Grenoble INP génie industriel

# Support de caméra GoPro

Livrable final

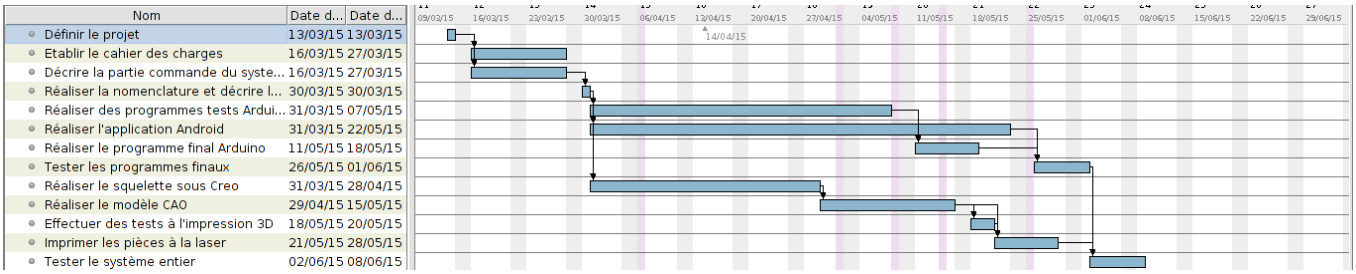
**Résumé du projet :** Notre objectif est de concevoir et fabriquer un support rotatif sur deux axes pour une caméra de la marque GoPro. De l'étude des attentes à la fabrication de la solution, nous effectuerons nous-même l'analyse fonctionnelle, la recherche de solutions techniques pertinentes ainsi que leur réalisation.

## Table des matières

# 1 Description du système

## 1.1 Gestion de Projet

Nous avons, tout d'abord, définit un planning du projet afin de respecter nos deadline pour les livrables dans les différents matières, mais aussi pouvoir avancer à un rythme convenable. Voici donc le planning que nous avions fixé :




Ensuite, nous avons répartis les tâches selon nos préférences et nos compétences. Voici donc la répartition des tâches, selon les ressources :

| Tâches | Lara | Guillaume | Gaël |
|---|---|---|---|
| Définir le produit | X | X | X |
| Établir le cahier des charges | X | X | X |
| Décrire la partie commande | X | X | X |
| Réaliser la nomenclature et décrire les composants | X | X | X |
| Réaliser l'application Android | | X | |
| Réaliser les programmes test Arduino | X | | |
| Réaliser le programme final Arduino | X | | |
| Tester le système électronique | X | X | |
| Réaliser le squelette sous Creo | X | | X |
| Réaliser le modèle CAO | X | | X |
| Imprimer les pièces | X | | X |
| Tester le produit final | X | X | X |

## 1.2 Cahier des charges

| Fonction de service | Critère d'appréciation | Niveau d'exigence | Flexibilité |
|---|---|---|---|
| FP1 : Orienter la caméra | Position angulaire en tilt | [ - 90° ; + 90° ] | aucune |
| | Position angulaire en pan | [ 0° ; 360° ] | aucune |
| FC1 : S'adapter au pied standard | Norme de fixation Standard | Vis universelle 1/4" | aucune |
| FC2 : S'adapter à l'environnement | Norme Maison | | aucune |
| FC3 : S'adapter aux dimensions des composants | Dimensions du boitier | 59*42*75 | aucune |
| FC4 : S'adapter à la caméra | Norme de fixation GoPro | Vis de serrage GoPro | aucune |
| FC5 : Être commandé par le téléphone | Norme de connexion Bluetooth | Shield Bluetooth | aucune |
| | Compatibilité Androïd | Versions 3 et ultérieures | aucune |

FIGURE 1 – Cahier des charges du projet GoPro

## 1.3 Schéma bloc de notre système

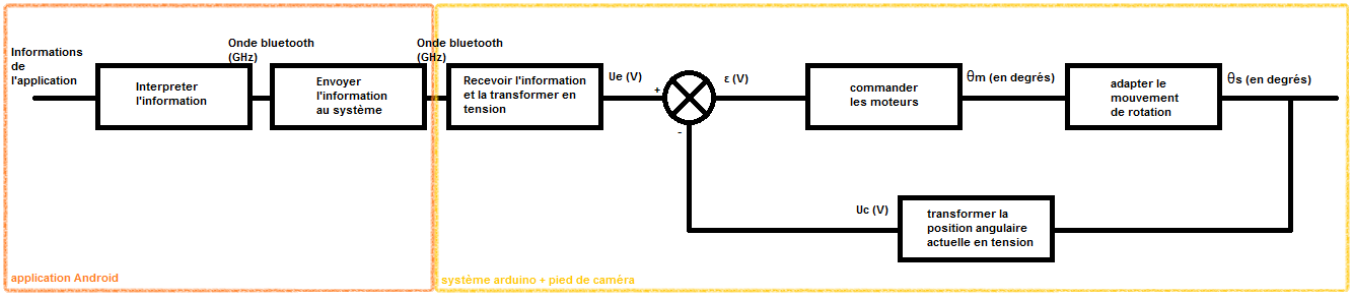



FIGURE 2 – Schéma bloc du projet GoPro

## 1.4 Nomenclature

Nous avons choisi de créer un système permettant la commande en rotation suivant l'axe de pan et de tilt d'une caméra GoPro, par le biais d'une application Android, d'un Arduino Uno et d'une interface de communication bluetooth. Rappelons les différents composants dont nous avons besoin pour notre produit final :

| | |
|---|---|
| 1 | Arduino Uno |
| 1 | Module bluetooth Pololu 2725 |
| 1 | Smartphone sous Android 4.0.0 (au minimum) |
| 2 | Servomoteur |
| 10 | Câbles de connexion |
| 1 | Système pour brancher une pile 9V |
| 1 | Pile 9V |

## 1.5 Description des composants

**Arduino Uno :** C'est une carte électronique programmable permettant de faire le lien entre des capteurs et des composants électroniques afin de faire effectuer à ceux ci des actions en fonction de la valeur mesurée par les capteurs.

Cette carte posséde 14 broches entrée/sortie, 6 entrées analogiques, une prise USB pour l'alimentation et la connexion à l'ordinateur (pour faire la programmation) et un microprocesseur ATmega328.

Elle doit être alimentée entre 7 et 12 volts et mesure 68.6 mm de longueur et 53.4 mm de largeur et pése 25 g.

`http://www.datasheetarchive.com/dl/Datasheets-UD9/DSARS0044624.pdf`

**Module Bluetooth Pololu 2725 :** Un shield est une carte électronique qui s'utilise de maniére couplée avec la carte Arduino Uno afin d'en augmenter les capacités, dans le cas du shield Bluetooth, il permet à l'Arduino Uno de recevoir des ondes bluetooth. Cela permet la communication de l'Arduino avec un autre périphérique bluetooth.

Alimentée à 5V, le module que nous avons choisi se branche sur une liaison série avec la carte Arduino Uno. Son interface de communication se fait donc par des

`https://www.pololu.com/product/2725`

**Smartphone sous Android 4.0.0 :** Il existe différentes version d'Android, et le choix de la version est la 4.0.0. En effet, cette version est plutôt récente mais permet de développer des applications plus sophistiquée. Nous l'avons donc choisie afin de s'abstenir de bug dû au poids de l'application, et la necessité de la réactivité du smartphone.

Tous nos tests seront effectués à partir d'un Nexus 6, développé par Google.

`http://www.google.com/nexus/6/`

**Servomoteur :** Le servomoteur est un moteur permettant de maintenir sa position qui est vérifiée en continu et corrigée en fonction de la mesure. C'est un système asservi. On peut donc ainsi contrôler le mouvement du moteur d'un angle défini par l'utilisateur.

Le servomoteur utilisé et le MG90s, il fonctionne sous une tension de 4.8 V - 6.0 V, il mesure : $22.5 \times 12 \times 35.5$ mm et pése 13.4 g.

`http://www.electronicoscaldas.com/datasheet/MG90S_Tower-Pro.pdf`

**Câbles :** Afin de relier nos différents composants électronique entre eux, mais aussi avec la carte Arduino, nous allons utiliser des câbles de type ceux que nous avions dans les mallettes. En effet, nous n'allons pas manipuler de grandes puissances électriques, nous n'avons donc pas besoin de câbles plus performant.

Du point de vue esthétique, notre produit devra tout de même paraître solide afin de satisfaire le client. Ainsi, bien que les câbles soient fins, nous allons définir leurs longueurs de manière à ce que les branchements soient lisses et fixes.

**Pile 9V :** Afin d'assurer une alimentation continue et suffisante de notre système électronique, nous alimentons notre carte Arduino à l'aide d'une pile basique de 9V.

Les dimensions de la pile choisie sont : 48,5 mm x 26,5 mm x 17,5 mm. Et la capacité typique de la pile Alcaline 9V est de 550mAh.

`http://fr.wikipedia.org/wiki/Pile_9_volts`

# 2 Programmation

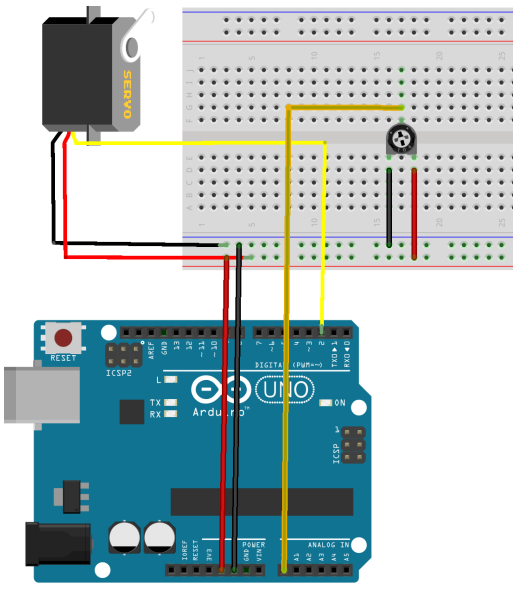
## 2.1 Programmes tests

### 2.1.1 Commande d'un servomoteur via un potentiomètre

Nous avons choisi de tout d'abord simuler la commande du servomoteur par l'application Android grâce à un potentiomètre. En effet, de la même manière que le potentiomètre branché en entrée analogique, l'application va en continu envoyer une information à l'arduino par l'interface bluetooth.

Voici donc le programme Arduino ainsi que son schéma de branchement pour cette simulation :

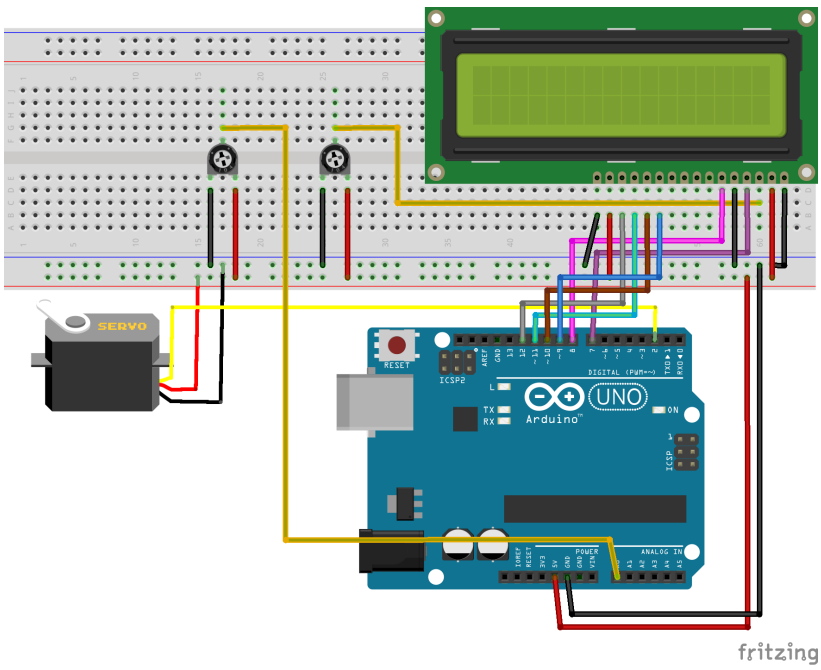
```
#include <Servo.h> // bibliothque des servomoteurs

int potentiometre = 0; // fixe le potentiometre sur l'entree analogique 0;
int pinServo = 2; // fixe l'entree du sevomoteur sur la sortie 2

Servo servomoteur; // creation du servomoteur
double val = 0.0; // initialise la valeur val  0.0

void setup() {
  servomoteur.attach(pinServo); // lie le servomoteur avec la sortie pinServo
  Serial.begin(9600); // fixe la serie de lecture  9600 bauds
}

void loop() {
  int sensorValue=analogRead(potentiometre); // lecture de la valeur du potentiometre
  val = sensorValue * (180.0/1023.0); // converssion en angle
  servomoteur.write(val); // commande du moteur
  Serial.println(val); // affiche la valeur de l'angle du moteur sur le moniteur serie
  delay(100);
}
```



### 2.1.2 Commande d'un servomoteur via un potentiomètre et affichage sur un écran LCD

Nous avons voulu ajouter une fonctionnalité supplémentaire au programme précédent. En effet, le potentiomètre permettait de commander le servomoteur mais nous ne disposions d'aucun moyen de controler et d'avoir un retour de l'angle du servomoteur. C'est pourquoi nous avons décider d'ajouter un écran LCD qui permet ainsi d'afficher la position actuelle du servomoteur. Cet écran LCD et le potentiomètre simulent ainsi le smartphone et les fonctionnalités de l'application Android.

```
#include <LiquidCrystal.h>
#include <Servo.h>

int potentiometre = 0;
int pinServo = 2;

LiquidCrystal lcd(7,8,9,10,11,12);
Servo servomoteur;
double val = 0.0;

void setup() {
  // put your setup code here, to run once:
  servomoteur.attach(pinServo);
  lcd.begin(16,2);
  lcd.print("L'angle est de");
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  int sensorValue=analogRead(potentiometre);
  val = sensorValue * (180.0/1023.0);
  servomoteur.write(val);
  Serial.println(val);
  lcd.setCursor(0,1);
  lcd.print(val);
  lcd.print(" degres");

  delay(100);
}
```



### 2.1.3 Mise en place de l'application android : test pour l'écran de contrôle

Afin d'apprendre à faire des applications android, nous avons commencer par faire l'écran de contrôle mais sans envoyer les données via le bluetooth afin de voir le fonctionnement d'une application basique avec quelques boutons :

**Code Java :** les fonctionnalités de cette application sont basiques, juste permetre d'aumgmenter et diminuer deux entiers à l'aide de boutons et les afficher à l'écran afin de représenter la postion en pan et tilt.

```
package com.example.guillaume.projetgopro;

import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;



public class MainActivity extends Activity implements View.OnTouchListener, View.OnClickListener {
    //la position en tilt par dfaut
    private int posTilt = 0;
    //la position en pan par dfaut
    private int posPan = 0;

    TextView tilt = null;
    TextView pan = null;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // rcupration  des vues
        Button raz = (Button) findViewById(R.id.raz);
        Button tiltDown = (Button)findViewById(R.id.tiltDown);
        Button tiltUp = (Button)findViewById(R.id.tiltUp);
        Button panLeft = (Button)findViewById(R.id.panLeft);
        Button panRight = (Button)findViewById(R.id.panRight);

        tilt = (TextView)findViewById(R.id.tilt);
        pan = (TextView)findViewById(R.id.pan);

        //mis en plance des listeners
        tiltUp.setOnClickListener((android.view.View.OnClickListener) this);
        tiltDown.setOnClickListener((android.view.View.OnClickListener) this);
        panLeft.setOnClickListener((android.view.View.OnClickListener) this);
        panRight.setOnClickListener((android.view.View.OnClickListener) this);
        raz.setOnClickListener((android.view.View.OnClickListener) this);
    }

    @Override
    public boolean onTouch(View view, MotionEvent event){
        return true;
    }

    @Override
    public void onClick(View view){
        Button bouton = (Button)view;
        switch (bouton.getId()){
            case R.id.tiltUp:
                if(posTilt<90){
                    posTilt += 5;
                }
                break;
            case R.id.tiltDown:
```

```
                if(posTilt>-90){
                    posTilt -= 5;
                }
                break;
            case R.id.panLeft:
                if(posPan>-90){
                    posPan -= 5;
                }
                break;
            case R.id.panRight:
                if(posPan<90){
                    posPan += 5;
                }
                break;
            case R.id.raz:
                posPan = 0;
                posTilt = 0;
        }

        tilt.setText("Pan : " + posPan);
        pan.setText("Tilt : " + posTilt);
    }
}
```

**Code XML :** On positionne les boutons et le texte sur l'écran grâce au code suivant :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tilt"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Tilt : "
        android:textStyle="bold"
        android:textColor="#000000"
        android:gravity="center"
        android:padding="10dp"
        />
    <TextView
        android:id="@+id/pan"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Pan : "
        android:textStyle="bold"
        android:textColor="#000000"
        android:gravity="center"
        android:padding="10dp"
        android:layout_below="@id/tilt"
        />
    <Button
        android:id="@+id/raz"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RAZ"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        />
    <Button
```

```
        android:id="@+id/tiltUp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="tilt up"
        android:layout_above="@id/raz"
        android:layout_centerHorizontal="true"
        />
    <Button
        android:id="@+id/panLeft"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="pan left"
        android:layout_toLeftOf="@id/raz"
        android:layout_below="@id/tiltUp"
        />
    <Button
        android:id="@+id/panRight"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="pan right"
        android:layout_toRightOf="@id/raz"
        android:layout_below="@id/tiltUp"
        />
    <Button
        android:id="@+id/tiltDown"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="tilt down"
        android:layout_below="@+id/raz"
        android:layout_centerHorizontal="true"
        />

</RelativeLayout>
```

On obtient l'interface graphique donnée en figure 3.


FIGURE 3 – interface graphique du premier test d'application android

### 2.1.4 Mise en place de la connexion bluetooth sous android

Ensuite nous avons du tester la façon de mettre en place la connexion bluetooth sous un terminal android, pour cela nous avons commencer par une application simple : allumer ou éteindre un LED à distance.

**Code Java :** Cette application se compose d'une page de connexion et d'une page de contrôle de la LED.
La page de connexion sert à mettre en place la liaison bluetooth :

```
package com.example.guillaume.ledarduino;

import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Set;


public class MainActivity extends ActionBarActivity
{
    //widgets
    Button btnPaired;
    ListView devicelist;
    //Bluetooth
    private BluetoothAdapter myBluetooth = null;
    private Set<BluetoothDevice> pairedDevices;
    public static String EXTRA_ADDRESS = "device_address";

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Calling widgets
        btnPaired = (Button)findViewById(R.id.peri);
        devicelist = (ListView)findViewById(R.id.deviceList);

        //if the device has bluetooth
        myBluetooth = BluetoothAdapter.getDefaultAdapter();

        if(myBluetooth == null)
        {
            //Show a mensag. that the device has no bluetooth adapter
            Toast.makeText(getApplicationContext(), "Bluetooth Device Not Available",
                Toast.LENGTH_LONG).show();

            //finish apk
            finish();
        }
        else if(!myBluetooth.isEnabled())
        {
            //Ask to the user turn the bluetooth on
            Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(turnBTon,1);
        }
```

```
    btnPaired.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v)
        {
            pairedDevicesList();
        }
    });

}

private void pairedDevicesList()
{
    pairedDevices = myBluetooth.getBondedDevices();
    ArrayList list = new ArrayList();

    if (pairedDevices.size()>0)
    {
        for(BluetoothDevice bt : pairedDevices)
        {
            list.add(bt.getName() + "\n" + bt.getAddress()); //Get the device's name and the address
        }
    }
    else
    {
        Toast.makeText(getApplicationContext(), "No Paired Bluetooth Devices Found.",
            Toast.LENGTH_LONG).show();
    }

    final ArrayAdapter adapter = new ArrayAdapter(this,android.R.layout.simple_list_item_1, list);
    devicelist.setAdapter(adapter);
    devicelist.setOnItemClickListener(myListClickListener); //Method called when the device from the
        list is clicked

}

private AdapterView.OnItemClickListener myListClickListener = new AdapterView.OnItemClickListener()
{
    public void onItemClick (AdapterView<?> av, View v, int arg2, long arg3)
    {
        // Get the device MAC address, the last 17 chars in the View
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);

        // Make an intent to start next activity.
        Intent i = new Intent(MainActivity.this, ledControl.class);

        //Change the activity.
        i.putExtra(EXTRA_ADDRESS, address); //this will be received at ledControl (class) Activity
        startActivity(i);
    }
};


@Override
public boolean onCreateOptionsMenu(Menu menu)
{
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
```

```
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Puis la seconde activité permet d'allumer ou d'éteindre la LED ou bien de régler sa luminosité :

```
package com.example.guillaume.ledarduino;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
        import android.view.Menu;
        import android.view.MenuItem;

        import android.bluetooth.BluetoothSocket;
        import android.content.Intent;
        import android.view.View;
        import android.widget.Button;
        import android.widget.SeekBar;
        import android.widget.TextView;
        import android.widget.Toast;
        import android.app.ProgressDialog;
        import android.bluetooth.BluetoothAdapter;
        import android.bluetooth.BluetoothDevice;
        import android.os.AsyncTask;

        import java.io.IOException;
        import java.util.UUID;


public class ledControl extends ActionBarActivity {

    Button btnOn, btnOff, btnDis;
    SeekBar brightness;
    TextView lumn;
    String address = null;
    private ProgressDialog progress;
    BluetoothAdapter myBluetooth = null;
    BluetoothSocket btSocket = null;
    private boolean isBtConnected = false;
    //SPP UUID. Look for it
    static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        Intent newint = getIntent();
        address = newint.getStringExtra(MainActivity.EXTRA_ADDRESS); //receive the address of the bluetooth
            device

        //view of the ledControl
        setContentView(R.layout.activity_led_control);

        //call the widgtes
        btnOn = (Button)findViewById(R.id.on);
        btnOff = (Button)findViewById(R.id.off);
        btnDis = (Button)findViewById(R.id.deco);
        brightness = (SeekBar)findViewById(R.id.seekBar);
        lumn = (TextView)findViewById(R.id.lumn);
```

```
    new ConnectBT().execute(); //Call the class to connect

    //commands to be sent to bluetooth
    btnOn.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            turnOnLed();    //method to turn on
        }
    });

    btnOff.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v)
        {
            turnOffLed(); //method to turn off
        }
    });

    btnDis.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            Disconnect(); //close connection
        }
    });

    brightness.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
            if (fromUser==true)
            {
                lumn.setText(String.valueOf(progress));
                try
                {
                    btSocket.getOutputStream().write(String.valueOf(progress).getBytes());
                }
                catch (IOException e)
                {

                }
            }
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {

        }
    });
}

private void Disconnect()
{
    if (btSocket!=null) //If the btSocket is busy
    {
        try
        {
            btSocket.close(); //close connection
        }
```

```
        catch (IOException e)
        { msg("Error");}
    }
    finish(); //return to the first layout

}

private void turnOffLed()
{
    if (btSocket!=null)
    {
        try
        {
            btSocket.getOutputStream().write("TF".toString().getBytes());
        }
        catch (IOException e)
        {
            msg("Error");
        }
    }
}

private void turnOnLed()
{
    if (btSocket!=null)
    {
        try
        {
            btSocket.getOutputStream().write("TO".toString().getBytes());
        }
        catch (IOException e)
        {
            msg("Error");
        }
    }
}

// fast way to call Toast
private void msg(String s)
{
    Toast.makeText(getApplicationContext(),s,Toast.LENGTH_LONG).show();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_led_control, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

private class ConnectBT extends AsyncTask<Void, Void, Void> // UI thread
{
```

```
        private boolean ConnectSuccess = true; //if it's here, it's almost connected

        @Override
        protected void onPreExecute()
        {
            progress = ProgressDialog.show(ledControl.this, "Connecting...", "Please wait!!!"); //show a
                progress dialog
        }

        @Override
        protected Void doInBackground(Void... devices) //while the progress dialog is shown, the connection
            is done in background
        {
            try
            {
                if (btSocket == null || !isBtConnected)
                {
                    myBluetooth = BluetoothAdapter.getDefaultAdapter();//get the mobile bluetooth device
                    BluetoothDevice dispositivo = myBluetooth.getRemoteDevice(address);//connects to the
                        device's address and checks if it's available
                    btSocket = dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID);//create a
                        RFCOMM (SPP) connection
                    BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                    btSocket.connect();//start connection
                }
            }
            catch (IOException e)
            {
                ConnectSuccess = false;//if the try failed, you can check the exception here
            }
            return null;
        }
        @Override
        protected void onPostExecute(Void result) //after the doInBackground, it checks if everything went
            fine
        {
            super.onPostExecute(result);

            if (!ConnectSuccess)
            {
                msg("Connection Failed. Is it a SPP Bluetooth? Try again.");
                finish();
            }
            else
            {
                msg("Connected.");
                isBtConnected = true;
            }
            progress.dismiss();
        }
    }
}
```

**Code XML :** Ensuite on défini l'interface graphique de ces activité grâce au code suivant :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Liste des priphriques bluetooth"
        android:id="@+id/devices"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" Priphriques "
        android:id="@+id/peri"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/deviceList"
        android:layout_above="@+id/peri"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/devices" />
</RelativeLayout>
```

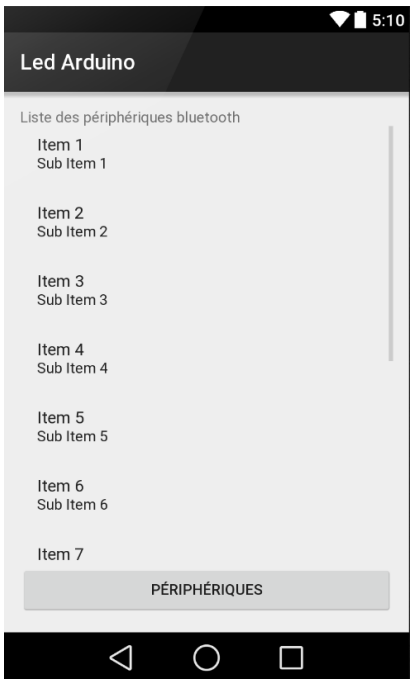



FIGURE 4 – Interface graphique de l'activité 1 du test 2

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.example.guillaume.ledarduino.ledControl">

    <Button
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="On"
        android:id="@+id/on"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="OFF"
        android:id="@+id/off"
        android:layout_below="@+id/on"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignRight="@+id/on"
        android:layout_alignEnd="@+id/on" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="DISCONNECT"
        android:id="@+id/deco"
        android:layout_centerVertical="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <SeekBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/seekBar"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignRight="@+id/deco"
        android:layout_alignEnd="@+id/deco" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Luminosit :"
        android:id="@+id/lumn"
        android:layout_above="@+id/seekBar"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

</RelativeLayout>
```

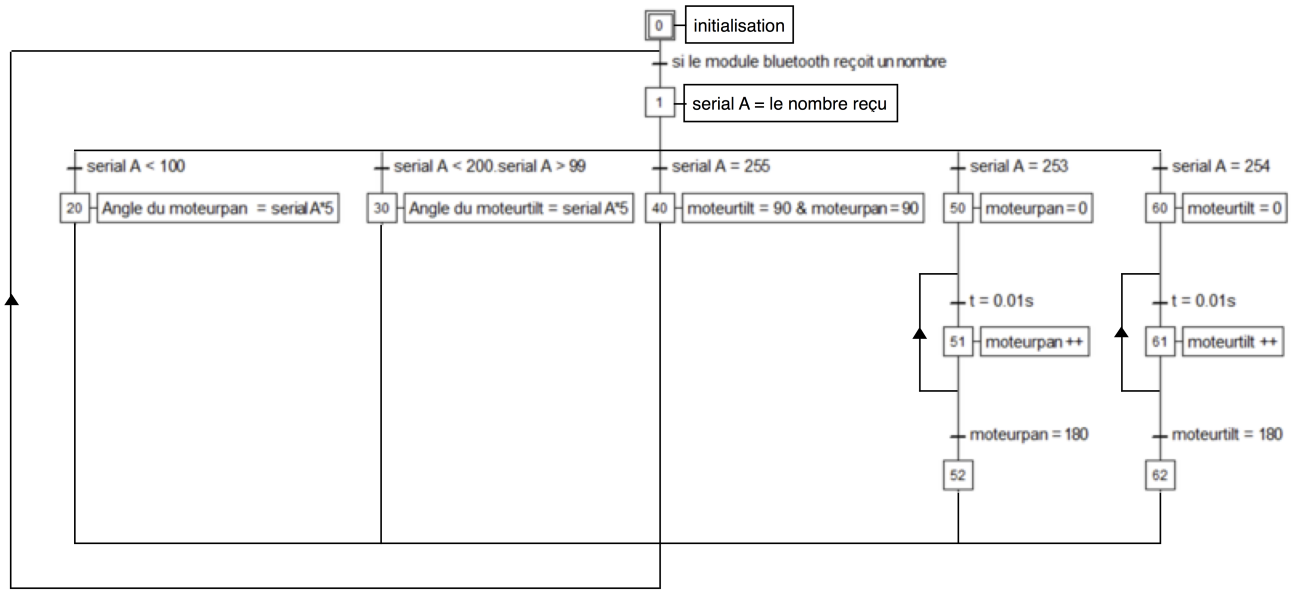

FIGURE 5 – Interface graphique de l'activité 2 du test 2

## 2.2 Programmes finaux

### 2.2.1 Commande des deux servomoteurs via le module bluetooth

Voici enfin notre programme “final”. Le principe est de recevoir un nombre à 6chiffres par bluetooth, et de récupérer les 3premiers (de droite à gauche) qui correspondent à l’angle en pan puis les 3suivants qui correspondent à l’angle en tilt. Le programme comprend une fonction qui permet d’initialiser la connexion bluetooth.

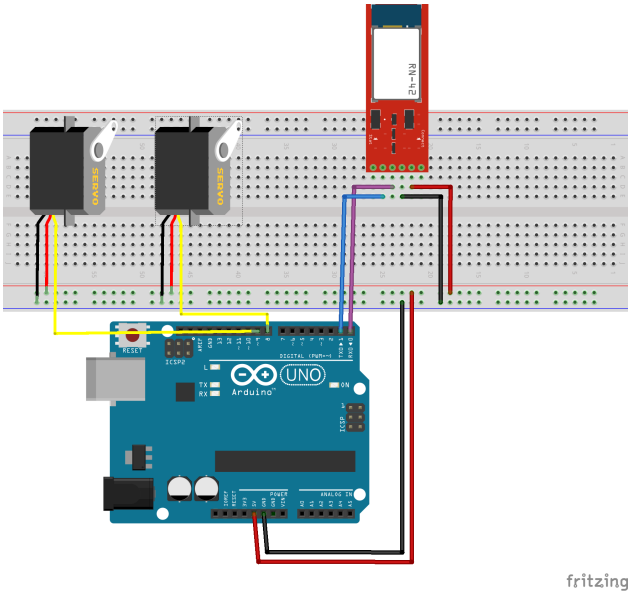
Tout d’abord l’architecture globale du programme Arduino, pour simplifier la compréhension du programme :




```
#include <Servo.h> // bibliothque du servomoteur

#include <SoftwareSerial.h>

Servo pan ; // cration du servomoteur de rotation en pan
Servo tilt; // cration du servomoteur de rotation en tilt

SoftwareSerial mavoieserie(10, 11);
long serialA; // variable de reception de donne via RX

int anglePan = 90;
int angleTilt = 90;

void setup()
{
  mavoieserie.begin(115200); // initialisation de la connexion srie (avec le module bluetooth)
  Serial.begin(9600);
  setupBlueToothConnection(); // dmarrage liason srie bluetooth cf fonction en bas

  pan.attach(8); // initialisation du moteur pan, il est branch sur la sortie 8
  tilt.attach(9); // initialisation du moteur tilt, il est branch sur la sortie 9
```

```
}

void loop() {
  if (mavoieserie.available()){

      serialA = mavoieserie.read();
      Serial.println(serialA);
      if (serialA <100){
        anglePan = serialA*5; // l'angle en pan voulu est les trois premiers chiffres du nombre recu. Si
            serialA = 120105 alors anglePan = 120
        pan.write(anglePan); // commande du moteur pan
        tilt.write(angleTilt); // commande du moteur tilt
      }
      else if (serialA == 255){
        if (anglePan <90){
           for (int i = 0; i <= 90-anglePan; i++){
              pan.write(anglePan+i);
              delay(10);
            }
        } else if (anglePan > 90){
           for (int i = 0; i <= anglePan-90; i++){
              pan.write(anglePan-i);
              delay(10);
            }
        }
        if (angleTilt<90){
           for (int i = 0; i <= 90-angleTilt; i++){
              tilt.write(angleTilt+i);
              delay(10);
            }
        } else if (angleTilt > 90){
           for (int i = 0; i <= angleTilt-90; i++){
              tilt.write(angleTilt-i);
              delay(10);
            }
        }
        anglePan = 90;
        angleTilt = 90;
      }
      else if (serialA == 253){
        for (int i = 0; i <= anglePan; i++){
          pan.write(anglePan-i);
          delay(10);
        }
        for (int i = 0; i<= 180 ; i++){
          pan.write(i);

          tilt.write(90);
          delay(50);
        }
        anglePan = 180;
      }
      else if (serialA == 254){
         for (int i = 0; i <= angleTilt; i++){
          tilt.write(angleTilt-i);
          delay(10);
        }
        for (int i = 0; i<= 180 ; i++){
          tilt.write(i);
          pan.write(90);
          delay(50);
        }
        angleTilt = 180;
      }
      else {
        angleTilt = (serialA-100)*5; // l'angle en tilt voulu est les trois derniers chiffres du nombre
            recu. Pour l'exemple prcedent, angleTilt = 105
```

```
        tilt.write(angleTilt); // commande du moteur tilt
        pan.write(anglePan); // commande du moteur pan
      }
      Serial.println(anglePan);
      Serial.println(angleTilt);
    }
}

void setupBlueToothConnection() // fonction de configuration du module bluetooth
{
  mavoieserie.begin(115200); //vitesse de bluetooth
  mavoieserie.print("\r\n+STBD=115200\r\n"); // fixe la vitesse du bluetooth
  mavoieserie.print("\r\n+STWMOD=0\r\n"); // bluetooth en mode esclave
  mavoieserie.print("\r\n+STNA=Arduino"); // nom de l'appareil
  mavoieserie.print("\r\n+STPIN=0000\r\n");// code pin "0000"
  mavoieserie.print("\r\n+STOAUT=1\r\n"); // permet les appareils de se connecter  l'arduino
  mavoieserie.print("\r\n+STAUTO=0\r\n"); // l'autoconnexion est empche
  delay(2000);
  mavoieserie.print("\r\n+INQ=1\r\n"); // rend le mode esclave non modifiable
  delay(2000);
  mavoieserie.flush();
}
```




### 2.2.2 Programme de l'application Android

Nous avons choisi de créer une application Android qui servira de télécommande pour commander la position de la caméra à distance en communiquant par Bluetooth avec l'arduino.

Pour cela l'application se décompose en deux activités (pages actives de l'application) :

— la première servant de page de connexion Bluetooth (vérifie si le Bluetooth est activé et l'active le cas échéant et donne la liste des périphériques connus)

— La seconde permet de sélectionner la position de la caméra et d'envoyer cette information par Bluetooth au périphérique précédemment sélectionné.

**Activité de connexion Bluetooth :** Une activité Android se décompose en deux parties :

— Une partie codée sous java qui permet de coder les fonctionnalités de l'activité (que faire dans quel cas)

— Une partie codée sous XML qui permet de coder l'interface graphique de l'activité (quel bouton à quel endroit)

**Code java :** Ce code permet donc d'initialiser le serveur Bluetooth et de se connecter à un périphérique sur le réseau.

```
package com.example.guillaume.projet_gopro_bluetooth;

import android.app.Activity;
import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Set;


public class DeviceList extends Activity {
    //widgets
    Button btnPaired;
```

```
ListView devicelist;

//Bluetooth
private BluetoothAdapter myBluetooth = null;
private Set<BluetoothDevice> pairedDevices;
public static String EXTRA_ADDRESS = "device_address";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_device_list);

    //Calling widgets
    btnPaired = (Button)findViewById(R.id.button);
    devicelist = (ListView)findViewById(R.id.listView);

    //if the device has bluetooth
    myBluetooth = BluetoothAdapter.getDefaultAdapter();

    if(myBluetooth == null)
    {
        //Show a mensag. that the device has no bluetooth adapter
        Toast.makeText(getApplicationContext(), "Bluetooth Device Not Available",
            Toast.LENGTH_LONG).show();

        //finish apk
        finish();
    }
    else if(!myBluetooth.isEnabled())
    {
        //Ask to the user turn the bluetooth on
        Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(turnBTon,1);
    }

    btnPaired.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v)
        {
            pairedDevicesList();
        }
    });
}

private void pairedDevicesList()
{
    pairedDevices = myBluetooth.getBondedDevices();
    ArrayList list = new ArrayList();

    if (pairedDevices.size()>0)
    {
        for(BluetoothDevice bt : pairedDevices)
        {
            list.add(bt.getName() + "\n" + bt.getAddress()); //Get the device's name and the address
        }
    }
    else
    {
        Toast.makeText(getApplicationContext(), "No Paired Bluetooth Devices Found.",
            Toast.LENGTH_LONG).show();
    }

    final ArrayAdapter adapter = new ArrayAdapter(this,android.R.layout.simple_list_item_1, list);
    devicelist.setAdapter(adapter);
    devicelist.setOnItemClickListener(myListClickListener); //Method called when the device from the
         list is clicked
```

```
    }

    private AdapterView.OnItemClickListener myListClickListener = new AdapterView.OnItemClickListener()
    {
        public void onItemClick (AdapterView<?> av, View v, int arg2, long arg3)
        {
            // Get the device MAC address, the last 17 chars in the View
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);

            // Make an intent to start next activity.
            Intent i = new Intent(DeviceList.this, goProControl.class);

            //Change the activity.
            i.putExtra(EXTRA_ADDRESS, address); //this will be received at ledControl (class) Activity
            startActivity(i);
        }
    };
}
```

**Code XML :** Ce code permet de définir l'interface graphique de la première activité (placement du texte, de la liste des périphérique et du bouton d'actualisation).

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".DeviceList">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Appareils connects :"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Appareils connects"
        android:id="@+id/button"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/textView"
        android:layout_above="@+id/button" />
</RelativeLayout>
```

**Activité de commande de la caméra :**

**Code Java :** Cette activité permet d'envoyer via Bluetooth un entier de type int (entre 0 et 255) qui sera interprété par l'arduino afin de faire bouger les servomoteurs. Voici un tableau récapitulatif de l'interprétation des nombres envoyés :

| Nombres | Signification |
|---|---|
| $n \in [\|0, 36\|]$ | une mise en position en pan de $5n°$ |
| $n \in [\|100, 136\|]$ | une mise en position en tilt de $5n°$ |
| 253 | balayage pan |
| 254 | balayage tilt |
| 255 | déconnexion du périphérique Bluetooth |

```
package com.example.guillaume.projet_gopro_bluetooth;

import android.app.Activity;
import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.util.UUID;


public class goProControl extends Activity implements View.OnTouchListener, View.OnClickListener {

    //la position en tilt par dfaut
    private long posTilt = 0;
    //la position en pan par dfaut
    private long posPan = 0;
    //le nombre  six chiffres  envoyer en bluetooth
    private long finalnb = 0;

    TextView tilt = null;
    TextView pan = null;

    String address = null;
    private ProgressDialog progress;
    BluetoothAdapter myBluetooth = null;
    BluetoothSocket btSocket = null;
    private boolean isBtConnected = false;

    //SPP UUID. Look for it
    static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_go_pro_control);

        Intent newint = getIntent();
        address = newint.getStringExtra(DeviceList.EXTRA_ADDRESS); //receive the address of the bluetooth
            device
```

```
// rcupration  des vues
Button raz = (Button) findViewById(R.id.raz);
Button tiltDown = (Button)findViewById(R.id.tiltDown);
Button tiltUp = (Button)findViewById(R.id.tiltUp);
Button panLeft = (Button)findViewById(R.id.panLeft);
Button panRight = (Button)findViewById(R.id.panRight);
Button dis = (Button)findViewById(R.id.button2);
Button bpan = (Button)findViewById(R.id.bpan);
Button btilt = (Button)findViewById(R.id.btilt);
SeekBar seekBar = (SeekBar)findViewById(R.id.seekBar);
SeekBar seekBar1 = (SeekBar)findViewById(R.id.seekBar2);

tilt = (TextView)findViewById(R.id.tilt);
pan = (TextView)findViewById(R.id.pan);

new ConnectBT().execute(); //Call the class to connect

//mis en plance des listeners
tiltUp.setOnClickListener((android.view.View.OnClickListener) this);
tiltDown.setOnClickListener((android.view.View.OnClickListener) this);
panLeft.setOnClickListener((android.view.View.OnClickListener) this);
panRight.setOnClickListener((android.view.View.OnClickListener) this);
raz.setOnClickListener((android.view.View.OnClickListener) this);
dis.setOnClickListener((android.view.View.OnClickListener) this);
bpan.setOnClickListener((android.view.View.OnClickListener) this);
btilt.setOnClickListener((android.view.View.OnClickListener) this);
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    int progress = 0;

    @Override
    public void onProgressChanged(SeekBar seekBar, int progresValue, boolean fromUser) {
        progress = progresValue;
        posPan +=progress;
        tilt.setText("Pan : " + posPan);
        pan.setText("Tilt : " + posTilt);
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});

seekBar1.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    int progress = 0;

    @Override
    public void onProgressChanged(SeekBar seekBar, int progresValue, boolean fromUser) {
        progress = progresValue;
        posTilt += progress;
        tilt.setText("Pan : " + posPan);
        pan.setText("Tilt : " + posTilt);
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
```

```
        }
    });
}

@Override
public boolean onTouch(View view, MotionEvent event){
    return true;
}

@Override
public void onClick(View view){
    Button bouton = (Button)view;
    switch (bouton.getId()){
        case R.id.tiltUp:
            if(posTilt<180){
                posTilt += 5;
                finalnb = posTilt/5 + 100;
            }
            break;
        case R.id.tiltDown:
            if(posTilt>0){
                posTilt -= 5;
                finalnb = posTilt/5 + 100;
            }
            break;
        case R.id.panLeft:
            if(posPan>0){
                posPan -= 5;
                finalnb = posPan/5;
            }
            break;
        case R.id.panRight:
            if(posPan<180){
                posPan += 5;
                finalnb = posPan/5;
            }
            break;
        case R.id.raz:
            posPan = 90;
            posTilt = 90;
            finalnb = 255;
            break;
        case R.id.bpan:
            posTilt = 90;
            posPan = 180;
            finalnb = 253;
            break;
        case R.id.btilt:
            posPan = 90;
            posTilt = 180;
            finalnb = 254;
            break;
        case R.id.button2:
            if (btSocket!=null) //If the btSocket is busy
            {
                try
                {
                    btSocket.close(); //close connection
                }
                catch (IOException e)
                { msg("Error");}
            }
            finish(); //return to the first layout
    }

    tilt.setText("Pan : " + posPan);
```

```
    pan.setText("Tilt : " + posTilt);
    if (btSocket!=null)
    {
        try
        {
            btSocket.getOutputStream().write((int) (finalnb));
        }
        catch (IOException e)
        {
            msg("Error");
        }
    }

}

// fast way to call Toast
private void msg(String s)
{
    Toast.makeText(getApplicationContext(),s, Toast.LENGTH_LONG).show();
}

private void Disconnect()
{
    if (btSocket!=null) //If the btSocket is busy
    {
        try
        {
            btSocket.close(); //close connection
        }
        catch (IOException e)
        { msg("Error");}
    }
    finish(); //return to the first layout

}

private class ConnectBT extends AsyncTask<Void, Void, Void> // UI thread
{
    private boolean ConnectSuccess = true; //if it's here, it's almost connected

    @Override
    protected void onPreExecute()
    {
        progress = ProgressDialog.show(goProControl.this, "Connecting...", "Please wait!!!"); //show a
            progress dialog
    }

    @Override
    protected Void doInBackground(Void... devices) //while the progress dialog is shown, the connection
        is done in background
    {
        try
        {
            if (btSocket == null || !isBtConnected)
            {
                myBluetooth = BluetoothAdapter.getDefaultAdapter();//get the mobile bluetooth device
                BluetoothDevice dispositivo = myBluetooth.getRemoteDevice(address);//connects to the
                    device's address and checks if it's available
                btSocket = dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID);//create a
                    RFCOMM (SPP) connection
                BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                btSocket.connect();//start connection
            }
        }
        catch (IOException e)
        {
            ConnectSuccess = false;//if the try failed, you can check the exception here
```

```
            }
            return null;
        }
        @Override
        protected void onPostExecute(Void result) //after the doInBackground, it checks if everything went
            fine
        {
            super.onPostExecute(result);

            if (!ConnectSuccess)
            {
                msg("Connection Failed. Is it a SPP Bluetooth? Try again.");
                finish();
            }
            else
            {
                msg("Connected.");
                isBtConnected = true;
            }
            progress.dismiss();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_go_pro_control, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

**Code XML :** La partie graphique de cette activité est très simple, seulement des boutons, des sliders et des zones de texte qui se mettent à jour en fonctions angles en pan et tilt.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tilt"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Tilt : "
        android:textStyle="bold"
        android:textColor="#000000"
```

```
    android:gravity="center"
    android:padding="10dp"
    />
<TextView
    android:id="@+id/pan"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Pan : "
    android:textStyle="bold"
    android:textColor="#000000"
    android:gravity="center"
    android:padding="10dp"
    android:layout_below="@id/tilt"
    />
<Button
    android:id="@+id/raz"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="RAZ"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    />
<Button
    android:id="@+id/tiltUp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="tilt up"
    android:layout_above="@id/raz"
    android:layout_centerHorizontal="true"
    />
<Button
    android:id="@+id/panLeft"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="pan left"
    android:layout_toLeftOf="@id/raz"
    android:layout_below="@id/tiltUp"
    />
<Button
    android:id="@+id/panRight"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="pan right"
    android:layout_toRightOf="@id/raz"
    android:layout_below="@id/tiltUp"
    />
<Button
    android:id="@+id/tiltDown"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="tilt down"
    android:layout_below="@+id/raz"
    android:layout_centerHorizontal="true"
    />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Dconnecter"
    android:id="@+id/button2"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignRight="@+id/pan"
    android:layout_alignEnd="@+id/pan"
    />
<Button
    android:id="@+id/bpan"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Balayage pan"
        android:layout_above="@+id/button2"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_toStartOf="@+id/raz"
        android:layout_toLeftOf="@+id/raz"
        />
    <Button
        android:id="@+id/btilt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="balayage tilt"
        android:layout_above="@+id/button2"
        android:layout_alignRight="@+id/button2"
        android:layout_alignEnd="@+id/button2"
        android:layout_toRightOf="@+id/raz"
        android:layout_toEndOf="@+id/raz"
        />

    <SeekBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/seekBar"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_above="@+id/bpan"
        android:layout_alignRight="@+id/button2"
        android:layout_alignEnd="@+id/button2" />

    <SeekBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/seekBar2"
        android:layout_above="@+id/seekBar"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

</RelativeLayout>
```

# 3 Conclusion et Perspective

Ce projet fut une grande expérience de groupe : nous avons apprécié la grande liberté dont nous avons pu jouir dans la répartition du travail, la réalisation des tâches, la rectification de nos erreurs. La supervisation de nos professeurs nous permettait de trouver par nous-même de nouvelles pistes d'évolution à chaque entrevue, à chaque confrontation des idées.

Le libre choix qui nous était laissé de nous concentrer essentiellement sur un aspect ou un autre du développement de notre produit nous a permis de nous épanouir dans cette tâche. Chacun a pu trouver son domaine d'efficacité - indispensable pour achever au plus vite le projet - tout en partageant avec les deux autres membres pour garantir une avance commune et équitable dans notre travail, sans que nous restions cloisonnés dans nos domaines favoris respectifs. Nos compétences étaient complémentaires (informatique, conception, mécanique) et nos outils de travail (informatiques avant tout, LaTeX) étaient les mêmes depuis les semaines qui ont précédées la mise en marche du projet ; ces éléments nous ont permis d'appréhender le travail de la même manière ont donc beaucoup apporté à la bonne conduite de notre entreprise.

Si nous étions enthousiastes avant tout à l'idée de travailler ensemble, nous avons rapidement compris que les libertés qui nous étaient données dès le début du projet mais aussi les conditions de développement du produit nous mettraient dans des conditions proches de celles du milieu professionnel. Nous avons donc décidé de penser notre produit comme un bien que nous pourrions mettre sur le marché, du moins mettre à la disposition du public. C'est dans cet esprit de "challenge" que nous avons été amené à développer le programme de commande sur un support Android et que nous avons porté un grand soin à l'architecture du pied pour, en lui apportant les modifications qui pourraient s'imposer mais qui dépassent nos moyens, éventuellement le proposer à une entreprise et lancer sa fabrication.

Notre prototype est cependant loin d'être parfait mais sommes certains que quelques simples évolutions le rendraient sérieux. Parmi celles-ci, améliorer la connexion en Bluetooth pour rendre son fonctionnement plus stable et changer la solution technique adoptée pour la réalisation de la liaison pivot entre le plateau et le boitier sont les plus pertinentes à court terme.