Departament d'Informàtica de Sistemes i Computadors

Universitat Politècnica de València

WirelessRemote
Mando universal Wi-Fi

Trabajo Fin de Máster

Máster Universitario en en Ingeniería de Computadores

Autor: Daniel Aparicio Martín

Tutor: Manuel Agustí i Melchor

Cotutor: José Oliver Gil

2014-2015

Indice de contenido

Resumen

La mayoría de los programas informáticos tienen atajos de teclado o “hotkeys” para realizar acciones predeterminadas, ahorrando así al usuario tener que usar el ratón para buscar el botón o tener que navegar por el menú que permitiría ejecutar esa acción.

Nuestro mando universal se aprovecha de estos atajos de teclado que tienen los programas informáticos para controlarlos de forma remota.

Para ello basta con ejecutar el programa servidor en la máquina que se quiere controlar remotamente y este será capaz de “pulsar” los atajos de teclado que sean indicados por el cliente.

Resum

La majoria dels programes informàtics tenen atall de teclat o “hotkeys” per realitzar accions predeterminades, estalviant així a l'usuari haver d'usar el ratolí per cercar el botó o haver de navegar pel menú que permetria executar eixa acció.

El nostre comandament universal s'aprofita dels atalls del teclat que tenen els programes informàtics per controlar-los de forma remota.

Per a això basta amb executar el programa servidor en la màquina que es vol controlar remotament i aquest serà capaç de “polsar” els atalls del teclat que siguen indicats pel client.

Abstract

Most computer programs have keyboard shortcuts or “hotkeys” to perform predetermined actions, thus saving the user from having to use the mouse to find the button or having to navigate through the menu that would carry out this action.

Our universal remote takes advantage of these keyboard shortcuts to remotely control computer programs.

To do this you simply have to run the server program on the machine to be controlled remotely and this will be able to "press" the keyboard shortcuts that are indicated by the user.

1. Introducción

1.1. Resumen de las ideas clave

En este Trabajo de Fin de Máster se pretende desarrollar un control remoto universal y totalmente personalizable, para controlar cualquier software de Windows o Linux, mediante la "pulsación" remota del teclado.

La máquina en la que se arranque el servidor deberá estar conectada vía Wifi o Ethernet a la misma red LAN que lo está el cliente vía Wifi.

La parte del servidor, programado en Java, se encargará de "pulsar" las teclas solicitadas por la aplicación. También ejecuta algunas órdenes predefinidas de la línea de órdenes.

1.2. Motivación

La idea de desarrollar WirelessRemote surgió inicialmente como una idea para cubrir una necesidad personal, ya que aún existiendo aplicaciones muy similares, la idea era tener una interfaz tan fácil de usar como de personalizar.

Sin embargo, no se había desarrollado por falta de tiempo, hasta que surgió la oportunidad de comenzar su desarrollo en la asignatura de Fundamentos de Sistemas Multimedia, siendo aquí donde se presentó una primera versión.

Una vez finalizada esta versión aún necesitaba pulir varios detalles, por lo que se decidió realizar el TFM sobre esta mejorando la versión inicial.

1.3. Objetivos

- Comunicación cliente-servidor vía wifi entre distintas plataformas: Cliente (Android) y Servidor (Java).
- Pulsación remota de teclas ("hotkeys") que ejecutan alguna función en programas determinados ej: pausa en Spotify = 'espacio', pantalla completa en VLC = 'f'...
- Ejecución de órdenes remotas de la línea de órdenes de Windows y Linux, como abrir otros programas (ej: Spotify) o apagar el ordenador.
- Botones en la parte del cliente totalmente editables: tanto los "hotkeys" que envía cada botón, como la imagen que muestran los botones en el mando.
- Uso del acelerómetro para enviar órdenes al servidor según la posición.
- Interfaz de usuario con número limitado de botones y fácil de usar.

1.4. Estructura de la memoria

Capítulo 2: Estado del arte

Expone diferentes aplicaciones y dispositivos existentes con funcionalidad similar a la de WirelessRemote.

Capítulo 3: Funcionamiento de WirelessRemote

Explica el funcionamiento del cliente y del servidor y se mostrará un diagrama de su funcionamiento.

Capítulo 4: Desarrollo

Profundiza en los detalles del desarrollo de aplicación y servidor.

Capítulo 5: Resultados

Muestra los resultados obtenidos, los dispositivos de pruebas empleados y los problemas que aparecieron durante el desarrollo del proyecto.

Capítulo 6: Conclusiones y trabajos futuros

Concluye con las lecciones aprendidas durante el desarrollo de este proyecto y las posibles mejoras para este en futuras versiones.

2. Estado del arte

Como soluciones similares a WirelessRemote podemos fijarnos en todo tipo de aplicaciones para dispositivos móviles cuya finalidad es controlar el ordenador, así como dispositivos físicos con el mismo fin.

2.1. Aplicaciones y dispositivos existentes

2.1.1 Remote Control Collection

Remote Control Collection es una aplicación para controlar el PC de forma remota y sin cables.

El cliente está disponible para Android, iOS y BlackBerry. El servidor, sin embargo, actualmente sólo está disponible para Windows.

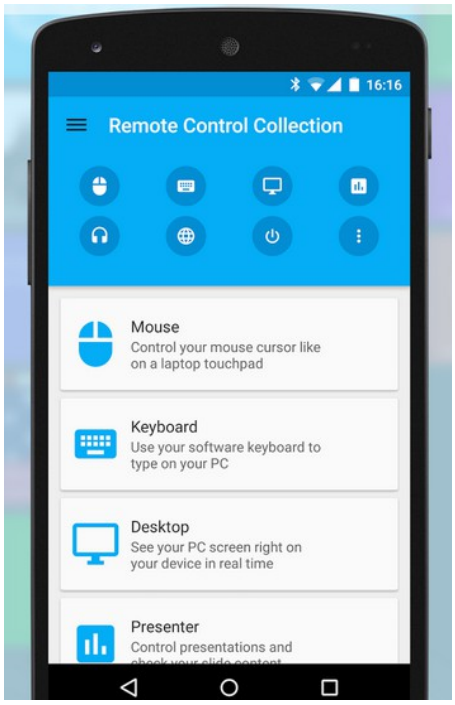


Figura 1. Remote Control Collection App

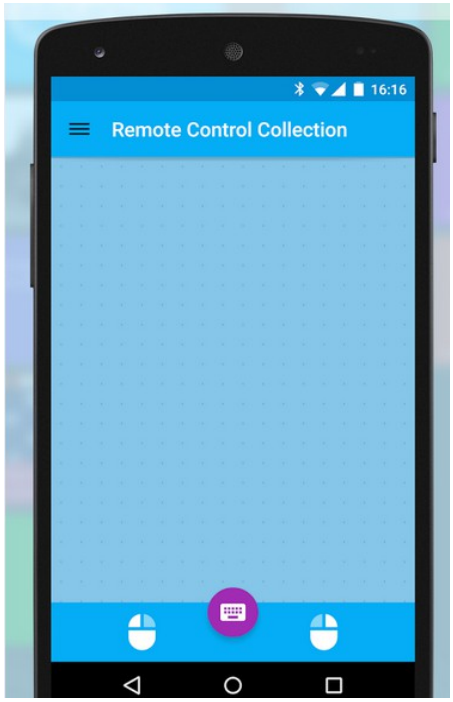


Figura 2. Ratón

Dentro de la aplicación pueden elegirse diferentes modos para controlar el PC:

- Ratón: Controla el ratón usando el teléfono como un touchpad
- Teclado: Utiliza el teclado del teléfono para escribir en el PC.
- Pantalla en vivo (Pro): Obtiene una vista en tiempo real del PC en la pantalla del teléfono
- Control de reproductor multimedia (Pro)
- Diapositivas (Pro): Controla presentaciones de diapositivas
- Reconocimiento de voz: Transcribe en el PC lo que se dicte en el teléfono
- Control vía Android Wear

Las opciones que se mencionan como Pro, son de la versión de pago, por lo que la opción gratuita no incluiría control de reproductor multimedia, pantalla en vivo ni paso de diapositivas.

El método de conexión es similar al de nuestra aplicación, el smartphone/tablet y el PC deben estar en la misma red , de lo contrario los dispositivos no pueden comunicarse entre sí.

2.1.2 Remote Mouse

Remote Mouse es una aplicación con cliente para iOS, Windows Phone y Android para el control remoto de PCs.

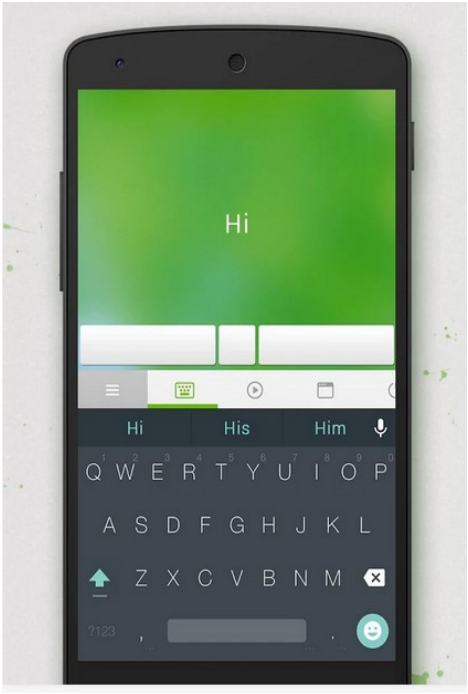


Figura 3. Teclado y ratón

Figura 4. Control multimedia

Con esta aplicación se pueden controlar tanto dispositivos con Windows como MacOS, siempre y cuando el servidor se encuentre en funcionamiento.

Remote Mouse ofrece los siguientes controles:

-Teclado y ratón inalámbricos

-Admite gestos táctiles

-Control de reproductores multimedia

El método de conexión entre cliente y servidor en este caso no tiene porque estar conectado ala misma red, sino que realiza su comunicación a través de internet, por lo que el cliente también podrá funcionar a través de 3G. Sin embargo en la primera conexión ambos dispositivos deben estar conectados a la misma red Wifi.

2.1.3 Unified Remote

Esta es una de las aplicaciones más completas en lo que a control remoto se refiere, además es la que funciona con más plataformas, ya sea el cliente (Android, iOS y Windows Phone) como el servidor (Windows, Linux, Mac OS, Raspberry Pi y Arduino Yún).

La comunicación cliente-servidor puede ser vía Wifi, Bluetooth, Infrarrojos, Internet y NFC.

Existen 2 versiones de esta aplicación, la versión gratuita y la de pago. La versión gratuita disfruta de 18 controles remotos diferentes, añadiendo en la de pago más de 40 controles nuevos.

Existe una opción de crear mandos personalizables, pero sólo está disponible en la versión de pago.

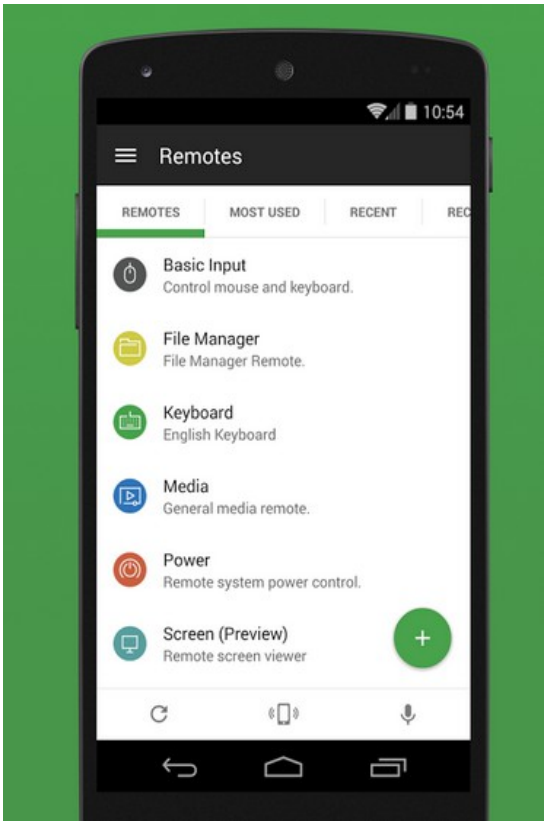


Figura 5. Controles de Unified Remote

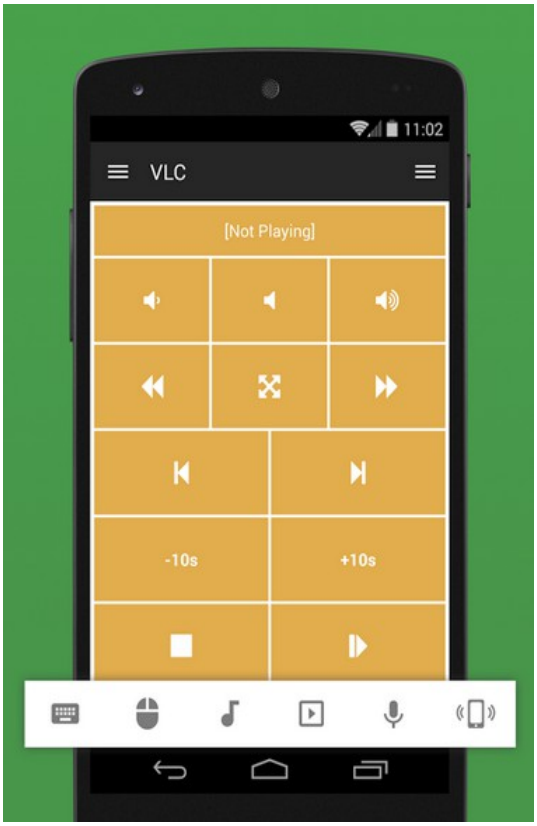


Figura 6. Control remoto para VLC

Figura 7. Teclado y ratón

Cabe destacar que también dispone de control por voz, pantalla en vivo, encendido y apagado remoto, control vía Android Wear... Siendo con todo esto la más completa y también la más compleja.

2.1.4 Wifi Remote

Se trata de una app para Android de control remoto inalámbrico para reproductores multimedia en Windows.

Figura 8. Wifi Remote App

Figura 9. Reproductores de Wifi Remote

Es necesario instalar el servidor remoto el PC, así como conectar cliente y servidor a la misma red para poder música y vídeos. Además ofrece una opción para encender y apagar la pantalla del ordenador y otra para silenciar la el reproductor si se recibe alguna llamada.

Los reproductores multimedia que controla por completo esta aplicación son limitados, mientras que puede controlar otras muchas de forma parcial.

2.1.5 Chrome Remote Desktop

Es una herramienta de escritorio remoto desarrollado por Google que permite a un usuario controlar de forma remota otro equipo mediante un protocolo Chromoting desarrollado por Google. Transmite los eventos de teclado y ratón de un ordenador a otro, transmitiendo las actualizaciones de pantalla gráfica de vuelta en la otra dirección, a través de una red.

Requiere el uso de Google Chrome, junto con la instalación de una extensión de Chrome Web Store, lo que funcionaría como servidor.

El cliente puede ser el propio navegador de Chrome, como la versión para Android o Iphone, funcionando todas como una vista real del dispositivo que se está controlando.

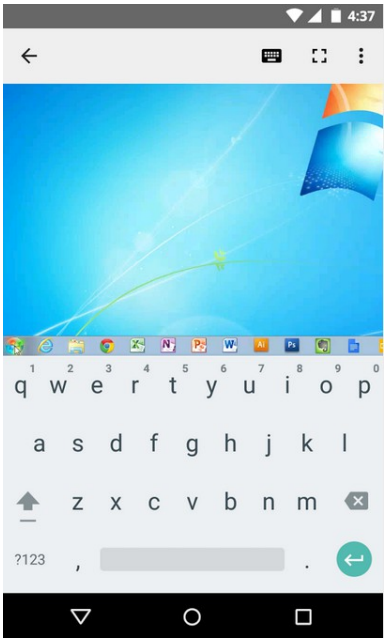


Figura 10. Vista de la App Chrome Remote Desktop

2.1.6 TeamViewer

Es una aplicación de control de escritorio remoto para Windows, Linux y Mac. El cliente de control está disponible para Windows, Linux, Mac OS, iOS, Windows Phone, Android y Blackberry.

Está disponible en su versión gratuita para usuarios privados, mientras que para empresas sólo está disponible en su versión de pago.

La funcionalidad es similar a Chrome Remote Desktop, ofrece una vista en vivo del dispositivo que se está conectando, pudiendo controlar teclado y ratón y además ofrece videoconferencia para la asistencia remota.

Figura 11. Vista de la App Team Viewer

2.1.7 Control Remoto Hardware

Existen muchos modelos diferentes de mandos a distancia físicos, haciendo el mismo uso que una app de control remoto, sin embargo el cliente en este caso es un dispositivo físico que se comunica con el servidor, en su mayoría, por infrarrojos o por bluetooth. Los hay en un amplio abanico de precios.

Figura 12. Distintas gamas de controles remoto

Algunos se ciñen al uso de los botones, mientras que otros añaden un touchpad para también controlar el ratón, como el de NEC que podemos ver en la figura 12.

Figura 13. Control remoto NEC

Este tipo de dispositivos están limitados físicamente, ya que no se pueden añadir ni quitar botones, sin embargo en muchos modelos si que se puede editar la funcionalidad de los mismos.

Además tienen el hándicap de que el teléfono es algo que siempre se tiene cerca, por lo que resulta más fácil controlarlo desde una aplicación que tener el mando físico al alcance en todo momento.

2.1.8 Flic, el botón inalámbrico

Flic es un dispositivo que va conectado a un dispositivo móvil a través de red WiFi o Bluetooth. Su alcance, de hasta 150 pies de distancia en caso de no estar conectado por bluetooth. Tiene una autonomía de 5 años, resistencia al polvo y agua y es configurable con hasta 3 funciones distintas según pulsemos una vez, dos veces o mantengamos pulsado.

Figura 14. Flic

Este botón no es directamente un mando para controlar el ordenador, pero empleando un smartphone como puente de comunicación entre botón y ordenador puede convertirse en un control inalámbrico para el mismo.

2.2. Comparación con trabajos previos

Observando las aplicaciones ya existentes WirelessRemote se ha desarrollado para ser más sencilla y minimalista que el resto.

Servidor: WirelessRemote tiene un servidor desarrollado en Java, lo que ha supuesto que el desarrollo para Windows y Linux haya ido prácticamente paralelo en lo que a código se refiere. Al estar desarrollado en Java supondría una baja carga de trabajo el hecho de portarlo a otros sistemas operativos.

Como comunicación se ha seleccionado Wifi o Ethernet en la misma red LAN. Esto es debido a que se pretende que funcione como mando a distancia y fijandonos en los dispositivos que puede llegar a controlar hay un alto porcentaje de ellos que puede que no tengan conexión Bluetooth (ordenadores de sobremesa, Raspberri Pi...).

Para el control se ha considerado que la forma más rápida para un usuario de controlar algo es haciendo click en un botón y que se ejecute la acción que desea, se ha considerado que tener que utilizar un teclado o ratón desde el cliente Android podría ser demasiado pesado para el usuario.

Considerando todo esto la aplicación UnifiedRemote sigue siendo similar a WirelessRemote, pero más completa, sin embargo, está cargada de opciones y demasiados detalles editables lo que podría sobrecargar al usuario, mientras que WirelessRemote ofrece una interfaz de usuario totalmente minimalista y fácil de editar, es muy “user-friendly”.

3. Funcionamiento de WirelessRemote

WirelessRemote está basado en el trabajo de NextSlide [7], podemos ver en la Figura 15 el diagrama de funcionamiento del servidor y el del cliente en la Figura 16.

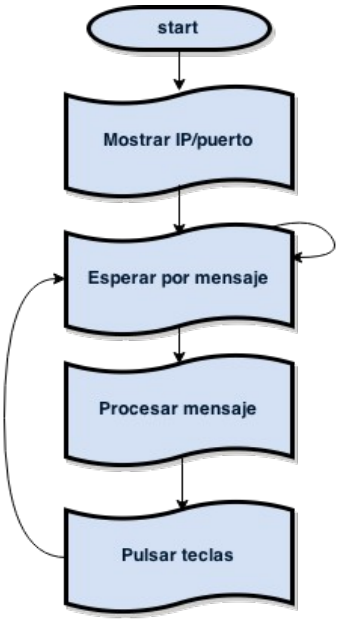


Figura 15. Diagrama servidor

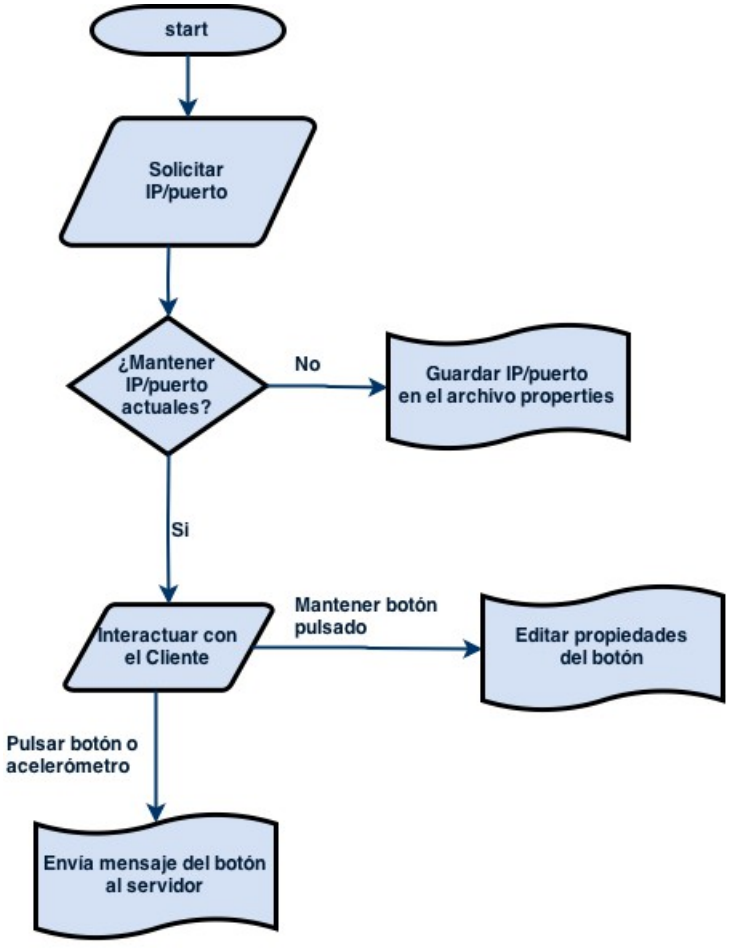
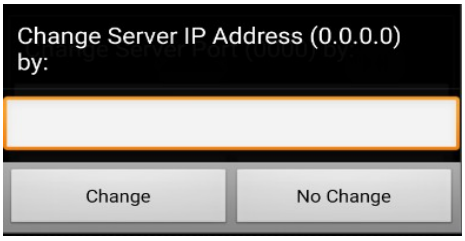
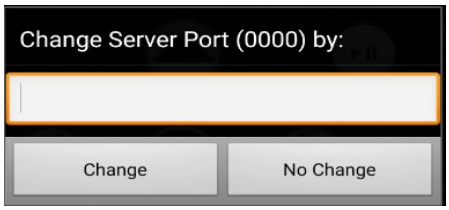


Figura 16. Diagrama cliente

La aplicación cliente tiene que estar ejecutándose en primer plano, mientras que el servidor controlará el programa que se encuentre en primer plano en la máquina en la que se esté ejecutando.

Al inicio de la aplicación hay que indicarle la IP y el puerto del servidor (Figuras 17 y 18) al cliente para que se puedan comunicar, los diálogos para indicar estos datos aparecen de manera automática al ejecutar la aplicación y se puede seleccionar un nuevo valor o que se mantenga el valor que tiene almacenado. IP y puerto se obtienen cuando arranca el servidor (Figura 19), mostrando la IP de la red local a la que se encuentra conectado y el puerto en el que escucha.





Figuras 17 y 18. Captura de los diálogos de cambio de IP y puerto.

Figura 19. Captura de ejemplo de ejecución del servidor.

Figura 20. Pantalla principal

Tras introducir IP y puerto la aplicación pasará a la pantalla principal, en la que muestra 12 botones con valores predefinidos (Figura 20), con los que se podrá interactuar para comunicarse con el servidor (véase 4.2) o para editar (véase 4.4) su funcionalidad y apariencia (Figura 22).

En el lado del servidor una consola mostrará los mensajes recibidos por los botones pulsados (Figura 21).

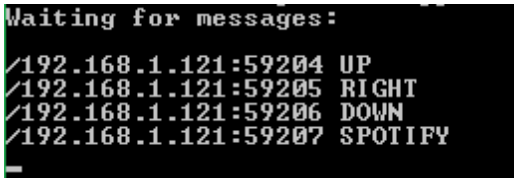


Figura 21. Recepción de mensajes en el servidor

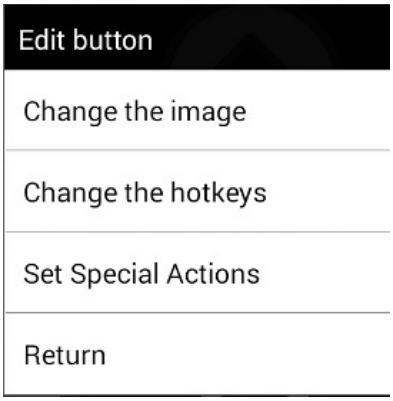


Figura 22. Diálogo para edición de los botones

Otro modo de comunicarse entre cliente-servidor, además de la pulsación de botones, es el uso del acelerómetro. La configuración del acelerómetro se puede modificar a través del menú (Figura 23), desde el que también se pueden ocultar/mostrar los mensajes que enviará cada botón al servidor (véase 4.6).

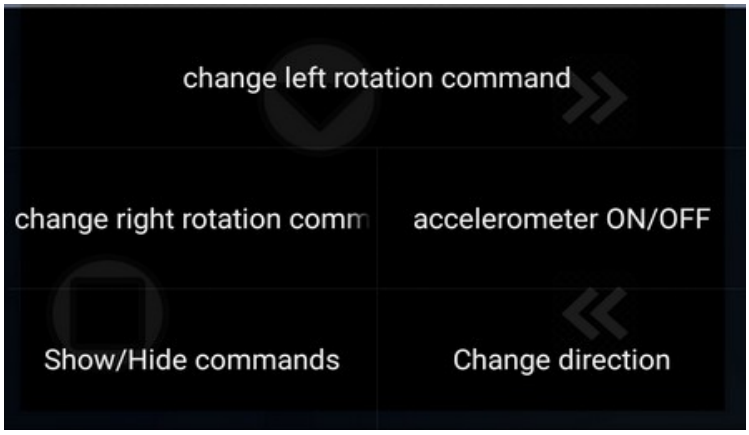


Figura 23. Menú de configuración de las acciones del acelerómetro

La aplicación viene preparada para enviar hotkeys a Spotify, Foxit Reader (visor de PDF) y presentaciones de diapositivas con libreOffice/openOffice. Podemos ver un ejemplo de las "hotkeys" que WirelessRemote tiene predefinidas en la Tabla 1.

Acción	Tecla
Play/pause	Space
Siguiente canción	Ctrl-Right
Canción anterior	Ctrl-Left
Subir volumen	Ctrl-Up
Bajar volumen	Ctrl-Down
Siguiente página/diapositiva	Right
Anterior página/diapositiva	Left
Ver presentación de diapositivas	F5
Pantalla completa	F11
Salir de presentación/pantalla completa	Esc

Tabla 1. Hotkeys.

La rotación del acelerómetro en el eje viene preparada con las órdenes para subir volumen (CTRL+UP) y bajar volumen (CTRL+DOWN) de Spotify (véase apartado 4.5).

Las funciones especiales, las cuales no son editables, ofrecen la posibilidad de apagar el ordenador/cerrar sesión y la de abrir el programa Spotify (véase 4.7).

4. Desarrollo

Según Plumbr [10] las versiones de Java más utilizadas son la 1.6 y 1.7 (Figura 24), por lo que el servidor ha sido compilado para la versión 1.6 y comprobado su compatibilidad en máquinas con la 1.7 y 1.8.

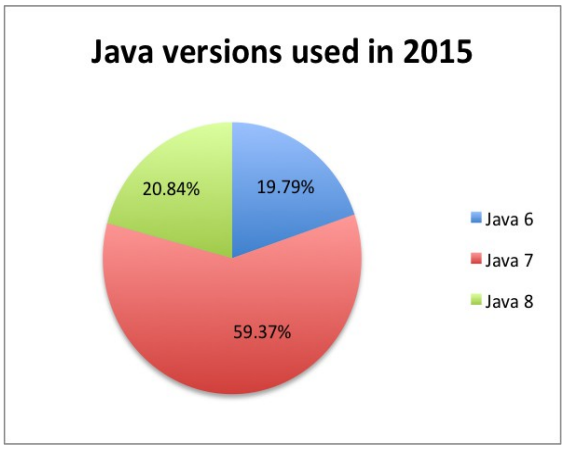


Figura 24. Distribuciones de Java 2015

Tanto cliente como servidor han sido desarrollados y compilados en el entorno Eclipse Juno. Eclipse permite cambiar en las propiedades del proyecto la versión de Java para la que se quiere compilar (Figura 25), por lo que el servidor de este proyecto podría fácilmente compilarse para otras versiones de Java.

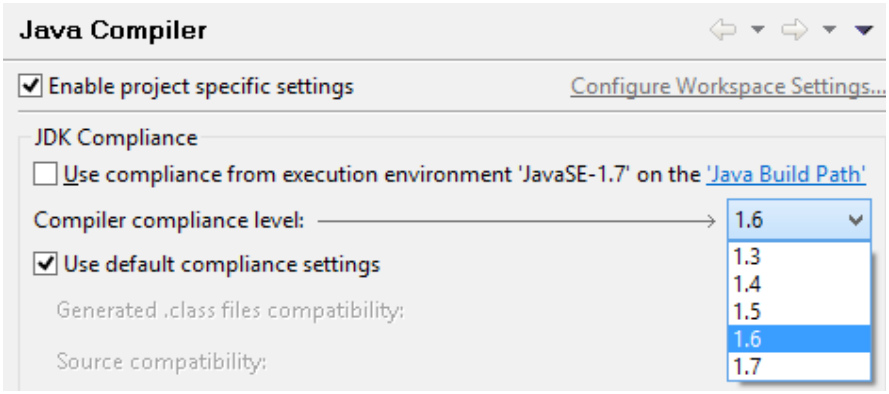


Figura 25. Selección de “target” de compilación.

Para el cliente Android ha sido seleccionado el nivel de API 8 como nivel mínimo, por lo que la aplicación sólo funcionará en versiones de Android igual o superior a la 2.2 Froyo [8].Se ha seleccionado este nivel de API porque según AndroidPolice [9], la versión más baja de Android que utilizan los usuarios es la 2.2 (Figura 26).

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.7%
4.1.x	Jelly Bean	16	16.5%
4.2.x		17	18.6%
4.3		18	5.6%
4.4	KitKat	19	41.4%
5.0	Lollipop	21	5.0%
5.1		22	0.4%

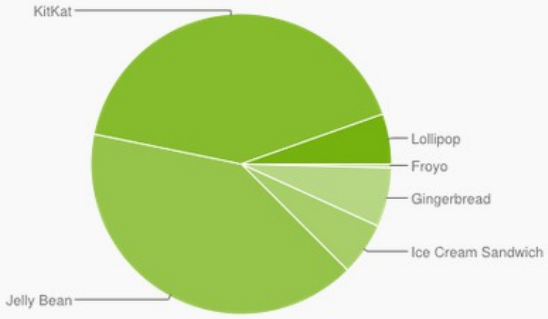


Figura 26. Distribuciones de Android por AndroidPolice

4.1. UML

Para una mejor comprensión del trabajo se muestran los diagramas UML de la aplicación cliente (Figura 27) y del servidor(Figura28), donde se podrán visualizar las relaciones entre las clases que involucran el sistema, así como los métodos de las mismas. Estos diagramas han sido generados con la herramienta The ObjectAid UML Explorer for Eclipse [13].

-**CLIENTE**:

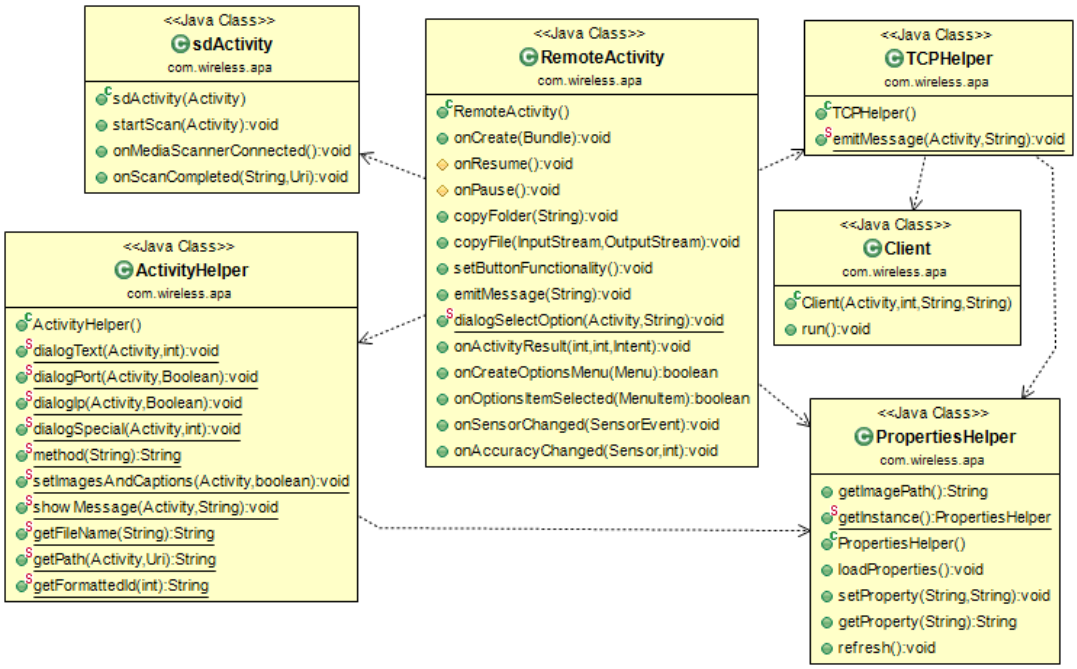


Figura 27. Diagrama UML del cliente

-**SERVIDOR**:

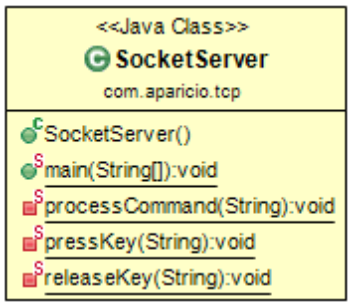


Figura 28. Diagrama UML del servidor

4.2. Comunicación

En una versión inicial de esta aplicación se utilizaba el protocolo UDP para la comunicación entre cliente-servidor, la conexión UDP facilita la conexión inicial, ya que se puede enviar un mensaje de Broadcast por toda la red local para que llegue a todos los dispositivos que se encuentren en la misma con tan solo indicar el puerto de escucha.

Sin embargo la implementación final se ha realizado con una conexión mediante sockets TCP, lo que supone indicar tanto el puerto de escucha del servidor, como su IP.

¿Por qué se ha optado finalmente por la conexión TCP?

El protocolo TCP (Figura 29) está orientado a conexión. Cuando una máquina A envía datos a una máquina B, la máquina B es informada de la llegada de datos, y confirma su buena recepción. A su llegada se comprueba la integridad de los datos transmitidos. De este modo, si los datos recibidos son corruptos, el protocolo TCP permite que los destinatarios soliciten al emisor que vuelvan a enviar los datos corruptos.

Mientras que UDP (Figura 30) es un protocolo no orientado a conexión. Es decir, cuando una máquina A envía paquetes a una máquina B, el flujo es unidireccional. La transferencia de datos es realizada sin haber realizado previamente una conexión con la máquina de destino. El destinatario recibirá los datos sin enviar una confirmación automática al emisor.

```
client = new Socket();
InetSocketAddress Address = new InetSocketAddress(server_IP,port);
client.connect(Address, 1000);
printwriter = new PrintWriter(client.getOutputStream(), true);
printwriter.write(msg);
```

Figura 29. Comunicación TCP

```
client = new DatagramSocket();
DatagramPacket packet = null;
packet = new DatagramPacket(msg.getBytes(), msg.length(), BAddress, port);
client.send(packet);
```

Figura 30. Comunicación UDP con Broadcast

Realizando pruebas con ambos tipos de conexión se opta finalmente por el protocolo TCP ya que en lugares con gran cantidad de redes Inalámbricas (como la UPV), provocan interferencias en la comunicación, lo que provoca que con el protocolo UDP se perdían 5 de cada 10 mensajes, también en pruebas realizadas en lugares con menor cantidad colisiones de redes Inálambricas se perdía 1 de cada 10 mensajes, mientras que con TCP los mensajes llegan siempre a su destino.

Para las pruebas mencionadas se ha realizado el envío de 100 mensajes obteniendo los resultados mostrados en la Figura 31.

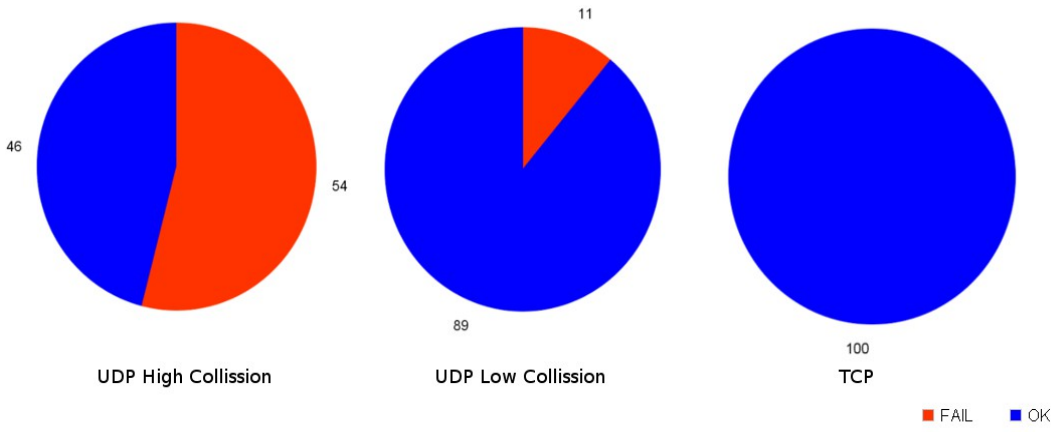


Figura 31. prueba de pérdida de mensajes

Estas pruebas han sido realizadas con los propios cliente y servidor de WirelessRemote de manera manual.

Cabe destacar que en las pruebas realizadas el tiempo de espera para TCP que ha sido empleado es de infinito, mientras que en WirelessRemote este valor es bastante bajo.

Esto es debido a que se pretende replicar el modo en que actua un mando a distancia, por ejemplo, si se pulsa un botón del mando a distancia de la televisión y esta no recibe la señal por alguna razón, lo más lógico es que el usuario vuelva a pulsar ese botón, pero su intención no es que la señal llegue dos veces a la televisión, sino que la televisión ejecute la señal que se le ha enviado al instante.

Por ello el tiempo de espera ha sido reducido a 1 segundo, siendo ese tiempo el límite para realizar retransmisiones.

4.3. Mensaje

La pulsación de un botón en la parte cliente supone el envío de un mensaje a la parte del servidor, este mensaje puede ser el identificador de una o varias teclas seguidos y separados por comas (Ej: CTRL,ALT,SUPR) o un identificador de acción especial (Ej: SPOTIFY).

4.4. Botones editables

La información que se carga en cada botón se almacena en memoria, pudiendo así guardar los cambios si se modifica el valor de los botones y volver a cargarlos al abrir la aplicación de nuevo (Figura 32).

```
public final static String TEXT_VIEW_ID_ROOT = "TextView";
public final static String IMAGEBUTTON_ID_ROOT = "ImageButton";
public final static String DOT_TITLE = ".title";
public final static String DOT_SRC = ".src";
public static void setImagesAndCaptions(Activity a) {

    TextView selText = null;
    ImageButton selImageButton = null;

    for (int i=1; i<=PropertiesHelper.getInstance().getMAX_NUMBER_OF_ELEMENTS(); i++) {
        String finalSufix =  getFormattedId(i);

        try {
            selText = (TextView) a.findViewById(R.id.class.
                    getDeclaredField(TEXT_VIEW_ID_ROOT+finalSufix).getInt(null));
            selText.setText(PropertiesHelper.getInstance().
                    getProperty(IMAGEBUTTON_ID_ROOT+finalSufix+DOT_TITLE));
            selImageButton = (ImageButton) a.findViewById(R.id.class.
                    getDeclaredField(IMAGEBUTTON_ID_ROOT+finalSufix).getInt(null));
            String imgUrl = PropertiesHelper.getInstance().
                    getProperty(IMAGEBUTTON_ID_ROOT+finalSufix+DOT_SRC);
            selImageButton.setImageDrawable(Drawable.
                    createFromPath(PropertiesHelper.getInstance().getImagePath() + imgUrl));

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Figura 32. Código de carga los valores del archivo “properties” a los botones

La funcionalidad cada botón se basa en la información guardada en el archivo “properties”, en este se determina la imagen que mostrará en ImageButton y el mensaje que enviará.

El tema de la edición de mensajes se ha realizado con un alertbox con checkboxes para seleccionar los hotkeys que enviará cada botón, mientras que la edición de imágenes ha sido un tema más complicado donde se han encontrado algunas dificultades.

Para seleccionar las imágenes se ha usado un `Intent.ACTION_PICK` (Figura 33) permitiendo así seleccionar cualquier imagen de la galería, por lo que se pueden añadir imágenes para los botones a gusto del usuario.

```
Intent intent = new Intent(
        Intent.ACTION_PICK,
        android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
intent.setType("image/*");
finalActivity.startActivityForResult(intent, id);
break;
```

Figura 33. código de selección de imágenes

En la llamada a startActivityForResult() se almacena en el archivo properties la imagen seleccionada para el botón editado y se carga para que sea mostrada en el mando.

Las imágenes predefinidas del mando se encuentran en la carpeta WirelessRemote, la cual se crea en la memoria del dispositivo Android la primera vez que es lanzada la aplicación, guardando también en esta el archivo properties.

Para que se mostrasen en la galería y así poder seleccionar las imágenes predefinidas de la aplicación, se ha utilizado una clase que implementa `MediaScannerConnectionClient` (Figura 34), que sirve para escanear la memoria en busca de nuevas imágenes sin tener que reiniciar el dispositivo.

```
class sdActivity implements MediaScannerConnectionClient {

        sdActivity()
        {
                StartScan(WirelessRemote);
        }

        StartScan(folder)
        {
                Create MediaScanner on folder
        }

        onMediaScannerConnected(folder)
        {
                Scann all files on folder
        }
}
```

Figura 34. Pseudocódigo de escaneo de imágenes

4.5. Acelerómetro

La clase para la ventana principal de la aplicación implementará SensorEventListener (Figura 35) para el manejo de las actualizaciones de los acelerómetros.

```
public class RemoteActivity extends Activity implements
        SensorEventListener {

    private SensorManager sensorManager;
    private Sensor accelerometer;


    public void onCreate(Bundle savedInstanceState) {

        sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);

        if (sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) != null) {
            // success! we have an accelerometer
            accelerometer = sensorManager
                    .getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
            sensorManager.registerListener(this, accelerometer,
                    SensorManager.SENSOR_DELAY_NORMAL);
        }

    }
}
```

Figura 35. código para el registro de eventos del acelerómetro

Al implementar esta interfaz debemos sobrecargar el método OnSensorChanged (Figura 36), que es donde manejamos la información obtenida del acelerómetro para enviar los mensajes según la orientación del giro del dispositivo.

```
public void onSensorChanged(SensorEvent event) {

        if ((event.values[1] > 2.5) && (difference > 2)) {
                emitMessage(left);
        } else if ((event.values[1] < -2.5) && (difference > 2)) {
                emitMessage(right);
        }
}
```

Figura 36. código para el manejo de eventos del acelerómetro

4.6. Menú

Desde el menú “settings” pueden modificarse varias opciones de la aplicación (Figura 23):

- Accelerometer ON / OFF:

  Permite activar o desactivar el envío de mensajes con el movimiento del acelerómetro.

- Change direction:

  Cambia la dirección de rotación del dispositivo con la que es activado el envío por acelerómetro.

- Change left rotation command:

  Permite cambiar el mensaje que envía con el giro a la izquierda por acelerómetro.

- Change right rotation command:

  Permite cambiar el mensaje que envía con el giro a la derecha por acelerómetro.

- Show/hide commands:

  Muestra/oculta los mensajes que envía cada botón.

4.7. Servidor

Las pruebas de funcionamiento del servidor se han realizado en Ubuntu y en dos versiones diferentes de Windows, 7 y 8, siendo el funcionamiento sobre Ubuntu y Windows 7 completamente funcional, mientras que en Windows 8 se ha encontrado un problema a la hora de simular pulsaciones de teclas para determinadas órdenes (apartado 5.1.3 Problemas con el envío de órdenes).

El servidor recibe el mensaje y lo procesa (Figura 37), esto significa que de la cadena de texto recibida se obtienen los identificadores de teclas/acción especial quitando las comas (como vimos en el apartado 4.3).

Después de obtener los identificadores se procederá a la pulsación virtual de las teclas o la ejecución de la acción especial correspondiente a los identificadores obtenidos, y una vez pulsadas todas se pasa a su liberación.

Para la pulsación y liberación de las teclas se hace uso de la clase Robot (Figuras 38 y 39) y para la ejecución de las acciones especiales se emplea la clase Runtime (Figura 40) .

```
private static void processCommand(String command) {
    try {
        int i;

        String[] parts = command.split(",");

        for(i= 0; i < parts.length; i++)
        {
            pressKey(parts[i]);
        }

        for(i= parts.length-1; i >= 0; i--)
        {
            releaseKey(parts[i]);
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figura 37. Simulación de pulsación simultánea de teclas en el servidor

```
if(key.equals("CTRL"))
{
     robot.keyPress(KeyEvent.VK_CONTROL);
}
```

Figura 38. Código para simulación de pulsación de la tecla CTRL

```
if(key.equals("CTRL"))
{
     robot.keyRelease(KeyEvent.VK_CONTROL);
}
```

Figura 39. Código para simulación de liberación de la tecla CTRL

```
else if(key.equals("SHUTDOWN"))
{
    try {
        proc = runtime.exec("shutdown -s -t 0");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figura 40. código para ejecutar la orden “shutdown” (Windows)

Con respecto a la ejecución de órdenes hay algunas pequeñas diferencias dependiendo del sistema operativo en el que se esté ejecutando el servidor.

Si el servidor se ejecuta en una máquina Linux el identificador SHUTDOWN hace un cierre de sesión en lugar de apagar la máquina, mientras que en una máquina Windows si que se apagará. Por otro lado en una máquina Windows habrá que indicar la ruta exacta de Spotify para poder ejecutarlo, mientras que en Linux basta con indicar el nombre del programa ejecutar, en este caso Spotify.

El servidor crea un ServerSocket indicando el puerto 0 (Figura 41), lo que supone que el sistema le asignará un puerto libre cualquiera.

```
serverSocket = new ServerSocket(0);
clientSocket = serverSocket.accept();
inputStreamReader = new InputStreamReader(clientSocket.getInputStream());
bufferedReader = new BufferedReader(inputStreamReader);
message = bufferedReader.readLine();
processCommand(message);
```

Figura 41. Creación de ServerSocket, aceptación de conexión y lectura de mensaje

Al iniciar el servidor, este nos mostrará la información mostrada en la Figura 42.

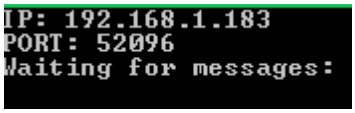


Figura 42 Captura de pantalla del servidor

Para que así podamos introducir la IP y puerto necesarios en el cliente para una conexión satisfactoria.

5. Resultados

El resultado cumple lo esperado, una interfaz sencilla, editable y funcional como podemos comprobar en la figura 43, en la que se está controlando el Spotify con la aplicación desarrollada.

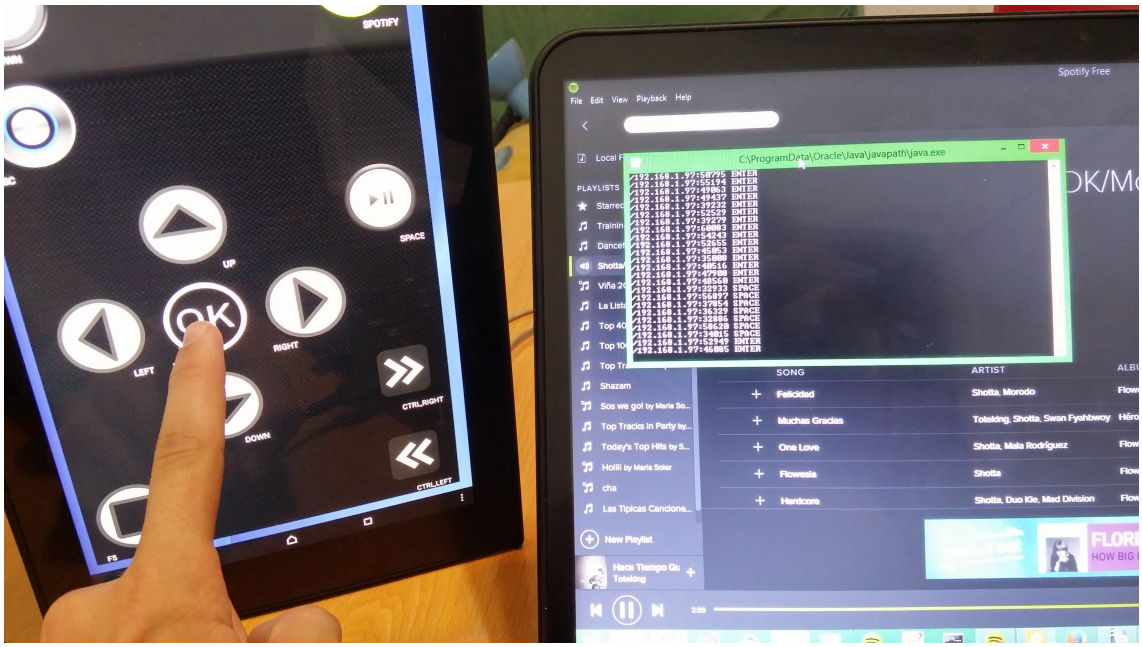


Figura 43. WirelessRemote controlando Spotify

Lo más característico de la aplicación frente a otras, es que no necesita comunicarse con la aplicación de escritorio que se desea controlar, sino que se aprovecha el uso de las hotkeys haciendola más universal.

Durante el desarrollo y las pruebas sobre los distintos dispositivos surgieron varios problemas, los cuales se comentan a continuación.

5.1. Problemas

5.1.1 Problemas de conexión

En una primera versión de la aplicación, cuando esta se iba a basar en una conexión UDP, se iba a utilizar el "Edimax EW-7811Un Wireless 802.11b/g/n nano USB Adapter" para hacer las pruebas de conexión, sin embargo en el momento de abrir el programa de configuración, obtenido para Windows 8 desde la propia web de Edimax[11], aparecía un error (Figura 44).

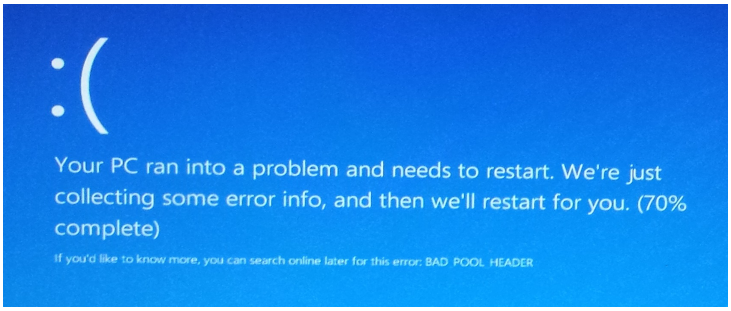


Figura 44. Captura de pantalla de Error de Windows 8

Esto en su momento supuso un problema, ya que en redes como la de la universidad no se permite el envío de mensajes Broadcast por UDP, para lo que necesitábamos este hardware

intermedio. A raíz de esto se dio el paso a mensajes UDP indicando la IP a la que iban dirigidos, sin embargo tras realizar pruebas con TCP y UDP (véase 4.2) finalmente se eligió el protocolo TCP.

5.1.2 IP y VirtualBox

El servidor obtiene la IP con el método getLocalHost de la clase InetAddress [12], la cual debería devolver la dirección IP real de la máquina en la que se está ejecutando el servidor.

Sin embargo, si VirtualBox está instalado en la máquina y el modo de red "host-only" se encuentra activo, el método getLocalHost nos devolverá la IP de esta red virtual en lugar de la IP real que de la máquina[1].

Se puede solucionar desactivando el modo de red "host-only" o comprobando la IP real de la máquina manualmente.

5.1.3 Probemas de ejecución de órdenes en Windows 8

La ejecución de órdenes compuestas con el uso de la tecla ALT (ej: ALT+TAB, CTRL+ALT+SUPR...) a través de la clase Robot no está permitido en Windows 8[2].

5.2. Dispositivos de pruebas

El servidor ha sido probado en los siguientes sistemas operativos con sus diferentes versiones de Java:

- Ubuntu 14.04 LTS
  java version "1.7.0_55"
  OpenJDK Runtime Environment (IcedTea 2.4.7) (7u55-2.4.7-1ubuntu1)
  OpenJDK 64-Bit Server VM (build 24.51-b03, mixed mode)

- Ubuntu 12.04.5 LTS
  java version "1.7.0_76"
  Java(TM) SE Runtime Environment (build 1.7.0_76-b13)
  Java HotSpot(TM) 64-Bit Server VM (build 24.76-b04, mixed mode)

- Ubuntu 12.04.2
  java version "1.6.0_35"
  OpenJDK Runtime Environment (IcedTea6 1.13.7) (6b35-1.13.7-1ubuntu0.12.04.2)
  OpenJDK 64-Bit Server VM (build 23.25-b01, mixed mode)

- Microsoft Windows 7 Enterprise 64 bits
  java "version 1.7.0_51"
  Java(TM) SE Runtime Environment (build 1.7.0_51_h13)
  Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)

- Microsoft Windows 8.1 pro 64 bits
  java version "1.8.0_25"

[1] Bug Conocido: http://stackoverflow.com/questions/8219664/java-gethostaddress-returning-virtualbox-ipv4-address

[2] Problema conocido: http://stackoverflow.com/questions/14549526/alttab-using-java-robot

Java(TM) SE Runtime Environment (build 1.8.0_25-b18)
Java HotSpot(TM) 64-Bit Server VM (build 25.25-b02, mixed mode)

La aplicación cliente ha sido probada en los siguientes dispositivos Android con sus diferentes versiones:

- Samsung Galaxy Note 10.1 GT-N8010
  Android 4.1.2

- LG G3, LG-D855
  Android 5.0

- LG Optimus Black, LG-P970
  Android 2.2.2

- Sony Xperia Tablet Z, SGP321
  Android 4.4.4

- Sony Xperia Tablet Z, SGP321
  Android 5.0.2

6. Conclusiones y trabajos futuros

Inicialmente para la aplicación WirelessRemote se había propuesto la opción de cambiar de perfiles para según que aplicación se quisiera controlar, sin embargo, en su lugar, se ha implementado un modo bastante amigable de editar los botones en cualquier momento.

Otra propuesta finalmente no incluida era la de encender el ordenador desde la aplicación, pero fue descartada debido a la necesidad de mantener el equipo conectado a la red mediante un cable de ethernet.

Como trabajo futuro se propone:

- comunicación bidireccional entre cliente-servidor
- mejorar el servidor con una mejor interfaz de usuario
- la creación de un protocolo para reconocer desde el cliente de manera automática si hay un servidor disponible en la misma red que este.
- uso de perfiles para un cambio más rápido de la configuración de los botones
- crear interfaz con menos botones para pantallas menores de 4 pulgadas
- compatible con android wear
- solucionar el problema con el envío de órdenes (tecla ALT) en Windows 8

Como conclusión de todo el trabajo realizado se podría decir que a veces menos es más, a ojos de un usuario muchas veces es mejor una interfaz sencilla, minimalista y fácil de usar que una interfaz demasiado complicada, a pesar de que esta ofrezca un mayor abanico de opciones.

Bibliografía

[1] MOOC gratuito “Android: Introducción a la Programación”

Disponible en: http://www.androidcurso.com/

[2] Java, Clase Robot

Disponible en: http://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html

[3] Java, Clase Runtime

Disponible en:http://docs.oracle.com/javase/7/docs/api/java/lang/Runtime.html

[4] Android Sockets

Disponible en: http://developer.android.com/reference/java/net/Socket.html

[5] Spotify Hotkeys

Disponible en: https://support.spotify.com/es/learn-more/faq/#!/article/Keyboard-shortcuts

[6] Obtención de la ruta de usuario específica de un archivo sin conocer el Username:

Disponible en: http://stackoverflow.com/questions/11361742/getting-a-user-specific-path-without-knowing-the-username

[7] NextSlide, de Albert Pardo Badia,

Control de diapositivas desde el Smartphone a través de bluetooth.

[8] Niveles de API

Disponible en: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels

[9] AndroidPolice. Distribuciones de Android 2015

Disponible en: http://www.androidpolice.com/2015/04/07/platform-distribution-numbers-updated-android-5-1-already-on-the-board-with-0-4-while-5-0-gains-1-7/

[10] Plumbr. Distribuciones de Java 2015

Disponiible en: https://plumbr.eu/blog/java/java-version-statistics-2015-edition

[11] Edimax, EW-7811Un

Disponible en
http://www.edimax.com/au/produce_detail.phppd_id=347&pl1_id=1&pl2_id=44#03

[12] Java, Getlocalhost()

Disponible en:
http://download.java.net/jdk7/archive/b123/docs/api/java/net/InetAddress.html#getLocalHost

[13] The ObjectAid UML Explorer for Eclipse

Disponible en: http://www.objectaid.com/