PROYECTO FIN DE MÁSTER

POSGRADO EN INFORMÁTICA

# UNIVERSIDAD DE ALMERIA

ESCUELA POLITÉCNICA SUPERIOR

MÁSTER EN INFORMÁTICA INDUSTRIAL

# "Diseño, desarrollo e implementación de un sistema de publicidad empotrado, de hardware y software libre"


**Curso 2012/2013**

**Alumno/a:**

Joaquín Cuéllar Padilla

**Director/es:**

Dra. Nuria Novas Castellano

Dr. José Antonio Gázquez Parra

UNIVERSIDAD DE ALMERÍA
ESCUELA POLITÉCNICA SUPERIOR
Y
FACULTAD DE CIENCIAS EXPERIMENTALES
Departamento de Ingeniería


PROYECTO FIN DE MÁSTER
MÁSTER EN INFORMÁTICA INDUSTRIAL
POSGRADO EN INFORMÁTICA

# DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE PUBLICIDAD EMPOTRADO, DE HARDWARE Y SOFTWARE LIBRE

Joaquín Cuéllar Padilla

Dirigida por:

Dra. Nuria Novas Castellano

Dr. José Antonio Gázquez Parra

Almería, Septiembre 2013

PROYECTO DE FIN DE MÁSTER
MÁSTER EN INFORMÁTICA INDUSTRIAL
POSGRADO EN INFORMÁTICA


# DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE PUBLICIDAD EMPOTRADO, DE HARDWARE Y SOFTWARE LIBRE

por

**Joaquín Cuéllar Padilla**

Para la obtención del

**Título del Máster en Informática Industrial**

**Posgrado en Informática**

| **Directora** | **Director** | **Autor** |
| --- | --- | --- |
| Dra. Nuria Novas Castellano | Dr. José Antonio Gázquez Parra | Joaquín Cuéllar Padilla |

# Agradecimientos

Quiero agradecer este trabajo primero a mi familia, estoy orgulloso de ser el socio 5º de este club selecto de 6 miembros. Espero que crezcamos pronto, tardemos en menguar y seamos tan felices y nos apoyemos tanto como hasta ahora.

En segundo lugar agradecer todo el apoyo recibido y la amistad a mi segunda familia, que es mi casa, La Pimentera, los que estamos, los que fuimos y los que seremos, no me olvido de nadie, y es que no paro de aprender (e intentar mejorar el mundo) con vosotros.

He hecho también mi trabajo siempre con mi amorcito presente, Ana, y es que así da gusto *frikear*, con una persona tan inteligente, cariñosa, preciosa, y maravillosa, que hace que los días oscuros se iluminen, y convierte la ansiedad en risas.

Agradecer a mi jefe, Pedro, creador y capitán del barco Prointegra S.L. Que es un hacker con todas las de la ley, y que siempre está dispuesto a enseñarme, y con el que he compartido (y espero compartir) numerosos proyectos.

A mis directores de proyecto José Antonio Gázquez y Nuria Novas, y a los profesores José Luis Guzmán y Leocadio G. Casado por la ayuda suministrada, y el valioso conocimiento que han querido compartir conmigo, y su profesionalidad.

Y finalmente también a todos mis amigos almerienses, tanto del máster como Javi, Mauricio y Luis, como ajenos a él, como Pau, Manu, Carmen, y Victor. Habéis hecho que el año que pasé me sintiera en familia.

*“I don't have a problem with someone using their talents to become successful, I just don't think the highest calling is success. Things like freedom and the expansion of knowledge are beyond success, beyond the personal. Personal success is not wrong, but it is limited in importance, and once you have enough of it it is a shame to keep striving for that, instead of for truth, beauty, or justice.”*

Richard Stallman

# Índice

# Índice de Figuras

# Índice de tablas

# DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE PUBLICIDAD EMPOTRADO, DE HARDWARE Y SOFTWARE LIBRE

**Joaquín Cuéllar Padilla.**

Máster en Informática Industrial

Posgrado en Informática

Escuela Politécnica Superior y Facultad de Ciencias Experimentales,

Universidad de Almería.

***Abstract*** One of the biggest problems a national medium size enterprise faces, in the present economic and technological context, is: to depend too much on foreign technologies; which on the other side means that there is a lack of resources to research on own technologies. This problem results in a position of dependence, either towards multinational enterprises and abusive subcontracting agreements or as a simple distributor of products on which, small enterprises have no control, towards the international market instability.

In this context, we developed the present project, based on embedded systems, and free hardware and software. Using a free hardware embedded system, one of the cheapest ones in the market, the Rasperry Pi, we tried to facilitate the I+D needed profiting from a strong and low cost embedded system.

We have designed a publicity system based on two programs: a configuration program and a publicity monitor program. The device is presented in a small case attachable to any screen.

***Resumen*** Uno de los grandes problemas a los que se presenta una empresa nacional de tamaño medio, en el contexto económico y tecnológico actual, es el de acabar dependiendo en demasía de tecnología extranjera, carecer del dinero necesario para poder investigar tecnología propia y acabar así sobreviviendo o mediante contratos abusivos de subcontratación a cargo de multinacionales, o como mero distribuidor a merced de los caprichos del mercado internacional en productos de los que no se tiene control alguno.

En este marco surge la idea del proyecto que se muestra, cuyos pilares principales son los sistemas empotrados, y el hardware

y software libre. Mediante un sistema empotrado de Hardware libre de los más económicos del mercado, el Raspberry Pi, se pretende aliviar la carga de I+D necesaria para un desarrollo ambicioso y aprovechar así la potencia y flexibilidad de un sistema embebido potente y de bajo coste.

Se ha diseñado un sistema de publicidad compuesto por un programa de configuración de la presentación y otro de visualización de la misma. El dispositivo se inserta en una caja cerrada capaz de poderse conectar en el monitor que se desee.

***Index terms--*** Embedded system, Free as in freedom software, Free as in freedom Hardware, GNU/Linux, ARM architecture.

***Palabras clave--*** Sistema empotrado, software libre, Hardware libre, GNU/Linux, arquitectura ARM.

## I. INTRODUCCIÓN

Vivimos actualmente una expansión y proliferación trepidante de los sistemas empotrados microcomputadores, sistemas que concentran en una sola PCB (*Printed Circuit Board*) la potencialidad de un PC (*Personal Computer*).

Éstos, con una gran potencia en muy reducido espacio han sido introducidos en móviles de última generación, tabletas electrónicas, libros electrónicos, televisiones, aplicaciones médicas, etc. Abarcando además infinidad de arquitecturas, incluyendo X86, VIA, ARM (*Acorn RISC Machine*),etc. [1-2].

Esta expansión se ha podido llevar a cabo gracias a modelos nuevos de mercado como los llevado a cabo por la compañía *ARM Holdings* [3], invirtiendo en investigación y cediendo la producción de sus *chipsets* a terceros mediante concesiones comerciales, y el mundo del hardware y el software libre, que ha facilitado el proceso abaratando costes de investigación y mejorando el flujo de conocimiento entre desarrolladores. Este proceso es muy notable en la cantidad de comunidades que han ido emergiendo en los últimos 10 años, formadas tanto por profesionales como por principiantes, que motivados por plataformas de bajo coste, estándares abiertos y la facilidad de intercambio de conocimiento que ofrece Internet, están compartiendo día a día proyectos novedosos y dándole el impulso a este tipo de plataformas, en lo que a investigación y desarrollo se refiere.

En este contexto comercial no es de extrañar la proliferación de sistemas de la arquitectura ARM ya mencionada, que poco a poco han ido ocupado un nicho de mercado históricamente de otras

compañías como Intel, AMD (*Advanced Micro Devices*) [4] o incluso Nvidia en plataformas de muy bajo coste y compatibles con sistemas Linux tipo Android o GNU/Linux (*GNU is Not Unix*) [3, 5-6].

El proyecto que se describe a continuación ha consistido en un estudio de opciones comerciales de sistemas embebidos microcomputadores, para el desarrollo de una plataforma de hardware y software libre utilizable como producto local de venta a pequeños y medianos comerciantes. El diseño consiste en un dispositivo de reducido precio, fácilmente transportable, ligero y sumamente sencillo, para la proyección de publicidad e información. Con el objetivo de economizar, y simplificar las plataformas de proyección de publicidad actuales, tales como ordenadores completos, reproductores de DVD, etc.

Con estas premisas se ideó, diseñó y construyó un sistema empotrado pequeño, basado en el sistema microcomputador Raspberry Pi de arquitectura ARM [1], que viene ya de serie con un sistema GNU/Linux basado en Debian [7], al cual se preparó para poder utilizar las librerías libres GTK (*GIMP Tool Kit*) [8] para la creación de las interfaces gráficas de la aplicaciones a desarrollar, Imagemagick, que es un conjunto de utilidades de código abierto para mostrar, manipular y convertir imágenes, capaz de leer y escribir más de 100 formatos [9], y GStreamer [10], framework multimedia multiplataforma de código abierto, para crear aplicaciones audiovisuales. En las cuales se basó el software de visualización, y que se ha pulido para tener un funcionamiento más suave y sencillo con características tales como auto-arranque, configuración personalizada del servidor X, ahorro de recursos, etc.

## II. SISTEMAS EMPOTRADOS COMERCIALES

El mercado de los microcomputadores ha experimentado una expansión exponencial en los últimos años, esto se ha debido a la expansión de dispositivos de pequeño tamaño tales como *smartphones, tablets, e-books*, etc. Unido al desarrollo en paralelo de los microprocesadores ARM de bajo coste y consumo con alto rendimiento [1].

A su vez al ser todas estas plataformas de arquitectura ARM [1] (o VIA) basan sus sistemas operativos en el núcleo linux, dejándonos la posibilidad de usar sistemas operativos tipo Android o sistemas operativos tipo GNU/Linux tales como Debian, Gentoo, etc. La utilización de estos sistemas operativos da flexibilidad, fiabilidad y seguridad tanto para configurar como en la programación de estos equipos, permitiendo un total control del dispositivo.

Antes de optar por una plataforma empotrada concreta para el desarrollo del proyecto es necesario establecer unos requisitos previos para su selección, entre estos se consideraron principalmente los siguientes:

- Hardware libre para poder modificar el diseño de futuros proyectos y tener control absoluto dela plataforma.
- Una amplia comunidad para obtener así mejor soporte.
- Una plataforma potente.
- Una plataforma de reducido coste,
- Capacidad de expansión mediante buses interesantes tales como el GPIO (*General Purpose Input/Output*) o el I2C (*Inter-Integrated Circuit*).

Se estudiaron varios sistemas comerciales entre los que se analizaron los siguientes:

## A. Beagleboard

La placa Beagleboard (véase Figura 1) [11] es de los diseños pioneros en lo referente a los sistemas computadores de reducido tamaño, tiene ya años de desarrollo y testeo, así como una comunidad potente, alentada por todo el software y hardware libre que rodea a los diseños realizados con este tipo de dispositivo.

A su vez es un dispositivo de dimensiones: 85x85 mm, interesante gracias a la amplia capacidad de expansión y actualización que tiene, principalmente por su puerto I2C, que abre así un amplio abanico de posibilidades de desarrollo. Como desventaja de este dispositivo cabría mencionar que es caro (alrededor de 100 euros) para la potencia que ofrece, pues es de reducida RAM (*Random Access Memory*) y velocidad de procesamiento.

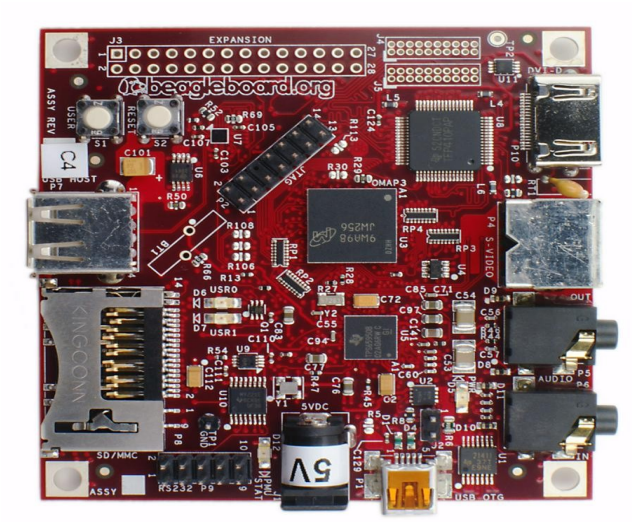


Figura 1. Detalle de la Beagleboard.

## B. Hackberry

La placa Hackberry (véase Figura 2) es un ordenador de dimensiones 85.60 x 53.98 mm [12]. Es un dispositivo muy potente, proveniente de desarrolladores Chinos de tablets y sistemas de televisión. Con un microprocesador de 1.2 GHz y una RAM que se puede obtener de 1Gb de tamaño.

Entre sus desventajas podemos considerar su precio (aproximadamente 65 $


Figura 2. Detalle de la Hackberry.

dependiendo de las prestaciones) y carece de puerto I2C o GPIO. A su vez, no disfruta de una comunidad activa como otros diseños más populares.

## C. PandaBoard

La placa Pandaboard (véase Figura 3) es otra de las grandes dentro del mundo de las placas de microcomputadores del mundo del hardware y software libre, posee un modelo básico y otro avanzado, una comunidad potente que da soporte, actualiza, y comparte información y desarrollos [13]. Su *hardware* es avanzado con un procesador de doble núcleo ARM 9 [1], y un 1 Gb de RAM, aunque con unas dimensiones posiblemente excesivas para la aplicación a desarrollar, 101.6 x 114.3 mm.

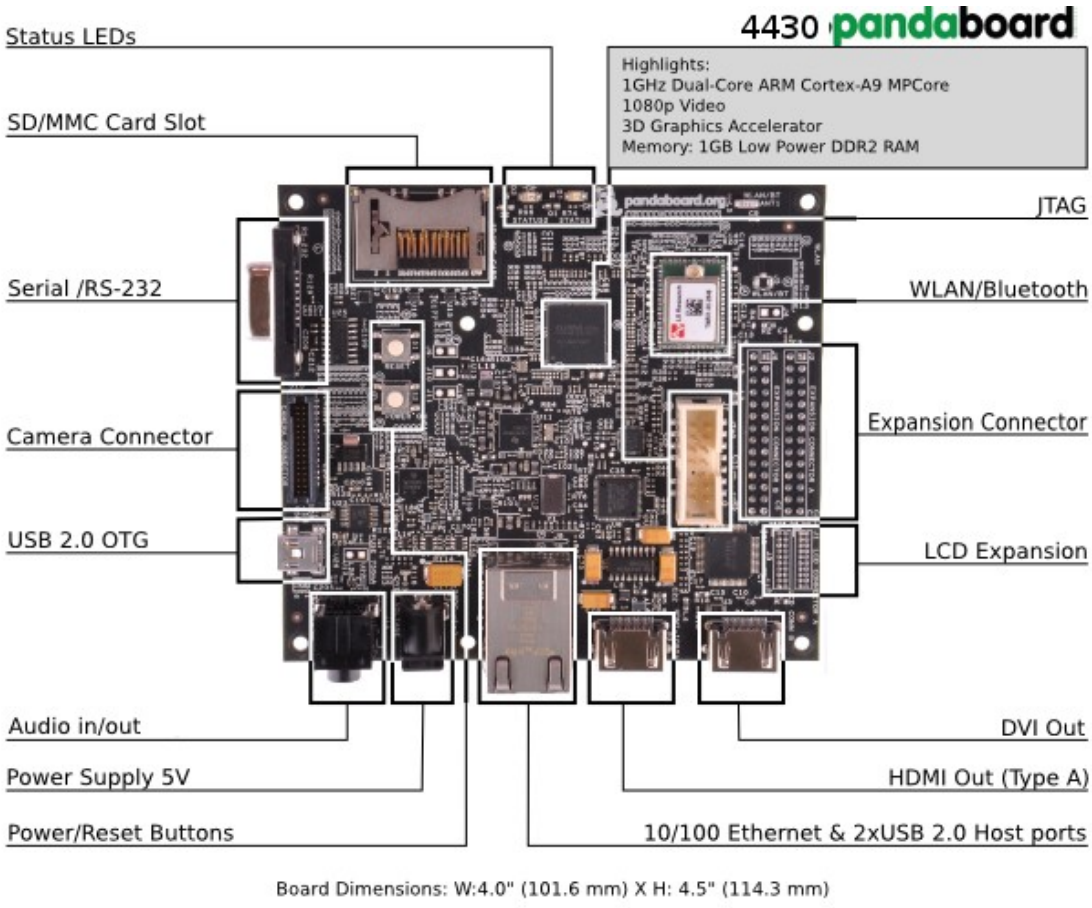


Figura 3. Detalle de la Pandaboard.

Aunque su principal desventaja es el precio (170 $). Entre sus ventajas se consideran su documentación amplia y actualizada, disfruta de una comunidad muy activa y gran soporte, posee además WiFi (*Wireless Fidelity*) integrado y más capacidades que el Raspberry Pi.

### D. *Raspberry P*i

La placa Raspberry Pi (véase Figura 4) es en sí un dispositivo especial, dispone de un hardware muy potente con un procesador ARM v11 [1] y 512 Mb de RAM y unas dimensiones de 85.6 x 56 mm. Con la potencia adicional de una GPU (*Graphics Processing Unit*) modelo Broadcom VideoCore4 de altas prestaciones [14].
Entre sus ventajas se considera su precio que ronda los 30 euros.

Figura 4. Detalle del Raspberry Pi.

Aunque carece de muchos extras, tales como Wifi, salida DVI (*Digital Visual Interface*), Bluetooth, entre otros. Pero tiene la posibilidad del I2C para adherirle cualquier diseño creado de manera fácil y eficaz.
A su vez posee una comunidad muy amplia, y gran soporte. Habiendo ya librerías muy maduras, avanzadas y documentadas para casi todo (I2C, python, etc.).

### *E. Elección del dispositivo empotrado*

Centrando la elección en los requerimientos del proyecto a desarrollar y considerando los dispositivos analizados en los apartados anteriores se ha optado por utilizar la Raspberry Pi. La elección de este dispositivo se ha visto condicionada por futuros productos, al final, desarrollando en este tipo de plataformas también se busca a efectos prácticos sus posibilidades futuras, y expansibilidad. Al ser un dispositivo de Hardware libre se tiene la certeza de que siempre habrá un proveedor para garantizar la adquisición de nuevos equipos. Además, la sencillez del dispositivo garantiza más fiabilidad.

## III. MATERIALES Y MÉTODOS

### A. Descripción del sistema hardware

#### 1. Raspberry Pi

La PCB Raspberry Pi fue creada a

principios del año 2012 en Gran Bretaña, a través de la fundación sin ánimo de lucro Raspberry Pi Foundation. Su objetivo es el de crear una plataforma con fines educativos muy expansionable y flexible a muy bajo coste [14].

El modelo del proyecto es el más potente de los dos ofertados en un principio por la fundación. Entre sus características podemos encontrar:

- El procesador principal pertenece a la familia ARM v11 [1], modelo ARM1176JZF-S a 700 MHz de frecuencia de reloj.
- El procesador gráfico es el Broadcom VideoCore IV a 250 MHz de frecuencia de reloj.
- Posee 512 Mb de RAM, compartida con la GPU.
- Dos puertos USB (*Universal Serial Bus*).
- Conectividad de video mediante HDMI (*High Definition Multimedia Interface*) y RCA (*Radio Corporation of America*) y DSI (*Display Serial Interface*).
- Memoria de almacenaje SD (*Secure Digital*).
- Puertos GPIO e I2C.
- Salida de audio, tanto mediante el HDMI antes comentado, como mediante su salida de tipo Jack de 3.5 mm.
- Alimentación de 700 mA (3.5 W).

## 2. Adaptador de corriente

La primera necesidad que surge a la hora de encapsular el dispositivo empotrado es la de unificar la alimentación a 220 V en alterna.

Esta decisión se toma con la idea de futuros proyectos con el mismo dispositivo. Sería interesante desvincularse de la alimentación por USB a 5 V, para tener otra mediante un transformador estándar de 220 V en alterna a 9 V en continua. Permitiendo una alimentación más común así, y más fácilmente reemplazable.

El Raspberry Pi también adolece de un consumo excesivo de intensidad, por lo que se garantiza corrientes máximas de 1.5 A. El circuito adaptador de la señal continua de 9 V a 5 V se realiza con el chip TSR 1-2450 (véase Figura 5).

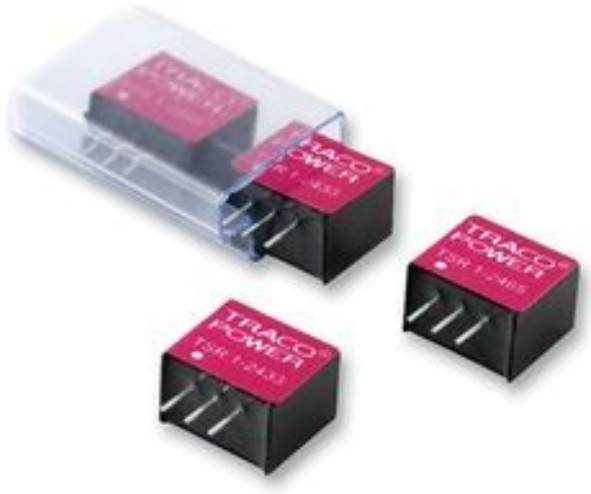


Figura 5. TSR 1-2450 De Traco Power.

Y su circuito adaptador (véase Figura 6) es sencillo, basado en un par de condensadores y una bobina, muy bien

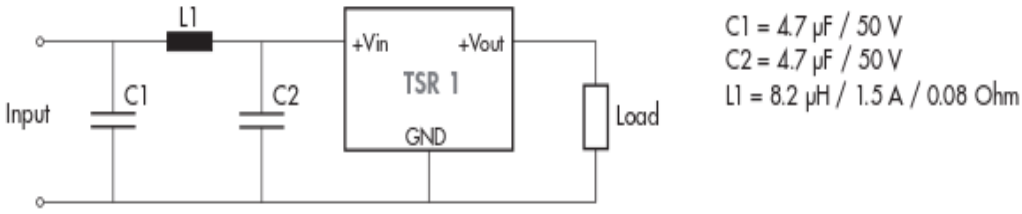


Figura 6: Detalle del circuito adaptador con el TSR 1-2450.

documentados dentro del manual de usuario del *chipset*.

Para la creación de la PCB se uso una placa perforada con las denominadas comúnmente islas de cobre, se insertaron así en ella los componentes y se fijaron las pistas mediante uniones de estaño soldado, o con porciones de cable. Siendo este método muy rápido y eficaz para pequeños diseños de prototipos.

## 3. Conjunto final

Finalmente se tiene un conjunto formado por la PCB del Raspberry Pi junto su tarjeta SD más un adaptador de corriente al que se le ha agregado un conector I2C para su fácil adhesión a la placa de la CPU (*Central Process Unit*).

La parte del encapsulado es la que posiblemente lleve más controversia, ya que se subestimó en un principio su importancia y coste, y se fue haciendo patente con el tiempo el error.

En un principio se barajó usar una de las múltiples opciones que hay en el mercado (véase Figura 7).

Esa decisión ofrecía ventajas e inconvenientes. Por un lado ofrecía una solución industrial al problema de la presentación del producto, con un acabado fino. Pero impedía unir dentro de la caja por falta de espacio el circuito adaptador eléctrico, y a su vez el precio de la misma disparaba el precio final del producto.

Se decidió la compra de una caja genérica de 140 x 80 x 32 mm. a un fabricante local de envases de plástico, con un precio 10 veces inferior al de las soluciones encontradas en Internet para el Raspberry Pi.

Figura 7. Ejemplo de caja para el Raspberry Pi.

A este envase se le hizo después el mecanizado y adaptación para el almacenaje de las dos placas creadas con

os huecos necesarios para los conectores. Y con la visión futura de poder agregar botones al envase (*reset*, encendido, botones de acción, etc.).

El resultado final (véase Figuras 8, 9, 10 y 11) no fue del todo profesional, y se quedó patente la pérdida de tiempo y la dificultad de enfrentarse a un desarrollo mecánico. Aunque para un prototipo inicial se da por exitoso, considerando modificar su envase en el prototipo comercial.

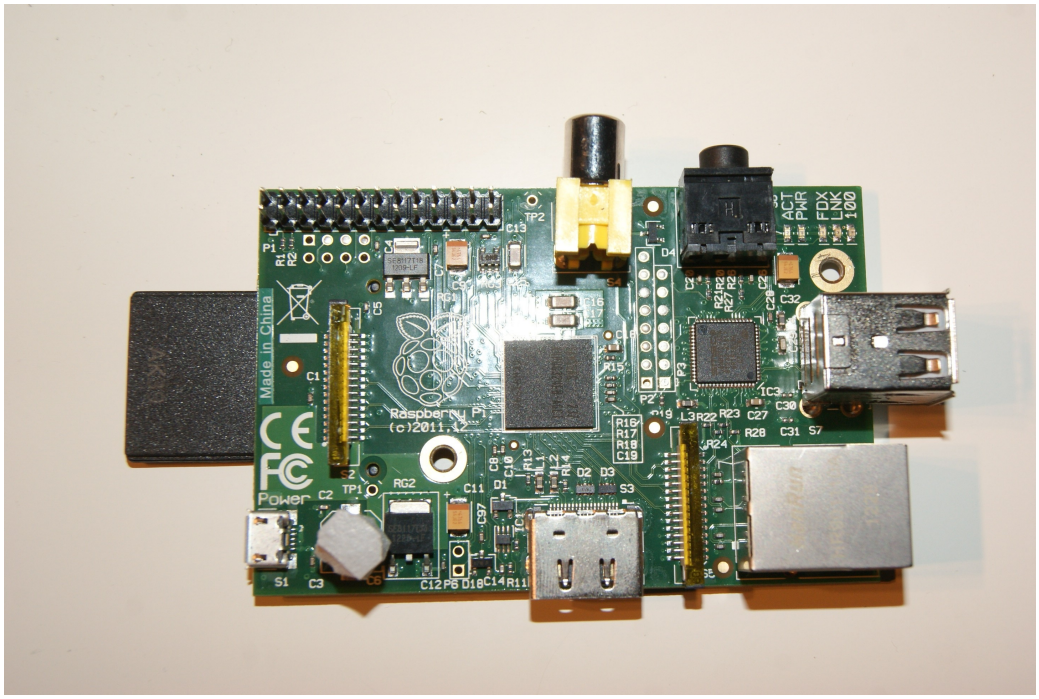


Figura 9: PCB del Raspberry Pi del proyecto.

Figura 8: Detalle del montaje final.

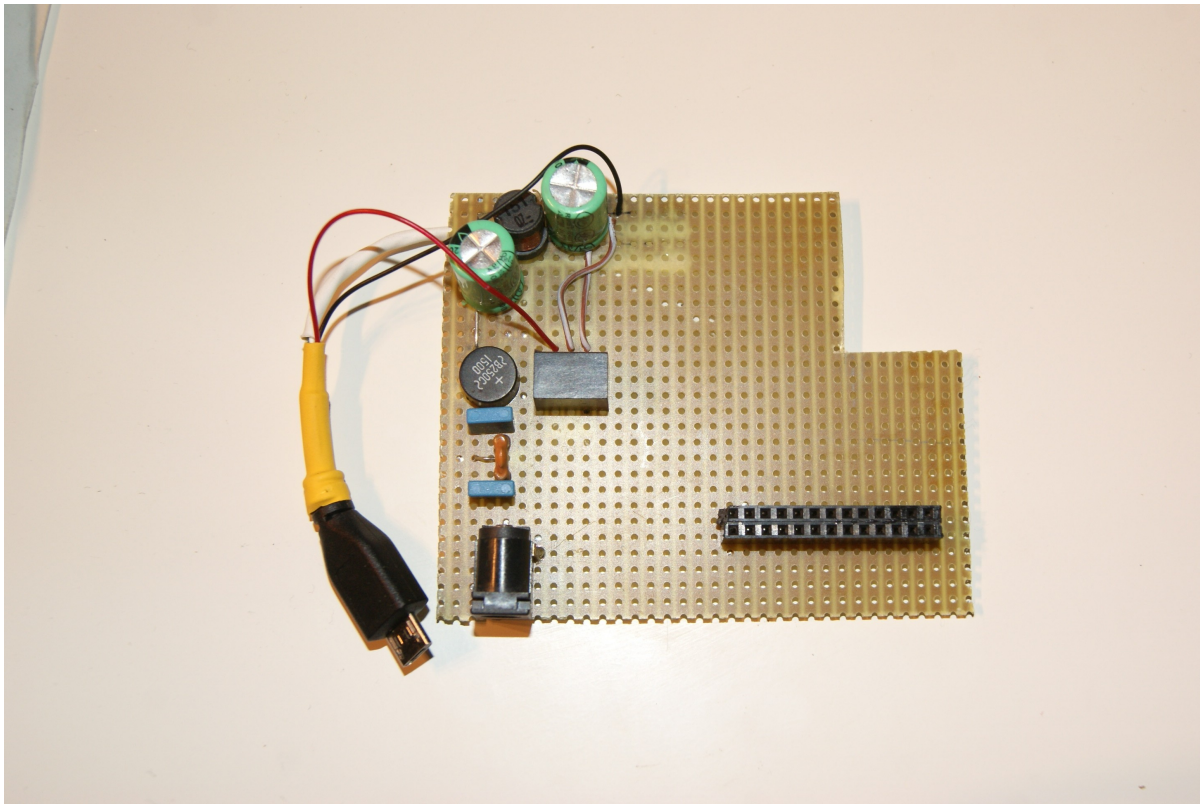

Figura 10. Detalle de la placa construida para suministrar la alimentación requerida por la placa del Raspberry Pi.

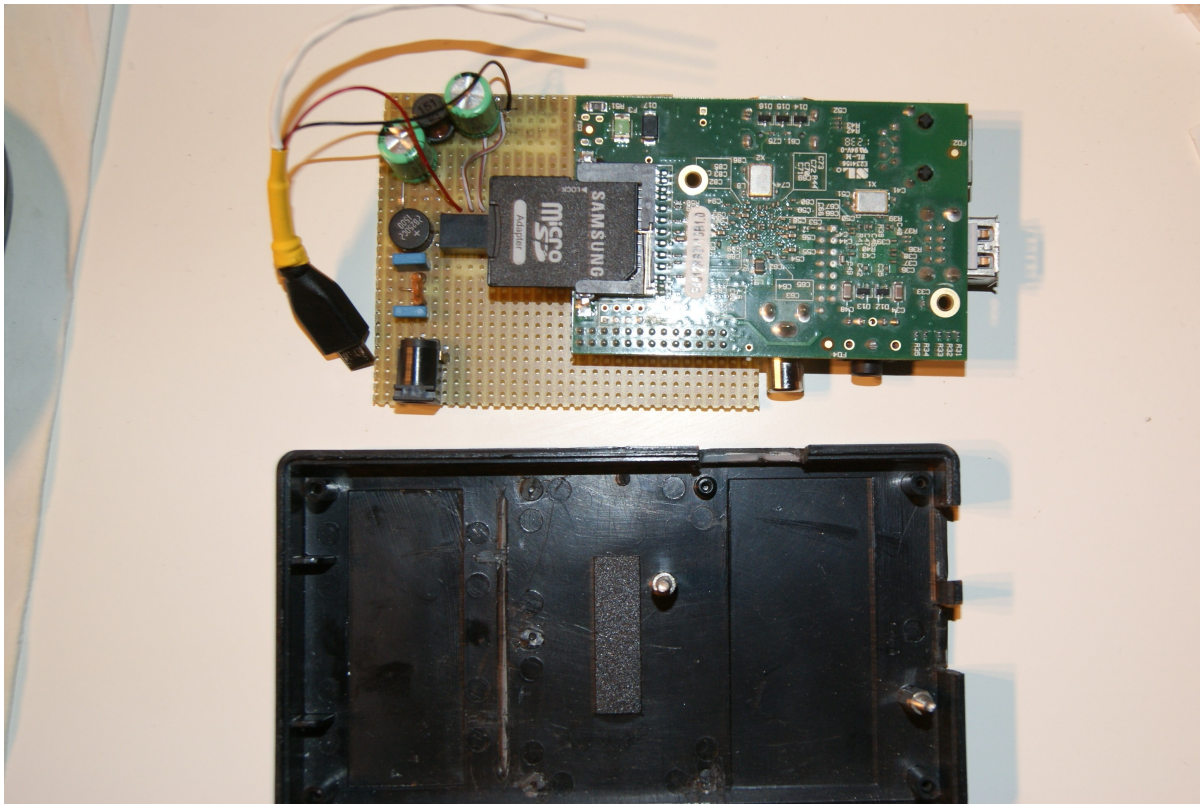


Figura 11. Detalle arriba del ensamblaje de las dos PCBs utilizadas en el desarrollo. Abajo, detalle de la caja ya mecanizada.

## B. Descripción del sistema software

La elección del lenguaje de programación, su entorno, y sus herramientas y librerías se ha visto siempre supeditada al hecho de ser software libre, unas elecciones están centradas por su potencia y facilidad de programación y otras por su adaptación a la aplicación y dispositivo a desarrollar.

Finalmente se eligió entre las distintas opciones:

- El lenguaje C.
- El entorno de desarrollo Emacs.
- Para la creación del entorno gráfico, GTK (*Gimp Tool Kit*).
- Para el tratamiento multimedia GStreamer e Imagemagick.
- Para el tratamiento de datos de configuración Libxml2.

### 1. Entorno de programación, Emacs y lenguaje C

El entorno de programación elegido es extremadamente simple, sencillo y potente a la vez, decantándose por el programa libre Emacs [15] y tener un control manual del archivo de configuración de la compilación (*makefile*).

Emacs es un sencillo editor de texto extensible para la programación en C.

Así, se evita el uso de pesados entornos de desarrollo IDEs (*Integrated Development Environment*) que pueden ralentizar el proceso de testeo, compilación entre otros dentro del Raspberry Pi, ya que de esta forma no hay necesidad alguna de usar algún método especial como la compilación cruzada, y va a ser dentro del mismo sistema empotrado donde se trabaje en el desarrollo.

El compilador elegido es el potente y maduro GCC (*GNU Compiler Collection*), pudiéndose apoyar en un momento determinado en su debugger, GDB (*GNU Debugger*).

Una vez compilado el programa de configuración en su versión GNU/Linux, este se exportó a Windows inmediatamente mediante el entorno Codeblocks y el simil de GCC en los sistemas operativos de Microsoft, MinGW [6].

La elección del lenguaje C para este proyecto se debe al conocimiento previo que se posee, por lo estándar que es, su potencia y eficiencia al compilarse para la arquitectura, y lo extendido que está su uso y lo bien documentado donde todas las librerías que se han considerado necesarias existen para este lenguaje.

Una opción viable también hubiera sido Python. La comunidad del Raspberry Pi está desarrollando mucho software en Python, pero su implementación en el proyecto conllevaría tiempo extra para un correcto manejo de dicho lenguaje de programación.

## 2. Librerías utilizadas

Los dos programas desarrollados se basan en librerías para facilitar la inclusión de características especiales (véase Figura 12).

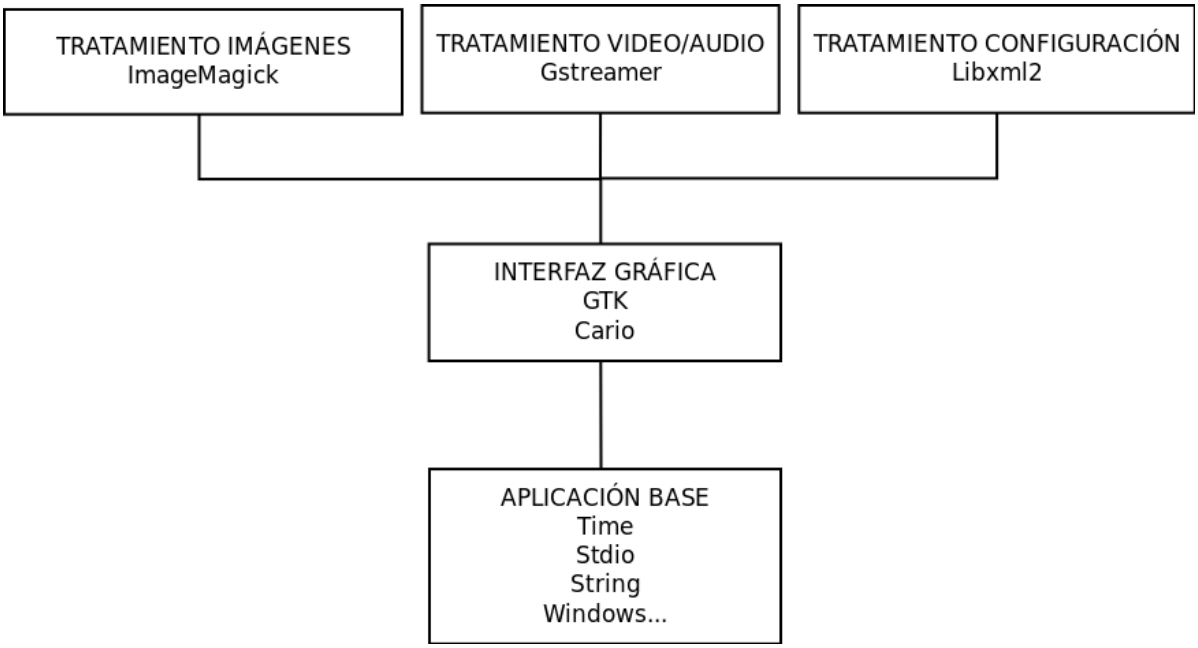


Figura 12. Resumen de librerías utilizadas.

Las librerías utilizadas son:

### a. GTK (*Gimp Tool Kit*)

Debido a la evidente necesidad de que la aplicación se ejecute de forma gráfica, se decide basarse en una de las numerosas librerías de creación de sistemas de ventanas que hay.

Se elige así GTK , la elección se debe a que se deseaba una misma librería tanto para el programa de configuración como el de proyección de la presentación publicitaria, que ejecutara de forma nativa en GNU/Linux (sistema operativo del sistema empotrado).

Esta debía ser compatible con otros sistemas operativos tales como Microsoft Windows, para que el programa de configuración pudiera ejecutarse en dichas plataformas también.

A su vez, GTK es la librería estándar de la GNU, y es la base de programas de software libre potentes como GNOME o GIMP (*GNU Image Manipulation Program*), hecho que da seguridad de su mantenimiento y desarrollo continuo, así como el soporte de una comunidad vasta de desarrolladores.

**b. Libxml2**

Libxml2 es el conjunto de herramientas desarrolladas por el proyecto GNOME, que forma parte de la fundación GNU [16], y consiste en el desarrollo de un entorno de escritorio completo, posiblemente el más usado en entornos GNU/Linux, para analizar y construir estructuras XML.

Esta librería está programada en C, y se ha usado en el proyecto para el almacenaje de parámetros.

Es más cómodo construir archivos *.xml* que en texto plano, ya que permite la estandarización de la introducción y salida de datos a través de etiquetas, lo que conlleva una aceleración del proceso.

**c. ImageMagick**

ImageMagick es un conjunto de herramientas informáticas (junto con una librería en C), que permite entre otras muchas cosas: editar, componer, y convertir imágenes de mapas de bits.

Puede a su vez trabajar con más de 100 formatos diferentes de imágenes incluyendo DPX, EXR, GIF, JPEG, JPEG-2000, PDF, PhotoCD, PNG, Postscript, SVG, TIFF, etc.

Permite ajustar el tamaño, rotar, espejar, y transformar imágenes, ajustar sus colores, etc.

Esta librería será de gran utilidad en la aplicación a la que este proyecto hace referencia, ya que permitirá homogeneizar las imágenes usadas en el mismo. Así será introducida en un conjunto de herramientas y funciones del programa que primero llevarán las imágenes a un formato conocido y presentable por la librería gráfica GTK, y segundo ajustarán sus tamaños al tamaño de presentación.

**d. GStreamer**

GStreamer es una librería para el tratamiento multimedia, donde el objetivo es construir una capa de abstracción en la reproducción de audio y vídeo.

Esta librería será usada en el programa tanto para el acondicionamiento de los archivos multimedia subidos por el usuario a formatos conocidos, como para su posterior reproducción.

# IV. DESARROLLO DE LA APLICACIÓN

Para sacar provecho del dispositivo desarrollado se diseñó un paquete informático compuesto por dos programas (véase Figura 13): programa configurador y otro de proyección. Con el objetivo de facilitar la preparación de la proyección por el cliente, permitiendo que éste la editara

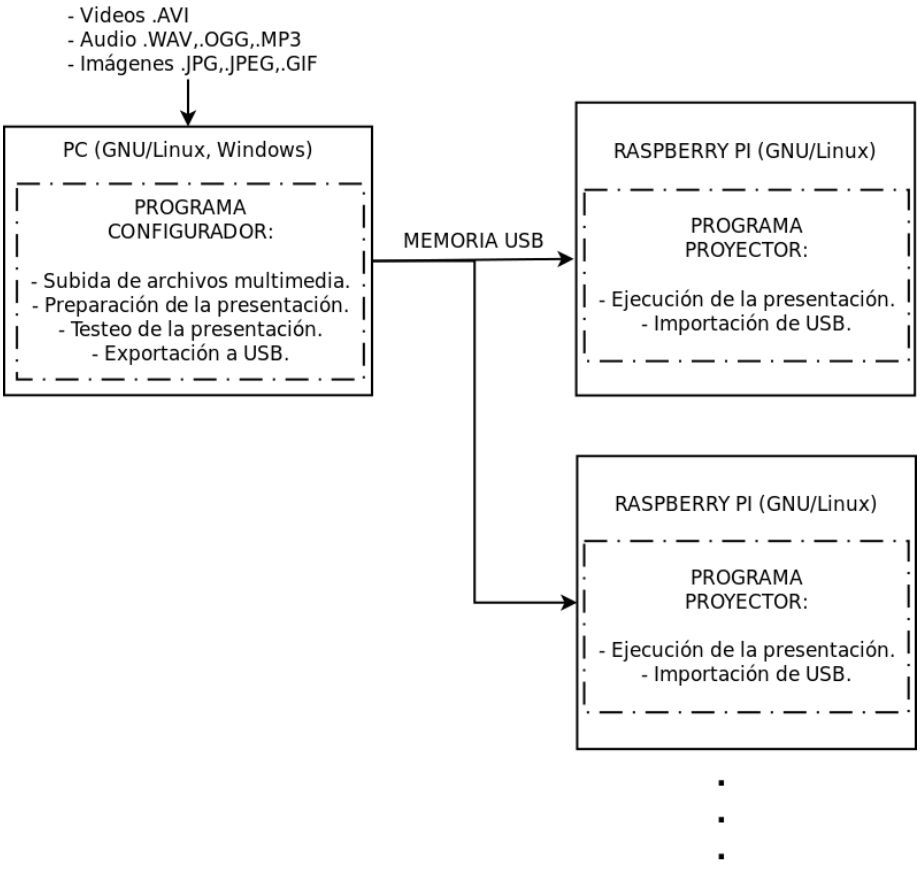


Figura 13. Diagrama genérico del paquete de software desarrollado.

en cualquier ordenador de su elección, de tal forma, que exportara la proyección a una memoria USB suministrada que transfiriera la información a los dispositivos desarrollados, de su elección.

## A. Mahimo, el programa de configuración

El objetivo del programa de configuración es conformar la plataforma de preparación de la exposición a representar, de tal forma que esta se pueda quedar totalmente establecida y configurada en una memoria USB, que sea ejecutable en pocos pasos y sin complicaciones por la aplicación de ejecución, ya dentro del sistema empotrado.

La potencialidad de este programa no sólo recae en poder preparar la presentación, si no que además constituye una potente herramienta para cargar vídeos, hilos musicales e imágenes estandarizándolas para su posible visualización. Así, todo el juego de la configuración de la presentación se realizará con las respectivas carpetas de imágenes, vídeos y música que el programa posee en su instalación, donde se introducirán los archivos multimedia deseados respectivamente, para luego su

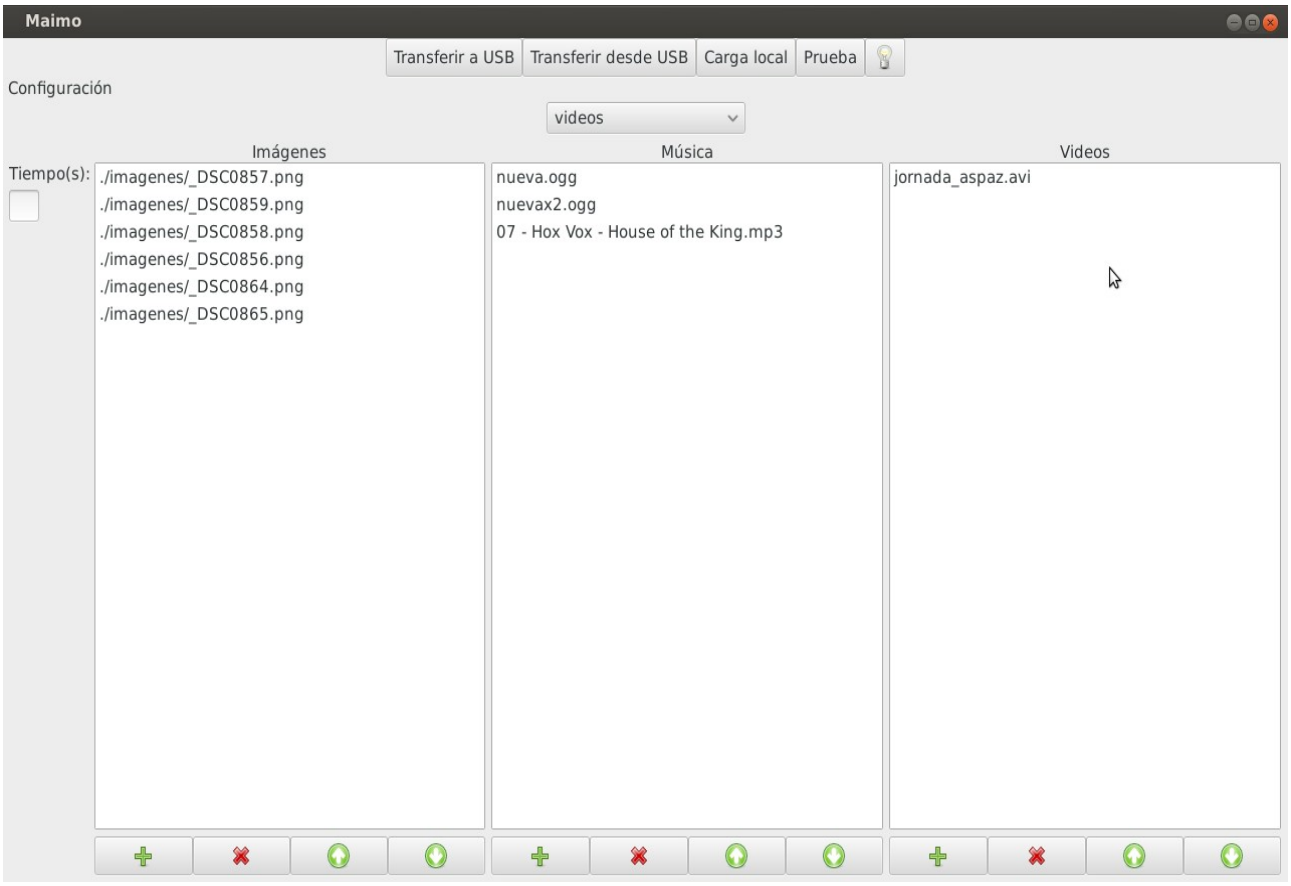


Figura 15. Interfaz principal de Mahimo.

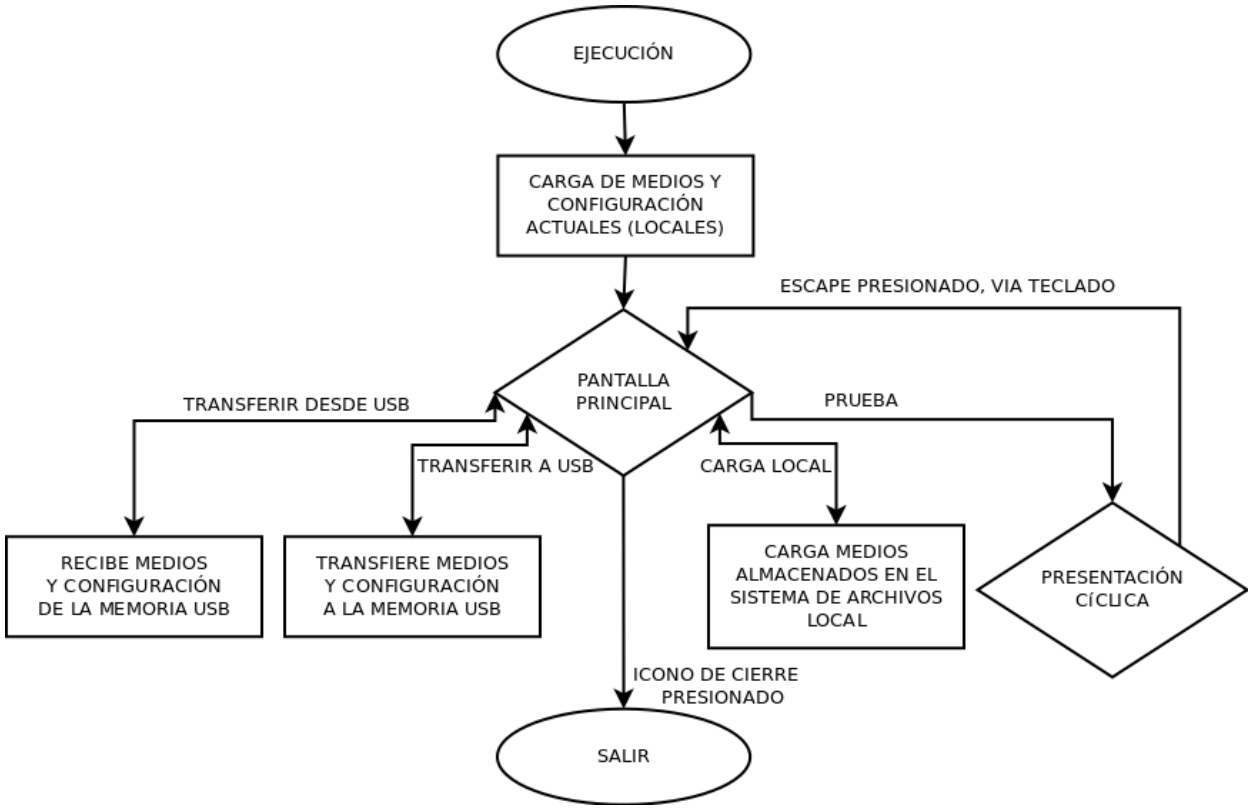


Figura 14. Diagrama lógico de la aplicación configuradora.

posterior carga en el programa.

Para ello el programa de configuración consta de una sencilla interfaz (véase Figuras 14 y 15), donde alrededor de una pantalla central, se muestran diferentes botones de acción para recuperar los datos desde el sistema de archivos, y recuperar desde/guardar hacia, nuestra memoria USB, así como una representación resumida de los distintos archivos multimedia, donde podremos ordenarlos, cambiar el tiempo de muestra de cada imagen, el tipo de presentación deseada, etc.

Se podrá transferir el tipo de presentación actual hacia el USB mediante el botón **transferir a USB**, donde para ello se deberá disponer de un USB formateado en FAT32 (*File Allocation Table32*) (o cualquier formato compatible con GNU/Linux si el programa de configuración no corre en una máquina Ms Windows).

Otro requisito será formatear el USB con la etiqueta: prointegra, aunque podrá ser configurable en futuras versiones.

Así, mediante esta acción, el sistema copiará las imágenes, vídeos, canciones y un archivo XML resumen de la presentación preparada.

EL botón siguiente es el de la **transferencia desde USB** mediante el cual, la aplicación cargará los archivos multimedia y su configuración al sistema de archivos locales, y los representará ordenados a modo resumen en sus respectivas columnas.

A continuación, la siguiente opción será la de la carga de los archivos del sistema de ficheros local. Mediante esta acción el programa cargará los vídeos de la carpeta vídeos, la música de la carpeta música, etc. así como el archivo de configuración XML. Y lo representará resumido y organizado en pantalla. Esta acción además analiza y prepara las imágenes, las canciones y vídeos para que sean representables por el programa. Esto es, cambia la resolución y el formato de las imágenes, comprueba el formato de las canciones, y hace lo mismo con los vídeos. Esto se hace para tener un control absoluto del formato de los archivos que utilizamos, limitando así la cantidad de tipos de archivo admitidos pero mejorando la estabilidad de la aplicación.

El formato de vídeo con el que el sistema es compatible hasta la fecha, es el formato AVI (*Audio Video Interleave*) y en audio los formatos Mp3, Ogg o WAV (*Waveform Audio File Format***)**. Para las imágenes el formato estándar será el PNG (*Portable Network Graphics*).

El penúltimo botón de acción será el de **prueba**, muy útil para comprobar la presentación configurada. Y que mostraría el resultado igual que si se fuera a visualizar en el Raspberry Pi.

Por último se puede acceder a la típica ventana de información con los créditos de

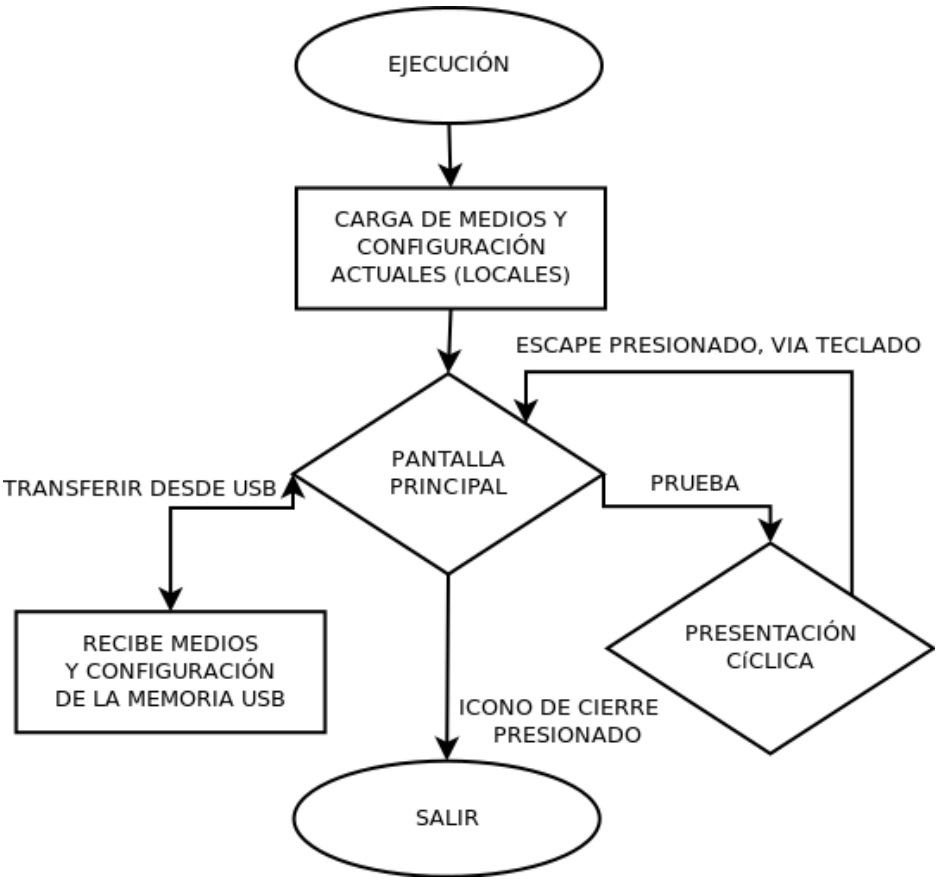


Figura 17. Diagrama lógico programa visualizador.


Figura 16. Interfaz principal de Nides.

la aplicación.

## B. Nides, el programa de visualización

Otro software de este proyecto lo conforma el programa Nides, y es el encargado de la visualización del proyecto dentro del Raspberry Pi.

Este programa también está concebido desde la simplicidad, para ejecutarse con robustez, de manera eficiente para el sistema empotrado, y de manera fácil de entender y utilizar para el usuario final.

Consta también de una pantalla central (véase Figuras 16 y 17) alrededor de la cual se accede a todas las posibilidades de esta aplicación. En la parte superior de esta pantalla se pueden realizar las siguientes tres acciones:

- **Transferencia desde USB***,* si se inserta un USB correctamente formateado como se explico en el apartado del programa de configuración, con una batería de archivos multimedia y un archivo de configuración provenientes del ejecutable Maimo, se tiene la opción de cargarlo, copiarlo al sistema de archivos locales, y establecerlo así como la presentación a cargar.
- Inmediatamente a la derecha, mediante el botón **Presentación** se procederá a la ejecución de la presentación tal como la tenemos concebida. Esta ejecución se llevará a cabo en pantalla completa de forma indefinida hasta que la proyección se vea interrumpida voluntariamente por el usuario.
- A su vez el programa posee un diálogo de información sobre la misma. Con la versión de la aplicación, los créditos, y más información referente a la misma.

Finalmente, en la parte central de la pantalla, se puede observar un recuadro donde se nos informará de todo el material cargado desde el USB, así como del tipo de presentación que se tiene preparada. Es una lectura del archivo de configuración en formato XML, sacando cada uno de los parámetros requeridos para la proyección.

## C. Ajustes necesarios en la configuración del Raspberry Pi

Este proyecto no consiste solamente en la compilación de sendos programas de configuración y visualización. Se debe realizar un trabajo previo de acondicionamiento del Raspberry Pi para un funcionamiento óptimo de la plataforma.

Entre los cambios necesarios para su acondicionamiento se realizaron los siguientes:

- Eliminación del apagado automático de la pantalla en modo ahorro de energía. Se realiza mediante la edición del fichero **config***,* en la ruta **/etc/kbd***.* Adjuntando a los parámetros:

***BLANK_TIME*** y ***POWERDOWN_TIME*** sendos valores de 0 (véase Figura 18).

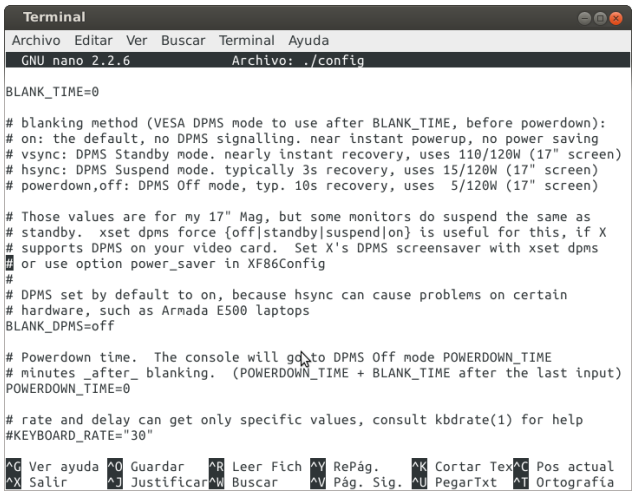


Figura 18. Detalle del archivo de configuración del paquete KBD.

- Configuración de una correcta resolución de pantalla. Mediante la modificación del archivo **config.txt** , en la carpeta **/boot**, estableciendo los valores deseados manualmente en los parámetros ***framebuffer_width*** y ***framebuffer_height***, resultando una resolución más que suficiente para trabajar de 1024x768 píxeles (véase Figura 19).

- Cambio del inicio de sesión por defecto.

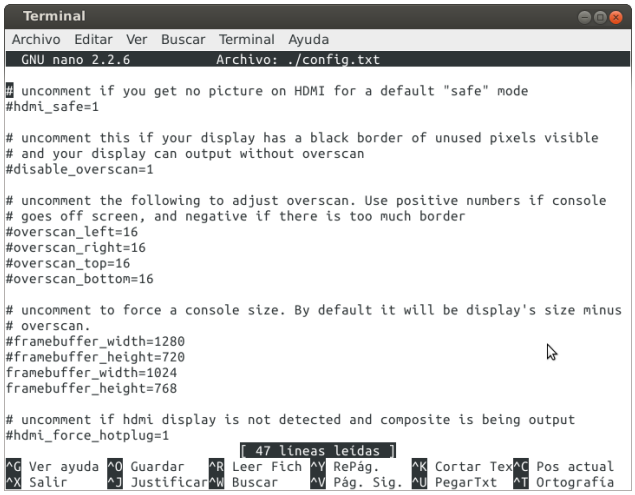


Figura 19. Contenido de /boot/config.txt.

Usando así el entorno minimalista **Openbox**, por encima de LXDE (*Ligh X Desktop Environment*), que es el que viene por defecto, para un uso más eficiente de la computadora, este cambio en la parametrización del Raspberry Pi se realizó mediante el comando en consola de Debian:

***sudo update-alternatives --config x-session-manager*** (véase Figura 20).

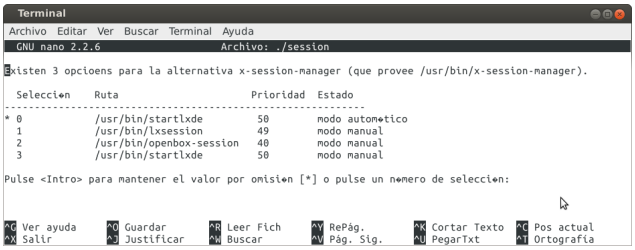


*Figura 20. Detalle del resultado devuelto por el comando* ***sudo update-alternatives --config x-session-manager***.

- Configuración del árbol de directorios de la aplicación de visualización, para un correcto almacenaje de los archivos multimedia y el archivo de configuración,

- Preparación del autoarranque de la aplicación. Mediante la edición del archivo **autostart**, en la ruta **~/.config/openbox** (véase Figura 21).


Figura 21. Detalle del archivo autostart.

Tabla 1.
Compendio resumido de la respuesta de los clientes.

| Cliente | Sistema Previo | Tiempo prueba | Puntos Débiles | Puntos Fuertes | Propuestas |
|---|---|---|---|---|---|
| Industrial: Empresa de fabricación de ámbito internacional | TV-DVD | 2 meses | -Incomodidad de tener que actualizar vía USB | -Reducido espacio<br>-Reducido consumo<br>-Un sistema novedoso bien valorado por los clientes<br>-Flexibilidad de modificación de la presentación.. | -Control vía red local de su funcionamiento y actualización<br>-Uso de una impresora 3D para la creación de un envase más estilizado |
| Bar: Comercio local | N/A | 2 meses | -Incomodidad de tener que actualizar vía USB<br>-Precio del prototipo elevado | -Interesante para la presentación del producto a los clientes<br>-Programas muy fáciles de usar<br>-Soporte técnico rápido<br>-Facilidad de conexión del dispositivo, permitiendo la utilización de la televisión del local cuando se desee | -Mejorar el sistema de modificación de las presentaciones<br>-Posibilidad de manejar el Raspberry Pi mediante un mando a distancia |
| Farmacia: Comercio local | N/A | 2 meses | -Aspecto rudimentario<br>-No hay versión para MacOs<br>-Hay vídeos no del todo fluidos<br>-imposibilidad de mezclar una presentación de vídeos e imágenes | -Facilidad de uso.<br>-Flexibilidad a la hora de cambiar de monitor, pudiendo recuperar equipos antiguos<br>-Soporte técnico local y especializado | -Soporte de plantillas para proyecciones<br>-Posibilidad de usar un programa de presentaciones tipo PowerPoint o Impress<br>-Una interfaz más estética.<br>-Posibilidad de alquilar el equipo en vez de su compra<br>-Posibilidad de controlarlo vía internet. |

# V. RESULTADOS

La prueba más importante para este tipo de sistema es la de un funcionamiento continuado de la aplicación (véase Figura 23). Para su realización se dejó el Raspberry Pi funcionar. Para ello, y gracias a la disposición de dos unidades Raspberry Pi, se pudo dejar un equipo trabajando continuamente mientras se ultimaba el desarrollo con la otra y se revisó periódicamente el estado del conjunto. El periodo de pruebas nunca excedió la semana, y como no se configuró ningún sistema de recuperación de la aplicación fue fácil comprobar su estabilidad, pues la presentación se podía interrumpir sin reinicializarse. A su vez testeó la

temperatura, y el uso de la memoria mediante un monitor de sistema de GNU/Linux.

Los fallos encontrados afectaban a la estabilidad, y eran relacionados con el uso de la librería GTK y su uso de memoria. Mediante la herramienta GDB (el depurador de la GNU) se solventaron dichos fallos. Y hasta la fecha no ha fallado la aplicación por su uso continuado. Las pruebas se dieron por finalizadas en Enero de 2013.

Cabe mencionar que no se ha usado ningún tipo de filosofía de control de fallos como podría ser un sistema de redundancia, o control de excepciones, ya que no se vio necesario para este tipo de desarrollo de electrónica de consumo.

La otra prueba realizada se enfocó a la mejora del producto, y consistió en la presentación del dispositivo a clientes potenciales, dueños de establecimientos como tiendas, bares, etc. Con el objetivo de que utilizaran la aplicación ellos mismos, y comentaran las bondades y desventajas del desarrollo, así como las mejoras que verían al producto final cara a sus necesidades individuales.

En este proceso de realimentación participaron 3 clientes potenciales distintos, con perfiles muy diferentes.

En ella se puso de manifiesto las distintas necesidades que tienen clientes de sectores distintos, y de empresas de distinto tamaño.

El producto resultó bien valorado, pero lo más importante fueron las demandas y necesidades que transmitieron. (véase Tabla 1).

# VI. CONCLUSIONES

Debido a lo extenso de este trabajo de ingeniería donde hay involucrados potenciales compradores y un producto comercial, se pueden extraer diferentes conclusiones desde el punto de vista del prototipo, de la elección de hardware y software utilizado y finalmente desde el posible comercio potencial.

## A. Valoración del producto final

El dispositivo final es un desarrollo concienzudo que pone de manifiesto la dificultad de desarrollo de productos concretos para su venta directa.

Hay muchas características que se han dejado apartadas, y puede que aún sea pronto para una etapa de comercialización, pero cabe mencionar el enriquecimiento que ofrece una aventura tal.

En una primera etapa se le dio demasiada importancia al desarrollo software, y se menospreció el desarrollo mecánico, cuando se ha dejado patente que puede haber una deriva considerable en el

desarrollo de la presentación final del dispositivo. Siendo además este apartado más que importante cara a un supuesto consumidor final.

Otro fallo de diseño fue considerar que el Raspberry Pi iba a funcionar a la perfección desde el principio, y es que, al ser un producto novedoso, aún a día de hoy no están perfectamente desarrollados los drivers para el *chipset* gráfico, y el sonido, lo que lleva a mal funcionamiento, ralentizaciones, etc. Con determinados formatos.

Aún así, hubiera sido infinitamente más complicado para una PYME (Pequeña y mediana empresa) desarrollar un dispositivo así desde cero, o sin basarse en librerías de software libre.

## B. Uso de software y hardware libre

Desde el punto de vista del desarrollador, las ventajas que este tipo de modelo deja a un desarrollo como el acontecido en esta memoria, son numerosas, y cabría destacar las siguientes:

1. Control total de la aplicación y las librerías y herramientas utilizadas. Esto da la posibilidad del desarrollador de conocer absolutamente todo lo que ocurre dentro del dispositivo. Dejando poco espacio al azar.
2. Independencia. Puesto que al tener el control total del dispositivo no se depende de empresas externas, normalmente grandes empresas con deficientes servicios técnicos. Al ser el software libre, software con un desarrollo continuado y normalmente bastante estable tampoco se cae en la trampa de productos novedosos incompatibles con versiones anteriores y de poco testado como los que suelen ofrecer las grandes compañías convencionales. Así, se ha garantizado la compatibilidad de las librerías utilizadas en el proyecto, su mantenimiento y mejora e inclusión de nuevas características, al igual que la conservación de los cambios realizados en el sistema operativo.

   También la PYME se asegura un suministro infinito de placas de Raspberry Pi, al no depender este diseño de una empresa que pudiera desaparecer, o considerar no rentable su desarrollo y venta en un momento dado.
3. Gran soporte y desarrollo colaborativo. Una PYME jamás podrá competir en desarrollo de aplicaciones, debido a su escasa potencia económica y humana. Por lo que la solución pasa por compartir conocimiento, donde a igualdad de

productos se puede competir en mejora de servicios, soporte o en venta de desarrollos personalizados. A su vez, el compartir conocimiento da el mejor soporte técnico posible, donde miles de desarrolladores ponen siempre en común sus conocimientos en proyectos afines. Esta ventaja cuesta verla en una etapa tan temprana como la del proyecto que nos acontece, pero con una buena campaña de soporte y publicidad, y con un diseño atractivo a desarrolladores, el proyecto puede acabar "rodando sólo" en poco tiempo.

4. Abundancia de documentación e información sobre modificaciones o reparaciones del sistema hardware. Ya que el Raspberry Pi está desarrollado con una filosofía de hardware libre, posee una gran comunidad de desarrolladores motivados por un bien común, que documentan, desarrollan, expanden el dispositivo. Así, la etapa de documentación en este desarrollo ha sido bastante corta, y fácil, pues es extremadamente sencillo obtener/ participar/mejorar las documentaciones también libres que existen en Internet.

Desde el punto de vista del cliente las ventajas son también numerosas, entre ellas cabe destacar:

1. Evita mediante este tipo de desarrollo "casarse" con una empresa concreta, y depender de ella. Así puede contratar a un tercero para que le desarrolle o mantenga la aplicación.
2. Fiabilidad del producto. El cliente está comprando un producto popular, público y documentado, garantía para evitar mal funcionamientos imprevistos u obsolescencia programada.

### C. Modelo de negocio basado en servicios

Así el modelo de negocio de una empresa embarcada en este tipo de producto sería, tanto la puesta en servicio del dispositivo y su venta, como el mantenimiento, como el desarrollo de soluciones concretas para clientes concretos (desarrollo de plantillas para publicidad personalizadas, desarrollo de características nuevas).

## VII. FUTUROS TRABAJOS

Un producto de estas características dista mucho de ser perfecto, así, después y durante del desarrollo, como ya se ha comentado en apartados anteriores, se

trabajó de la mano de potenciales clientes, que evaluaron la aplicación y dieron abiertamente su opinión.

Así fueron surgiendo sinergias y posibles mejoras del dispositivo.

En este último epígrafe se nombrarán las que resultaron más interesantes y que por falta de tiempo no fueron introducidas. Abriendo la puerta a futuras mejoras del sistema:

1. Compatibilidad con presentaciones **Impress** de **Libreoffice/Openoffice**.
   Punto muy interesante, ya que da la posibilidad de crear una presentación al cliente con una herramienta gratuita, que permite la inserción de audio, imagen y vídeo en un mismo archivo. Se podría, usando las librerías de **libreoffice** embeber la presentación dentro de la aplicación, como si se tratara de un tipo de presentación más. Otra opción sería usar tuberías para conectar la aplicación con el **libreoffice**, y que fuera este el que ejecutara la presentación, solución más sencilla pero presumiblemente más inestable.
   La inclusión de esta característica puede ser muy interesante,como ya se ha comentado, por la facilidad que da al usuario final de crear presentaciones de su establecimiento.
2. Posibilidad de poseer un servidor Web al cual acceder para cambiar los parámetros de la presentación, tipo de presentación, subir y bajar archivos al dispositivo, etc.
   Esta idea es muy atractiva también, ya que permite al usuario conectarse al dispositivo desde el móvil, desde casa, u otro lugar distinto del trabajo y cambiar desde allí la presentación a mostrar, sin tener que estar presente en la modificación de la misma mediante la memoria USB.
   Este problema no parece crítico aún, con el tipo de usuario final al que se ha enfocado. Pero sería un problema grande si hablamos de volúmenes de negocio donde se manejan más de una pantalla.
   Es interesante también, porque abre la puerta a más diseños con el Raspberry Pi, donde tener un servidor Web y poder acceder a él es un valor añadido a tener en cuenta.
3. El programa podría reproducir medios provenientes de Internet (como vídeos de servidores de reproducciones), ya que la librería GStreamer lo permite, pero no se vio necesario a corto plazo
4. Se vio interesante la inclusión en la caja, unos botones, conectados a través del bus I2C al Raspberry Pi. Se Tenía localizada la librería que

permite interpretar el I2C en el Raspberry PI, y su inclusión dentro del programa de visualización. Pero por falta de tiempo, y complicación de instalación en la caja y la PCB, se desechó por el momento. Dejando un ratón simplemente, conectado por USB.

5. Por último se estudió dar la posibilidad de transición entre imágenes mediante cortinillas y transiciones elegibles. Pero se ha apartado por el momento por la dificultad a priori de ser codificado (siendo además más fácil cargar una presentación **Impress**, que solucionaría el problema en parte).

## REFERENCIAS


[1] Steve Furber. ARM System-on-chip Architecture. 2ª Ed. Pearson Education Limited. 2000. pp. 413.

[2] Chen Tie-jun, Wei Chao, Jia Dong-ming. Electrocardiogram monitoring system based on ARM9. Cnki [Revista en Internet] 2011. [acceso en Julio 2012] Disponible en http://en.cnki.com.cn/Article_en/CJFDTOTAL-XDKF201117031.htm.

[3] Ashlee Vance. British Chip Designer Prepares for Wider Demand. The New York Times. 19 de septiembre del 2010. Sección B1.

[4] Advanced Micro Devices Company, AMD Geode LX Processor Family. [Actualizada en 2012, accedida en Febrero 2012]. Disponible en http://www.amd.com/us/products/embedded/processors/geode-lx/Pages/geode-lx-processor-family.aspx.

[5] Wang Xiao-ning, Wang Zhen-chen, Zhang Shao-bing, Yao Fan. The Transplanting of Linux Operating System to ARM9 Processor. Cnki [Revista en Internet] 2010. [acceso en Julio 2012] Disponible en http://en.cnki.com.cn/Article_en/CJFDTOTAL-HGZD201002020.htm.

[6] Fundación GNU, Sistema Operativo GNU. [actualizada en agosto 2012, accedida en agosto 2012]. Disponible en http://gnu.org.

[7] Fundación Raspberry Pi, Raspberry Pi y Debian, hacen Raspbian. [Actualizada en julio 2012, accedida en julio 2012]. Disponible en http://raspbian.org.

[8] Fundación GNOME, The GTK+ Project. [Actualizada en octubre 2012, accedida en noviembre 2012]. Disponible en http://gtk.org.

[9] ImageMagick Studio LLC, Imagemagick. [actualizada en abril 2012, accedida en noviembre 2012]. Disponible en http://www.imagemagick.org.

[10] Equipo GStreamer, GStreamer open source multimedia framework. [Actualizada en diciembre 2012, accedida en diciembre 2012]. Disponible en http://gstreamer.freedesktop.org.

[11] Fundación Beagleboard, Beagleboard -Community supported open hardware computers for making. [Actualizada en julio 2012, accedida en julio 2012]. Disponible en http://beagleboard.org.

[12] Tienda online Miniand, Miniand – your one stop shop for mini Pcs and hobby tech [actualizada en marzo 2012, accedida en julio 2012]. Disponible en https://www.miniand.com.

[13] Comunidad mantenida por voluntarios, Pandaboard.org [actualizada en julio 2012, accedida en julio 2012]. Disponible en http://pandaboard.org.

[14] Fundación Raspberry Pi, Raspberry Pi. [Actualizada en julio 2012, acceso en julio 2012]. Disponible en http://raspberrypi.org.

[15] Stallman R,et al.. GNU Emacs Manual. Ed. Free Sotware Foundation. 2013. pp. 579.

[16] Fundación GNOME, The XML C parser and toolkit of Gnome libxml. [actualizada en septiembre 2012, accedida en diciembre 2012]. Disponible en http://www.xmlsoft.org.

## ANEXO I. Makefiles del proyecto

El makefile es el archivo de configuración de la compilación del programa, que entiende el compilador GCC, en él definimos los archivos fuente, los objetos, las dependencias, el archivos de salida, etc. El makefile para compilar este proyecto es extremadamente sencillo.

Programa Maimo:

```
PI-RPI: main.c

        gcc -Wall -g -O2
-DCONFIGURADOR -o Maimo

main.c
main_win.c
../frontends/proyeccion.c
../frontends/proyeccion_vid.c
../frontends/progressbar.c
../backends/carga_biblioteca.c
../backends/carga_imagenes.c
../backends/carga_audio.c
../backends/carga_video.c
../common/pi_xml_parser.c

-lgthread-2.0 `xml2-config --cflags`
`pkg-config --cflags MagickWand
gstreamer-0.10 gtk+-3.0 gmodule-export-2.0
gstreamer-interfaces-0.10` `pkg-config --libs
MagickWand gstreamer-0.10 gtk+-3.0
gmodule-export-2.0 gstreamer-interfaces-0.10`
`xml2-config --libs`
```

Programa Nides

```
PI-RPI: main.c

gcc  -Wall -g -O2 -DPUBLICISTA -o Nides

main.c
main_win.c
../frontends/proyeccion.c
../frontends/proyeccion_vid.c
../backends/carga_biblioteca.c
../backends/carga_imagenes.c
../backends/carga_audio.c
../backends/carga_video.c
../frontends/progressbar.c
../common/pi_xml_parser.c

-lgthread-2.0 `xml2-config --cflags`
`pkg-config --cflags MagickWand
gstreamer-0.10 gtk+-3.0 gmodule-export-2.0
gstreamer-interfaces-0.10` `pkg-config --libs
MagickWand gstreamer-0.10 gtk+-3.0
gmodule-export-2.0 gstreamer-interfaces-0.10`
`xml2-config --libs`
```

En ellos hay marcados en verde las opciones de compilación, en rojo los archivos fuente del proyecto, y en azul las dependencias.

## ANEXO II. Estructura del archivo .xml de configuración

```
<?xml version="1.0" encoding="UTF-8"?>
<settings>
<tipo_de_presentacion>20</tipo_de_presentacion>
 <no_images>12</no_images>
 <no_videos>4</no_videos>
 <no_audios>20</no_audios>
 <imágenes>
 <n1>
<nombre>./imagenes/1024x768.png</nombre>
 <tiempo>1</tiempo>
 </n1>
...
 </imágenes>
 <videos>
 <n1>
<nombre>Canon_PowerShot_SD850_IS.avi</nombre>
 </n1>
 <n2>
 <nombre>DSCF8582.avi</nombre>
 </n2>
...
 </videos>
<audios>
 <n1>
 <nombre>nueva.ogg</nombre>
 </n1>
 <n2>
 <nombre>nuevax2.ogg</nombre>
 </n2>
…
</audios>
</settings>
```

El archivo XML está estructurado de tal forma que da información del tipo de presentación (1 música e imágenes, 2 sólo imágenes, 20 vídeos), del número de archivos multimedia insertados (útil para la reserva de memoria dinámica), y luego una lista de los archivos, en el caso de las imágenes no sólo ofrece el nombre, si no también el tiempo de muestra de cada una.

UNIVERSIDAD DE ALMERÍA

# Resumen/Abstract

Resumen-Uno de los grandes problemas a los que se presenta una empresa nacional de tamaño medio, en el contexto económico y tecnológico actual, es el de acabar dependiendo en demasía de tecnología extranjera, carecer del dinero necesario para poder investigar tecnología propia y acabar así sobreviviendo o mediante contratos abusivos de subcontratación a cargo de multinacionales, o como mero distribuidor a merced de los caprichos del mercado internacional en productos de los que no se tiene control alguno.
En este marco surge la idea del proyecto que se muestra, cuyos pilares principales son los sistemas empotrados, y el hardware y software libre. Mediante un sistema empotrado de Hardware libre de los más económicos del mercado, el Raspberry Pi, se pretende aliviar la carga de I+D necesaria para un desarrollo ambicioso y aprovechar así la potencia y flexibilidad de un sistema embebido potente y de bajo coste.
Se ha diseñado un sistema de publicidad compuesto por un programa de configuración de la presentación y otro de visualización de la misma. El dispositivo se inserta en una caja cerrada capaz de poderse conectar en el monitor que se desee.

Abstract-One of the biggest problems a national medium size enterprise faces, in the present economic and technological context, is: to depend too much on foreign technologies; which on the other side means that there is a lack of resources to research on own technologies. This problem results in a position of dependence, either towards multinational enterprises and abusive subcontracting agreements or as a simple distributor of products on which, small enterprises have no control, towards the international market instability.
In this context, we developed the present project, based on embedded systems, and free hardware and software. Using a free hardware embedded system, one of the cheapest ones in the market, the Rasperry Pi, we tried to facilitate the I+D needed profiting from a strong and low cost embedded system.
We have designed a publicity system based on two programs: a configuration program and a publicity monitor program. The device is presented in a small case attachable to any screen.