Aplicación Android para el control Bluetooth de un sistema domótico

Efrén García Buitrago
Javier Santos Paniego
Francisco Buitrago Pavón

ÍNDICE

1. INTRODUCCIÓN

En este proyecto que está enclavado en el ámbito de la domótica y para ello vamos a aclarar una serie de conceptos claves antes de empezar a explicar el contenido de dicho trabajo.

- ***Domótica*:** es el conjunto de sistemas capaces de automatizar una vivienda, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. Se podría definir como la *integración de la tecnología en el diseño inteligente de un recinto cerrado*.

- ***Arduino:*** es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

- ***Android:*** es un sistema operativo basado en el núcleo Linux diseñado originalmente para dispositivos móviles.

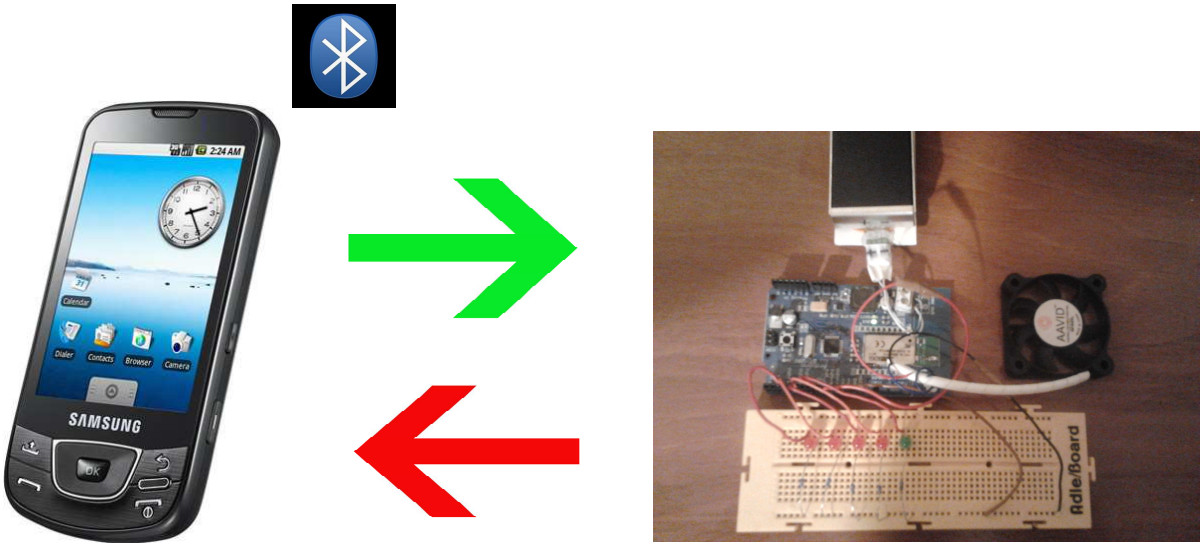
El proyecto consiste en un sistema que está compuesto por dos placas una board y una de Arduino. La placa board es la que se encarga de simular el hogar y tiene cuatro leds y un ventilador. El módulo de Arduino es el encargado de mandar las señales para que los leds y el ventilador se enciendan simulando las luces y el aire acondicionado de una casa.

El programa que hemos realizado parece simple pero tiene una cierta dificultad a la hora de programar con las clases Bluetooth de las librerías de Java para entenderlas.

Su funcionamiento es el siguiente, este programa al ejecutarse nos mostrará un botón para escanear los dispositivos Bluetooth que se encuentren activos. Si el bluetooth del dispositivo Android tuviera el Bluetooth apagado pediría encenderlo en la configuración. Una vez hecho esto escanearía los dispositivos y si encuentra la mac del BT2 (Bluetooth placa de Arduino), entonces mostrará mediante un mensaje en la pantalla la mac y se enlazará el dispositivo móvil con la placa de Arduino. Seguidamente cargará una imagen de una casa con sus bombillas, aire acondicionado y botón de encendido/apagado. Estos botones cambiarán de color según estén encendido o apagadas los elementos en la realidad, es decir tocando cada imagen podremos apagar y encender cada elemento de la placa board conectada a la placa Arduino. El botón de encendido/apagado nos servirá para activar todo a la vez o apagar todo a la vez.

Para que al tocar y apagar o encender los elementos, veamos como cambia la imagen en nuestro programa de Android, lo que hacemos es recoger los comandos que devuelve la placa de Arduino y así saber el estado de cada elemento. También al volver arrancar el programa si se dejó algún elemento encendido pide un estado para saber como tiene los elementos nada más arrancar el programa.

Durante la explicación de las partes más importantes del código explicaremos todo más detalladamente.



2. HITOS DE LA PRÁCTICA

Hito 1) *Lee atentamente toda la documentación de este guión y realiza la instalación descrita en el apartado 2.*
Hito 2) *Realizar los ejemplos descritos en el apartado 2.*
Hito 3) *Modificar el ejemplo del apartado 2 para que realice un bucle de 50 iteracciones.*
Hito 4) *Realizar un programa que se comunique con la placa de las prácticas y le envíe @11 (encender led 1).*
Hito 5) *Realizar un programa para la comunicación con la placa de Arduino, o la placa domótica bluetooth, que integre un protocolo para enviar una serie de comandos para controlar el encendido y apagado de los led's de la placa.*

Hito 5.1) *El programa deberá enviar @11 y al cabo de 5 segundos @10, al cabo de 5 segundos @21 y al cabo de 5 segundos @20.*

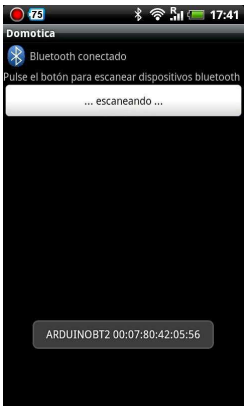
Hito 5.2) *El programa dará la posibilidad al usuario de introducir el comando a enviar en una caja de texto. Además, existirá un botón que cuando se pulse enviará el contenido de la caja de texto.*

Hito 5.3 – optativo) *Incluir iconos en el interfaz que simbolicen encendido/apagado de bombillas. Cuando se pulse uno de los iconos de cada bombilla se enviará automáticamente el comando.*

Cada led simboliza lo siguiente:

- *Led1: luz del patio.*
- *Led2: luz del salón.*
- *Led3: luz del dormitorio.*
- *Led4: luz del aseo.*

Aquí podemos observar una muestra del interfaz del programa en Android.



Un ejemplo del protocolo está explicado en los anexos del guión de la práctica 2-B.
Hito 6 - optativo) *Si en la placa se ha incorporado un sensor de temperatura, incluir en el interfaz el envío de un comando que ordene el envío de la temperatura desde la placa Arduino al móvil. Una vez recibido aparecerá en una segunda caja de texto este valor.*

Nosotros en vez de poner un sensor de temperatura hemos añadido un ventilador simulando un aire acondicionado de tal forma que al pulsar en el icono del aire acondicionado este puede encenderse o apagarse según lo deseemos. También hemos añadido un botón para apagar el sistema domótico completo y dejar la casa sin luz y con el aire acondicionado apagado, ya que esto te facilita la tarea del apagado para no tener que ir apagando las estancias de la casa una a una.

3. DESARROLLO

Después de leer los documentos que se nos facilitan para esta Práctica procedemos a la instalación de Eclipse, ya que el desarrollo en Android se hace mediante Java y Xml. Una vez instalado Eclipse procedemos a bajarnos de la página de desarrolladores de Android el módulo Sdk necesario para las librerías que utiliza Android y descargaremos todo lo necesario para las diferentes versiones de Android. Por último se configura un emulador de Android en Eclipse para poder probar nuestro programa al compilarlo si todo es correcto. Si el programa es correcto se exporta el .apk (instalador de Android) con firma para así mandárnoslo al dispositivo móvil y ejecutarlo.

Nota: A veces dependiendo de la velocidad del pc/emulador es más aconsejable después de compilarlo directamente exportar el apk siempre que creamos que ninguna función del programa pueda afectar a la integridad del dispositivo móvil.

4. ESTRUCTURA DEL CÓDIGO - ANDROID

Como ya hemos explicado antes la estructura para un desarrollo de un programa de Android consta de programación en java, por lo que tendremos paquetes con archivos .java.

Archivos JAVA

- Main.java
- BluetoothConnection.java

En el siguiente apartado detallaremos el código.

Por otro lado necesitaremos un sitio donde guardar imágenes, sonidos, vídeos, todo el material que el programa vaya a utilizar desde el mismo sin necesitar Internet para descargarlo, para todo ello se suele utilizar la carpeta src, usualmente las imágenes suelen ir dentro de res/drawable, se pueden crear otras carpetas con el nombre que queramos y meter sonidos, vídeos etc…

Después necesitamos la interfaz gráfica de nuestro programa para ello dentro de la carpeta res/Layout tendremos nuestro archivo XML el cual podremos mediante ventanas añadirle elementos como botones, imágenes, y luego en el código del xml tendremos su identificador y demás opciones de los elementos.

Layout

- Nuestra aplicación dispondrá de **un principal y único** Layout en el cual aparecerá un botón **Escanear**. Una vez pulsado este botón y después de encontrar la mac del BT2 (placa de Arduino Bluetooth) y enlazarse con ella se harán visibles el resto de elementos del Layout que ya hemos descrito en la introducción.

(Aquí 2 imágenes, una de antes de encontrar el BT2, y otra después de encontrarlo con la casita luces y demás)

Por último el archivo **Manifest.xml**, este archivo es un archivo fundamental en un programa para Android ya que lleva la configuración principal, donde diremos los

- Permisos que tiene el programa:

```
    <uses-permission
android:name="android.permission.BLUETOOTH"></uses-
permission>
    <uses-permission
android:name="android.permission.BLUETOOTH_ADMIN"></uses-
permission>
```

- El sdk que tiene (versión desde la cual se puede utilizar en un dispositivo Android, en este caso la 10 se referiría a Android 2.3.

```
    <uses-sdk android:minSdkVersion="10" />
```

- Nombre aplicación, icono, versión de programa, etc...

5. CÓDIGO DEL PROGRAMA - ANDROID

En este apartado vamos a ir viendo nuestro código explicado por partes, funciones….

Primero vemos que tenemos bien diferenciados **MAIN** de la clase **BluetoothConnection**. Hemos hecho esto para implementar todas las clases relacionadas con el Bluetooh en la clase **BluetoothConnection** y así la clase **MAIN** quede mucho más clara solo llamando alas funciones que necesita de la clase **BluetoothConnection**.

5.1. BluetoothConnection

// Paquete y librerías

```
package com.domotica;

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

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
```

// Aquí comenzaría nuestra clase BluetoothConnection tenemos unas constantes para cada error que se pueda producir.

```
public class BluetoothConnection {

	public static final int BLUETOOTH_NOT_AVAILABLE = 1;
	public static final int BLUETOOTH_NOT_CONNECTED = 2;
	public static final int BLUETOOTH_SOCKET_CREATION_ERROR = 3;
	public static final int BLUETOOTH_CONNECTION_ERROR = 4;
	public static final int BLUETOOTH_CREATION_SUCCESS = 5;
	public static final int BLUETOOTH_OUTPUTSTREAM_FAIL = 6;
	public static final int BLUETOOTH_SEND_FAIL = 7;
	public static final int BLUETOOTH_SEND_SUCCESS = 8;
	public static final String BLUETOOTH_INPUTSTREAM_FAIL=
		"Error_DataInputStream";
	public static final String BLUETOOTH_RECIEVE_FAIL=
		"Error_Recieve";
```

// Éstas variables son para la creación del objeto device que será el dispositivo que unamos, el socket para poder abrir la conexión/enlace y los objetos de entrada y salida que serán los caracteres de los comandos que irán a través de un buffer.

```
	private BluetoothAdapter mBtAdapter =
		BluetoothAdapter.getDefaultAdapter();
	private String mac;
	private BluetoothDevice device;
	private BluetoothSocket socket_device;
	private DataOutputStream tmpOut;
```

```
private DataInputStream tmpIn;

public BluetoothConnection(String mac){
      this.mac=mac;
}
```

// En esta función lo que se hace es primero inicializar el objeto Bluetooth Adapter es nuestro manejador del Bluetooth, comprobamos si tenemos el Bluetooth activado o no, una vez hecho esto adjudicamos la mac que ya tenemos guardada de nuestro BT2 (placa de Arduino Bluetooth) a la variable socket_device para así crear el socket con el dispositivo BT2. Por último al objeto socket_device ya hecho para la mac de BT2 le creamos la conexión es decir enlazarlo a nuestro dispositivo Android. En todo lo dicho siempre comprobamos si lo hace correcto o no, retornando en cada caso un valor que podremos utilizar para mostrar errores.

```
public Integer doConnect() {
      //INICIALIZACION
      mBtAdapter = BluetoothAdapter.getDefaultAdapter();
      if (mBtAdapter == null) {
            return BLUETOOTH_NOT_AVAILABLE;
            //Devuelve FALSO en caso de no estar disponible el
      Bluetooth
      }
      if (!mBtAdapter.isEnabled()) {
            return BLUETOOTH_NOT_CONNECTED;
      }
      //OBTENIENDO DISPOSITIVO
      this.device=mBtAdapter.getRemoteDevice(this.mac);

      //CREANDO SOCKET
      try {
            this.socket_device =
            this.device.createRfcommSocketToServiceRecord(UUID.fr
      omString("00001101-0000-1000-8000-00805F9B34FB"));
      } catch (IOException e) {
            return BLUETOOTH_SOCKET_CREATION_ERROR;
      }

      //CREANDO CONEXION
      try {
            this.socket_device.connect();
      } catch (IOException e) {
            try {
                  this.socket_device.close();
            } catch (IOException e2) {
                  return BLUETOOTH_CONNECTION_ERROR;
            }
      }
      return BLUETOOTH_CREATION_SUCCESS;
}
```

// Esta función se refiere dentro del canal de la comunicación entre los 2 dispositivos, al envío de datos. Primero tenemos que crearnos un buffer y un objeto:

```
tmpOut = new DataOutputStream(this.socket_device.getOutputStream());
```

que servirá para escribir los datos que queramos enviar al dispositivo. Es decir luego cuando tengamos el objeto creado, en el buffer le daremos el valor de cada mensaje en cada momento (comandos que le mandaremos al BT2) y para así luego mandar lo que hay en ese buffer con la función write del objeto que hemos creado en este caso tmpOut.

```
public Integer sendInfo(String message) {
	byte[] buffer;
	try {
		tmpOut = new DataOutputStream
			(this.socket_device.getOutputStream());
	} catch (IOException e) {
		return BLUETOOTH_OUTPUTSTREAM_FAIL;
	}
	try {
		buffer = (message + '\r').getBytes();
		tmpOut.write(buffer);
		tmpOut.flush();
	} catch (IOException e) {
		return BLUETOOTH_SEND_FAIL;
	}
	return BLUETOOTH_SEND_SUCCESS;
}
```

// Esta función se refiere dentro del canal de la comunicación entre los 2 dispositivos, al recibimiento de datos, viene a ser muy parecida a la anterior. Primero tenemos que crearnos un objeto

```
tmpIn = new DataInputStream(this.socket_device.getInputStream());
```

En este caso el nombre ya indica que el objeto es para la entrada de datos (Input). Una vez hecho esto haremos un bucle para ir leyendo byte por byte cada carácter que se manda con la función readByte del objeto que hemos creado tmpIn hasta encontrarnos con el carácter especial '#'. Aquí es donde primero nos llegará un carácter para que guardemos el número de caracteres del comando, para luego al leer del buffer podamos especificarle cuantos caracteres tendrá que leer y así tener el comando exacto que se nos envió.

```
public String receiveInfo(){
	try {
		tmpIn = new DataInputStream
			(this.socket_device.getInputStream());
	} catch (IOException e) {
		return "BLUETOOTH_INPUTSTREAM_FAIL";
	}
	try {
		char readChar=(char)tmpIn.readByte();
		while(readChar!='#'){
			readChar=(char)tmpIn.readByte();
		}
		Integer cBytes=
			Integer.parseInt("" + (char)tmpIn.readByte());
		byte buffer[] = new byte[cBytes];
		tmpIn.read(buffer,0,cBytes);
		return new String(buffer);
	} catch (IOException e) {
		return "BLUETOOTH_RECIEVE_FAIL";
	}
}
```

// Aquí simplemente es una función que dependiendo el entero que le pasemos (nuestras constantes para errores) devolverá un string con el nombre del error.

```
	public static String showResult(Integer result) {
		switch (result){
			case BLUETOOTH_NOT_AVAILABLE:{
				return ("BLUETOOTH_NOT_AVAILABLE");}
			case BLUETOOTH_NOT_CONNECTED:{
				return ("BLUETOOTH_NOT_CONNECTED");}
			case BLUETOOTH_SOCKET_CREATION_ERROR:{
				return ("BLUETOOTH_SOCKET_CREATION_ERROR");}
			case BLUETOOTH_CONNECTION_ERROR:{
				return ("BLUETOOTH_CONNECTION_ERROR");}
			case BLUETOOTH_CREATION_SUCCESS:{
				return ("BLUETOOTH_CREATION_SUCCESS");}
			case BLUETOOTH_OUTPUTSTREAM_FAIL:{
				return ("BLUETOOTH_OUTPUTSTREAM_FAIL");}
			case BLUETOOTH_SEND_FAIL:{
				return ("BLUETOOTH_SEND_FAIL");}
			case BLUETOOTH_SEND_SUCCESS:{
				return ("BLUETOOTH_SEND_SUCCESS");}
			default: {return ("Not understood!");}
		}
	}
}
```

5.2. Main

// Paquete, liberías y variables

```
package com.domotica;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

public class Main extends Activity implements OnClickListener{
      private static final int REQUEST_ENABLE_BT = 0;

      private Button button_scaner;
      private String statusBotonTodo="ENCENDIDO";
      private ImageView botonTodo;
      private ImageView bulb1Image;
      private ImageView bulb2Image;
      private ImageView bulb3Image;
      private ImageView bulb4Image;
      private ImageView bulb5Image;
      private ImageView fanImage;

      private ImageView bluetoothIndicator;
      private TextView bluetoothStatus;
      private View arduinoLayout;

      private BluetoothAdapter mBtAdapter;
      private BluetoothConnection btC;

      private String macArduinoBT2="";
      private int[] boardStatus = {0,0,0,0,0,0};

      private final BroadcastReceiver mReceiver =
            new BroadcastReceiver() {
```

// Aquí es donde detectaremos el evento para encontrar el dispositivo y comprobará el nombre y si coincide con ARDUINOBT2 cogeremos su mac para luego mostrarla por pantalla. Después intentaremos conectarnos al dispositivo encontrado y si es posible ya pasaremos a hacer visible en el Layout los elementos que referencian a la placa board, es decir las imágenes del salón con las bombillas, aire acondicionado, etc… Seguidamente mandaremos un comando para preguntar por el estado de los elementos de la placa y así saber como debemos de tener las imágenes.

```
        @Override
        public void onReceive(Context arg0, Intent arg1) {
          String action = arg1.getAction();
          if (BluetoothDevice.ACTION_FOUND.equals(action)) {
              //Detecta evento de descubrimiento de dispositivo.
```

```
                BluetoothDevice device = arg1.getParcelableExtra
                    (BluetoothDevice.EXTRA_DEVICE);
                if (device.getName().contains("ARDUINOBT2")==true)
                    macArduinoBT2=device.getAddress();
                Toast.makeText(getApplicationContext(),
                    device.getName() + " " + device.getAddress(),
                    Toast.LENGTH_LONG).show();
            }else if
(BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
                //Detecta evento de fin de búsqueda de dispositivos.
                Toast toast =
                    Toast.makeText(getApplicationContext(),
                    "Fin de búsqueda", Toast.LENGTH_SHORT);
                toast.show();

button_scaner.setText(getResources().getString(R.string.scan));
                    if (macArduinoBT2!="")
                      if (tryConnectArduino()==true){
                          arduinoLayout.setVisibility(0);
                          //Hacemos visible el layout del comando.
                          String status= sendCommand("@ST");
                          //preguntamos por el estado de los elementos de
                    la placa
                          for (int i = 0;
                                i<status.substring(1).length();
                                i++){
                                boardStatus[i] = Character.digit
                                      (status.substring(1).charAt(i),
                                10);
                          }
                          updateImages();
                    }
            }
          }
        };
```

// Los objetos referidos a imágenes les asociamos a la variable el identificador y la acción de clickar sobre ellos. Cambiamos la imagen del bluetooth para hacer referencia a que estamos conectados.

```
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT)
;
        setContentView(R.layout.main);

        this.botonTodo = (ImageView)findViewById(R.id.botonTodo);
        this.botonTodo.setOnClickListener(this);
        this.bulb1Image = (ImageView)findViewById(R.id.bulb1);
        this.bulb1Image.setOnClickListener(this);
        this.bulb2Image = (ImageView)findViewById(R.id.bulb2);
        this.bulb2Image.setOnClickListener(this);
        this.bulb3Image = (ImageView)findViewById(R.id.bulb3);
        this.bulb3Image.setOnClickListener(this);
        this.bulb4Image = (ImageView)findViewById(R.id.bulb4);
        this.bulb4Image.setOnClickListener(this);
        this.bulb5Image = (ImageView)findViewById(R.id.bulb5);
        this.bulb5Image.setOnClickListener(this);
        this.fanImage = (ImageView)findViewById(R.id.fan);
        this.fanImage.setOnClickListener(this);
```

```
        this.button_scaner = (Button)findViewById(R.id.button1);
        this.button_scaner.setOnClickListener(this);

        this.bluetoothIndicator =
(ImageView)findViewById(R.id.btIndicator);
        this.bluetoothStatus =
(TextView)findViewById(R.id.connectedStatus);

        this.arduinoLayout= (View)findViewById(R.id.linearLayout2);

        //Comprueba la conectividad del bluetooth para actualizar el
icono.
      mBtAdapter = BluetoothAdapter.getDefaultAdapter();
        if ((mBtAdapter == null) || (!mBtAdapter.isEnabled())){

      bluetoothIndicator.setImageDrawable(getResources().getDrawable(R
.drawable.bluetooth_disc));

      bluetoothStatus.setText(getResources().getString(R.string.bt_dis
connected));
        }

    }
```

// Esta es la función para los eventos de click, (táctiles en este caso). Entonces según que botón/imagen toque en la pantalla realizará un case u otro. El elemento más especial aquí sería el button1 que llama ala función startScanner que es la función que realizamos al principio para encontrar nuestro dispositivo bluetooth. Lo demás serán botones para encender y apagar elementos que mandarán un comando para activar o desactivar el elemento de la placa board. A la vez actualizaremos la imagen de nuestro Layout.

```
      @Override
      public void onClick(View v) {
            // TODO Auto-generated method stub
            String response="";
            switch(v.getId()){
                  case (R.id.button1):
                        startScanner();
                        break;
                  case(R.id.botonTodo):
                        if (statusBotonTodo=="ENCENDIDO"){
                              response=sendCommand("@P1");
                              if (response.compareTo("#OKPON")==0)
                                    for (int i =
0;i<boardStatus.length;i++)
                                          boardStatus[i]=1;

      this.botonTodo.setImageDrawable(getResources().getDrawable(R.dra
wable.apagado));
                              this.statusBotonTodo="APAGADO";
                        }else{
                              response=sendCommand("@P0");
                              if (response.compareTo("#OKPOFF")==0)
                                    for (int i =
0;i<boardStatus.length;i++)
                                          boardStatus[i]=0;

      this.botonTodo.setImageDrawable(getResources().getDrawable(R.dra
wable.encendido));
                              this.statusBotonTodo="ENCENDIDO";
```

```
                }
                break;
            case(R.id.bulb1):
                if (boardStatus[0]==0){
                    response=sendCommand("@11");
                    {if (response.compareTo("#OK1ON")==0)
                        boardStatus[0]=1;}}
                else{
                    response=sendCommand("@10");
                    {if (response.compareTo("#OK1OFF")==0)
                        boardStatus[0]=0;}}
                break;
            case(R.id.bulb2):
                if (boardStatus[1]==0){
                    response=sendCommand("@21");
                    {if (response.compareTo("#OK2ON")==0)
                        boardStatus[1]=1;}}
                else{
                    response=sendCommand("@20");
                    {if (response.compareTo("#OK2OFF")==0)
                        boardStatus[1]=0;}}
                break;
            case(R.id.bulb3):
                if (boardStatus[2]==0){
                    response=sendCommand("@31");
                    {if (response.compareTo("#OK3ON")==0)
                        boardStatus[2]=1;}}
                else{
                    response=sendCommand("@30");
                    {if (response.compareTo("#OK3OFF")==0)
                        boardStatus[2]=0;}}
                break;
            case(R.id.bulb4):
                if (boardStatus[3]==0){
                    response=sendCommand("@41");
                    {if (response.compareTo("#OK4ON")==0)
                        boardStatus[3]=1;}}
                else{
                    response=sendCommand("@40");
                    {if (response.compareTo("#OK4OFF")==0)
                        boardStatus[3]=0;}}
                break;
            case(R.id.bulb5):
                if (boardStatus[4]==0){
                    response=sendCommand("@51");
                    {if (response.compareTo("#OK5ON")==0)
                        boardStatus[4]=1;}}
                else{
                    response=sendCommand("@50");
                    {if (response.compareTo("#OK5OFF")==0)
                        boardStatus[4]=0;}}
                break;
            case(R.id.fan):
                if (boardStatus[5]==0){
                    response=sendCommand("@F1");
                    {if (response.compareTo("#OKFON")==0)
                        boardStatus[5]=1;}}
                else{
                    response=sendCommand("@F0");
                    {if (response.compareTo("#OKFOFF")==0)
                        boardStatus[5]=0;}}
                break;
        }
        updateImages();
    }
```

// Función ESCANER, esta función es la que utiliza la mayoría de las funciones que hemos explicado en la clase BluetoothConnection. Primero nos creamos un objeto BluetoothAdapter para crearnos el manejador de Bluetooth. Y con el comprobamos si no existe Bluetooth en nuestro dispositivo Android o está apagado, y en ese caso lo activamos. Una vez activado cambiaremos la imagen del Layout para mostrar que estamos conectados, y el texto del botón de Escanear cambiará a Escaneando para ir realizando las siguientes acciones:

Se lanza el evento para poder descubrir el dispositivo BT2
- Registramos el evento en el objeto BroadcastReceiver.
Se lanza el evento para indicar la finalización del descubrimiento de dispositivos.

- *Registramos el evento en el objeto BroadcastReceiver.*

Por últimos creamos un objeto BluetoothConnection con la mac del BT2. Y realizamos la conexión con la función doConnect.

```
      public void startScanner(){
            mBtAdapter = BluetoothAdapter.getDefaultAdapter();
//Inicializa manejador Bluetooth. Android no nos da un manejador
        if (mBtAdapter == null) {    //Entra si el dispositivo
Bluetooth no está disponible o si no hay dispositivo Bluetooth en
nuestro terminal
            Toast.makeText(this, "Bluetooth no está disponible",
Toast.LENGTH_LONG).show();
            finish();
            return;
        }
        if (!mBtAdapter.isEnabled()) {    //Entra si no es capaz de
habilitar el manejador. El blueetooth está apagado
            Intent enableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);    //Lanza un Intent
de solicitud de activación del dispositivo Bluetooth.
            startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
//Recoge el resultado en REQUEST_ENABLE_BT previamente declarada por
nosotros.
        }

        while(!mBtAdapter.isEnabled()){}


bluetoothIndicator.setImageDrawable(getResources().getDrawable(R.drawa
ble.bluetooth));

bluetoothStatus.setText(getResources().getString(R.string.bt_connected
));


button_scaner.setText(getResources().getString(R.string.scanning));
        mBtAdapter.startDiscovery();     //Empieza el escaner de
dispositivos bluetooth.
        IntentFilter filter = new
IntentFilter(BluetoothDevice.ACTION_FOUND);  //Evento descubrimiento
dispositivo
        this.registerReceiver(mReceiver, filter);    //Registra evento
en BroadcastReceiver.
        filter = new
IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);    //Evento
fin de búsqueda de dispositivo.
        this.registerReceiver(mReceiver, filter);    //Registra evento
en BroadcastReceiver.
      }
```

```
      public boolean tryConnectArduino() {
            btC=new BluetoothConnection(macArduinoBT2);
      if
(btC.doConnect()==BluetoothConnection.BLUETOOTH_CREATION_SUCCESS)
                  return true;
      else
            return false;
      }
```

// Función para enviar comandos y recibirlos a través de las funciones sendInfo y receibeInfo

```
    private String sendCommand(String command){
      btC.sendInfo(command);
      return (btC.receiveInfo());
    }
```

// Esta función será la que utilizamos para actualizar las imágenes del Layout dependiendo del estado que tengan, que siempre guardaremos cada vez que hagamos un evento OnClick tocando algún botón de la pantalla.

```
    private void updateImages(){
      if (boardStatus[0]==0)

      this.bulb1Image.setImageDrawable(getResources().getDrawable(R.dr
awable.b1off));
      else

      this.bulb1Image.setImageDrawable(getResources().getDrawable(R.dr
awable.b1on));
      if (boardStatus[1]==0)

      this.bulb2Image.setImageDrawable(getResources().getDrawable(R.dr
awable.b2off));
      else

      this.bulb2Image.setImageDrawable(getResources().getDrawable(R.dr
awable.b2on));
      if (boardStatus[2]==0)

      this.bulb3Image.setImageDrawable(getResources().getDrawable(R.dr
awable.b3off));
      else

      this.bulb3Image.setImageDrawable(getResources().getDrawable(R.dr
awable.b3on));
      if (boardStatus[3]==0)

      this.bulb4Image.setImageDrawable(getResources().getDrawable(R.dr
awable.b4off));
      else

      this.bulb4Image.setImageDrawable(getResources().getDrawable(R.dr
awable.b4on));
      if (boardStatus[4]==0)

      this.bulb5Image.setImageDrawable(getResources().getDrawable(R.dr
awable.b5off));
      else

      this.bulb5Image.setImageDrawable(getResources().getDrawable(R.dr
awable.b5on));
      if (boardStatus[5]==0)
```

```
      this.fanImage.setImageDrawable(getResources().getDrawable(R.draw
able.fan_off));
      else

      this.fanImage.setImageDrawable(getResources().getDrawable(R.draw
able.fan_on));
    }

    /*
    private void show(String message){
      Toast.makeText(this, message, Toast.LENGTH_LONG).show();
    }
    */
}
```

6. CONCLUSIONES

Para terminar con la memoria de la práctica en este apartado veremos las conclusiones que hemos sacado tras realizar el trabajo y de comprarlo con su realización en lenguaje ensamblador.

En primer lugar hay que destacar que el uso de Arduino facilita mucho la labor a la hora de programar la placa, en clase hemos visto un ejemplo de cómo sería la misma practica pero en lenguaje ensamblador y cada instrucción en Arduino supone tres o cuatro instrucciones en ensamblador, hay que destacar también que para comparar valores en Arduino puedes hacer un '= =' como en cualquier lenguaje de programación y la misma operación en ensamblador consiste en hacer una resta y comparar su resultado si es igual a cero.

Respecto a la programación en Android gracias a la documentación existente y al ejemplo de comunicación Bluethooth hemos podido desarrollar la práctica sin que nos costara demasiado tan solo hemos tenido algunos problemillas al recibir los asentimientos por partes de la placa de Arduino, una vez solventado estos problemas podemos decir que realizar una aplicación en Android no es nada complicado y queremos agradecer haber podido trabajar con tecnologías punteras como son Android y Arduino.

7. ROLES DE LA PRÁCTICA

Los roles que se han desempeñado en esta práctica son, diseño interfaz, diseño funcional, implementación del código y memoria.

- Diseño interfaz: Este rol ha sido el encargado de hacer el diseño del interfaz del programa.
    - Javier 25%
    - Efrén 50%
    - Fran 25%
- Diseño funcional: Encargado del diseño de las funciones y estructura del código para la implementación.
    - Javier 25%
    - Efrén 25%
    - Fran 50%
- Implementación de código: Encargado de implementar el código del programa en Android y sus pruebas.
    - Javier 50%
    - Efrén 25%
    - Fran 25%
- Memoria: rol encargado de realizar la memoria de la práctica.
    - Javier 33%
    - Efrén 33%
    - Fran 33%

8. BIBLIOGRAFÍA

Para la ayuda en el desarrollo de esta práctica hemos utilizado el material adicional de la práctica, video tutoriales y la página de desarrolladores de Android.

http://developer.android.com/guide/topics/wireless/bluetooth.html
http://es.wikipedia.org/wiki/Dom%C3%B3tica
http://es.wikipedia.org/wiki/Arduino