Segundo Desafío Tecnológico del DTSTC

# PROPUESTA DE PROYECTO



**Grupo I⁴**

Iván Fernández Bermejo

Iván López Espejo

Santiago Prieto Calero

# 1. Introducción

Este documento de propuesta de proyecto recoge la línea de trabajo a partir de su definición en pos de conseguir diseñar e implementar un buen sistema para la sintetización de partituras musicales para coro. Este problema recoge un gran espectro de soluciones, pudiéndose descomponer en la suma de elementos funcionales independientes, a su vez, con diferentes posibilidades. Por ello, no se ha fijado a priori una ruta de trabajo estricta y esta es flexible, comentándose en este documento diferentes técnicas con las que se experimentará a partir de entonces.

La secuencia básica y general que el sistema debe cumplir trata de la extracción y codificación de la información de la fuente (contemplándose que esta sea una partitura en formato imagen o un fichero en formato MIDI) para la posterior sintetización sonora. A continuación se explican someramente las diferentes metodologías con las que se pretende experimentar, siendo alguna de ellas complementaria o suplementaria, según lo que se decida a lo largo de la fase de trabajo en función del rendimiento.

# 2. Técnicas de Desarrollo

## 2.1 Extracción de la Información Musical

Comenzará desarrollándose el sistema para la extracción de la información a partir de ficheros en formato MIDI. Paralelamente, se trabajará en el diseño e implementación de un sistema OMR (Optical Music Recognition) para la lectura de partituras. En el caso de que este último sistema no satisfaga suficientemente nuestras necesidades, se partirá del empleo de música codificada según el estándar MIDI. Sin embargo, en caso satisfactorio, pueden mantenerse las dos vías de obtención de la información musical, decidiendo el usuario cuál emplear y haciendo así más versátil el sistema final.

## 2.2 Sintetización sin Letra

Trataremos de abordar la sintetización con la letra de la pieza en el caso de existir. Sin embargo, si la dificultad y, primordialmente, el tiempo no lo permitiesen, se propone emplear *wavetables* para, a partir de la información extraída de la fuente (cuestión trivial mediante el uso de MIDI (se propone hacer uso de un *toolbox* dedicado de MatLab)) según lo mencionado en el apartado anterior, generar las diferentes líneas melódicas del coro. *Wavetables*, dada la capacidad computacional existente, permite obtener un resultado fidedigno a partir de la concatenación de unidades básicas de voz real recogidas en una base de datos. Dicha base de datos sería una matriz de dos dimensiones, correspondiéndose una de ellas al timbre (posibilidad de registrar voces de diferentes cantantes (según el

sexo, la tesitura, etc)) y la otra al pitch, evitando así generar una excitación artificial (modulación constante de una muestra de pitch basal) que proporcionaría un peor resultado, ya que la respuesta del tracto vocal varía conjuntamente con la altura a la que se esté entonando.

### 2.3 Sintetización con Letra

También se abordará, como se ha mencionado, esta cuestión. Se estudiará la forma de obtención de la letra, aunque ya se piensa en diferentes soluciones. Numerosas partituras de coro incluyen la letra bajo las propias notas musicales (quedando así secuenciada y modelada su duración), por lo que en el caso de llevarse finalmente a cabo el sistema OMR de lectura de partituras, se propone que este, extensiblemente, se combine con un sistema OCR (Optical Character Recognition) para la detección de la letra. En el caso de emplear MIDI, se conoce de la existencia de ficheros *.kar* que codifican la secuenciación de la letra de la música, por lo que, en el caso de estar disponibles, esta es fácilmente extraíble de los mismos. Aunque estos son los dos mecanismos principales que se consideran a día de hoy, no se descarta el empleo de otros.

En esta tesitura, la sintetización de cada una de las líneas melódicas mediante *wavetables* no es a priori útil. Por ello, se proponen dos variantes que estudiar: síntesis LPC y HMMs (modelos ocultos de Márkov). Para el primero de los casos, consideremos que hemos extraído la información musical correspondiente necesaria (altura, duración e intensidad). Se trataría ahora de emplear el modelo LPC de producción de voz para generar cada una de las líneas melódicas con letra. En función del tiempo,

la señal excitación es variable a partir de la información de altura tonal extraída (pitch), así como variables son los coeficientes del filtro todo-polos LPC que modelan el tracto vocal en función de la letra y del tiempo, obteniendo a la salida voz sintética donde se ha modelado el par de características mencionadas. No obstante no se esperan grandes resultados con esta metodología, por lo que la otra propuesta es el uso de HMMs para la sintetización de voz. Es conocida ampliamente la aplicación de esta herramienta estadística al reconocimiento de voz, pero en los últimos tiempos también viene usándose con gran eficiencia y espectaculares resultados para la tarea inversa, es decir, sintetizar voz a partir de texto. En el caso de emplear esta última tecnología, el resultado, cuya duración ya debe haber quedado modelada, debería pasar por un bloque de modulación de pitch para incluir la información relativa a la altura tonal.

## 2.4 Modelado de la Dinámica, Sintetización y Obtención del Resultado Final

Sea cual sea el camino elegido en función de las anteriores propuestas u otras que pudieran surgir durante la realización de nuestro trabajo, el tramo final, una vez obtenida la señal *en bruto* que modela el pitch, la duración y la voz (timbre y quizás letra), consiste en el modelado de la dinámica (variaciones en la intensidad sonora) y en la sintetización de las diferentes líneas melódicas para obtener el resultado final de la coral.

Para la cuestión de la dinámica se pretende emplear la información extraída de la fuente en conjunción con la aplicación de un filtrado en frecuencia (ponderación de la forma de onda en el dominio del tiempo) para

el modelado típico de la dinámica de un sonido cantado, siguiendo posiblemente el esquema ADSR (Attack/Decay/Sustain/Release).

Finalmente se trata de aplicar, posiblemente, sintetización aditiva de cada una de las líneas melódicas que componen el coro y que ahora modelan las cuatro cualidades del sonido de la forma más fidedigna posible (timbre, altura, intensidad y duración) con el fin de obtener el resultado final. No obstante, se tratará de incluir información y tecnología psicoacústica en este proceso para obtener un resultado lo suficientemente bueno y realista, por lo que no se limitará a una simple adición de señales en el tiempo.

Una herramienta software usable es el objetivo final del proyecto que implemente los resultados de diseño e investigación propios de nuestro trabajo, de tal forma que el usuario proporcione un fichero MIDI o una partitura en formato imagen, y obtenga a la salida un fichero de audio con la pieza coral sintetizada de la forma más realista posible.

Segundo Desafío Tecnológico del DTSTC

# MEMORIA DE PROYECTO (v1.)



**Grupo I⁴**

Iván Fernández Bermejo

Iván López Espejo

Santiago Prieto Calero

## Sumario

1. Introducción ..... 3
2. Procesamiento MIDI y de audio ..... 4
   2.1 Lectura de archivo MIDI ..... 4
   2.2 Síntesis con wavetables ..... 5
   2.3 Mejorando la acústica ..... 8
3. OMR y síntesis de partituras ..... 10
   3.1 Preprocesamiento de imagen ..... 10
   3.2 Corrección de la inclinación ..... 11
   3.3 Segmentación de pentagramas ..... 12
   3.4 Segmentación de símbolos ..... 12
   3.5 Reconocimiento de símbolos ..... 13
   3.6 Detección de tono ..... 16
   3.7 Síntesis de partitura ..... 18
4. Interfaz de usuario ..... 19
5. Conclusiones y trabajo futuro ..... 23
6. Referencias ..... 25

# 1. Introducción

Tras haber superado la primera fase del II Desafío Tecnológico planteado por el Departamento de TSTC (Teoría de la Señal, Telemática y Comunicaciones) de la Universidad de Granada, se realiza para esta segunda fase un prototipo basado en las ideas presentadas en la mencionada primera fase.

Dicho prototipo se articula en torno a la lectura de un archivo MIDI y la síntesis de este mediante wavetables, obteniendo como resultado un archivo *wav* para su posterior reproducción. El prototipo también permite la lectura de imágenes en formato JPEG de partituras sencillas para su síntesis en términos análogos a los que serán descritos para el caso de poseer un fichero MIDI de partida.

También se ha incorporado en dicho prototipo una interfaz gráfica de usuario para hacer más cómodo e intuitivo el manejo de los algoritmos implementados y la modificación de algunos de sus parámetros, como pueden ser la velocidad de reproducción o el cambio de acústica del recinto, entre otros.

Sin más dilación pasamos a explicar el desarrollo del trabajo realizado durante esta segunda etapa del II Desafío Tecnológico del DTSTC.

# 2. Procesamiento MIDI y de audio

El procesamiento de archivos MIDI se realiza gracias a la herramienta MIDI Toolbox [1], que podemos encontrar para MatLab. Este paquete nos permitirá tanto leer los archivos MIDI como extraer la información contenida en dichos archivos. El procesamiento de audio es directamente llevado a cabo sobre MatLab. A continuación se desglosa el procedimiento.

## 2.1 Lectura de archivo MIDI

Mediante MIDI Toolbox se extrae la información del archivo MIDI. Dicha información viene ordenada en forma matricial, indicándonos cada una de sus filas una nota, siendo las columnas:

*(1) - note start in beats*

*(2) - note duration in beats*

*(3) - channel*

*(4) - midi pitch (60 → C4 = middle C)*

*(5) - velocity*

*(6) - note start in seconds*

*(7) - note duration in seconds*

*(8) - track*

Una vez que tenemos dicha matriz en MatLab, se pasa a realizar la síntesis de audio. Para llevar a cabo dicha síntesis, lo primero que se hace es una separación de canales del MIDI, obteniendo con esto una matriz para cada uno de ellos.

## 2.2 Síntesis con wavetables

Una vez que tenemos separados los canales, realizamos la síntesis para cada uno de ellos independientemente. Para esto, se pasa a la lectura de las filas de la matriz del canal, de las que se extraerán la nota (pitch), el comienzo de esta, la duración de la misma y su intensidad.

A continuación, se lleva a cabo una síntesis mediante wavetables. Se dispone de una serie de archivos *wav*, conteniendo cada uno de ellos la entonación de una nota por un conjunto coral real. Estos se extrajeron de la famosa base de datos Colossus de EastWest, muy empleada en estudios de grabación profesionales para el *sampling*, con sonidos de instrumentos reales, de líneas melódicas generadas por computador (pues dicha base de datos se compone de una ingente cantidad de sonidos capturados directamente de instrumentos reales de alta calidad). Gracias al uso de la herramienta software "UnNKS", los ficheros con extensión *nks* que componen Colossus pudieron ser decodificados en los ficheros *wav* mencionados y finalmente usados para la constitución de la wavetable. En total se dispone de 40 archivos con voz de hombre que van desde la nota 38 hasta la 77, y 34 archivos con voz de mujer que van desde la 55 hasta la 88, ambos rangos dados en notación MIDI.

Dado que tenemos la nota (pitch), su inicio, duración e intensidad, realizamos una lectura del fichero *wav* correspondiente de duración la deseada, agregando dicho fragmento sonoro a la señal final sintetizada en el espacio correspondiente, dado que conocemos el tiempo en el que esta nota debe agregarse. Para que a la hora de ir concatenando las diferentes notas

no existan saltos abruptos de señal (discontinuidades de fase), se generó un conjunto de matrices que contienen información referida a los cruces por cero de los diferentes ficheros *wav*. Esto nos permite ensamblar las señales siempre por un punto en el que la amplitud de ambas notas valen aproximadamente cero, lo que suaviza la discontinuidad de fase en los cambios de pitch con la consiguiente disminución del ruido que percibe el oyente en dichos instantes respecto de si no se aplicase esta metodología.

Una vez extraída y segmentada la señal del archivo *wav*, a esta se le aplica una ponderación por el factor de intensidad codificado en el fichero MIDI, seguido de un filtro ADSR para conseguir más realismo en la síntesis. Esto se realiza para todas las notas del canal, consiguiéndose finalmente una señal por cada banda, siendo todas ellas de igual duración e igual a la de la pieza final sintetizada, la cual es obtenida como composición aditiva de todas las señales generadas (una por canal). Finalmente, al resultado se le aplica un filtrado de acústica para obtener un resultado todavía más natural. Notar que, previo a ello, sobre cada canal se lleva a cabo la adición de una especie de suave *dithering* con el fin de aleatorizar levemente la voz, si bien el comienzo de la segmentación de cada nota sobre cada uno de los ficheros *wav* de la base de datos se aleatoriza en cada iteración.

Podemos ver este proceso más esquematizado en el diagrama de bloques de la página siguiente.

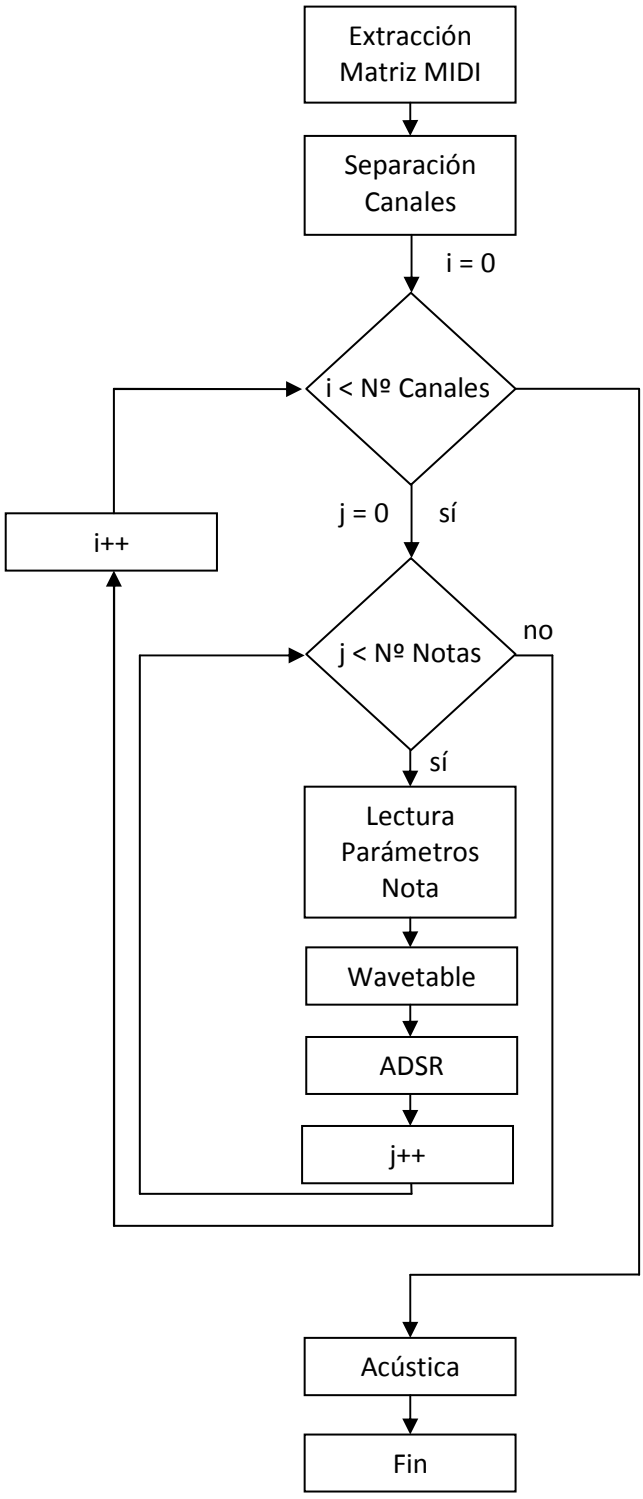


**Figura 1.** Diagrama de bloques del proceso de síntesis coral partiendo de un fichero MIDI.

## 2.3 Mejorando la acústica

Tal y como se ha mencionado en el anterior punto, tras obtener la pieza coral final sintetizada se aplica un filtrado para mejorar la acústica del resultado. El objetivo de este paso es triple:

- Suavizado de las discontinuidades en las interfases de notas musicales.
- Incremento de la correlación de las diferentes líneas melódicas de la coral tal y como ocurriría en la práctica, con el consiguiente incremento del realismo en términos de percepción psicoacústica.
- Emulación de la acústica del recinto, lo que confiere un mayor realismo y un sonido final más agradable.

El mencionado filtrado de acústica se implementa en el dominio de la frecuencia con la finalidad de minimizar el tiempo de cómputo respecto del derivado del cálculo de la convolución en el dominio temporal. Para ello, se dispone de un conjunto de respuestas al impulso obtenidas a través del software de edición de audio Adobe Audition. Estas se generan a partir de cargar un impulso unidad (generado con MatLab y exportado en formato *wav*) en la aplicación de Adobe y aplicar alguno de los presets de acústica disponibles. El resultado (la respuesta impulsiva de la sala) es almacenado en un fichero *wav* y cargado de nuevo en MatLab. Los coeficientes obtenidos son almacenados en un fichero *mat* que el prototipo de síntesis carga cuando precisa llevar a cabo el filtrado de acústica.

Sea $x(n)$ la señal de la pieza coral sintetizada y $h(n)$ la respuesta impulsiva que emula la acústica de un determinado recinto, la pieza musical realzada, $x_r(n)$, se obtiene como

$$x_r(n) = h(n) * x(n) = \text{FFT}^{-1}[H(k)X(k)].$$

Notar que, dado que $x(n)$ suele ser de mayor longitud respecto de $h(n)$, previo a calcular $H(k)$ se lleva a cabo una etapa de zero-padding en el dominio temporal sobre $h(n)$ con el fin de que ambas secuencias temporales sean de igual dimensión y poder aplicar el producto elemento a elemento en el dominio espectral.

La lista de acústicas contempladas en esta primera versión del prototipo son las siguientes:

| | |
|---|---|
| Club pequeño | En la ducha |
| Gimnasio | Gran auditorio |
| Gran cañón | Gran hall |
| Habitación seca y pequeña | Habitación tibia |
| Iglesia | Marcianos |
| Sala de conciertos (abierta) | Sala de conciertos (caliente) |
| Sala de conciertos (crujiente) | Sala de estar amueblada |
| Sala de estar vacía | Teatro |

**Tabla 1.** Acústicas de recintos disponibles en la primera versión del prototipo.

# 3. OMR y síntesis de partituras

En este apartado se describe el reconocedor automático de partituras y la síntesis de estas utilizando wavetables. Para el reconocedor de partituras se utiliza el Image Toolbox de MatLab, que consta de una gran variedad de funciones para el procesamiento digital de imágenes. Esta herramienta será útil para el reconocimiento de partituras sencillas, que contendrán la melodía con figuras como negras, blancas y redondas incluyendo sus respectivos silencios. En la siguiente figura se observa una partitura simple con la que se obtienen buenos resultados con el sistema OMR desarrollado hasta la fecha.

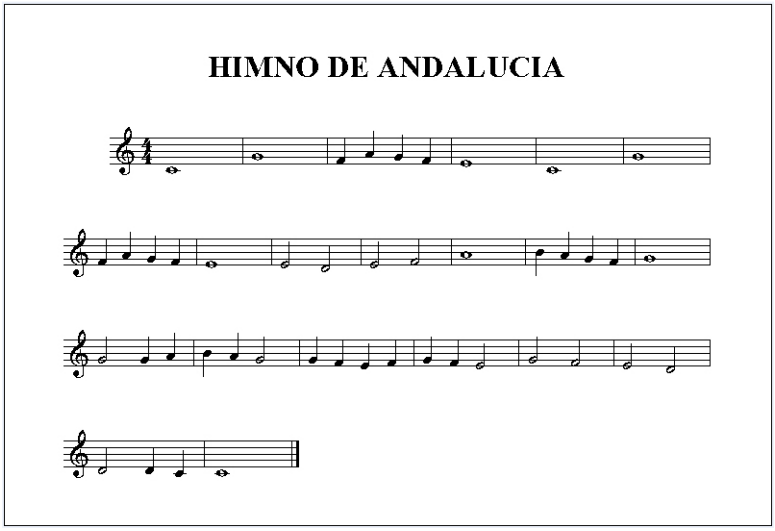


**Figura 2.** Ejemplo de partitura sencilla apta para el sistema OMR actualmente desarrollado.

## 3.1 Preprocesamiento de imagen

En la primera fase del OMR adaptamos la partitura para su tratamiento. Una vez obtenemos la imagen de la misma, supuesta su entrada

con el modelo de color RGB, se transforma a niveles de gris para seguidamente invertirla. La misión de la inversión es la de un tratamiento más sencillo mediante la contemplación de la información contenida en la partitura como niveles de gris relativamente grandes frente al fondo tendente a nivel cero. Finalmente aplicamos binarización para que los píxels de interés se mantengan en blanco frente al fondo negro.

## 3.2 Corrección de la inclinación

La segunda etapa consiste en la corrección de la inclinación que pueda presentar la imagen de partida. Este paso es de vital importancia dado que los métodos de segmentación y reconocimiento desarrollados no operarían correctamente ante la presencia de este defecto.

El método seguido para llevar a cabo la corrección de la inclinación se basa principalmente en la transformada de Hough. En primer lugar se aplica la transformada de Hough con una resolución en la componente theta de 0.05º. El espacio de parámetros de salida usado es el polar. La ventaja que aporta (aparte de los problemas que nos evitamos con las líneas verticales) es que obtenemos directamente una aproximación del ángulo de inclinación de la imagen.

Dado que estamos ante la imagen de una partitura musical que presenta numerosos pentagramas, los máximos de la transformada de Hough se corresponderán con las líneas que componen dichos pentagramas. En consecuencia, uno de los máximos de la transformada de Hough nos proporcionará la ecuación de la recta en coordenadas polares asociada a una de las

líneas de los pentagramas y, a partir de la misma, podremos obtener directamente una estimación del ángulo de inclinación.

Posteriormente, por si acaso la imagen presenta aún cierta inclinación, se opta por un ajuste de tanteo. Este consiste en aplicar inclinaciones a la imagen en los márgenes de error que tenemos tras aplicar el paso de la transformada de Hough. La realimentación para determinar si las correcciones que estamos realizando están surtiendo efecto o no (y en definitiva para obtener convergencia hacia la solución) se basan en el estudio de la proyección horizontal del histograma de la imagen resultado.

### 3.3 Segmentación de pentagramas

En la siguiente etapa se segmenta la imagen separando pentagramas y título de la partitura. Esto lo logramos con ayuda de la proyección horizontal de histograma. Dado que el fondo es completamente negro, las zonas inter-pentagrama se caracterizarán en la proyección horizontal de histograma por una secuencia continua de ceros por lo que, mediante un algoritmo de búsqueda de ceros basado en comparación con el valor anterior y posterior, establecemos el punto medio entre dos pentagramas para que estos sean separados.

### 3.4 Segmentación de símbolos

Para cada uno de los pentagramas, así como para el título, la siguiente etapa se encarga de segmentar cada uno de los símbolos que com-

ponen la línea con la ayuda de la proyección vertical de histograma de cada una de las líneas por separado. En el caso de un pentagrama, en su proyección vertical de histograma se podrá observar un nivel mínimo de continua correspondiente a la contribución de las cinco líneas del pentagrama. Este nivel, que conocemos como el mínimo no-cero, es detectado para cada uno de los pentagramas individualmente de modo que, con un algoritmo similar al anteriormente presentado para la segmentación de la proyección horizontal de histograma, segmentamos los símbolos de cada una de las líneas detectadas (con la diferencia del uso de un algoritmo de búsqueda del mínimo no-cero comentado). Estos límites intersimbólicos podrían aprovecharse para la segmentación directa del elemento en la imagen. No obstante, se prefirió realizar una caracterización del símbolo mediante el estudio del contorno de la proyección vertical de histograma del mismo, ya que podemos limitar su extensión mucho más fácilmente respecto de la extensión del símbolo en el interior del pentagrama (comentar en este sentido que la caracterización del símbolo mediante el estudio de seis momentos estadísticos arrojó peores resultados respecto del método finalmente implementado).

## 3.5 Reconocimiento de símbolos

A continuación se procede a la clasificación de cada uno de los símbolos detectados según el siguiente patrón. Ya hemos dicho que realmente estamos segmentando la proyección de histograma vertical del símbolo. Pues bien, con esta información hacemos lo siguiente:

- Eliminación de la componente de continua detectada como el mínimo no-cero anteriormente comentado.

- Normalización de los valores de la proyección vertical de histograma del símbolo.
- Obtención de 24 coeficientes del módulo de la FFT de la proyección vertical de histograma del símbolo.
- Comparación con cada uno de los patrones base almacenados por clase en la base de datos. La selección de la clase a la cual pertenece el símbolo se realiza en base a la minimización del error lineal calculado como la acumulación de las diferencias en valor absoluto entre cada par de coeficientes de los módulos de las FFT's del símbolo en estudio y del patrón de la base de datos con el cual se compara.

El estudio del símbolo en base a la FFT se basa, junto con la normalización, en la importancia de caracterizar con un número standard de coeficientes el símbolo, de modo que se contemple la propiedad de la invarianza a la escala.

Es importante notar que la función de MatLab asume que el símbolo que se estudia se va a encontrar en la base de datos, por lo que se selecciona mediante minimización del error y no se establece ningún umbral necesario para poder clasificarse o no. Otra limitación importante es que la función también asume que el título sólo ocupa la primera línea segmentada mediante la proyección horizontal de histograma. Esto se ha pensado así debido a la forma de segmentar un símbolo dentro de un pentagrama. Recordemos que el símbolo se lleva sobre el nivel cero (eliminación de la componente de continua introducida por las cinco líneas de pentagrama). Pues bien, el programa hace una excepción sobre el cálculo del nivel mínimo de continua para la primera línea ya que, si se selecciona un mínimo no-cero

en la línea del título, podemos perder y distorsionar la información no proveyendo un resultado coherente. Comentar también que sobre un conjunto de test con partituras sencillas se obtienen unos resultados excelentes con este método de clasificación.

Los símbolos contemplados en la base de datos aparecen recogidos en la siguiente figura. Notar que, para mayor precisión en la selección, por cada símbolo se incluyen varias caracterizaciones en función de si este puede aparecer reflejado, invertido (piénsese en una figura como una blanca). El reconocimiento de más símbolos aparte de los incluidos en la base de datos sería inmediatamente extensible, pudiendo cubrir con este método partituras de mayor complejidad si fuese necesario.

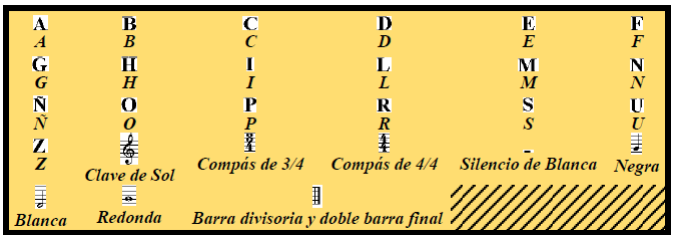


**Figura 3.** Conjunto de figuras contemplado en la base de datos del sistema OMR.

Como se observa, tampoco se ha incluido el reconocimiento de armaduras por simplicidad (sólo se admiten partituras cuya tonalidad sea de Do Mayor o su relativo menor, La menor) ni alteraciones accidentales. Por tanto, las partituras que se proporcionan de test están todas transportadas a la tonalidad de Do Mayor.

### 3.6 Detección de tono

Una vez tenemos segmentada la imagen y se han reconocido las figuras musicales, se procede a la detección de tono donde se requiera. El método seguido para la detección de tono está basado en el uso de la proyección horizontal del histograma. Dado el trozo de la imagen, hallamos su proyección horizontal (sobre el eje $y$). A continuación, se sigue un método en el que se determina tanto la posición de las líneas del pentagrama como las líneas virtuales (se pueden definir 13 líneas desde el Re 7 al Sol 3, 5 visibles y 8 virtuales). Lo que hacemos es recorrer la proyección horizontal hasta determinar las líneas del pentagrama. Como las líneas del pentagrama estarán representadas en blanco y ocuparán todo el trozo de la imagen, se corresponderán con máximos. Así determinamos sus posiciones y las distancias de separación entre cada par de líneas. Haciendo un promedio de la separación entre pares de líneas adyacentes visibles podemos obtener una estimación de la separación (aplicando redondeo por estar en un espacio discreto) entre líneas. Con esta separación estimada podemos determinar las posiciones de las 8 líneas virtuales restantes. Una vez determinadas todas las posiciones, vamos recorriendo la proyección del histograma situándonos en las líneas estimadas y en el punto medio de ambas. En cada posición haremos la suma de todos los elementos del histograma en un entorno igual a la mitad de la separación estimada. Finalmente, en aquella posición (de las 26 posibles: líneas y puntos medios entre dos líneas) donde se obtenga una suma mayor se considera que se encuentra la figura musical.

Mediante la siguiente expresión se puede asociar la frecuencia fundamental de vibración (pitch) a la nota para corroboración inmediata de la corrección del resultado obtenido:

$$f = 440e^{\left(\left(\left\lfloor \log_2\left(\frac{f}{32.7}\right)+1\right\rfloor -4\right)+\left(\frac{n-10}{12}\right)\right)\log(2)} \Longrightarrow$$

$$\Longrightarrow \log\left(\frac{f}{440}\right) = \left(\left(\left\lfloor \log_2\left(\frac{f}{32.7}\right)+1\right\rfloor -4\right)+\left(\frac{n-10}{12}\right)\right)\log(2) \Longrightarrow$$

$$\Longrightarrow \log\left(\frac{f}{440}\right) = \log(2)\left\lfloor \log_2\left(\frac{f}{32.7}\right)+1\right\rfloor - 4\log(2) + \left(\frac{n-10}{12}\right)\log(2) \Longrightarrow$$

$$\Longrightarrow \boldsymbol{n = 10 + \frac{12}{\log(2)}\left[\log\left(\frac{f}{440}\right) - \log(2)\left[\left\lfloor \log_2\left(\frac{f}{32.7}\right)+1\right\rfloor - 4\right]\right]}.$$

| n | Nota |
|---|---|
| 1 | Do |
| 2 | Do# |
| 3 | Re |
| 4 | Re# |
| 5 | Mi |
| 6 | Fa |
| 7 | Fa# |
| 8 | Sol |
| 9 | Sol# |
| 10 | La |
| 11 | La# |
| 12 | Si |

**Tabla 2.** Numeración empleada para referencias las notas musicales.

La anterior tabla relaciona el número natural $n$ obtenido de la aplicación de la anterior expresión en función de la frecuencia con la nota en cuestión. Por otro lado, la octava en la que dicha nota se encuentra también se puede obtener a partir de:

$$o = \left\lfloor \log_2\left(\frac{f}{32.7}\right)+1\right\rfloor.$$

Todos los símbolos y letras reconocidos en la partitura son guardados en un fichero de texto para hacer la síntesis posteriormente.

### 3.7 Síntesis de partitura

Una vez que el reconocedor automático de partituras ha extraído la información relevante, las notas y su duración, será posible la síntesis de ésta utilizando wavetables. Para ello, se debe leer el archivo de texto proporcionado por el OMR y extraer un vector con la información del pitch y su duración.

Tras recopilar los datos propios del vector de notas y duración, se realiza la síntesis de la partitura extrayendo información de la base de datos de wavetables. Es decir, para cada par [pitch, duración], se selecciona del archivo correspondiente del wavetable un número de muestras proporcional a la duración de la nota. El resto del procesado es igual al descrito en el punto 2.2.

# 4. Interfaz de usuario

En este apartado se desglosa la interfaz gráfica de usuario realizada para el manejo de todas las funciones relacionadas con la síntesis de partituras y archivos MIDI. El aspecto de la interfaz de usuario tras arrancar el programa es el que se muestra en la siguiente imagen.

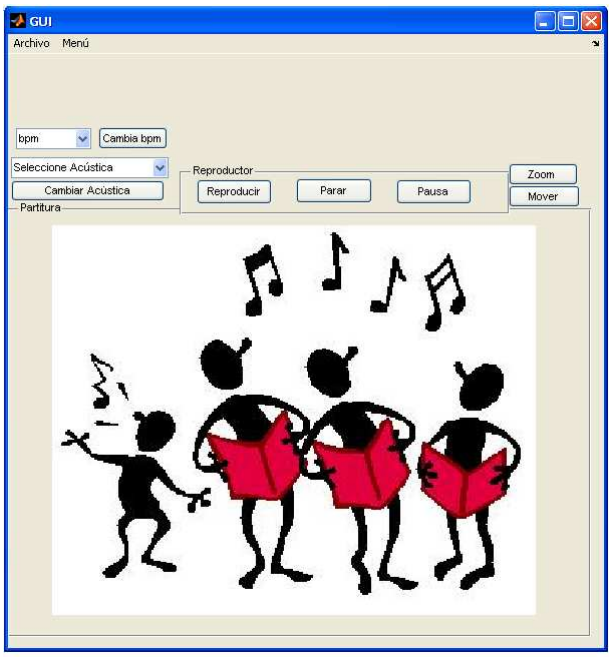


**Figura 4.** Interfaz de usuario del prototipo tras ser arrancado.

A través del menú *Archivo* se pueden realizar dos funciones básicas: lectura de archivos MIDI y lectura de partituras en formato JPEG, para posteriormente poder sintetizar las líneas melódicas de la partitura coral.


**Figura 5.** Menú *Archivo*.

Una vez leído el formato especificado, aparecerá en el panel la partitura del archivo seleccionado. Esto nos permitirá su visualización mientras es posible escucharla. A continuación se muestra un ejemplo de la interfaz gráfica de usuario tras leer el archivo indicado en el directorio de trabajo.

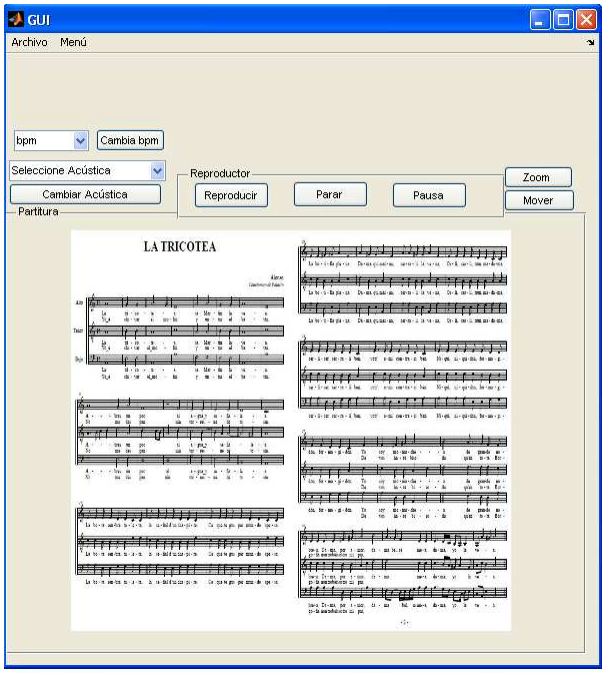


**Figura 6.** Aspecto del programa tras la lectura de un fichero MIDI o partitura.

Tras llevarse a cabo la lectura del fichero, el siguiente paso será seleccionar en *Menú* la opción *Samplear Midi...* o *Samplear Partitura...* para que sea sintetizado y crear un archivo *wav* con el resultado.

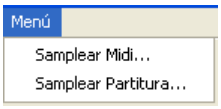


**Figura 7.** Menú *Menú*.

Cuando se realiza esta operación, podemos llevar a cabo una serie de acciones sobre el resultado de la síntesis, como seleccionar la acústica del

recinto de interpretación y cambiarla a nuestra elección con ayuda del menú desplegable que se muestra a continuación.

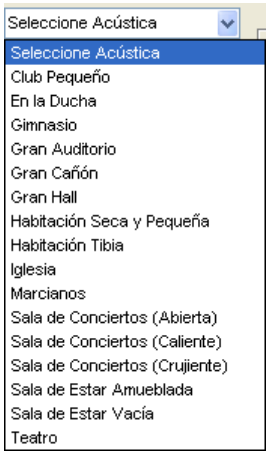


**Figura 8.** Desplegable con acústicas de recintos disponibles para su selección.

Una vez seleccionada la acústica deseada, se pincha sobre el botón *Cambiar Acústica* para poder modificar el fichero sintetizado. Con el panel de reproducción podemos escuchar el archivo resultante a nuestro gusto, ya que nos permite reproducirlo, pausarlo y pararlo. A continuación se observa el panel del reproductor.


**Figura 9.** Panel de reproducción.

Otra de las funciones que se ha desarrollado en la interfaz es la posibilidad de hacer zoom sobre la partitura y movernos por ella para poder visualizarla mejor. Dichas funciones se pueden realizar con los botones *Zoom* y *Mover*, que se encuentran en la parte superior derecha de la interfaz de usuario.

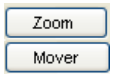


**Figura 10.** Botones de función de partitura.

Para la síntesis de partituras, es decir, cuando se lee un archivo JPEG y se aplica el sistema OMR desarrollado, es posible cambiar la velocidad de reproducción del archivo sintetizado. Esto se realiza con ayuda del menú desplegable *bpm* (beats per minute), en el que aparecen números del 60 al 120 con paso 10. Estas cantidades significan que, por ejemplo, se van a reproducir 60 negras por segundo de la partitura reconocida como mínimo, o 120 negras por segundo en el caso máximo. A continuación se observa una captura del menú desplegable.

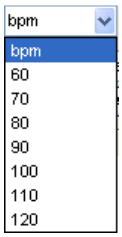


**Figura 11.** Desplegable con *bpm*'s seleccionables para la síntesis a través de partitura.

Una vez seleccionada la velocidad de reproducción, se pulsa sobre el botón *Cambia bpm*, creándose tras ello un archivo resultado con la métrica seleccionada. Con ayuda del menú de reproducción, a continuación se podrá escuchar la pieza sintética pinchando sobre el botón *Reproducir*.

# 5. Conclusiones y trabajo futuro

Tras la realización del "Prototipo I4", la calidad de audio conseguida y la percepción global obtenida por el grupo I4 es satisfactoria. No obstante, el trabajo no está concluido, por lo que el equipo pretende realizar futuras mejoras. Dado que el proyecto se dividió en sus inicios en la síntesis con letra y la síntesis sin letra, se podría decir que la rama de síntesis sin letra está finalizada a falta de unos pequeños matices que se comentan a continuación. Por tanto, nos enfrentaremos en el futuro, eminentemente, al problema de la síntesis de piezas corales letradas.

Como se ha comentado en el anterior párrafo, uno de los trabajos futuros más importantes estará basado en el reconocimiento y síntesis de partituras más complejas, incluyendo la letra.

Otra de las mejoras que se pretende incorporar, es un modelo más complejo de dinámica ADSR, ya que el actual empleado es uno basado en parámetros lineales, siendo este poco realista para el modelado de la voz humana.

En penúltimo lugar, y menos importante, se incluirán nuevos símbolos para mejorar el sistema OMR, ya que el actual utilizado sólo está pensado para el reconocimiento de partituras simples, poseyendo la mayoría de partituras corales una estructura más compleja con un mayor número de símbolos que reconocer. Otra cuestión derivada del aumento de la dimensión de la base de datos es la mejora de eficiencia de cómputo en el proceso de clasificación de símbolo y la reducción del consumo de memoria de dicha

base de datos. Para tal fin, se contempla modificar el descriptor actualmente utilizado fundamentado en el cálculo de 24 coeficientes de magnitud de la FFT de la proyección vertical de histograma segmentada y normalizada. Se experimentará con la aplicación de la transformada discreta del coseno (DCT) sobre el anterior conjunto de coeficientes (o uno más amplio) con el fin de decorrelar y compactar la información contenida en ellos en un menor número de valores. De este modo se reduce el conjunto de números almacenado en la base de datos y la cantidad de cómputo necesaria durante el proceso de clasificación.

Finalmente, y si el tiempo nos lo permite, se desarrollará una aplicación standalone funcional sobre el sistema operativo Windows.

## 6. Referencias

[1] MIDI Toolbox, Departamento de Música, Universidad de Jyväskylä, Finlandia. https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/miditoolbox/

[2] X. Fernández Hermida y C. Sánchez-Barbudo y Vargas, "Development of an Optical Music Recognizer".

[3] A. Sánchez, J. J. Pantrigo y J. I. Pérez, "Extracción de Líneas Melódicas a partir de Imágenes de Partituras Musicales".

Segundo Desafío Tecnológico del DTSTC

# MEMORIA DE PROYECTO (v2.)



**Grupo I⁴**

Iván Fernández Bermejo

Iván López Espejo

Santiago Prieto Calero

## Sumario

1. Introducción ..... 3
2. Implementación para Android ..... 4
   2.1 Estructura de la aplicación ..... 5
   2.2 Definición de servicio ..... 11
3. Actualizaciones del sistema OMR ..... 12
4. Conclusiones y trabajo futuro ..... 14
5. Referencias ..... 15

# 1. Introducción

Este documento tiene por objeto el plasmar en sus páginas los avances acontecidos en torno a la realización del proyecto propuesto para el II Desafío Tecnológico planteado por el Departamento de TSTC (Teoría de la Señal, Telemática y Comunicaciones) de la Universidad de Granada. Como tal, únicamente son mencionadas someramente las nuevas contribuciones y actualizaciones acerca del proyecto ya presentado en la anterior fase, por lo que esta memoria es únicamente un documento complementario al último presentado y, por ende, para su correcta comprensión es preciso tener en cuenta lo ya expuesto durante la segunda fase.

Debido a lo extenso y complejo del problema, para esta última etapa se ha decidido priorizar dos aspectos: por un lado, llevar a cabo correctamente la síntesis de los ficheros MIDI y partituras propuestas y, por otro, concluir una primera implementación de una aplicación destinada a funcionar sobre un dispositivo Android con fines eminentemente didácticos. Esta aplicación es una versión sencilla (actualmente monofónica) que nos permite construir una partitura simple con el fin de que esta sea sintetizada en voz.

Debemos mencionar asimismo que la síntesis de los ficheros MIDI no supuso ningún contratiempo y que, únicamente, hubo de adaptarse, mejorarse y ampliarse el sistema de reconocimiento óptico de partituras u OMR. Es por ello que las páginas de a continuación desglosan brevemente estos dos aspectos mencionados.

## 2. Implementación para Android

Esta sencilla primera versión para Android pretende ser el caballo de batalla de esta última fase. En la actualidad, las aplicaciones para dispositivos Android están teniendo una gran capacidad de penetración en la sociedad, resolviendo estas de un modo específico numerosos problemas cotidianos y relacionados con la necesidad de interacción con el mundo que nos rodea. Particularmente, la implementación desarrollada ha sido concebida con fines eminentemente didácticos pues, un alumno introductorio en música, puede escribir partituras simples monofónicas y escuchar el resultado de su síntesis en voz coral. Esto le puede permitir mejorar su oído musical y verificar conceptos básicos de lenguaje musical, reforzando la lectura de partituras y permitiendo la corrección de errores en términos de ritmo o entonación. Versiones posteriores incluirán la posibilidad de construir partituras más complejas y polifónicas.

Una vez el usuario haya escrito la partitura a partir de la sucesiva selección de figuras y notas musicales, esta es transmitida a un servidor externo, encargado este de realizar su síntesis. El servidor también es el responsable de devolverle a la aplicación que corre sobre el dispositivo móvil el fichero *wav* que codifica la síntesis de la partitura. Una vez disponible el audio en nuestro terminal, este puede ser reproducido desde la propia aplicación para su escucha. Las partituras se guardan en una lista de proyectos, permitiéndonos eliminarlas o modificarlas.

## 2.1 Estructura de la aplicación

La idea inicial de la aplicación consistía en realizar una fotografía de una partitura escrita en papel, la cual sería procesada y enviada al servidor, recibiendo en el terminal el fichero *wav* que codifica la señal con la voz sintética, pudiendo ser así almacenada y reproducida. Pese a tener la parte de control de la cámara realizada, obteniéndose correctamente la imagen de la partitura, esta idea tuvo que desecharse dado que el sistema OMR implementado hasta el momento no funcionaba correctamente con imágenes reales debido a la dificultad de llevar a cabo una umbralización apropiada.

Este problema nos hizo replantear el enfoque de la aplicación, de tal modo que propició el desarrollo de una pizarra virtual en la que nos encontramos un pentagrama donde es posible añadir figuras musicales a partir de seleccionarlas mediante una serie de botones. Esta pizarra facilita la generación de un documento *txt*, el cual es transmitido al servidor, obteniendo así el archivo *wav* correspondiente.

Obsérvese a continuación la figura 1, la cual recoge un diagrama de actividades de la aplicación finalmente desarrollada.

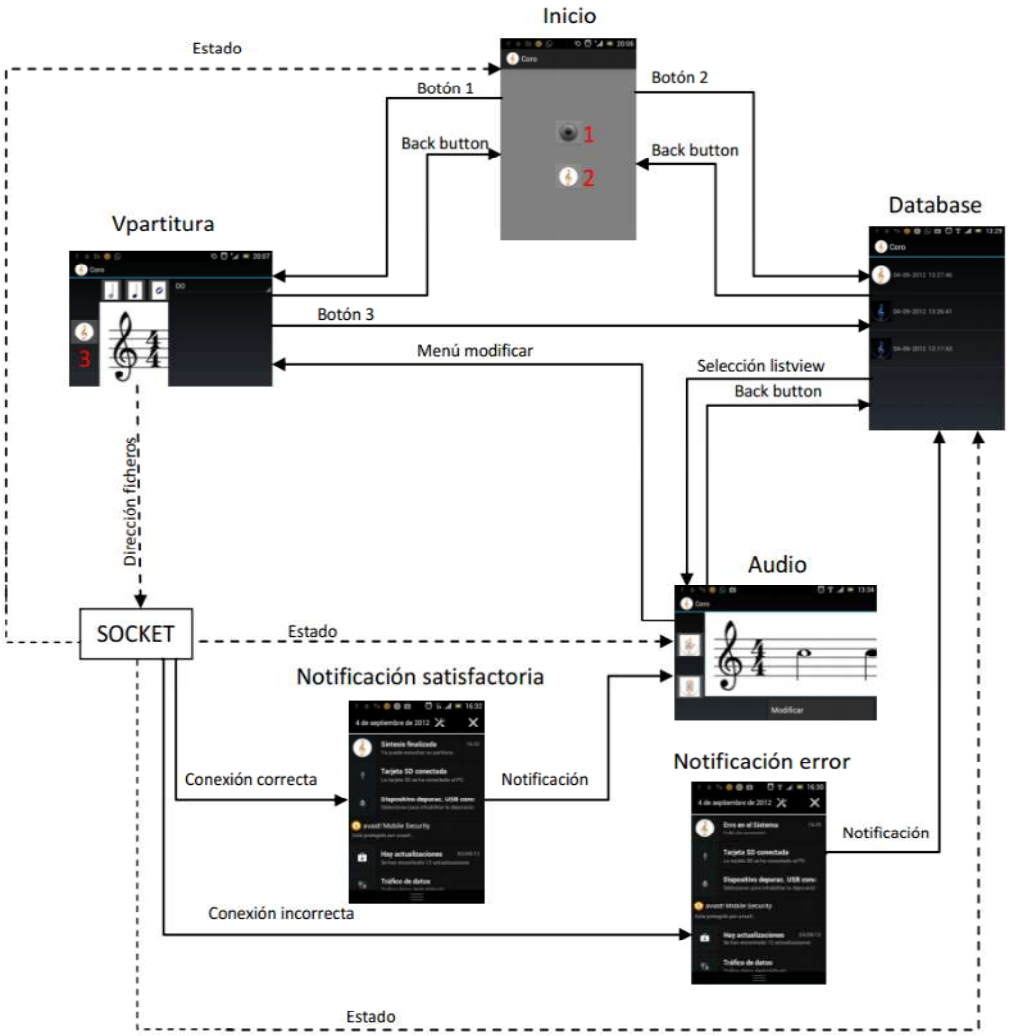


**Figura 1.** Diagrama de actividades.

## Coro.inicio

Este paquete está formado únicamente por la actividad *Inicio.java*, donde esta nos permite elegir entre la opción de irnos a una lista de los proyectos ya existentes o generar uno nuevo. El botón menú nos permitirá cambiar la dirección IP, usada para conectarnos al servidor. Podemos observar la vista de establecimiento de IP en la figura 2.


**Figura 2.** Cambiando la IP.

## Coro.partitura

Este paquete está formado por tres clases que son:

- Fichero.java: nos permite de manera cómoda escribir, leer y modificar los ficheros generados por esta aplicación, lo cuales serán de tipo *txt* por un lado, que codifican la partitura de manera adecuada para que sea enviada al servidor, y sin extensión por otro, en los que se guardará una lista de los IDs de las imágenes que conforman la partitura.
- HorizontalListView.java: nos permite tener un *listview* horizontal, siendo esto fundamental para la creación de la partitura.
- Vpartitura.java: esta actividad es la encargada de obtener la partitura gracias a su interfaz. Una vez que se desea enviar dicha partitura al servidor mediante el servicio de socket, el cual se verá más adelante, el *listview* que conforma nuestra partitura nos permitirá borrar notas no deseadas de un modo fácil a través de pulsar sobre estas. Esta opción puede visualizarse en la captura recogida en la figura 3. Esta actividad presenta también un menú, el cual nos permitirá seleccionar un valor para el parámetro *bpm*, así como elegir el entorno acústico. Podemos observar estos menús en la figura 4.

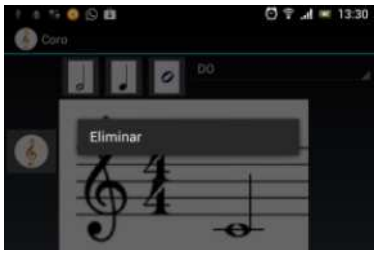


**Figura 3.** Eliminar figura.

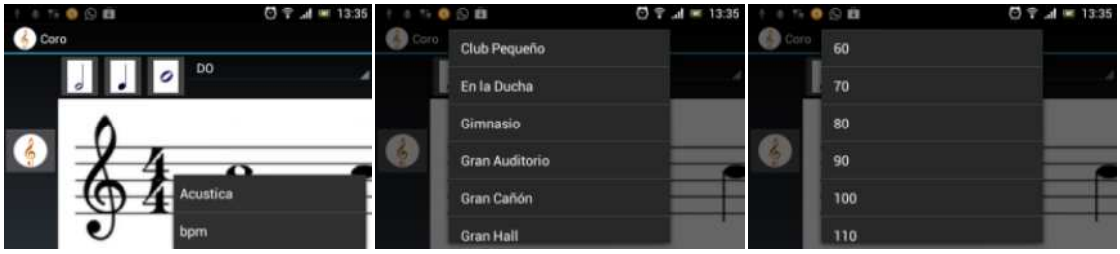


**Figura 4.** Diferentes capturas. De izquierda a derecha: menú de opciones, elección de acústica y selección de velocidad de reproducción (*bpm*).

## Coro.database

Este paquete está formado por cuatro clases que son:

- Partitura.java: los objetos de esta clase nos ayudan a manejar los atributos de la partitura.
- DatabaseRoad.java: debido a que la base de datos es actualizada y leída por casi todas las actividades de la aplicación, se terminó por generar esta clase que tiene métodos que nos facilitan la gestión de las llamadas a la base de datos.
- BaseDatosHelper.java: es en sí la clase que controla la base de datos.
- Database.java: actividad que nos permite ver en un *listview*, de manera cómoda y rápida, el contenido de la base de datos, dándonos opciones de eliminar o renombrar elementos de esta lista. Podemos visualizar estas particularidades en la figura 5.

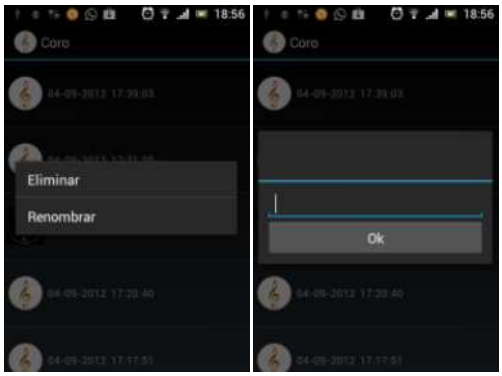


**Figura 5.** Varias capturas. De izquierda a derecha: menú con opciones y selección de cambio de nombre.

Esta última actividad también presenta un círculo de carga para el proyecto que se esté sintetizando, así como dos colores para la clave de sol: azul que nos indica que el proyecto no se ha podido sintetizar por algún motivo, y amarillo que nos especifica que dicho proyecto se encuentra sintetizado y puede escucharse. Estas últimas cuestiones pueden visualizarse en la captura de la figura 6.

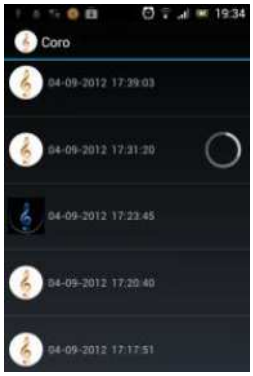


**Figura 6.** *Listview* con proyectos en carga y proyectos sin sintetizar.

## Coro.audio

Este paquete está formado por tres clases que son:

- MyFocus.java: este servicio nos permite manejar el "foco" puesto en la música que se escucha en el terminal. Así, si mientras el usuario

está escuchando uno de los resultados sintetizados recibe una llamada de teléfono, esta reproducción se detendrá.

- AudioServicio.java: este otro servicio nos permite llamar al mediaplayer para la reproducción de la música.
- Audio.java: actividad que nos permite iniciar, pausar y reanudar la reproducción de la melodía, así como también desplegar un menú que nos da la opción de modificar dicho fragmento musical.

**Coro.notificaciones**

Este paquete está sólo formado por la clase *Notificaciones.java*, la cual es la encargada de crear una notificación. Como se ha visto en la figura 1, ahora mismo la aplicación tiene un par de notificaciones disponibles (notificación satisfactoria y notificación de error). Estas notificaciones desaparecerán al pulsar sobre ellas, llevándonos a la actividad señalada en la figura 1.

**Coro.socket**

Este paquete está formado por tres clases que son:

- TomaFoto.java: objetos de esta clase son enviados con información como la acústica seleccionada o la velocidad de reproducción (*bpm*).
- TomaWav.java: cuando el servidor ha terminado de operar, un objeto de esta clase es recibido en la aplicación del terminal.
- SocketCliente.java: esta clase es la encargada de establecer una conexión con el servidor. Además de esto, también tiene la responsabilidad de lanzar las notificaciones cada vez que sea pertinente, así como de devolver su estado a las actividades.

**Coro.camara**

Paquete no utilizado en la actualidad, pues era el encargado de realizar una fotografía de una partitura para luego ser enviada mediante el servicio *SocketCliente.java*.

## 2.2 Definición de servicio

El servicio acepta un único cliente, pues no se ha implementado aún uno que haga uso de hilos, permitiendo esto último el que varios usuarios pudieran hacer uso del servicio de forma simultánea.

Como ya se ha mencionado, el servicio es externo a la aplicación que corre en el terminal. Uno de los motivos que hacen que esto sea así es el elevado consumo de memoria de los ficheros *wav* que constituyen la base de datos. Se está estudiando la posibilidad de reducir las necesidades de memoria de la misma a partir de la recodificación de los ficheros constituyentes al formato MP3.

El servicio está formado por un *jar* denominado *Receptor.jar*. Este programa no tiene interfaz de usuario, por lo que habrá que ejecutarlo desde la consola. En la figura 7 podemos ver cómo se ha llamado a dicho programa y lo que aparece en la línea de comandos una vez ejecutado correctamente.


**Figura 7.** Ejecución del servicio.

# 3. Actualizaciones del sistema OMR

En líneas generales el sistema de reconocimiento óptico de partituras permanece inmutable. No obstante, se han llevado a cabo pequeñas modificaciones con el fin de adaptar el sistema OMR al reconocimiento de partituras corales algo más complejas como las proporcionadas.

La etapa de preprocesamiento de la imagen continúa intacta con la salvedad de que ahora se hace uso de un método más sofisticado para la umbralización de la partitura, que es el método de Otsu. Con algunos pequeños nuevos matices de implementación irrelevantes para esta memoria, la segmentación de los pentagramas continúa llevándose a cabo mediante el estudio de la proyección horizontal de histograma, análogamente segmentándose las diferentes figuras por cada línea a partir del uso de la proyección vertical de histograma. Una vez segmentado cada símbolo, cada uno de ellos es comparado con cada uno de los patrones que conforman la base de datos a partir del estudio de la correlación cruzada (recordemos que la clasificación de la figura anteriormente se efectuaba a través del análisis de la distancia euclídea entre los vectores de coeficientes de magnitud de la FFT (Transformada Rápida de Fourier) de los patrones de la base de datos y de la figura que se deseaba clasificar). No obstante, antes de proceder con la comparación en sí, se aplica un preprocesamiento sobre el símbolo en cuestión con la finalidad de que sea extraído del fondo constituido por las líneas del pentagrama. Dicho preprocesamiento consiste en los siguientes pasos:

- Erosión con elemento estructural horizontal de 20 píxels.

- Sustracción de la imagen erosionada a la imagen original.
- Dilatación del resultado con elemento estructural vertical de 3 píxels con la finalidad de mitigar el daño causado por la erosión inicial sobre la figura de interés.

Como ejemplo real para comprender el efecto de los anteriores pasos, obsérvese el ejemplo recogido en la figura 8.

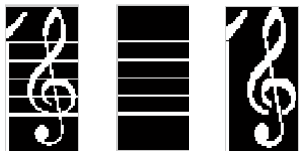

**Figura 8.** Ejemplo de preprocesamiento de figura segmentada previo a la comparación con los patrones de la base de datos. De izquierda a derecha: figura original segmentada, resultado de sustraer la imagen erosionada a la imagen original y figura final extraída del fondo y dilatada con el elemento estructural.

Nótese finalmente que la base de datos de figuras se ha ampliado consecuentemente con el fin de poder llevar a cabo un reconocimiento apropiado de las partituras proporcionadas. De otro lado, el tono de la figura se determina análogamente a lo ya expuesto en el anterior documento de memoria.

# 4. Conclusiones y trabajo futuro

Como trabajo futuro se desea abordar las líneas mencionadas en la anterior memoria y que no han sido solventadas en la presente fase, en especial lo que concierne al asunto de la síntesis con letra.

También se desea comentar que, lamentablemente, esta aplicación no ha podido ser alojada en Google Play debido a la no finalización del servicio de síntesis. Incluso, se puede plantear la posibilidad de cambiar la síntesis para no precisar de un servicio externo a dicha aplicación. Un escollo relacionado es el uso actual de una base de datos comercial que no podría ser integrada en la práctica debido a las restricciones derivadas de los derechos de propiedad.

## 5. Referencias

[1] D. Carretero de la Rocha, *Sistema de Reconocimiento de Partituras Musicales*. Proyecto fin de carrera, 2009.