUT Paris 13 Villetaneuse pour son accompagnement et son aide précieuse tout au long de la mise en place de ce projet.

Nous souhaitons dans un second temps remercier l'IUT de Villetaneuse pour la mise à disposition de son matériel qui nous a permis de mener à bien cette réalisation.

Pour finir nous remercions nos camarades, Rousseau Régis, Palanque Florent, Brissaud Paul, pour en citer quelques-uns; pour leurs réponses pertinentes et indispensables lorsque nous avions pu en avoir besoin durant la conception de notre projet et lors de cette écriture.

1) Projet : Pilotage de drone

1.1) Cahier des charges

Le but de notre projet était de programmer des trajectoires (circulaires, rectangulaires, …) pour un drone roulant « le Jumping Sumo » en utilisant le kit de développement fourni par le constructeur Parrot utilisant les langages de programmation suivant : C, C++, Java.

Dans notre cas nous utilisons le langage Java.

Nous avons choisi d'utiliser le logiciel Android Studio afin de créer une application Android qui aura pour but de contrôler le drone via WiFi par l'intermédiaire de boutons contenant les commandes nécessaire à l'exécution d'un parcourt précis par celui-ci.

Figure 1 : Logo Java

Figure 2 : Logo Android Studio

Figure 3 : Logo Parrot SDK

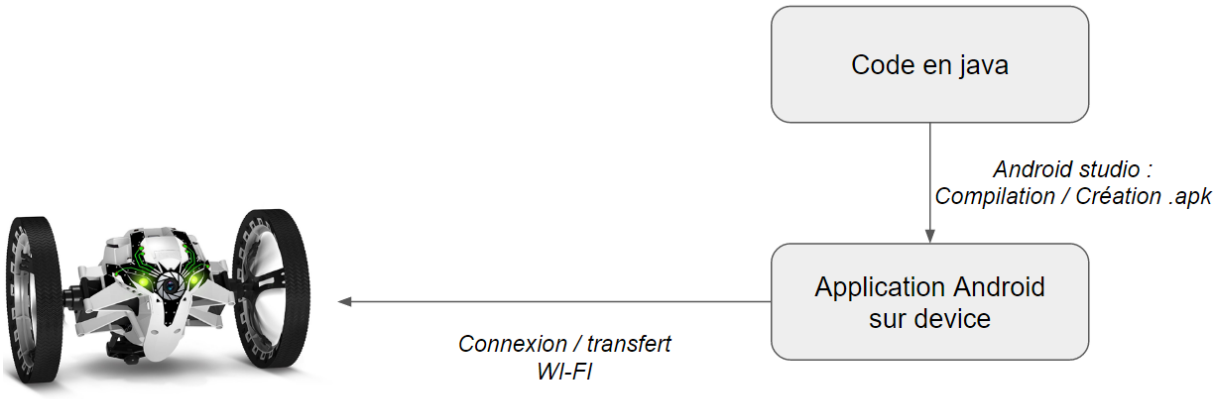

Figure 4 : illustration idée de conception

1.2) Caractéristique du drone : Parrot Jumping Sumo

- Vitesse : 7 km/h
- Saut : 80 cm
- Roues ajustables : plus de vitesse et de stabilité
- Caméra : 640 x 480 pixels / 15 ips
- Portée: jusqu'à 50 mètres
- Autonomie : 20 minutes
- Recharge complète : 90 minutes
- Dimensions roues écartées : 185 x 150 x 110 mm
- Dimensions roues rétractées : 145 x 150 x 110 mm
- Poids : 180 g
- Capacité de la batterie : 550 mAH

1.3) Diagramme bête à cornes

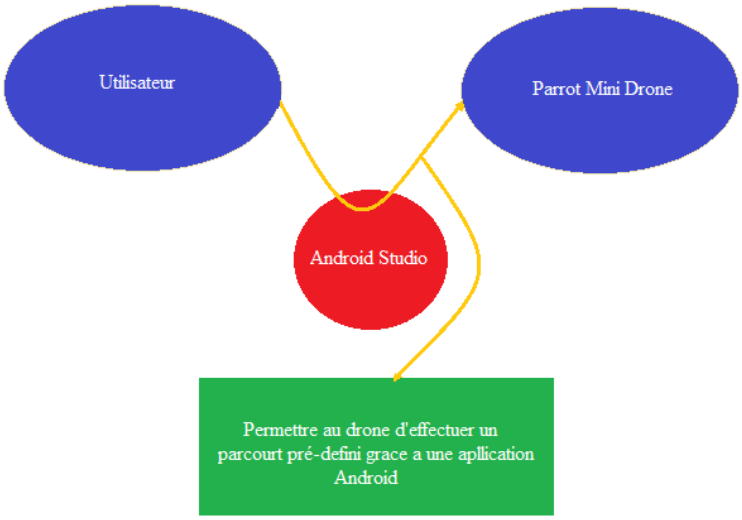

Figure 5 : Diagramme Bête a corne du projet

2) Présentation d'Android Studio

Android Studio est un logiciel open source créé par Google en 2014.

Il s'agit d'un IDE complet permettant la création d'application mobile Android et étant principalement utilisé pour éditer des fichiers java étant le langage d'une application android native ainsi que des fichiers de mise en page xml.

Les fichiers que nous utilisons principalement pour le développement de notre application sont : MainActivity.java qui contient le code brut de l'application (classes, méthodes etc.) et le activity_main.xml qui permet de faire la mise en page de notre application (placement des boutons, TextView etc).

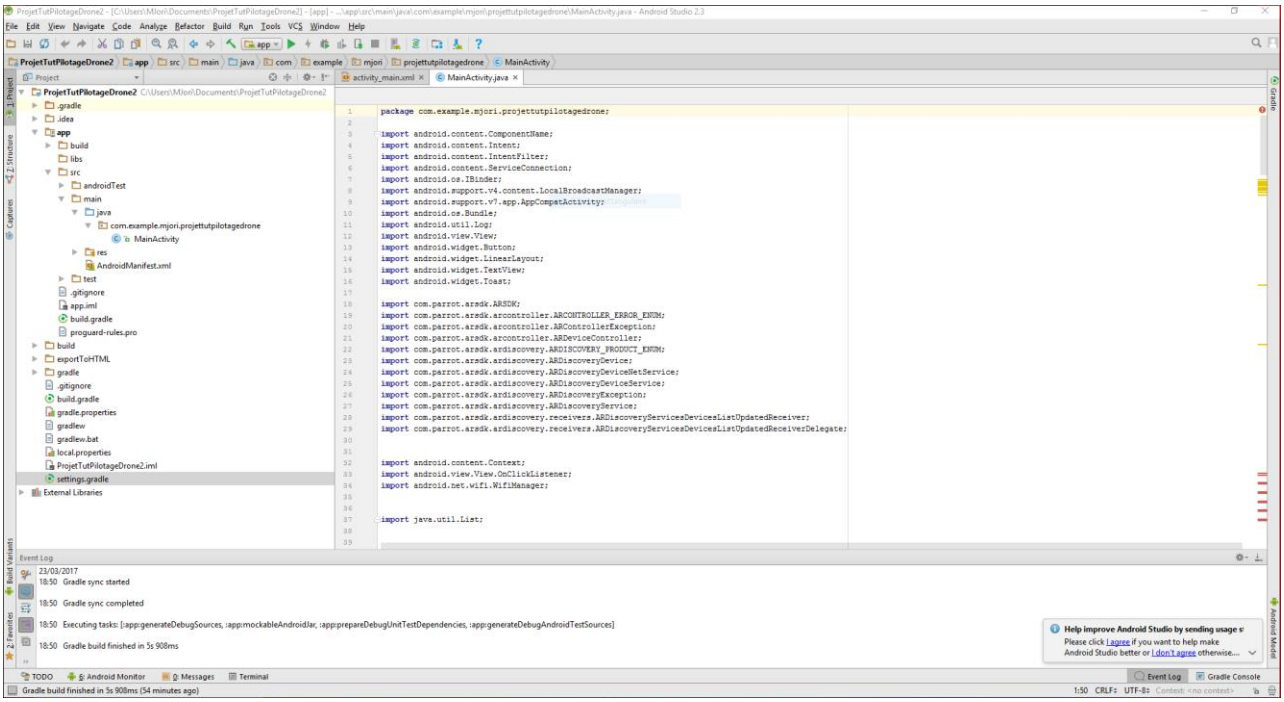

Figure 6 : Fenêtre Android studio fichier MainActivity.jav

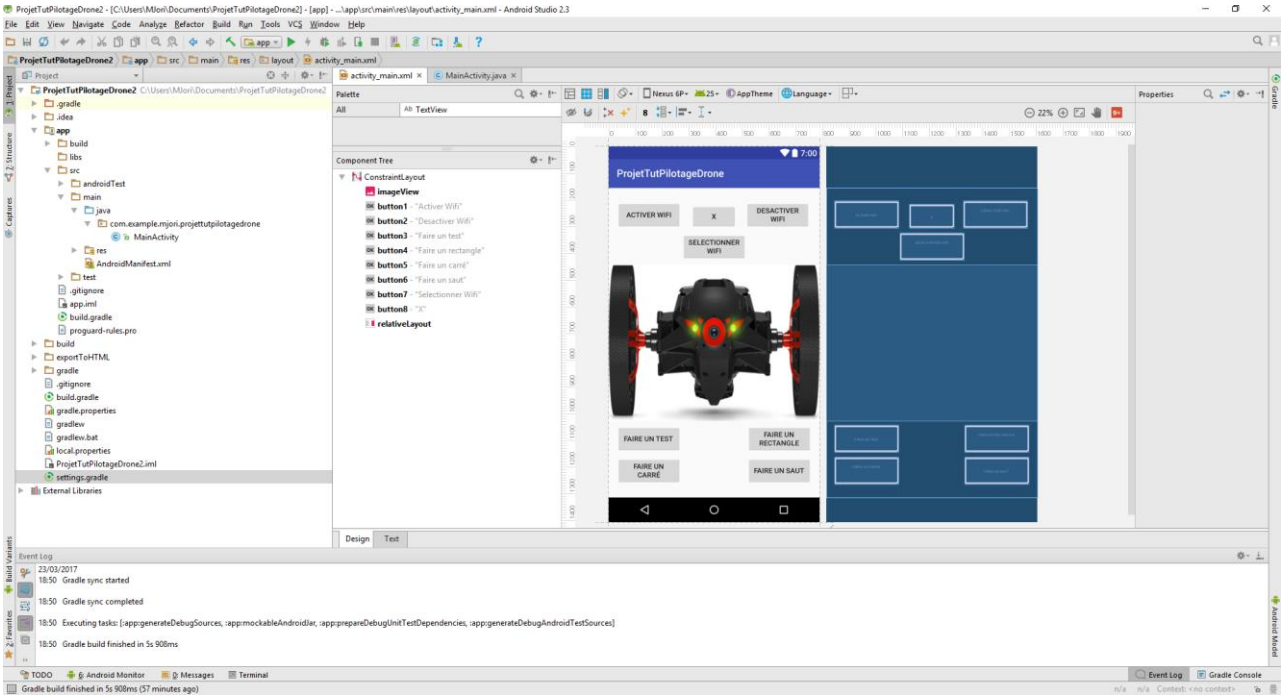

Figure 7 : Fenêtre Android studio fichier activity_main.xml

3) Le code

Dans cette partie nous allons afficher notre code d'application et expliquer brièvement ce que fait chaque partie du programme (classes, méthodes etc.)

3.1) Fichier MainActivity.java

• Dans un premier temps nous importons tous les modules nécessaires au bon fonctionnement de l'application Android :

```
package com.example.mjori.projettutpilotagedrone;
import android.content.ComponentName;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.ServiceConnection;
import android.os.IBinder;
import android.support.v4.content.LocalBroadcastManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import android.content.Context;
import android.view.View.OnClickListener;
import android.net.wifi.WifiManager;
import java.util.List;
```

• Nous importons par la suite tous les modules du SDK de Parrot nécessaires au contrôle du drone :

```
import com.parrot.arsdk.ARSDK;
import com.parrot.arsdk.arcontroller.ARCONTROLLER_ERROR_ENUM;
import com.parrot.arsdk.arcontroller.ARControllerException;
import com.parrot.arsdk.arcontroller.ARDeviceController;
import com.parrot.arsdk.ardiscovery.ARDISCOVERY_PRODUCT_ENUM;
import com.parrot.arsdk.ardiscovery.ARDiscoveryDevice;
import com.parrot.arsdk.ardiscovery.ARDiscoveryDeviceNetService;
import com.parrot.arsdk.ardiscovery.ARDiscoveryDeviceService;
import com.parrot.arsdk.ardiscovery.ARDiscoveryException;
import com.parrot.arsdk.ardiscovery.ARDiscoveryService;
import com.parrot.arsdk.ardiscovery.receivers.ARDiscoveryServicesDevicesListUpdatedReceiver;
import com.parrot.arsdk.ardiscovery.receivers.ARDiscoveryServicesDevicesListUpdatedReceiverDelegate;
```

• Nous créons la classe MainActivity qui implémente AppCompatActivity, qui est une classe de base pour les activités utilisant des fonctions de barre d'action :

```
public class MainActivity extends AppCompatActivity implements
ARDiscoveryServicesDevicesListUpdatedReceiverDelegate
```

• Ensuite nous initialisons l'ensemble de nos objets en leur attribuant un nom et un type correspondant :

```
    private Button faireTest;
    private Button enableButton;
    private Button disableButton;
    private Button selectWifiButton;
    private Button fairerectangleButton;
    private Button faireunsautButton;
    private Button fairecarreButton;
    private Button exitappButton;
    private TextView tv;
    private Toast test;
    private LinearLayout groupeDeVue;
```

• La variable TAG est une chaine de caractère qui ne pourra pas être changée après sa création due au mot clé « final » et qui prend en valeur « MainActivity » :

```
    private static final String TAG = "MainActivity";
```

• Ces variables seront utilisées pour la découverte et connexion aux services :

```
private ARDiscoveryService mArdiscoveryService;
private ServiceConnection mArdiscoveryServiceConnection;
```

• Nous chargeons l'ensemble de la librairie du SDK de Parrot :

```
static {
    ARSDK.loadSDKLibs();
}
```

• La méthode initDiscoveryService permet d'initialiser la recherche des réseaux wifi et de créer ou de lier le service en fonction de son existence :

```
private void initDiscoveryService() {
    // create the service connection
    if (mArdiscoveryServiceConnection == null) {
        mArdiscoveryServiceConnection = new ServiceConnection() {
            @Override
            public void onServiceConnected(ComponentName name, IBinder service) {
                mArdiscoveryService = ((ARDiscoveryService.LocalBinder) service).getService();

                startDiscovery();
            }

            @Override
            public void onServiceDisconnected(ComponentName name) {
                mArdiscoveryService = null;
            }
        };
    }

    if (mArdiscoveryService == null) {
        // if the discovery service doesn't exists, bind to it
        Intent i = new Intent(getApplicationContext(), ARDiscoveryService.class);
        getApplicationContext().bindService(i, mArdiscoveryServiceConnection,
Context.BIND_AUTO_CREATE);
    } else {
        // if the discovery service already exists, start discovery
        startDiscovery();
    }
}
```

• La méthode startDiscovery a pour fonction de démarrer la recherche de services si la variable de service n'est pas vide :

```
private void startDiscovery() {
    if (mArdiscoveryService != null) {
        mArdiscoveryService.start();
    }
}
```

• La méthode registerReceivers permet d'organiser les services trouvés dans une liste :

```
private void registerReceivers() {

    ARDiscoveryServicesDevicesListUpdatedReceiver mArdiscoveryServicesDevicesListUpdatedReceiver =
new ARDiscoveryServicesDevicesListUpdatedReceiver(this);
    LocalBroadcastManager localBroadcastMgr =
LocalBroadcastManager.getInstance(getApplicationContext());
    localBroadcastMgr.registerReceiver(mArdiscoveryServicesDevicesListUpdatedReceiver, new
IntentFilter(ARDiscoveryService.kARDiscoveryServiceNotificationServicesDevicesListUpdated));
}

@Override
public void onServicesDevicesListUpdated() {
    Log.d(TAG, "onServicesDevicesListUpdated ...");

    if (mArdiscoveryService != null) {
        List<ARDiscoveryDeviceService> deviceList = mArdiscoveryService.getDeviceServicesArray();


    }

}
```

• La méthode suivante permet de créer un ARDiscoveryDevice à l'aide du ARservice choisi s'il n'existe pas et le configure en lui donnant un nom, une adresse IP et un numéro de port :

```
private ARDiscoveryDevice createDiscoveryDevice(ARDiscoveryDeviceService service) {
    ARDiscoveryDevice device = null;

    if ((service != null) &&
(ARDISCOVERY_PRODUCT_ENUM.ARDISCOVERY_PRODUCT_ARDRONE.equals(ARDiscoveryServic
e.getProductFromProductID(service.getProductID())))) {
        try {
            device = new ARDiscoveryDevice();

            ARDiscoveryDeviceNetService netDeviceService = (ARDiscoveryDeviceNetService)
service.getDevice();

            device.initWifi(ARDISCOVERY_PRODUCT_ENUM.ARDISCOVERY_PRODUCT_ARDRONE,
netDeviceService.getName(), netDeviceService.getIp(), netDeviceService.getPort());
        } catch (ARDiscoveryException e) {
```

```
            e.printStackTrace();
            Log.e(TAG, "Error: " + e.getError());
        }
    }

    return device;
}
```

• La méthode connecte_drone permet de gérer toutes les étapes de la connexion en faisant appel aux méthodes précédente :

```
public ARDeviceController connecte_drone()
{
    ARDeviceController deviceController = null;

    initDiscoveryService();
    startDiscovery();
    registerReceivers();
    onServicesDevicesListUpdated();
    for (int i=0; i==mArdiscoveryService.getDeviceServicesArray().size(); i++)
    {
        try
        {
            deviceController = new ARDeviceController
(createDiscoveryDevice(mArdiscoveryService.getDeviceServicesArray().get(i)));
            ARCONTROLLER_ERROR_ENUM error = deviceController.start();
        }
        catch (ARControllerException e)
        {
            e.printStackTrace();
        }
    }
    return deviceController ;
}
```

• Nous créons les attributions des boutons en donnant un ID propre afin de pouvoir les différentier les uns des autres par la suite comme par exemple dans le activity_main.xml :

```
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    enableButton = (Button) findViewById(R.id.button1);
    disableButton = (Button) findViewById(R.id.button2);
    faireTest = (Button) findViewById(R.id.button3);
    fairerectangleButton = (Button) findViewById(R.id.button4);
    fairecarreButton = (Button) findViewById(R.id.button5);
    faireunsautButton = (Button) findViewById(R.id.button6);
    selectWifiButton = (Button) findViewById(R.id.button7);
```

```
exitappButton = (Button) findViewById(R.id.button8);
```

• On associe la classe OnClick aux boutons « enable, disable, selectWifi et fairecarré etc.» afin de pouvoir attribuer une action et aussi une notification lors de l'utilisation de ce bouton :

```
enableButton.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        WifiManager wifi = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        wifi.setWifiEnabled(true);
        afficherNotifEnableButton();
    }
});

disableButton.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        WifiManager wifi = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        wifi.setWifiEnabled(false);
        afficherNotifDisableButton();
    }
});

selectWifiButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(Intent.ACTION_MAIN, null);
        intent.addCategory(Intent.CATEGORY_LAUNCHER);
        ComponentName cn = new ComponentName("com.android.settings",
"com.android.settings.wifi.WifiSettings");
        intent.setComponent(cn);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }

});

faireTest.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {

        ARDeviceController controller = connecte_drone();
        controller.getFeatureJumpingSumo().setPilotingPCMDTurn((byte) 50);

        afficherNotifFairecarréButton();
    }
});

fairerectangleButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        /*ARDeviceController controller = connecte_drone();*/
```

```
            afficherNotifFairerectangleButton();
        }
    });

    fairecarreButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            /*ARDeviceController controller = connecte_drone();*/


            afficherNotifFairelosangeButton();
        }
    });

    faireunsautButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            /*ARDeviceController controller = connecte_drone();*/


            afficherNotifFaireunsautButton();
        }
    });

    exitappButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {

            finish();
        }
    });
}
```

• C'est la méthode utilisée pour afficher la notification "La WIFI a été activée" lorsque le bouton est actionné :

```
public void afficherNotifEnableButton() {
    Toast.makeText(this, "La WIFI a été activée", 10);
    test = Toast.makeText(this, "La WIFI a été activée", 10);
    test.show();
}
```

• C'est la méthode utilisée pour afficher la notification "La WIFI a été désactivée" lorsque le bouton est actionné :

```
public void afficherNotifDisableButton() {
    Toast.makeText(this, "La WIFI a été désactivée", 10);
    test = Toast.makeText(this, "La WIFI a été désactivée", 10);
    test.show();
}
```

• C'est la méthode utilisée pour afficher la notification "Le drone execute une action de test" lorsque le bouton est actionné :

```
public void afficherNotifFairecarréButton() {
    Toast.makeText(this, "Le drone execute un carré", 10);
    test = Toast.makeText(this, "Le drone execute une action de test", 10);
    test.show();
}
```

• C'est la méthode utilisée pour afficher la notification "Le drone execute une rectangle" lorsque le bouton est actionné :

```
public void afficherNotifFairerectangleButton() {
    Toast.makeText(this, "Le drone execute un rectangle", 10);
    test = Toast.makeText(this, "Le drone execute un rectangle", 10);
    test.show();
}
```

• C'est la méthode utilisée pour afficher la notification "Le drone execute un carré " lorsque le bouton est actionné :

```
public void afficherNotifFairelosangeButton() {
    Toast.makeText(this, "Le drone execute un carré", 10);
    test = Toast.makeText(this, "Le drone execute un carré", 10);
    test.show();
}
```

• C'est la méthode utilisée pour afficher la notification "Le drone execute un saut" lorsque le bouton est actionné :

```
public void afficherNotifFaireunsautButton() {
    Toast.makeText(this, "Le drone execute un saut", 10);
    test = Toast.makeText(this, "Le drone execute un saut", 10);
    test.show();
}

}
```

3.1) Fichier activity_main.xml

Ce fichier permet la mise en page de notre application :

Pour chaque bouton nous définissons une taille (width/height), un text à afficher sur le bouton, une position sur l'écran etc.

Nous faisons le lien entre un bouton à afficher sur l'écran et le code correspondant à ce bouton (action, notif) dans le fichier MainActivity.java en utilisant les id (ex : **@+id/button1).**

Ce fichier nous a permis aussi de mettre en place un fond d'écran (<RelativeLayout>)

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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"
   tools:context="com.example.mjori.projettutpilotagedrone.MainActivity"
   tools:layout_editor_absoluteY="73dp"
   tools:layout_editor_absoluteX="0dp">


   <Button
      android:id="@+id/button1"
      android:layout_width="0dp"
      android:layout_height="54dp"
      android:layout_alignEnd="@+id/imageView"
      android:layout_centerVertical="true"
      android:text="Activer Wifi"
      android:layout_marginBottom="8dp"
      app:layout_constraintBottom_toTopOf="@+id/button7"
      tools:layout_constraintRight_creator="1"
      app:layout_constraintRight_toRightOf="@+id/button3"
      tools:layout_constraintLeft_creator="1"
      app:layout_constraintLeft_toLeftOf="@+id/button3" />

   <Button
      android:id="@+id/button2"
      android:layout_width="0dp"
      android:layout_height="54dp"
      android:layout_alignEnd="@+id/imageView"
      android:layout_centerVertical="true"
      android:text="Desactiver Wifi"
      android:layout_marginBottom="8dp"
      app:layout_constraintBottom_toTopOf="@+id/button7"
      tools:layout_constraintRight_creator="1"
      android:layout_marginStart="18dp"
      android:layout_marginEnd="19dp"
      app:layout_constraintRight_toRightOf="parent"
      tools:layout_constraintLeft_creator="1"
      app:layout_constraintLeft_toRightOf="@+id/button8" />
```

```
<Button
    android:id="@+id/button3"
    android:layout_width="0dp"
    android:layout_height="54dp"
    android:layout_alignEnd="@+id/imageView"
    android:layout_centerVertical="true"
    android:text="Faire un test"
    android:layout_marginTop="8dp"
    app:layout_constraintTop_toBottomOf="@+id/relativeLayout"
    tools:layout_constraintRight_creator="1"
    app:layout_constraintRight_toRightOf="@+id/button5"
    tools:layout_constraintLeft_creator="1"
    app:layout_constraintLeft_toLeftOf="@+id/button5" />

<Button
    android:id="@+id/button4"
    android:layout_width="0dp"
    android:layout_height="54dp"
    android:layout_alignEnd="@+id/imageView"
    android:layout_centerVertical="true"
    android:text="Faire un rectangle"
    android:layout_marginTop="8dp"
    app:layout_constraintTop_toBottomOf="@+id/relativeLayout"
    tools:layout_constraintRight_creator="1"
    android:layout_marginEnd="16dp"
    app:layout_constraintRight_toRightOf="parent"
    tools:layout_constraintLeft_creator="1"
    app:layout_constraintLeft_toLeftOf="@+id/button6" />

<Button
    android:id="@+id/button5"
    android:layout_width="125dp"
    android:layout_height="54dp"
    android:layout_alignEnd="@+id/imageView"
    android:layout_centerVertical="true"
    android:text="Faire un carré"
    android:layout_marginTop="8dp"
    app:layout_constraintTop_toBottomOf="@+id/button3"
    android:layout_marginStart="16dp"
    tools:layout_constraintLeft_creator="1"
    app:layout_constraintLeft_toLeftOf="parent" />

<Button
    android:id="@+id/button6"
    android:layout_width="125dp"
    android:layout_height="54dp"
    android:layout_alignEnd="@+id/imageView"
    android:layout_centerVertical="true"
    android:text="Faire un saut"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintHorizontal_bias="0.944"
    android:layout_marginTop="8dp"
    app:layout_constraintTop_toBottomOf="@+id/button4" />
```

```
<Button
    android:id="@+id/button7"
    android:layout_width="125dp"
    android:layout_height="54dp"
    android:layout_alignEnd="@+id/imageView"
    android:layout_centerVertical="true"
    android:text="Selectionner Wifi"
    android:layout_marginBottom="8dp"
    app:layout_constraintBottom_toTopOf="@+id/relativeLayout"
    tools:layout_constraintRight_creator="1"
    app:layout_constraintRight_toRightOf="parent"
    tools:layout_constraintLeft_creator="1"
    app:layout_constraintLeft_toLeftOf="parent" />

  <Button
    android:id="@+id/button8"
    android:layout_width="88dp"
    android:layout_height="48dp"
    android:layout_alignEnd="@+id/imageView"
    android:layout_centerVertical="true"
    android:text="X"
    android:layout_marginBottom="8dp"
    app:layout_constraintBottom_toTopOf="@+id/button7"
    android:layout_marginStart="18dp"
    tools:layout_constraintLeft_creator="1"
    app:layout_constraintLeft_toLeftOf="@+id/button7" />

  <RelativeLayout
    android:id="@+id/relativeLayout"
    android:layout_width="0dp"
    android:layout_height="304dp"
    android:background="@drawable/background"
    android:orientation="vertical"
    tools:layout_constraintTop_creator="1"
    tools:layout_constraintRight_creator="1"
    tools:layout_constraintBottom_creator="1"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    tools:layout_constraintLeft_creator="1"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"></RelativeLayout>*/



</android.support.constraint.ConstraintLayout>
```

4) L'Interface graphique

Voici l'interface graphique finale de notre application :

Figure 8 : Interface Graphique

Elle est composée des éléments suivant :

- Un bouton « Activer WIFI » permettant d'activer le WiFi du téléphone.
- Un bouton « Désactiver WIFI » permettant de désactiver le WiFi du téléphone.
- Un bouton « Sélectionner WIFI » permettant d'ouvrir la fenêtre de paramètres de WiFi du téléphone afin de sélectionner le drone et de se connecter à celui-ci.
- Un bouton « X » permettant de fermer l'application.
- Un bouton « Faire un test » nous ayant servie de bouton test afin de tester le transfert des commandes de mouvement jusqu'au drone et ainsi sa réactivité aux commandes. (Voir 5)
- Trois boutons « Faire … » permettant normalement de faire exécuter au drone différents types de parcours. (Voir 5).

5) Erreurs rencontrées

Nous n'avons malheureusement pas réussis à établir le transfert entre notre device et notre drone.

En effet lorsque nous appuyons sur le bouton « FAIRE UN TEST » qui est seul bouton pour lequel nous avons attribué des commandes d'action pour le drone dans l'optique de tester le bon fonctionnement avant de faire de même avec les autres boutons, notre programme plante et notre application se ferme.

Nous n'avons malheureusement pas réussis à trouver la solution dans le temps imparti attribué à notre projet mais nous sommes encore en ce moment même en train de chercher la solution à notre problème.

Cependant nous pouvons émettre une hypothèse sur l'origine du « bug ». En effet nous pensons qu'il s'agit très certainement de la boucle « for » se trouvant dans notre méthode « connecte_drone ».

Cette boucle for a pour objectif de créer pour chaque périphérique Wifi détecté par le smartphone à partir de ses informations un discoveryService puis un discoveryDevice pour instancier un ARDeviceController qui est utiliser pour la suite du programme.

Nous somme conscient que cette boucle n'est pas coder de manière optimale et qu'elle semble beaucoup trop « gourmande » pour fonctionner rien que du fait-est que le balayage wifi du smartphone se lance très régulièrement.

Comme une erreur en cache souvent une autre, nous ne pouvons pas admettre qu'il s'agit là de la seule erreur bloquant notre transfert Wifi avec le drone, mais n'ayant pas résolu notre première difficulté, nous sommes actuellement dans l'incapacité de vérifier nos dires.

Conclusion

Malgré que nous n'avons pas réussis à mener à bien l'intégralité de notre projet, cette expérience nous a ouvert davantage au monde du développement en java et plus particulièrement au développement sous Android.

En effet, n'ayant aucune connaissance du développement sous Android nous avons dû apprendre « sur le tas » à utiliser le logiciel Android Studio pour programmer notre application en java et cela nous a alors permis de développer de nouvelles compétences en matière de développement, ce qui est pour nous un véritable atout.

Cette réalisation nous a de plus appris l'importance de la cohésion au sein d'un groupe de travail, où l'échange des savoirs et l'écoute d'autrui restent indispensables à la réussite.

Nous tirons donc une réelle conclusion de cette expérience, aussi bien par le biais les compétences qu'elle a pu nous apporter, mais aussi du fait que ces compétences pourront et seront réutilisées dans un futur proche, pour de futurs projets ou durant notre vie active.

Annexe

Fichier MainActivity.java :

```
package com.example.mjori.projettutpilotagedrone;

import android.content.ComponentName;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.ServiceConnection;
import android.os.IBinder;
import android.support.v4.content.LocalBroadcastManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.parrot.arsdk.ARSDK;
import com.parrot.arsdk.arcontroller.ARCONTROLLER_ERROR_ENUM;
import com.parrot.arsdk.arcontroller.ARControllerException;
import com.parrot.arsdk.arcontroller.ARDeviceController;
import com.parrot.arsdk.ardiscovery.ARDISCOVERY_PRODUCT_ENUM;
import com.parrot.arsdk.ardiscovery.ARDiscoveryDevice;
import com.parrot.arsdk.ardiscovery.ARDiscoveryDeviceNetService;
import com.parrot.arsdk.ardiscovery.ARDiscoveryDeviceService;
import com.parrot.arsdk.ardiscovery.ARDiscoveryException;
import com.parrot.arsdk.ardiscovery.ARDiscoveryService;
import
com.parrot.arsdk.ardiscovery.receivers.ARDiscoveryServicesDevicesListUpdate
dReceiver;
import
com.parrot.arsdk.ardiscovery.receivers.ARDiscoveryServicesDevicesListUpdate
dReceiverDelegate;


import android.content.Context;
import android.view.View.OnClickListener;
import android.net.wifi.WifiManager;


import java.util.List;



public class MainActivity extends AppCompatActivity implements
ARDiscoveryServicesDevicesListUpdatedReceiverDelegate {

    private Button faireTest;
    private Button enableButton;
    private Button disableButton;
    private Button selectWifiButton;
    private Button fairerectangleButton;
    private Button faireunsautButton;
```

```
    private Button fairecarreButton;
    private Button exitappButton;
    private TextView tv;
    private Toast test;
    private LinearLayout groupeDeVue;

    private static final String TAG = "MainActivity";


    static {
        ARSDK.loadSDKLibs();
    }


    private ARDiscoveryService mArdiscoveryService;
    private ServiceConnection mArdiscoveryServiceConnection;

    private void initDiscoveryService() {
        // create the service connection
        if (mArdiscoveryServiceConnection == null) {
            mArdiscoveryServiceConnection = new ServiceConnection() {
                @Override
                public void onServiceConnected(ComponentName name, IBinder
service) {
                    mArdiscoveryService = ((ARDiscoveryService.LocalBinder)
service).getService();

                    startDiscovery();
                }

                @Override
                public void onServiceDisconnected(ComponentName name) {
                    mArdiscoveryService = null;
                }
            };
        }

        if (mArdiscoveryService == null) {
            // if the discovery service doesn't exists, bind to it
            Intent i = new Intent(getApplicationContext(),
ARDiscoveryService.class);
            getApplicationContext().bindService(i,
mArdiscoveryServiceConnection, Context.BIND_AUTO_CREATE);
        } else {
            // if the discovery service already exists, start discovery
            startDiscovery();
        }
    }

    private void startDiscovery() {
        if (mArdiscoveryService != null) {
            mArdiscoveryService.start();
        }
    }

    private void registerReceivers() {

        ARDiscoveryServicesDevicesListUpdatedReceiver
mArdiscoveryServicesDevicesListUpdatedReceiver = new
ARDiscoveryServicesDevicesListUpdatedReceiver(this);
        LocalBroadcastManager localBroadcastMgr =
```

```
LocalBroadcastManager.getInstance(getApplicationContext());

localBroadcastMgr.registerReceiver(mArdiscoveryServicesDevicesListUpdatedRe
ceiver, new
IntentFilter(ARDiscoveryService.kARDiscoveryServiceNotificationServicesDevi
cesListUpdated));
    }

    @Override
    public void onServicesDevicesListUpdated() {
        Log.d(TAG, "onServicesDevicesListUpdated ...");

        if (mArdiscoveryService != null) {
            List<ARDiscoveryDeviceService> deviceList =
mArdiscoveryService.getDeviceServicesArray();


        }

    }

    private ARDiscoveryDevice
createDiscoveryDevice(ARDiscoveryDeviceService service) {
        ARDiscoveryDevice device = null;

        if ((service != null) &&
(ARDISCOVERY_PRODUCT_ENUM.ARDISCOVERY_PRODUCT_ARDRONE.equals(ARDiscoverySer
vice.getProductFromProductID(service.getProductID())))) {
            try {
                device = new ARDiscoveryDevice();

                ARDiscoveryDeviceNetService netDeviceService =
(ARDiscoveryDeviceNetService) service.getDevice();


device.initWifi(ARDISCOVERY_PRODUCT_ENUM.ARDISCOVERY_PRODUCT_ARDRONE,
netDeviceService.getName(), netDeviceService.getIp(),
netDeviceService.getPort());
            } catch (ARDiscoveryException e) {
                e.printStackTrace();
                Log.e(TAG, "Error: " + e.getError());
            }
        }

        return device;
    }

    public ARDeviceController connecte_drone()
    {
        ARDeviceController deviceController = null;

        initDiscoveryService();
        startDiscovery();
        registerReceivers();
        onServicesDevicesListUpdated();
        for (int i=0;
i==mArdiscoveryService.getDeviceServicesArray().size(); i++)
        {
            try
            {
                deviceController = new ARDeviceController
```

```
(createDiscoveryDevice(mArdiscoveryService.getDeviceServicesArray().get(i))
);
                ARCONTROLLER_ERROR_ENUM error = deviceController.start();
            }
            catch (ARControllerException e)
            {
                e.printStackTrace();
            }
        }
        return deviceController ;
    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        enableButton = (Button) findViewById(R.id.button1);
        disableButton = (Button) findViewById(R.id.button2);
        faireTest = (Button) findViewById(R.id.button3);
        fairerectangleButton = (Button) findViewById(R.id.button4);
        fairecarreButton = (Button) findViewById(R.id.button5);
        faireunsautButton = (Button) findViewById(R.id.button6);
        selectWifiButton = (Button) findViewById(R.id.button7);
        exitappButton = (Button) findViewById(R.id.button8);


        enableButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                WifiManager wifi = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE);
                wifi.setWifiEnabled(true);
                afficherNotifEnableButton();
            }
        });

        disableButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                WifiManager wifi = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE);
                wifi.setWifiEnabled(false);
                afficherNotifDisableButton();
            }
        });

        selectWifiButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_MAIN, null);
                intent.addCategory(Intent.CATEGORY_LAUNCHER);
                ComponentName cn = new
ComponentName("com.android.settings",
"com.android.settings.wifi.WifiSettings");
                intent.setComponent(cn);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
            }

        });
```

```
        faireTest.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {

                ARDeviceController controller = connecte_drone();

controller.getFeatureJumpingSumo().setPilotingPCMDTurn((byte) 50);

                afficherNotifFairecarréButton();
            }
        });

        fairerectangleButton.setOnClickListener(new View.OnClickListener()
{
            public void onClick(View v) {
                /*ARDeviceController controller = connecte_drone();*/


                afficherNotifFairerectangleButton();
            }
        });

        fairecarreButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                /*ARDeviceController controller = connecte_drone();*/


                afficherNotifFairelosangeButton();
            }
        });

        faireunsautButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                /*ARDeviceController controller = connecte drone();*/


                afficherNotifFaireunsautButton();
            }
        });

        exitappButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {

               finish();
            }
        });
    }


    public void afficherNotifEnableButton() {
        Toast.makeText(this, "La WIFI a été activée", 10);
        test = Toast.makeText(this, "La WIFI a été activée", 10);
        test.show();
    }

    public void afficherNotifDisableButton() {
        Toast.makeText(this, "La WIFI a été désactivée", 10);
        test = Toast.makeText(this, "La WIFI a été désactivée", 10);
        test.show();
    }
```

```
    public void afficherNotifFairecarréButton() {
        Toast.makeText(this, "Le drone execute un carré", 10);
        test = Toast.makeText(this, "Le drone execute une action de test",
10);
        test.show();
    }

    public void afficherNotifFairerectangleButton() {
        Toast.makeText(this, "Le drone execute un rectangle", 10);
        test = Toast.makeText(this, "Le drone execute un rectangle", 10);
        test.show();
    }

    public void afficherNotifFairelosangeButton() {
        Toast.makeText(this, "Le drone execute un carré", 10);
        test = Toast.makeText(this, "Le drone execute un carré", 10);
        test.show();
    }

    public void afficherNotifFaireunsautButton() {
        Toast.makeText(this, "Le drone execute un saut", 10);
        test = Toast.makeText(this, "Le drone execute un saut", 10);
        test.show();
    }

}
```

Fichier activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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"
    tools:context="com.example.mjori.projettutpilotagedrone.MainActivity"
    tools:layout_editor_absoluteY="73dp"
    tools:layout_editor_absoluteX="0dp">


    <Button
        android:id="@+id/button1"
        android:layout_width="0dp"
        android:layout_height="54dp"
        android:layout_alignEnd="@+id/imageView"
        android:layout_centerVertical="true"
        android:text="Activer Wifi"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toTopOf="@+id/button7"
        tools:layout_constraintRight_creator="1"
        app:layout_constraintRight_toRightOf="@+id/button3"
        tools:layout_constraintLeft_creator="1"
        app:layout_constraintLeft_toLeftOf="@+id/button3" />

    <Button
        android:id="@+id/button2"
        android:layout_width="0dp"
        android:layout_height="54dp"
        android:layout_alignEnd="@+id/imageView"
        android:layout_centerVertical="true"
        android:text="Desactiver Wifi"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toTopOf="@+id/button7"
        tools:layout_constraintRight_creator="1"
        android:layout_marginStart="18dp"
        android:layout_marginEnd="19dp"
        app:layout_constraintRight_toRightOf="parent"
        tools:layout_constraintLeft_creator="1"
        app:layout_constraintLeft_toRightOf="@+id/button8" />

    <Button
        android:id="@+id/button3"
        android:layout_width="0dp"
        android:layout_height="54dp"
        android:layout_alignEnd="@+id/imageView"
        android:layout_centerVertical="true"
        android:text="Faire un test"
        android:layout_marginTop="8dp"
        app:layout_constraintTop_toBottomOf="@+id/relativeLayout"
        tools:layout_constraintRight_creator="1"
        app:layout_constraintRight_toRightOf="@+id/button5"
        tools:layout_constraintLeft_creator="1"
        app:layout_constraintLeft_toLeftOf="@+id/button5" />
```

```
<Button
    android:id="@+id/button4"
    android:layout_width="0dp"
    android:layout_height="54dp"
    android:layout_alignEnd="@+id/imageView"
    android:layout_centerVertical="true"
    android:text="Faire un rectangle"
    android:layout_marginTop="8dp"
    app:layout_constraintTop_toBottomOf="@+id/relativeLayout"
    tools:layout_constraintRight_creator="1"
    android:layout_marginEnd="16dp"
    app:layout_constraintRight_toRightOf="parent"
    tools:layout_constraintLeft_creator="1"
    app:layout_constraintLeft_toLeftOf="@+id/button6" />

<Button
    android:id="@+id/button5"
    android:layout_width="125dp"
    android:layout_height="54dp"
    android:layout_alignEnd="@+id/imageView"
    android:layout_centerVertical="true"
    android:text="Faire un carré"
    android:layout_marginTop="8dp"
    app:layout_constraintTop_toBottomOf="@+id/button3"
    android:layout_marginStart="16dp"
    tools:layout_constraintLeft_creator="1"
    app:layout_constraintLeft_toLeftOf="parent" />

<Button
    android:id="@+id/button6"
    android:layout_width="125dp"
    android:layout_height="54dp"
    android:layout_alignEnd="@+id/imageView"
    android:layout_centerVertical="true"
    android:text="Faire un saut"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintHorizontal_bias="0.944"
    android:layout_marginTop="8dp"
    app:layout_constraintTop_toBottomOf="@+id/button4" />

<Button
    android:id="@+id/button7"
    android:layout_width="125dp"
    android:layout_height="54dp"
    android:layout_alignEnd="@+id/imageView"
    android:layout_centerVertical="true"
    android:text="Selectionner Wifi"
    android:layout_marginBottom="8dp"
    app:layout_constraintBottom_toTopOf="@+id/relativeLayout"
    tools:layout_constraintRight_creator="1"
    app:layout_constraintRight_toRightOf="parent"
    tools:layout_constraintLeft_creator="1"
    app:layout_constraintLeft_toLeftOf="parent" />
```

```
<Button
        android:id="@+id/button8"
        android:layout_width="88dp"
        android:layout_height="48dp"
        android:layout_alignEnd="@+id/imageView"
        android:layout_centerVertical="true"
        android:text="X"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toTopOf="@+id/button7"
        android:layout_marginStart="18dp"
        tools:layout_constraintLeft_creator="1"
        app:layout_constraintLeft_toLeftOf="@+id/button7" />

    <RelativeLayout
        android:id="@+id/relativeLayout"
        android:layout_width="0dp"
        android:layout_height="304dp"
        android:background="@drawable/background"
        android:orientation="vertical"
        tools:layout_constraintTop_creator="1"
        tools:layout_constraintRight_creator="1"
        tools:layout_constraintBottom_creator="1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        tools:layout_constraintLeft_creator="1"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"></RelativeLayout>*/



</android.support.constraint.ConstraintLayout>
```

Table des Illustrations