APPENDIX B: THESIS' SYNTHESIS IN SPANISH / SÍNTESIS DEL PROYECTO EN CASTELLANO

B.1. INTRODUCCIÓN

B.1.1 Descripción del Campo de Estudio

Uno de los logros más importantes de los pasados años ha sido la aparición y el amplio despliegue de infraestructura inalámbrica y de dispositivos portátiles. La movilidad inherente causada por la creciente incorporación de interfaces inalámbricos en estos dispositivos portátiles abre un nuevo mundo de posibilidades para el usuario que la infraestructura conectada por cable y los dispositivos fijos no podían ofrecer. A medida que los estudios en este innovador campo avanzan, muchos aspectos distintos en necesidad de mejora y numerosos obstáculos que superar emergen. Por ejemplo, el asegurar una conectividad constante, mientras el dispositivo se traslada de un lugar a otro, es un aspecto muy importante a considerar. Alternativamente, este proyecto demuestra que el usuario puede que no necesite conectividad constante mientras que la lista de reproducción del mismo tenga contenido suficiente para reproducir. Por lo tanto, un período en el que el usuario no esté conectado a la red, desde el momento actual hasta que se alcance el final de su lista personal de reproducción, es considerado viable.

Otros conceptos importantes, aparte de la movilidad, que considerar a la hora de diseñar nuevos dispositivos son la "sensibilidad al contexto" y el relacionado uso del estado del usuario como fuente de información para el dispositivo. Según lo descrito en [1] y tal y como se explicó anteriormente, las personas usan un número creciente de dispositivos móviles, como por ejemplo los ordenadores portátiles, los teléfonos móviles, las "Personal Digital Assistants" (PDAs), así como también dispositivos fijos, como ordenadores personales y teléfonos fijos, para una variedad de diversas tareas. Sin embargo, ninguno de esos dispositivos es consciente del entorno en el cual se encuentra el usuario y ninguno de ellos toma el estado del usuario (ocupado, disponible, en el trabajo, en casa...) como entrada de datos. Esto da lugar a un panorama en el cual se consume mucho tiempo atendiendo a las diversas clases de notificaciones de los diversos tipos de dispositivos (e-mail en los ordenadores personales o portátiles, llamadas en los teléfonos, mensajes en los buscapersonas [1],...).

Algunos de los problemas causados por el hecho de tener múltiples dispositivos son la carencia de diferenciación en las señales de notificación, el mínimo conocimiento del usuario y su entorno, dispositivos que no

aprenden de interacciones anteriores con el usuario, carencia de notificaciones coordinadas, e interrupciones en el lugar de trabajo [1]. Teniendo en cuenta todos estos problemas que pueden suceder al usar tales dispositivos, no es difícil imaginar porqué algunas personas se niegan a usarlos (o los mantienen desconectados muy frecuentemente) y porqué un nuevo enfoque en este campo es necesario para mejorar la experiencia del usuario.

"Nomadic Radio" es definida como: "…una plataforma informática móvil que proporciona un interfaz unificado de sólo audio a servicios y mensajes remotos, tales como e-mail, correo de voz, la recepción de noticias del día y de anotaciones de agenda personal…"[1]. *"Nomadic Radio"* puede ser una solución parcial a la situación de sobrecarga de información que se presenta con múltiples dispositivos, realizando una entrega de la información oportuna y filtrada, relevante al estado actual del usuario [2]. El desarrollo de *"Nomadic Radio"* se funda en el reconocimiento de voz ("speech and speaker recognition"), la síntesis texto-voz ("text-to-speech synthesis") y el audio espacializado ("spatial audio"). La detección continua del usuario y de su entorno, el priorizar la información que se recibe y poseer una adecuada infraestructura de red inalámbrica es también necesario.

Implementando siete niveles crecientes de notificación: silencio, señales de ambiente, notificación auditiva, resumen del mensaje, vista previa, mensaje completo e interpretación en primer plano, *"Nomadic Radio"* proporciona una presentación escalable, entregando información suficiente, mientras que reduce al mínimo la interrupción al usuario. Usando indicaciones contextuales, tales como prioridad de mensaje, nivel de uso y probabilidad de conversación, parámetros como el nivel de notificación y la latencia de presentación se calculan, dando como resultado notificaciones para los mensajes recibidos que están dinámicamente escaladas [1].

"SmartBadge versión 4" se considera una plataforma adecuada para una posible realización de todas las mejoras conseguidas con *"Nomadic Radio"*. PCMCIA, USB, infrarrojos y puertos compact flash están presentes en la SmartBadge, dándole así la capacidad de comunicarse de diferente forma en distintos entornos. Es pequeña y puede funcionar con pilas, por lo que la movilidad es inherente en la SmartBadge. Posee un procesador Intel Strong ARM, lo cual permite que se puedan desarrollar aplicaciones en ella y además posee variedad de sensores [3].

La posibilidad de tener y de usar múltiples interfaces inalámbricos que *"SmartBadge versión 4"* ofrece, le da mayor importancia al concepto de

sensibilidad al entorno ("context-awareness"). Este concepto se refiere al conocimiento del 'contexto' en cuál se encuentra el usuario; por lo tanto, sus actividades pueden influenciar la funcionalidad del dispositivo(s). El término 'contexto' es bastante amplio y puede abarcar la localización física actual del usuario ("location-awareness") y/o el estado del usuario (trabajando, descansando, durmiendo...).

Un buen sistema multimedia "sensible al contexto" combinado con los múltiples interfaces inalámbricos de *"SmartBadge versión 4"*, ofrece la posibilidad de asegurar conectividad constante (si es requerida), mientras que, al mismo tiempo, también permite elegir el tipo de interfaz inalámbrico que satisfaga lo mejor posible las necesidades actuales del usuario dentro de la infraestructura disponible.

Teniendo presente la enorme inversión en infraestructura inalámbrica que se está llevando a cabo actualmente, junto con la necesidad de mejorar el diseño de dispositivos móviles [2] y la posibilidad de usar *"SmartBadge versión 4"* como una plataforma real donde probar aplicaciones, un estudio más profundo de este tema se consideró apropiado.

B.1.2 Especificación del Problema

El principal objetivo de este proyecto es el de tratar los problemas expuestos anteriormente sobre el uso de múltiples dispositivos, considerando las atractivas mejoras introducidas por *"Nomadic Radio"*. Estudiando cómo utilizar y explotar el interfaz de audio de un dispositivo móvil y fundamentándose en la conversión texto-voz, el reconocimiento de voz y en una infraestructura sensible al contexto, se pretende explorar las posibilidades de este novedoso campo.

Para realizar el entorno de prueba, una aplicación cliente/servidor fue diseñada, usando el protocolo UDP. Esta aplicación consiste en un servidor-manager y varios clientes. El manager construye y modifica una lista de contenido de audio, la cuál determina qué se debe reproducir como audio para el usuario y cuándo debe ser hecha dicha reproducción. El manager gestiona esta lista de reproducción, que será modificada dinámicamente para incluir nueva información de interés para el usuario (canciones, notificaciones de audio,...), insertando este nuevo contenido en el lugar apropiado de la lista según ciertas prioridades prefijadas.

En principio se estudió cómo son construidas habitualmente las listas de reproducción, como por ejemplo las utilizadas para la difusión de radio comercial. A la hora de estudiar cómo estructurar los datos que una lista de

reproducción contiene normalmente, el protocolo XML fue tenido en cuenta, al considerarse que posibilita la representación externa de la lista de reproducción de una manera flexible y fácil. XML se fundamenta en los puntos fuertes de los protocolos SGML y HTML.

Uno de los clientes obvios y necesarios para la aplicación es un reproductor para la lista de reproducción construida por el manager. El cliente-reproductor genera salida de audio, seleccionando al reproductor apropiado para cada elemento de audio contenido en la lista de reproducción. Otro cliente proporciona un interfaz para el usuario, permitiendo a éste especificar los archivos de audio que quiere que el manager incluya en la lista. Para generar las notificaciones de audio que se incluirán en la lista, un cliente construido previamente por Sean Wong [19], que usa la herramienta de conversión de texto a voz Festival-Lite [16], fue modificado para que interactúe con el manager de manera apropiada.

El sistema resultante fue probado en un ordenador portátil cuyo sistema operativo era linux y que usaba una tarjeta inalámbrica para conectarse a una red inalámbrica privada (WLAN). Este ordenador se conecta vía WLAN con una *"SmartBadge versión 4"*. Este último ordenador es considerado un dispositivo móvil que también funciona con el sistema operativo linux y utiliza una tarjeta inalámbrica dentro de la misma WLAN privada. Basándose en pruebas realizadas en este sistema, con el servidor y los clientes iniciales, se presentan algunas conclusiones y posibles mejoras del proyecto, las cuales pueden ser la base de proyectos futuros.

B.1.3 Prerrequisitos

Para poder adquirir una comprensión plena, el lector debe poseer algún conocimiento previo y entender los conceptos básicos y los fundamentos de la transmisión de datos y la comunicación entre ordenadores, incluyendo la comunicación inalámbrica (específicamente las redes de área local inalámbricas (WLAN)), y los principios y funciones de los protocolos de comunicación, específicamente la pila TCP/IP.

B.2. DISEÑO

Teniendo en cuenta lo indicado en el capítulo de introducción, este capítulo presenta el diseño sugerido en el presente proyecto, el cual pretende ser un primer paso en la solución a los problemas que surgen al usar múltiples dispositivos, basándose para ello en estudios previos como el realizado en *"Nomadic Radio"*.

B.2.1 Descripción General

La solución pretende ser flexible y modular, de forma que permita mejoras y ampliaciones sucesivas de dicha solución y de forma que facilite posibles cambios en el futuro. Esta solución se basa también en el trabajo hecho anteriormente por Sean Wong [19]. Las modificaciones necesarias para la adaptación de su trabajo al diseño actual, específicamente en uno de los clientes creados en su aplicación cliente/servidor, "*client.c*", serán también descritas.

B.2.1.1 Metodología

Se comenzará describiendo la funcionalidad y las características generales de los componentes del diseño, para luego describir con mayor profundidad cada uno de dichos componentes. Se ha decidido diseñar una aplicación UDP usando el modelo cliente/servidor con la finalidad de crear un entorno de prueba. En esta aplicación se diseñará un servidor-manager junto con tres clientes. El servidor-manager y otro de los clientes, el reproductor, funcionarán en un dispositivo móvil, siendo *"SmartBadge versión 4"* el utilizado en este caso. Luego, el cliente generador de alertas, que es el cliente diseñado por Sean Wong y modificado para este proyecto para ser usado como generador de notificaciones de audio usando la herramienta de FLite; y el cliente que proporciona un interfaz para el usuario, funcionarán en un ordenador portátil cuyo sistema operativo es linux. El ordenador portátil y *"SmartBadge versión 4"* se comunican a través de un interfaz de WLAN en una red inalámbrica privada.

Como se especificó anteriormente, la aplicación se compone de cuatro programas:

- "manager"

  Este programa actúa como el servidor de la aplicación UDP. El manager construye y modifica una lista de contenido de audio, dando como resultado una lista de reproducción. Los datos necesarios para la construcción de la lista de reproducción serán estructurados usando el protocolo XML, el cuál proporciona una representación externa de la lista de reproducción flexible y fácil de usar. Esta lista de reproducción determina lo que el usuario debe obtener como salida de audio de su dispositivo y cuando debe obtener dicha salida. Nueva información (canciones, notificaciones de audio...) será agregada dinámicamente por el manager en el lugar apropiado de la lista de reproducción.

- "interfaz de usuario"

  El primer cliente que fue implementado fue un interfaz de usuario que permite a los mismos interactuar con el manager en la construcción y la modificación de la lista de reproducción. Este cliente proporciona seis comandos distintos al usuario y, a través de uno de éstos, puede decirle al manager qué contenido de audio desea escuchar y con qué prioridad. Otros comandos permiten que el usuario elimine contenido de audio de la lista de reproducción, pregunte al manager cuál es el primer, segundo, tercer,..., elemento a reproducir, haga que el manager guarde la lista de reproducción en un archivo para su posterior uso, o haga que el manager termine.

- "generador de alertas"

  Según lo mencionado anteriormente, el generador de alertas es el resultado de las modificaciones efectuadas al programa previamente escrito por Sean Wong [19]. Estas modificaciones fueron necesarias para que dicho programa pudiera interactuar correctamente con el manager. Este cliente fue incluido en la aplicación con el fin de tener un ejemplo de un generador de alertas de audio y por lo tanto de notificaciones de audio, las cuáles deben ser tratadas de distinta forma con respecto del resto del contenido de audio (primordialmente porque éstas pueden tener muy diversas prioridades). La herramienta de conversión de texto a voz Festival-Lite será utilizada por este cliente para alcanzar sus propósitos.

- "reproductor"

  Este cliente es obviamente necesario para poder reproducir el contenido de audio almacenado en la lista de reproducción. El reproductor pide constantemente al manager información sobre el primer elemento almacenado en la lista de reproducción. Si existe tal elemento, el manager envía el nombre del archivo a reproducir, el tipo de formato del archivo de audio (MPEG, WAVE o AU) y el momento en que debe ser reproducido. El cliente reproductor pide al manager que borre el elemento en cuestión de la lista de reproducción, selecciona el reproductor de entre los posibles dependiendo del tipo del archivo y reproduce el contenido de audio. Seguidamente, le vuelve a pedir al manager información sobre el primer elemento de la lista.

B.2.2 XML: Representación Externa de la Lista de Reproducción

Según lo explicado en la introducción, el primer paso en la construcción de la aplicación fue el diseño de una lista de contenido de audio a reproducir. Con este propósito, fue realizado un estudio de cómo las listas de reproducción son generalmente construidas. Este estudio reveló varios elementos que son comúnmente contenidos en una lista de reproducción y que por lo tanto tuvieron que ser tomados en cuenta al diseñar la lista. Un método para estructurar la información que la lista de reproducción debía contener era necesario. El protocolo XML fue considerado como uno de los más útiles de dichos métodos, pues podría definir una representación externa flexible de la lista de reproducción.

XML fue utilizado para definir inicialmente los elementos necesitados para la lista de reproducción. La información que fue seleccionada como parte de la lista de reproducción es principalmente de dos tipos distintos: información contenida generalmente por listas de reproducción, tales como una lista de reproducción de una estación de radio, y la información específicamente necesitada para la propia aplicación. Nombre, artista y álbum del contenido de audio (las canciones proporcionarán generalmente esta información), duración (en segundos) y tamaño (en bytes) son ejemplos de la información contenida en listas de reproducción comunes. Prioridad, tipo de formato del archivo de audio y tipo de audio (ya sea canciones o notificaciones de audio) son incluidos para satisfacer necesidades del diseño específico.

Usando XML, las etiquetas necesarias fueron definidas. Dado este documento de XML, una traducción manual desde XML a C dio como resultado el código para la lista de reproducción presente en “manager.c”. Según lo indicado antes, una mejora posible a la solución presentada sería un programa que procesa el documento de XML y produce el código C necesario para manipular la lista.

B.2.3 Descripción del Programa del Manager

Según lo indicado previamente, el manager funciona como el servidor de la aplicación UDP cliente/servidor. El propósito de este manager es construir y manipular una lista de contenido de audio que se reproducirá para un usuario. Como con cualquier servidor, el manager se comunica con sus (tres) clientes, reproductor, interfaz de usuario y generador de alertas, enviando y recibiendo datagramas. Un archivo de cabecera, “playlist.h”, se utiliza para definir las variables comunes a los cuatro programas. El nombre del host (“hostname”) que el manager usa es definido como

"badge41", ya que opera en *"SmartBadge versión 4"*; y la dirección IP que corresponde a este host es 10.0.1.6, una dirección privada en una red inalámbrica.

Otro archivo de cabecera, "manager.h", utilizado solamente por el manager, contiene todas las definiciones de constantes, variables y funciones, necesarias para el "manager". La primera definición que se encuentra al mirar en "manager.h" es la traducción del documento de XML de la lista de reproducción a la estructura de C de la lista de reproducción. Específicamente los atributos de una entrada de esta lista son:

```
struct song {
  char songfile[Maxdat];/*nombre del archivo a reproducir*/
  char songname[Maxdat];
  char artist[Maxdat];
  char album[Maxdat];
  int priority;       /*prioridad para reproducirlo*/
  int recordingtype;  /*tipo de formato del archivo*/
  int songnumber;     /*orden dentro de la lista*/
  float song_length;  /*duración en segundos*/
  unsigned long songsize;    /*tamaño en bytes*/
  long samplerate;    /* muestras por segundo */
  int audiotype;      /*tipo de audio*/
  double time_to_play;  /* momento de reproducción*/
  struct song *next_song;
  struct song *previous_song;
} list_entry;
```

Después, se definen tres variables globales: *first*, que es un puntero al primer elemento de tipo *song* de la lista de reproducción; *last*, un puntero al último elemento de tipo *song* de la lista; y existing_audioalert, que es una variable usada para almacenar las alarmas de audio, ya que éstas se tratan de distinta forma con respecto a las canciones. Finalmente, se encuentran las definiciones de las funciones usadas en manager.

Las acciones siguientes son las que ejecuta el manager para procesar una petición de uno de los clientes:

- El manager espera hasta que recibe un datagrama de uno de los clientes y, una vez que éste es recibido, comprueba el primer número en la cadena de caracteres recibida con el fin de determinar cuál es la petición del cliente.

- Si el cliente interfaz de usuario ha ordenado al manager terminar, entonces éste comprueba si hay contenido de audio aún por reproducir en la lista. Si es así, envía estos elementos al reproductor y, tras ello, termina al reproductor y a sí mismo.

- Si no hay orden de terminación, la acción tomada es llamar a una función para manejar la petición del cliente en cuestión. Todos los mensajes enviados de vuelta a estos clientes también contienen un número al principio que indica a qué petición corresponde la respuesta. Se asume que es siempre un par petición/respuesta la forma de comunicación entre servidor y cliente. La recuperación ante pérdidas no está implementada actualmente.

Las peticiones posibles son:

1) Un nuevo elemento se debe incluir en la lista

- El manager crea e inicializa una nueva estructura de tipo song, basándose en la información contenida en el datagrama. Esta información incluye el nombre del archivo en el cual se almacena el contenido de audio (songfile), nombre, artista y álbum (cuando esta información no está disponible estos campos se rellenan con las cadenas "nombre", "artista" y "álbum"), la prioridad, el tipo de formato del archivo de audio, el orden dentro de la lista (songnumber), la duración en segundos (song_length), el tamaño en bytes (songsize), las muestras por segundo (samplerate), y tipo del contenido de audio (audiotype).

- Todos los archivos de audio que utilizará la aplicación se almacenan en un sistema de ficheros NFS montado (en /opt/Badge4) en *"SmartBadge versión 4"*, pudiendo realizar dicha operación al estar ésta conectada lógicamente con el ordenador portátil, donde se encuentran los archivos a usar, vía WLAN. El manager tiene así acceso a estos archivos y los copia al directorio /tmp de *"SmartBadge versión 4"*. Observe que el dispositivo móvil podría utilizar otros medios de conseguir los archivos (tales como las peticiones "get" de HTTP).

- Para el siguiente paso se distingue entre las canciones y las alarmas de audio. Si el manager tiene que almacenar una *canción*, se almacena ésta basándose en una prioridad fijada por el usuario (1= se almacena en primer posición, 2= se almacena después del primer elemento, 3= se almacena donde ha especificado el usuario, si es posible, 4= se almacena al final de la lista). Si el elemento a almacenar es una *alarma de audio*, la cuál tiene la prioridad fijada a 1, se almacena la primera si no hay otra alarma de audio almacenada ya en la lista de reproducción. En caso de que ya existan alarmas de audio, se almacena después de la última alarma almacenada. Esto se hace para

mantener la información en correcto orden, en caso de que las alarmas de audio sean sobre el mismo acontecimiento. Además de almacenar el contenido, el parámetro de orden dentro de la lista ("songnumber") es actualizado y el parámetro del momento de reproducción ("time_to_play") se calcula, ya que depende del momento de reproducción y de la duración de los elementos anteriores en la lista. Un mensaje que indica que el archivo ha sido almacenado se envía al cliente.

2) El manager devuelve información sobre el primer elemento en la lista

Esta información es devuelta al cliente de quien ha recibido la petición. Si la lista está vacía, se le es indicado al cliente. Cuando la lista no está vacía, el nombre del archivo del primer elemento, su tipo de formato, las muestras por segundo, el tipo de audio, duración y momento de reproducción, se envían de nuevo al cliente.

3) El manager devuelve información sobre el elemento enésimo en la lista

El parámetro del número de orden ("songnumber") del elemento solicitado se lee de la petición recibida, se busca el elemento y la misma información que para la petición del primer elemento se devuelve si el elemento fue encontrado. Si el elemento no está presente en la lista de reproducción o si la lista de reproducción está vacía, se le es indicado al cliente.

4) El manager guarda la lista de reproducción en un archivo

El nombre de este archivo se define estáticamente como "playlist", pero los archivos guardados se nombran como "playlist-xxxxxxxxx", para tener un nombre distinto cada vez se guarde un archivo. La variable añadida al nombre es calculada llamando a la función "time". Los parámetros: nombre del archivo, nombre, artista, álbum, tipo de formato, duración, tamaño, muestras por segundo y tipo de audio de cada uno de los elementos de la lista de reproducción se guardan en el archivo. Si la operación es efectuada correctamente, un mensaje indicándolo se devuelve, pero, si hay algún problema al abrir el archivo o si la lista está vacía, el cliente es informado de este error.

5) Un elemento de la lista de reproducción debe ser eliminado.

El manager busca en la lista de reproducción dicho elemento por el nombre del fichero y, si está, se elimina (si el mismo archivo se encuentra más de una vez en la lista de reproducción, solamente el primer elemento encontrado es eliminado). Los parámetros número de orden ("songnumber") y momento de reproducción ("time_to_play") se actualizan en cada elemento, cuando es necesario. La contestación al cliente indica acción correcta o fallo.

B.2.4 Descripción del Programa del Interfaz de Usuario

Este cliente permite que el usuario pueda agregar manualmente o eliminar elementos de la lista de reproducción, pedir información sobre dichos elementos, guardar la lista de reproducción en un archivo, o hacer terminar al manager y al reproductor a su vez (cuando no hay elementos en la lista a reproducir). El nombre del host ("hostname") que este cliente utiliza es el que usa el ordenador portátil en el que funciona. La dirección IP que le corresponde a este nombre de host es 10.0.1.4, otra dirección privada dentro de la red inalámbrica que el ordenador portátil y *"SmartBadge versión 4"* usan para su comunicación.

El archivo de cabecera, "user_interface.h", utilizado solamente por este cliente, contiene todas las definiciones de constantes, variables y funciones necesarias. Se deben destacar dos de las definiciones dentro de este archivo de cabecera. En primer lugar, la definición de las estructuras para los "chunks" de los archivos wave (los archivos wave se estructuran en distintas partes llamadas "chunks": riff, format y data chunk,). Estas estructuras son necesarias para obtener los parámetros de muestras por segundo y bytes por segundo de la cabecera de un archivo wave. En segundo lugar, las estructuras necesarias para leer la información que se encuentra dentro de un archivo MP3. También se definen dos tablas con los posibles valores de sample-rate y de bitrate de un archivo MP3, otra estructura para leer una cabecera MPEG y algunas constantes y variables globales. Para informar al usuario sobre el rango de prioridades, una función de uso ("usage") también es definida aquí.

Este cliente muestra al usuario las opciones posibles y después le pide que introduzca peticiones, hasta que dicha petición sea la de terminar ("quit"). Tras ello, el cliente cierra el "socket" y termina. Las posibles peticiones son las siguientes:

1) Almacenar una canción nueva en la lista de reproducción

Éste es la petición más engorrosa ya que el cliente debe enviar toda la información necesaria para la creación de la estructura tipo *song* en el manager. El cliente continúa pidiendo canciones para almacenar hasta que el usuario introduce el carácter 'q', momento en el que se le pide al usuario que introduzca una nueva petición. Cuando el usuario ha introducido la canción a almacenar, el cliente pide que se introduzca a su vez la prioridad correspondiente a dicha canción y fija el número de orden en la lista ("songnumber") cuando es posible.

El cliente llama a una función para calcular el tipo de formato del archivo de audio. Si el nombre del archivo no incluye su extensión o es incluida, pero no reconocida (actualmente .mp3, .wav, y .au son las únicas extensiones usadas), un mensaje de error se le muestra al usuario y no se envía ninguna petición al manager.

Dependiendo del tipo de formato, se toman distintas acciones, todas con el mismo propósito: el cálculo del sample-rate, bitrate y del tamaño del archivo. En el caso de archivos wave, su tamaño en bytes se calcula y se lee parte de su cabecera ("format chunk") para determinar el número de muestras por segundo y el número de bytes por segundo. La duración (en milisegundos) se calcula con la siguiente fórmula:

```
song_length_ms=(file_size*1000)/bytepersecond
```

Sin embargo, si el archivo es un archivo mp3, el título, artista y el álbum de la canción son leídos primero. Seguidamente se calcula su tamaño en bytes y su sample-rate y bitrate se leen de su cabecera. Después de esto, la duración del archivo de audio se calcula. Para ello, la etiqueta TLEN, que contiene una secuencia numérica con la duración de la canción en milisegundos, se busca. Cuando se encuentra, el procesamiento del archivo MP3 se acaba. Sin embargo, no todos los archivos MP3 incluyen la etiqueta TLEN en su cabecera. Una manera alternativa de calcular la duración se utiliza en estos casos (usando el número de muestras por segundo y de bits por segundo). Si se trata de un archivo SUN au, se calcula su tamaño y su duración, usando 8000 muestras por segundo:

```
song_length_ms=(file_size*1000)/8000
```

Finalmente, una petición que contiene toda la información recopilada se envía al manager y el cliente queda esperando una confirmación de esta petición.

2) Conseguir la primera canción de la lista de reproducción

Una petición para conseguir información sobre el primer elemento de la lista de reproducción se envía al manager y se espera la respuesta del mismo.

3) Conseguir la canción enésima de la lista de reproducción

Se le pide al usuario el número de la canción cuya información solicita. Entonces, el mismo procedimiento que en la petición 2 se lleva a cabo.

4) Guardar la lista de reproducción en un archivo

El cliente hace que el manager guarde la lista de reproducción actual en un archivo. Se envía una petición y se espera una respuesta del manager.

5) Eliminar una canción de la lista de reproducción

Se le pide al usuario el nombre del archivo (songfile) a eliminar y una petición de eliminar este archivo de la lista de reproducción se envía al manager. Como anteriormente, se espera la respuesta del manager.

6) Terminar

Una petición de terminar se le envía al manager y se cierra el socket sin esperar respuesta alguna. Tras ésto, el programa de este cliente termina.

B.2.5 Descripción del Programa del Generador de Alertas

Este cliente es una modificación de un cliente creado previamente por Sean Wong [19]. Dichas modificaciones fueron necesarias para que el cliente de Sean pudiera comunicarse con el manager, puesto que su cliente y servidor se comunicaban vía TCP. Este cliente opera en el ordenador portátil y usa el mismo nombre del host y dirección IP que el cliente interfaz de usuario. Usando este cliente en la aplicación se proporcionan notificaciones de audio simples, que son un ejemplo de contenido de audio que se debe tratar de forma diferente a las canciones. El cliente utiliza la herramienta de conversión de texto a voz Festival-Lite [16] para la traducción a audio de las cadenas de caracteres que reciba.

Las modificaciones que fueron hechas son:

- Añadir el archivo de cabecera “playlist.h” y el programa “socket.c”.

- Cambiar el número de puerto del cliente de 50555 a 50558.
- Añadir las variables nombre, artista, álbum, etc., necesitadas para la futura construcción de la estructura *song*.
- Cambiar de TCP a UDP (se pasa a enviar y recibir datagramas).
- Añadir el cálculo de la duración del archivo en segundos.

Por defecto, las notificaciones de audio que resultan del uso de este cliente tienen su prioridad fijada a uno, lo cuál indica que la notificación será reproducida en el siguiente momento disponible (después de la canción que se está reproduciendo actualmente; observe que ésta ha sido eliminada ya de la lista de reproducción). Sin embargo, según lo explicado previamente, el manager inserta una notificación a la cabecera de la lista de reproducción solo si no hay otras notificaciones en la lista de reproducción ya. Cuando hay ya una o más, las notificaciones siguientes serán insertadas según su orden de llegada, siguiendo las notificaciones anteriores.

B.2.6 Descripción del Programa del Reproductor

Este cliente proporciona una manera de reproducir el contenido de audio almacenado en la lista de reproducción. El nombre de host usado es "badge41". La dirección IP que corresponde este host es 10.0.1.6, según lo indicado previamente para el manager.

Un archivo de cabecera, "player.h", utilizado solamente por este cliente, contiene todas las definiciones de constantes, variables y funciones necesarias. Algunas definiciones dentro de este archivo de cabecera merecen ser mencionadas. Las definiciones de las estructuras para los "chunks" de los archivos wave se incluyen aquí también. Estas estructuras son necesarias porque se desarrolló un reproductor para los archivos wave como parte de este cliente. Definiciones para las conversiones lineares PCM se incluyen también, junto con las funciones necesarias para las conversiones desde muestras 8 bit μlaw a muestras 16 bit lineares con signo o PCM. Finalmente, las funciones para fijar el número de muestras por segundo y abrir el dispositivo de audio también se incluyen en la cabecera.

El reproductor continuamente pide al manager información sobre el primer elemento de la lista de reproducción. Cuando el usuario le pide al manager que termine, el manager le pide a su vez al reproductor que termine, si no hay elementos en la lista de reproducción a reproducir. El funcionamiento de este cliente es el siguiente:

- Envía un datagrama al manager para pedirle el primer elemento de la lista de reproducción. Entonces, espera la respuesta del manager. Cuando ésta llega, el manager puede pedirle que acabe, decirle que no hay contenido de audio almacenado en la lista de reproducción o enviarle el primer elemento de la lista de reproducción. En este último caso, el reproductor extrae la información que el manager envió (que incluye el nombre del archivo a reproducir), comprueba si el tiempo actual es posterior al momento de reproducción ("time_to_play") y, si es así, el reproductor envía una petición al manager para eliminar el primer elemento. Como de costumbre, espera una respuesta del manager.

- Cuando todavía no existe audio en la lista de reproducción, el reproductor incrementa un contador y espera dos segundos antes de preguntar al manager otra vez.

- Cuando el reproductor debe reproducir un elemento de la lista, el primer parámetro comprobado es el tipo de formato de audio, pues diferentes tipos de archivos requieren un procesamiento diferente antes de ser reproducido. Si el archivo es mp3, un reproductor digital preexistente, mpg123, es utilizado, con una llamada de sistema simple ("mpg123/tmp/%s", songfile). Si no, para los archivos wave y AU, el reproductor procesa y reproduce directamente estos archivos, usando la misma función para abrir el dispositivo de audio del *"SmartBadge versión 4"* para ambos.

- En el caso de archivos wave, los "chunks" (llamados riff, format y data) se leen, ya que toda la información necesaria para reproducir correctamente un archivo wave se encuentra en la cabecera. Después de fijar el número de muestras por segundo, se lee el contenido de audio y se escribe en los "drivers" de audio del *"SmartBadge versión 4"*.

- En el caso de archivos AU, la función "sayfile", creada por Jef Poskanzer en 1990 y modificada por Gerald Maguire en 2003 para su adaptación al *"SmartBadge versión 4"*, se utiliza para reproducir estos archivos.

- Una vez reproducido el archivo, el reproductor pide otra vez al manager el primer elemento de la lista de reproducción, repitiendo este proceso hasta que el manager le dice que debe terminar.

B.2.7 Descripción de la Funciones en "socket.c"

Estas funciones son utilizadas por el servidor y todos los clientes para abrir un "socket" UDP para enviar y para recibir datagramas. Dentro de "socket.c" dos funciones para uso general, open_bind_UDP_socket y

setup_server_sockaddr_in_structure son definidas. La primera función recibe como parámetros el nombre del host y el número del puerto y abre un "socket" UDP usando el número del puerto y la dirección IP que corresponde al nombre del host dado. Después liga el "socket" a esa dirección IP. La función open_bind_UDP_socket comprueba que las operaciones realizadas estén hechas correctamente y devuelve el identificador del socket. La función setup_server_sockaddr_in_structure se utiliza para conseguir la dirección del servidor en el socket abierto.

B.2.8 Usar la Aplicación cliente/servidor

Para utilizar la aplicación, después de encender los dispositivos, establecer la red inalámbrica privada y montar el sistema de ficheros NFS, se debe compilar el código. Para ayudar en ésto, hay un archivo "make" que compila todos los programas incluidos en esta aplicación; simplemente introduciendo el comando *make all*. Aunque "manager.c" y "reproductor.c" se almacenan en el ordenador portátil, funcionan en SmartBadge 4, por lo que un compilador cruzado es necesario. La herramienta uClibc [32], instalada previamente en el ordenador portátil, fue la utilizada. Tanto el manager como el reproductor se ejecutan en el "background" del *"SmartBadge versión 4"* mediante las líneas de comando:

>> ./manager&
>> ./player&

El paso siguiente es poner a funcionar en el ordenador portátil a los otros dos clientes escribiendo los comandos en dos ventanas separadas:

>> ./user_interface
>> ./alert_generator "*cadena de caracteres*"

B.3. EVALUACIÓN DEL DISEÑO

Como el principal objetivo de este proyecto era realizar una aplicación de audio y utilizarla para estudiar cómo explotar el audio para los usuarios *"nómadas"*, una evaluación de la aplicación diseñada fue llevada a cabo.

B.3.1 Observaciones

Para poder evaluar la aplicación diseñada, un test práctico en el que se interactuara con todos los programas era necesario. La finalidad de este test es descubrir las características buenas de la aplicación y encontrar áreas que necesitaban ser mejoradas.

B.3.1.1 Características Buenas

Algunos de los puntos que fueron considerados positivos después de probar la aplicación son los siguientes:

- La aplicación se considera *fácil de usar*, ya que la mayoría del procesamiento se le oculta al usuario. El usuario no necesita saber cómo funciona el manager y el reproductor, puesto que ambos reciben la información que necesitan para su ejecución de otros programas y no del usuario. Ya que el cliente "interfaz de usuario" muestra al usuario las funciones disponibles, éste solo tiene que introducir un número para indicar el comando deseado y los datos solicitados (según el caso). Para tener un ejemplo de cómo las notificaciones escritas se pueden oír como notificaciones de audio, el usuario solo tiene que ejecutar el "generador de alertas" introduciendo en la línea de comandos el nombre del programa seguido de lo que desea escuchar entre paréntesis.

- Gracias al cliente "interfaz de usuario", éste puede *interactuar directamente* con el manager *en la construcción* y modificación *de la lista de reproducción*. Este hecho fue considerado importante en el test, puesto que el usuario puede controlar fácil y directamente la manera en la que se construye la lista de reproducción y, cuando lo desee, terminar al manager y al reproductor.

- El *sonido* digital que el CODEC de *"SmartBadge versión 4"* proporciona fue considerado de *alta calidad*, comparable con los lectores de MP3 o de CD.

- En el test, las ventajas de tener *notificaciones de audio* y usar *"SmartBadge versión 4"* como un dispositivo móvil de *sólo-audio* fueron apreciadas. Los dispositivos móviles de sólo-audio permiten que el usuario se mueva o tenga las manos libres, mientras que puede recibir notificaciones y escuchar música.

B.3.1.2 Características que Necesitan ser Mejoradas

Los aspectos de la aplicación que necesitaban ser mejorados son los siguientes:

- Cuando hay archivos MP3 incluidos en la lista de reproducción, la salida de estos archivos comienza después de un *retardo* apreciable. El cliente reproductor hace una llamada a *mpg123* y, después de eso, se aprecian dos retardos fijos. El primer retardo viene de la abertura de

/etc/sound/dsp y es de aproximadamente 15 segundos. El segundo retardo viene de la abertura del archivo en sí, tomando otros 15 segundos. Esta circunstancia se debe mejorar ya que, la mayoría de canciones digitales son archivos MP3, este retardo puede ser realmente molesto.

- *No todos* los tipos existentes de *archivos de audio* se consideran en la aplicación. Aunque dos de los tipos, archivos MP3 y wave, son los formatos más usados para almacenar música digital, el interfaz de usuario aceptará solamente archivos de los tipos considerados, restringiendo al usuario en los tipos de archivos de audio que puede escuchar con la aplicación actual. Algunos formatos adicionales podrían ser añadidos o SOX [37] se podría utilizar para conversión de tipos.

- En la aplicación diseñada, *no* se adquiere *información* alguna sobre el *entorno del usuario* como entrada para la aplicación. Por lo tanto, el dispositivo móvil no muestra ningún comportamiento inteligente. Sin embargo, la aplicación actual (junto con otros estudios realizados con *"SmartBadge versión 4"* [13], [18], [19]), se extiende fácilmente para aceptar información de contexto. Esto mejoraría el comportamiento del dispositivo móvil.

- Como el SmartBadge es solo un prototipo para la experimentación, *no* dispone de *monitor* ni *teclado* para mostrar e introducir datos. Esto restringe el uso y la movilidad deseados para esta aplicación, puesto que una conexión a un ordenador que sí disponga de monitor y teclado es necesaria para poder usar los clientes interfaz de usuario y generador de alertas.

- La *terminación* del cliente *interfaz de usuario* hace siempre que el *manager termine*. Esto era útil para la aplicación actual, puesto que era interesante tener una manera de terminar al manager y la terminación del interfaz de usuario fue considerada la mejor manera de hacerlo. Sin embargo, en una futura mejora de la aplicación en la que se agregaran más clientes, no es conveniente que el manager termine cuando el interfaz de usuario termina, puesto que habría otros clientes que estarían enviando datos todavía al manager.

- En toda la aplicación, cuando se envía una petición, se espera una respuesta, pero asumimos que no es posible la *pérdida* de esa *respuesta*. Por ejemplo, cuando el reproductor solicita al manager suprimir la canción que va a ser reproducida, espera la respuesta del manager. Pero si se pierde esa respuesta, el reproductor permanece esperando para

siempre y el archivo de audio no se reproducirá nunca. Por lo tanto, añadir *"timeouts"*, *reintentos* después de un tiempo establecido y *comprobaciones* de los datagramas recibidos es claramente conveniente para realzar la fiabilidad y el escalabilidad de la aplicación.

B.3.2 Nuevas Funciones

Después de experimentar con la aplicación, algunas ideas sobre nuevas funciones para mejorar el actual diseño surgieron. Las nuevas funciones que pueden contribuir a una mejora de la aplicación son las siguientes:

- Una función que permitiera usar *listas de reproducción previamente construidas.* El cliente interfaz de usuario tendría un comando adicional para seleccionar una lista de reproducción que ya existe. Las canciones en esa lista de reproducción serían copiadas al directorio temporal (/tmp) de *"SmartBadge versión 4"*, donde el manager y el reproductor puedan tener acceso a ellas. El manager tendría que comprobar si hay elementos en la lista y si es así agregaría la lista de reproducción después de los elementos existentes en la lista del manager. Otra posibilidad sería agregar una prioridad a la lista de reproducción que el usuario quisiera incluir para poder escucharla primero, aunque haya más elementos en la lista del manager.

- Otro posible comando a añadir (cuando se usaran listas de reproducción existentes) sería una opción de *shuffle*. El manager insertaría listas de reproducción nuevas en orden aleatorio. Sería similar a la opción de *shuffle* en muchos reproductores de MP3 y de CD.

- Una posible modificación del reproductor sería sustituir la función get_delete_first_song() por dos funciones: *get_first_song()*, *delete_song_played()*. El reproductor pediría que el manager eliminara el archivo de audio una vez que se haya reproducido y no antes. Así, el archivo de audio permanece en la lista de reproducción y se elimina solo cuando el manager sabe que ha sido reproducido. Esta modificación en el reproductor afecta al generador de alertas, puesto que la prioridad (fijada para reproducir la notificación en el siguiente momento disponible) tendría que ser cambiada para insertar la notificación después del primer elemento, que es el que se está reproduciendo todavía.

- Otra modificación que podría mejorar la aplicación sería extender el manager para que pueda aceptar peticiones para *reproducir* un archivo de audio *en* o *antes de un momento determinado.* La manera en la que el

interfaz de usuario y el generador de alertas envían información al manager tendría que cambiar, puesto que sería tiempo en vez de prioridad lo que determinaría cuando fijar el momento de salida de un archivo de audio. Una manera fácil de implementar ésto para el interfaz de usuario y el generador de alertas sería fijar el parámetro de prioridad a 0 cuando se requiere que el archivo de audio sea reproducido en un momento determinado. El manager entonces incluiría el archivo de audio en el lugar apropiado de la lista comparando el tiempo requerido con el momento de reproducción de los otros elementos de la lista. También tendría que comprobar que este archivo de audio está colocado en el lugar correcto cuando añade o elimina archivos nuevos de la lista de reproducción. Con este nuevo procedimiento, el generador de alertas también podría decir al manager si utiliza prioridad o tiempo para incluir un archivo en la lista de reproducción. Añadir la posibilidad de insertar un elemento *después* de un momento determinado sería sencillo una vez hecha la modificación recién expuesta.

- Una función que *eliminara más* de un archivo de audio a la vez. Esta función sería útil si el usuario quisiera, por ejemplo, eliminar todos los elementos de la lista de reproducción después el actual. También podría eliminar todos los archivos después del enésimo o eliminar un solo archivo pero en todas sus ocurrencias en la lista de reproducción.

- Una función para *detener* la *aplicación* sin tener que terminarla. Esto sería útil cuando el usuario no desea escuchar nada por un rato. Esta función necesita obviamente otra para *recomenzar* la aplicación otra vez, en el mismo punto en el que estaba cuando se detuvo, pero actualizando los tiempos.

B.3.3 Clientes Adicionales

Los clientes adicionales considerados útiles para mejorar la aplicación son los siguientes:

- Un cliente que proporcionara *reconocimiento de voz* y en concreto del *hablante* (“speaker recognition”), usando una de las herramientas ya existentes. Este cliente recibiría un archivo de voz como entrada y decidiría de qué usuario es, seleccionando de entre un grupo de usuarios cuyas voces han sido registradas. Entonces, pasaría esta información al manager y éste podría insertar una lista de reproducción ya construida por este usuario específico. Si mientras que usa una lista de reproducción, este cliente detecta un usuario distinto, el manager podría

cambiar la lista de reproducción para que tuviera ahora las selecciones del nuevo usuario.

- Un cliente de *reconocimiento de voz* y en concreto del *habla* ("speech recognition") . Este cliente esperaría órdenes y peticiones por la entrada de audio del dispositivo. El audio esperado incluiría al sistema de peticiones del cliente interfaz de usuario, como, por ejemplo, "introducir un archivo en la lista de reproducción" o "guardar la lista de reproducción". Este cliente reconocería estas elocuciones e iniciaría el procesamiento que interfaz de usuario debe hacer. Usando este cliente, la aplicación tendría un interfaz de usuario de sólo-audio, eliminando la necesidad de un cliente de interfaz de usuario basado en texto. La aplicación que resultaría sería una aplicación de sólo-audio. Por supuesto, el cliente interfaz de usuario podría permanecer en la aplicación en caso de que también se quisiera disponer de un interfaz visual.

- Un cliente dedicado a *capturar audio*. Este cliente estaría continuamente escuchando la entrada de audio del dispositivo y enviaría a los clientes de reconocimiento de voz el audio capturado para que éstos puedan realizar su procesamiento.

- Un cliente que proporcione información sobre el entorno del usuario, *información de contexto*. Este cliente podría decidir si el usuario está en una reunión o en su oficina, por ejemplo, analizando y evaluando la entrada de audio que la aplicación está recibiendo. Podría también decidir si conectarse a Internet vía uno de los interfaces disponibles o no conectarse si la lista de reproducción del usuario tiene ya bastantes elementos que no necesitan ser bajados de la red. El cliente de Sean Wong para proporcionar la *localización física del usuario* ("location awareness") [19] podría ser también agregado fácilmente a la aplicación actual.

- Si el dispositivo está conectado, entonces un cliente que *comprobara* la llegada de *e-mail* o de *SMS* podría ser útil. Este cliente podría enviar estos e-mails o SMSs al generador de alertas y el usuario podría oír estas notificaciones en vez de tener que leerlas en una pantalla. Al mismo tiempo, podría fijar la prioridad o el momento de reproducción a esos e-mails o SMSs dependiendo del remitente. La aplicación mantendría una base de datos en la cual el usuario especifica las prioridades deseadas para la gente con la que se comunica normalmente.

B.4. CONCLUSIONES

El objetivo de este proyecto era tratar los problemas que los usuarios afrontan al tener múltiples dispositivos, considerando las posibles mejoras expuestas por *"Nomadic Radio"*. Para explorar este campo en auge, un estudio de cómo utilizar y explotar el interfaz de audio de un dispositivo móvil, basándose en la síntesis texto-voz, el reconocimiento de voz y en una infraestructura sensible al contexto, fue realizado. Para poder tener un entorno real de prueba sobre el cual basar tal estudio, una aplicación cliente/servidor usando el protocolo UDP fue diseñada. *"SmartBadge versión 4"* fue el dispositivo móvil usado en esta aplicación, el cual se conectaba vía una WLAN privada con un ordenador portátil cuyo sistema operativo era linux. Este último ordenador actuaba como un servidor de contenido y en algunos casos proporcionaba un interfaz de texto a la aplicación.

En la aplicación, el manager construye, modifica y mantiene una lista de contenido de audio, la lista de reproducción, que incluye información de interés para el usuario (canciones, notificaciones de audio,...). El reproductor genera salida de audio seleccionando al reproductor apropiado y pasándole el archivo a reproducir. El cliente interfaz de usuario permite al usuario interactuar con el manager usando comandos textuales para construir la lista de reproducción. Este interfaz se habría podido construir como un interfaz web o gráfico, pero para los propósitos de este estudio era suficiente un interfaz textual simple. El último cliente se diseñó para generar notificaciones de audio usando la herramienta de conversión de texto a voz Festival-Lite [16]. Este cliente representa una de las muchas herramientas posibles que se podrían incluir, como SMS a audio, e-mail a audio y otras herramientas similares.

Después de implementar la aplicación, ésta fue probada y evaluada. La aplicación fue considerada realmente útil como base para el estudio de audio deseado para los usuarios "nómadas". Se destacaron algunas características buenas del diseño y otros aspectos donde la aplicación necesitaba ser mejorada fueron clarificados. Además, gracias al test realizado, ideas sobre nuevas funciones y clientes que mejorarían en gran medida la aplicación surgieron. Uno de los resultados más importantes de este proyecto fue el desarrollo de ejemplos concretos de cómo el audio se podría utilizar por un usuario "nómada". Es importante apuntar que la finalidad del actual proyecto era realizar un estudio de los cambios y mejoras potenciales para un usuario "nómada" y no desarrollar una implementación completa de una aplicación de audio que incluyera la plenitud de dichos cambios y mejoras.

B.5. POSIBLES AMPLIACIONES DEL PROYECTO

La solución presentada en este proyecto realiza un estudio de cómo utilizar y explotar el interfaz de audio de un dispositivo móvil. Este estudio proporciona una buena plataforma donde basarse para futuros proyectos en este campo. Posibles continuaciones del proyecto serían:

- Una continuación evidente sería tratar de *mejorar* todas las *características* en necesidad de mejora e *implementar* las *funciones y los clientes adicionales*, pudiendo ser ésto la base de un futuro proyecto puesto que el trabajo a realizar es bastante extenso y el resultado sería muy apreciado, según lo concluido en el actual proyecto.

- Combinar el trabajo hecho en este proyecto con el trabajo hecho por Giulio Mola [18] sería muy conveniente, puesto que se obtendrían *ahorros de energía* significativos. Teniendo en cuenta que otra mejora lógica de la presente aplicación sería incluir archivos de audio en la lista de reproducción que necesitan ser descargados al dispositivo móvil, el manager de audio tendría que pasar información al manager de interfaz de red para obtener un comportamiento correcto de la aplicación. Este último manager puede utilizar esta información para decidir si se necesita o no conexión en el momento actual (la lista de reproducción puede tener bastante contenido a reproducir sin necesidad de descargar canciones), y si es necesitada, el manager puede decidir cuando será necesaria (que debe ser con bastante anterioridad para tener el archivo descargado totalmente cuando le toque ser reproducido) y cuál de los interfaces disponibles usar (dependiendo de la anchura de banda necesitada).

- Este proyecto no se ha centrado en ningún aspecto de *seguridad* que debería haber sido tomado en cuenta. Esto abre una continuación de la aplicación que maneje la seguridad. En una implementación real de la aplicación, en la que el manager estaría siempre recibiendo datagramas UDP desde cualquier fuente que quiera enviarlos, una autentificación de que los comandos recibidos son legítimos sería necesaria. Una buena manera de hacer esto es tener un mensaje "hash" firmado (llamado HMAC). Ya existen estudios en este campo que se podrían agregar a la actual aplicación [38].

- Un campo de gran interés para mejorar la aplicación actual es la introducción del *audio especializado* ("spatial audio") , el cuál permite que la aplicación localice fuentes de audio como si estuvieran en lugares distintos en el espacio. Esto hace que el manager pueda superponer la

reproducción de diversas fuentes de audio al mismo tiempo, mientras que el usuario puede separar mentalmente estas diversas fuentes (mediante el efecto llamado "cocktail party"). Es posible implementar el audio espacial con el SmartBadge ya que tiene salida de audio estéreo de alta calidad.

- Otra posible continuación sería basarse en el trabajo hecho en *sensibilidad al contexto* ("context-awareness"), puesto que esta información de contexto adicional podría permitir a la aplicación tomar decisiones mejores sobre qué reproducir y cuando. Además, la generación de alarmas de audio basadas en la localización del usuario (según lo descrito ya en el informe de Sean Wong) proporciona un servicio que es aplicable al usuario nómada que quiere saber donde está, qué está situado cerca de él, etc.

- Según lo indicado previamente, una mejora podría ser incluir XML para tener un método alternativo de implementación de la lista de reproducción, manipulando el documento XML, por ejemplo usando la aplicaciones XSLT ("Extensible Stylesheet Language Transformations") [35], DOM ("Document Object Model") [33] o SAX ("Simple API for XML") [34].