Real-Time Audio Processor

Memoria del proyecto

Rubén Agustín
David Andrino
Estela Mora
Fernando Sanz

Ingeniería de Sistemas Electrónicos
Primavera 2024

Índice

Índice de figuras

1. Introducción

1.1. Especificaciones iniciales del sistema a diseñar y construir

El proyecto consiste en el desarrollo de un sistema de procesamiento de audio en tiempo real con un sistema centrado en un microprocesador.

El sistema permitirá entrada de audio a través de un módulo de radio FM y la lectura de canciones de una tarjeta microSD externa. Tras capturar dicho audio, se le aplicará un procesamiento digital implementado mediante la librería CMSIS DSP y se reproducirá por un altavoz o unos auriculares.

La interacción con el sistema se realizará a través de una pantalla táctil o una interfaz web, la cual requiere conexión a través de Ethernet. Dicha interfaz permite seleccionar la entrada de audio, controlar el volumen y ecualización de audio y elegir la salida. Además, permite controlar la emisora de radio sintonizada.

El sistema almacenará los parámetros seleccionados (entrada, salida y filtros) y una lista de emisoras favoritas en la tarjeta microSD, los cuales se cargarán al iniciar el equipo.

El sistema será completamente autónomo, contando con una batería con su correspondiente circuitería de carga, protección y medición de consumo. Se ofrecerá información sobre la batería en la interfaz gráfica del sistema. Además, el sistema contará con un modo de bajo consumo para alargar la duración de dicha batería.

Para la selección de canciones se permitirá el uso de tarjetas NFC preconfiguradas con canciones o emisoras preconfiguradas, para la interacción sin interfaz gráfica. Por último, el sistema utilizará el RTC integrado en la placa para mantener la hora y el protocolo SNTP para la sincronización.

1.2. Especificaciones finales del sistema diseñado y construido

2. Desarrollo de subsistemas

Para este proyecto hemos desarrollado dos subsistemas analógicos propios, sobre dos PCB distintas, una de alimentación y un amplificador de audio. Además, hemos utilizado tres subsistemas ya existentes para la recepción de audio, la reproducción de música en MP3 y la lectura de información NFC de un dispositivo móvil.

2.1. Subsistema de alimentación

2.2. Subsistema de audio

2.3. Módulo de radio

El modelo de radio elegido ha sido el Sintonizador FM RDA5807M, mostrado en la Figura 1, utilizado anteriormente en la asignatura de Sistemas Basados en Microprocesadores.

Dicho modelo se comunica con el microcontrolador mediante el protocolo `I2C`. El sintonizador cuenta con un decodificador MPX, salida de audio stereo y un rango de sintonización de 87 MHz a 108 MHz debido a nuestra situación geográfica.

En cuanto a la señal de audio de salida, hemos encontrado que presenta una componente continua de 1.65V y una amplitud de aproximadamente 1V. Debido a la conectividad que presenta el sintonizador, nos hemos encontrado el problema de que, al unirlo junto al resto del proyecto, cortocircuita la señal de masa analógica con la señal de masa virtual creada en nuestro circuito. Dicho problema se comentará en el **??**.

Por otra parte, necesita ser alimentado con una tensión entre 1.8V y 3.3V, soporta una corriente máxima de 10mA, cuenta con una relación señal-ruido típica de 55dB y presenta una distorsión armónica de audio total máxima del 0.2 %.

Todas las características de dicho sintonizador FM se han obtenido del datasheet ofrecido por el fabricante.

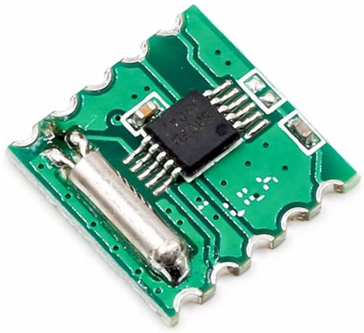

Figura 1: Sintonizador FM RDA5807M

2.4. Módulo MP3

El modelo de MP3 seleccionado ha sido el YX5300, el cual se muestra en la Figura 2.

Este reproductor se comunica con el microcontrolador mediante UART, con una velocidad de 9600 bps. También cuenta con una frecuencia de muestreo de 48 kHz y soporta tanto el formato MP3 como el formato WAV. Este modelo cuenta con un socket de tarjeta microSD en la cual se introducen las canciones, en los formatos antes mencionados, que se deseen reproducir. Dicha tarjeta deberá estar en formato Fat16 o Fat32 y tener como máximo 32G de almacenamiento.

En cuento a la señal de audio obtenida a la salida del reproductor, podemos observar una salida bipolar entrada en 0V. Este comportamiento es totalmente inesperado ya que, al estar alimentado entre 3.3V y 0V, es extraño que la señal de salida pueda presentar valores negativos. Esto ha supuesto un gran problema en la unificación con el resto de módulos. Dicho problema y su solución será comentada en el **??**.

Por otra parte, el reproductor debe ser alimentadocon una tensión entre 3.2V y 5.2V, soporta una corriente máxima de 200mA y cuenta con un LED rojo que indica si está reproduciendo alguna canción.

Todas las características del reproductor MP3 se han obtenido del datasheet ofrecido por el fabricante.

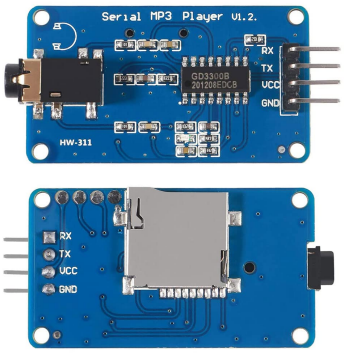

Figura 2: Reproductor MP3 YX5300

2.5. Módulo NFC

2.6. Entre Dos Tierras

3. Software

3.1. Interfaz de usuario

Hemos desarrollado dos interfaces de usuario, una sobre una web y otra sobre una pantalla táctil.

3.2. Interfaz web

Para la interacciónconel sistema, hemos diseñado un servidor web mediante el lenguaje `HTML`, estilizado mediante `CSS` y con algunas funciones creadas mediante JavaScript.

EL procesamiento de datos recibidos desde el servidor web y la creación dinámica de datos para la web se generan en diferentes archivos `CGI`, los cuales se explicarán en el **REFENCIA AL APARTADO DE LOS CGI**.

El servidor web cuenta con 4 páginas web diferentes, una principal, una para gestionar la radio, otra para gestionar el reproductor MP3 y otra para gestionar el procesado de audio. En todas estás páginas, en la esquina superior derecha, se encuentra la fecha y la hora del sistema. Todas estas páginas se explican a continuación.

3.2.1. Configuración General

Esta página web cuenta con una sección llamada *Camino de Audio*, la cual cuenta con 4 botones que nos permiten seleccionar tanto la entrada, Radio o MP3, tanto la salida de audio, Auriculares o Altavoz.

También cuenta con una sección llamada *Bajo Consumo* que cuenta con tan solo un botón que nos permitirá poner el microcontrolador en modo bajo consumo.

Por último, cuenta con una sección llamada *Consumo*, que contiene un widget, el cual se ha obtenido de **REFENCIA A DE DONDE SE SACO ESTO** que nos permite visualizar de forma dinámica en consumo medido en el sistema.

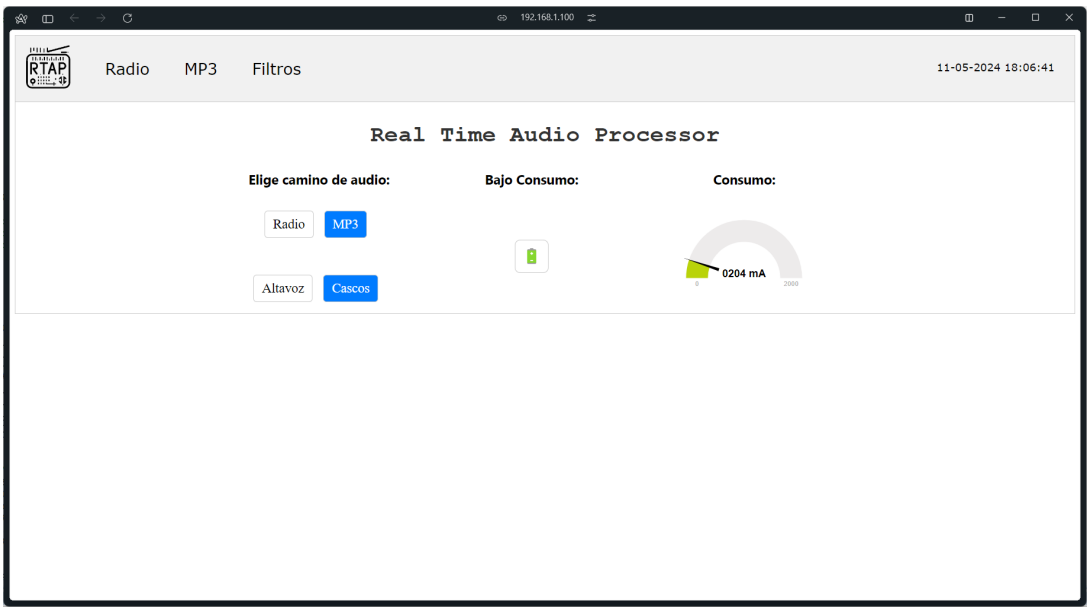

Figura 3: Página Principal

3.2.2. Página Radio

Esta página web cuenta con una primera sección llamada *Sintonizar una frecuencia* la cual nos permite introducir la frecuencia que deseemos sintonizar en el recuadro blanco. Luego, mediante el botón Sintonizar, podremos sintonizar dicha frecuencia en el Sintonizador FM.

A continuación, se encuentra una sección llamada *Seek*, en la cual se encuentran dos botones. El primero, que contiene una flecha hacia arriba, nos permite realizar un *SeekUp*, es decir, sintonizar una frecuancia mayor con más potencia que un determinado umbral. De forma análoga, el otro botón, ilustrado mediante una flecha hacia abajo, nos permite realizar un *SeekDown*, es decir, sintonizar una frecuencia menor, pero que tenga más potencia que dicho determinado umbral.

La siguiente sección llamada *Volumen*, contiene un slider horizontal que nos permite selecionar el volumen de la señal de audio. También encontramos un botón de mute, es decir, situar el volumen a 0.

Por último, encontramosla sección *Salida*, la cual cuenta con dos botones que nos permiten seleccionar la salida de auido deseada entre Altavoz o Auriculares.

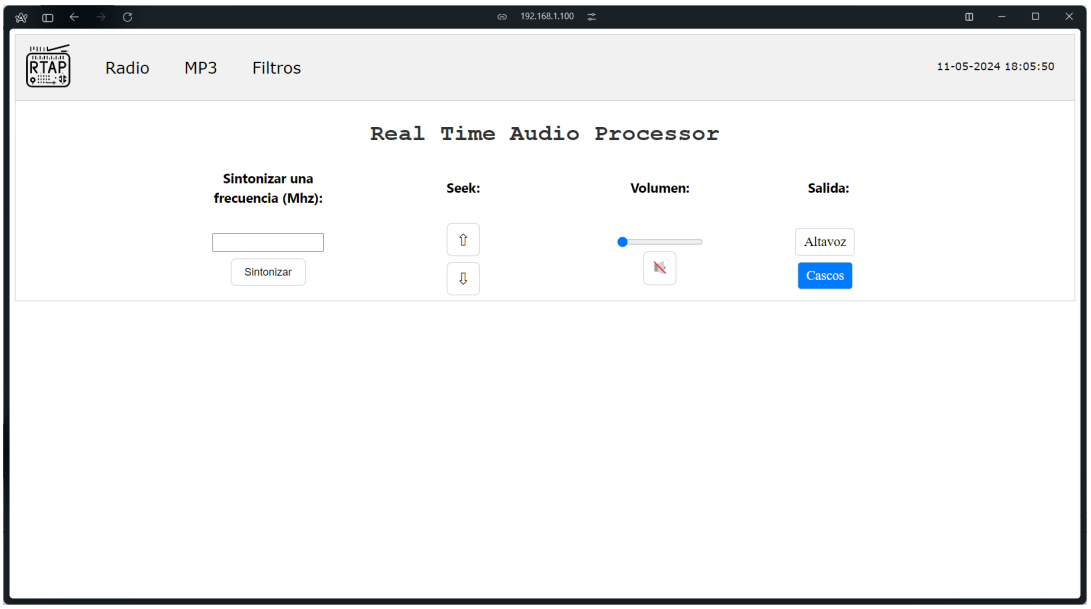

Figura 4: Página Radio

3.2.3. Página MP3

En primer lugar, nos econtramos con una sección llamada *Canciones*, la cual cuenta con un menú desplegable con lista, con sus correspondientes nombres, de las posibles canciones. Junto a dicho menú, se cuentra un botón que nos permite confimar la canción seleccionada.

A continuación, encontramos la sección demoninada *Control* la cual cuentra con 4 botones. El primero, nos permite seleccionar la canción anterior a la canción acutal. A continuación, nos encontramos con un botón que nos permite tanto pausar como continuar la reproducción de la canción actual. El siguiente botón nos permite seleccionar la siguiente canción de la lista. Por último, el botón de abajo, nos permite activar y desactivar la puesta en bucle de la canción actual.

De forma análoga a la página de la Radio, las siguientes dos secciones nos permiten modificar el volumen del sistema y la salida de la señal de audio.

3.2.4. Procesamiento de Audio

La última página web se encarga de todo el procesamiento de audio. En primer lugar, nos encontramos con la sección llamada *Ecualizador*, el cual nos permite elegir, mediante unos sliders vertiales, entre un rango de valores, la ecualización que se desee aplicar en las diferentes bandas posibles.

A continuación, en la sección denominada *Guardar Conf.*, nos encontramos un botón que nos permite guardar la configuración de los distintos filtros en la tarjeta microSD conectada al sistema.

Por último, de la misma manera que en las dos anteriores páginas, nos encontramos los controles que nos permiten modificar el volumen del sistema y seleccionar la salida de auido.

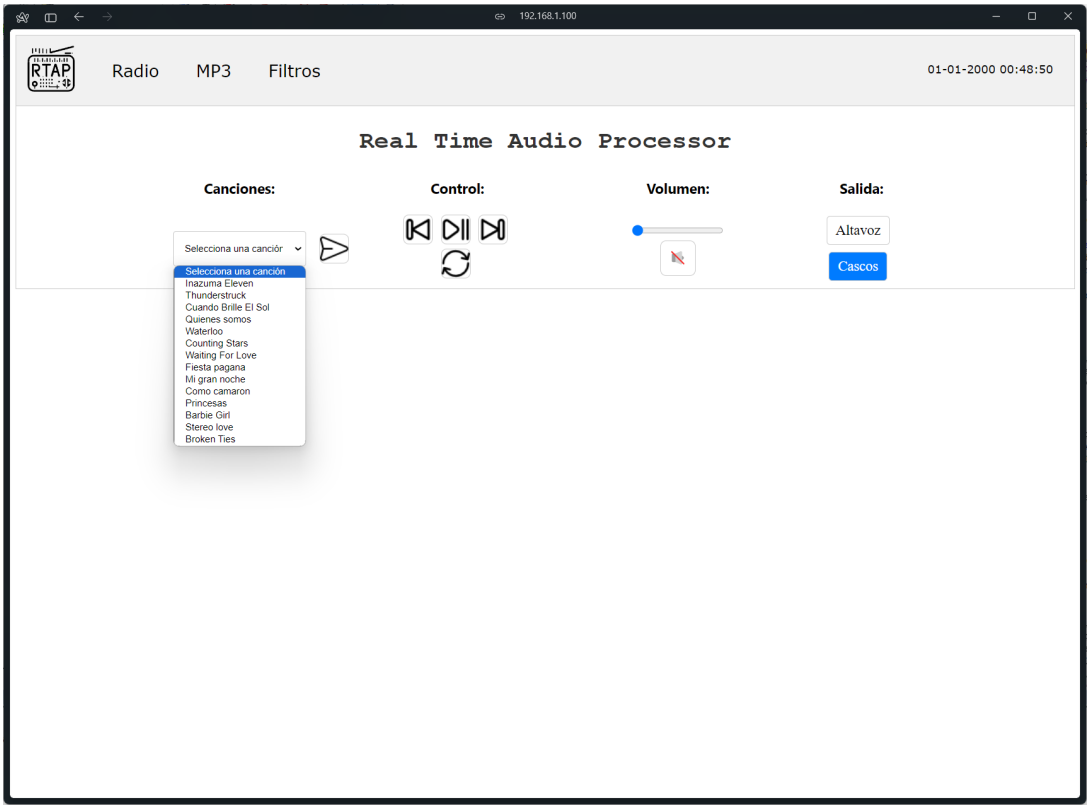

Figura 5: Página MP3

3.3. Interfaz táctil

3.4. Módulo RTC

El principal objetivo de este módulo es mostrar la hora y la fecha actual en el sistema. Esto se consigue mediante el periférico RTC del propio microcontrolador.

Dicho reloj se ha configurado para que se conecte al oscilador interno LSE, que cuenta con una frecuencia de 32.768kHz. Debido a esta frecuencia, se ha configurado el periférico con unos valores `AsynchPrediv = 127` y `SynchPrediv = 255` para obtener un periodo de 1 segundo.

Por otra parte, para obtener la hora actual, se ha optado por utilizar una sincronización con un servidor mediante SNTP.

También se ha configurado el RTC para generar una interrupción mediante una alarma con una frecuencia de 1Hz. En el callback de dicha alarma, se informa a este módulo, mediante una flag, que debe indicar al módulo principal, la fecha y la hora actuales. Dicha comunicación se realiza mediante una cola de mensajes.

3.5. Módulo Radio

El módulo del Sintonizador FM consiste en un *Thread* que se encarga de gestionar el funcionamiento del propio módulo. La comunicación entre el periférico y el microcontrolador se ha configuradao mediante el *Driver I2C* proporcionado por *CMSIS*.

Cuenta con dos colas de mensajes, una en la que el programa principal introduce los mensajes con los comandos que desea que ejecute el Sintonizador, como sintonizar una frecuencia, hacer un *Seekup* o un *SeekDown*, etc. La otra cola se utiliza para que la radio introduzca mensajes con información sobre las frecuencias sintonizadas. Los mensajes mandados por esta cola son del tipo *MSG_RADIO*.

También cuenta con un *timer* periódico, con un periodo de medio segundo, que se encarga de leer los registros del sintonizador para comprobar su correcto funcionamiento.

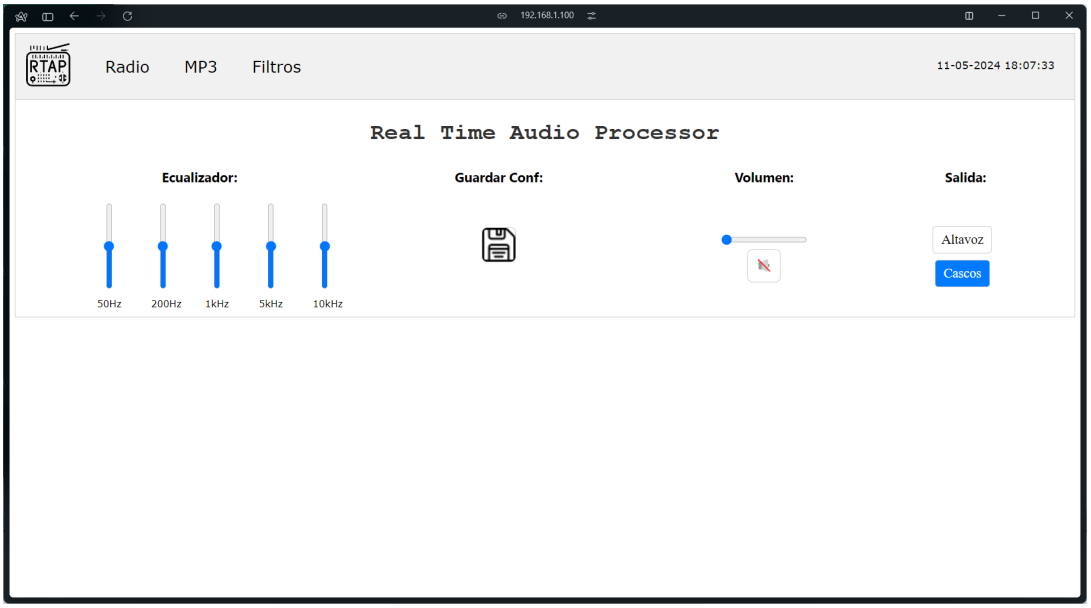

Figura 6: Página Filtros

Debido a que nuestro sistema cuenta con dos periféricos que utilizan el protocolo `I2C`, nos hemos visto obligados a utilizar un módulo de gestión del propio protocolo de comunicación, del cual se hablará en el **REFERENCIA AL APARTADO DEL I2C**.

El comportamiento básico de este módulo consiste en una espera mediante un *osMessageQueueGet* de un mensaje proporcionado por el programa principal. Una vez el mensaje es recibido, es procesado y en función de su contenido, se ejecutará el comando correspondiente. En caso de ser un *SeekUp* o un *SeekDown*, el módulo del sintonizador informará al *Thread* principal para que muestre la frecuencia en la que ha terminado.

3.6. Módulo MP3

El módulo del reproductor MP3 consiste en un *Thread* que se encarga de controlar la gestión del propio módulo.

Cuenta con una cola de mensajes en la que el programa principal introduce mensajes con el comando a ejecutar. Algunos de estos comandos son reproducir una cancion, pausar su reproducción, siguiente cancion, etc.

Para gestionar la comunicación entre este módulo y el principal, se ha utilizado la USART6 mediante el *Driver_USART6* proporcionado por CMSIS. Dicha USART se ha configurado atendiendo a las necesidades del reporductor, modo asíncrono, datos de 8 bits sin paridad, utlizando 1 bit de stop y con una velocidad de 9600 bps.

Este módulo no envía mensajes al programa principal debido a que el reproductor no ofrece la posibilidad de leer sus registros mediante la UART.

El comportamiento básico de este módulo consiste en una espera mediante un *osMessageQueueGet* de un mensaje proporcionado por el programa principal. Una vez el mensaje es recibido, es procesado y en función de su contenido, se ejecutará el comando correspondiente. Una vez la canción es seleccionada, se realiza otra espera mediante un *osThreadFlagsWait* hasta que la transferencia es completada, en cuyo caso el módulo continua con su funcionamiento. as

3.7. Módulo Web

Para la generación de las diferenctes páginas web del servidor y la creación de los diferentes scripts para interactuar con dichas páginas, hemos creados varios archivos con extension `CGI` como el que se adjunta a continuación:

```
t       <form action="index.cgi" method="post">
c i 1       <input type="radio" id="entrada_radio" name="entrada" value="radio"
    OnClick="submit();" %s>
t               <label for="entrada_radio" style="font-size: 20px;">Radio</label>
c i 2       <input type="radio" id="entrada_mp3" name="entrada" value="mp3"
    OnClick="submit();" %s>
t               <label for="entrada_mp3" style="font-size: 20px;">MP3</label>
t       </form>
t           <br><br>
t       <form action="index.cgi" method="post">
c i 3       <input type="radio" id="salida_altavoz" name="salida" value="altavoz"
    OnClick="submit();" %s>
t               <label for="salida_altavoz" style="font-size: 20px;">Altavoz</label>
c i 4       <input type="radio" id="salida_cascos" name="salida" value="cascos"
    OnClick="submit();" %s>
t               <label for="salida_cascos" style="font-size: 20px;">Cascos</label>
t       </form>
```

Fragmento 1: Ejemplo archivo .CGI

Como se puede observar, existen dos tipos direfentes de líneas de código, las que empiezan por "t" y las que empiezan por "c". Las que empiezan por "t" son ignoradas por el compilador y no se procesan, en cambio, las que empiezan en por "c" son atendidas y procesadas. A continuación, se comprueba la letra siguiente al espacio, en este caso la "i", debido a que nos encontramos en el archivo *index.cgi*. Por último, se obtiene el siguiente carácter a continuación del espacio en blanco y, el propio programa del servidor web, tomará unas medidas u otras dependiendo de dicha letra o número. En todos los casos, se sustituirá el "%s" que encontramos al final de dichas líneas de código por el conjunto de carácteres deseado mediante las función *snprintf()*. A continuación se adjunta un trozo de código del programa principal del servidor web en el que se realiza dicha acción:

```
switch(env[0]){
   case 'i':
      // Cases for index
      switch (env[2]){
         case '1':
         // Case for Radio Input
            len = sprintf (buf, &env[4], web_state.entrada == WEB_RADIO ? "checked" :"");
         break;
         case '2':
         // Case for MP3 Input
            len = sprintf (buf, &env[4], web_state.entrada == WEB_MP3 ? "checked" :"");
         break;
         case '3':
         // Case for Altavoz Output
            len = sprintf (buf, &env[4], web_state.salida == WEB_ALTAVOZ ? "checked" :"");
         break;
         case '4':
         // Case for Auriculares Output
            len = sprintf (buf, &env[4], web_state.salida == WEB_AURICULARES ? "checked" :"");
         break;
         }
     break;
}
```

Fragmento 2: Ejemplo procesamiento archivo .CGI

Por otra parte, para enviar los datos desde las páginas al programa principal del servidor, se ha optado por usar fótmularios con el método *post*, los cuales se envían si se pulsa algún botón mediante la función *OnClick="submit()"*.

Por último, vamos a comentar la función utilizada para actualizar periódicamente tanto la fecha y la hora como el consumo medido. Esto se ha realizado mediante una función creada en JavaScript llamada

peridicUpdate(). A continuación se adjunta dicha función en la página denomianada *index*:

```
<script language=JavaScript type="text/javascript" src="xml_http.js"></script>
<script>
    var timeUpdate = new periodicObj("time.cgx", 500);
    function periodicUpdateRTC(){
      updateMultiple(timeUpdate, plotRTCTime);
      rtc_elTime = setTimeout(periodicUpdateRTC, timeUpdate.period);
    }
    function plotRTCTime(){
      timeVal = document.getElementById("rtcTime").value;
      document.getElementById("rtc").textContent = timeVal;
      consumo = document.getElementById("cons_ref").value;
      jg1.refresh(consumo);
    }
</script>
```

Fragmento 3: Función updatePeriodic()

Como se puede observar, dicha función consiste en una llamada, cada 500ms, al archivo *time.cgx* de donde se obtienen periódicamente los valores de fecha y hora y de consumo. Cuando dichos valores son obtenido, se actualizan sus valores mostrados en las direfentes páginas web. A continuación se adjunta el código presente en el archivo *time.cgx*, el cual tiene un comportamiento idéntico a los archivos con extension `.CGI` antes mencionados:

```
t <form>
t <text>
t <id>rtcTime</id>
c h <value>%02d-%02d-20%02d %02d:%02d:%02d</value>
t </text>
t <text>
t <id>cons_ref</id>
c z <value>%04d</value>
t </text>
t </form>
.
```

Fragmento 4: Archivo time.cgx

4. Depuración y test

4.1. Pruebas software

4.1.1. Test Radio

El objetivo de esta prueba es la comprobación del correcto funcionamiento del módulo del sintonizador FM. Para navegar entre las direfentes pruebas de este test, se ha utilizador el botón azul (B1) de la propia placa.

Para ello, primero se manda el comando, mediante un *Thread* auxiliar, para encender la radio y se commprueba si se obtiene señal de audio a la salida.

A continuación, se sintoniza una frecuencia concreta y se comprueba, mediante una radio externa, si ambas señales de audio coincide.

Ahora, se realiza un *SeekUp* y se comprueba si la nueva frecuencia sintonizada es mayor a la anterior y si la calidad del audio a aumentado.

De manera análoga, se realiza un *SeekDown* y se comprueba si la frecuencia obtenida es menor a la anterior y también si la calidad de audio aumenta.

A continuación, se intenta sintonizar una frecuencia fuera de rango y se comprueba que solo se obtiene ruido a la salida.

Por último, se manda el comando para apagar la radio y se comprueba que ya no hay señal de auido a la salida.

4.1.2. Test MP3

El objetivo de esta prueba es la comprobación del correcto funcionamiento del módulo del reproductor MP3. Para navegar entre las direfentes pruebas de este test, se ha utilizador el botón azul (B1) de la propia placa.

Para ello, primero se inicia la reproducción de una canción concreta y se comprueba si a la salida se escucha esa canción.

Ahora, se manda el comando, mediante un *Thread* auxiliar, el comando que indica la reproducción de la siguiente canción de la lista y se comprueba si se obtiene a la salida.

A continucación, se indica al reproductor que ponga la anteior canción y se comprueba si se escucha dicha canción.

La siguiente comprobación es la puesta en pausa de la canción reproducida, para ella se manda dicho comando y se comprueba que no se obtiene salida.

De forma análoga, se le indica al reproductor que continue con la reproduccón de la canción y se comprueba que se obtiene la canción esperada.

Ahora, se intenta seleccionar una canción que no esté presente en la lista, comprobando que no se obitiene señal a la salida.

Por último, se comprueba el modo *loop*. Para ello se manda el comando indicado y se espera a que termine la canción actual y se comprueba que vuelve a comenzar y, de forma análoga, se indica al reproductor que termine dicho modo y se comprueba, al finalizar la canción actual, que no se obtiene señal a la salida.

4.1.3. Test RTC

El objetivo de esta prueba es la comprobación del correcto funcionamiento del módulo RTC.

Concretamente, se prueba si el reloj pasa el tiempo correctamente, si se pasa de minuto y hora adecuadamente y si la fecha se actualiza correctamente.

Implícitamente se comprueba el correcto funcionamiento de la alarma y, además, si la frecuencia a la que se activa es correcta.

También se comprueba si la sincronización con el servidor SNTP es correcta y si la hora actualizada coincide con la fecha y hora actuales.

Por último, se comprueba si los mensajes que envía al programa principal son del tipo correcto y si su contenido es el esperado.

4.1.4. Test Web

El objetivo de esta prueba es la comprobación del correcto funcionamiento del módulo del servidor web.

Para ello, se va a dividir este test en dos partes. En la primera, se comprobrá si las peticiones que se generan desde las diferentes páginas web se crean de manera correcta.

Para conseguir esto, se van pulsando sucesivamente todos los botones de todas lás páginas y se comprueba, en el módulo del servidor, si se generan de manera correcta.

También se comprueba si los mensajes que se generan para las diferenctes peticiones creadas son correctos y se envían de forma exitosa al programa principal.

A continuación, se procederá a comprobar si los valores mostrados en las páginas web se actualizan de manera correcta. Para ello, se enviará, desde un *Thread* auxiliar, diferentes modificaciones en los datos mostrado y se comprobrára si se actualizan de manera correcta.

Por último, se comprobará si tanto la hora y la fecha como el consumo se actualizan en tiempo real, implicitamente comprobrando el correcto funcionamiento de las funciones desarrolladas en JavaScript, por lo que se envían durante un cierto periodo de timepo y con una frecuencia de 1Hz, los valores de tiempo, fecha y consumo comprobando que en las diferentes páginas web se muestra de manera correcta.

4.2. Pruebas hardware

5. Presupuesto final

El presupuesto total del proyecto está recogido en la Tabla 1. Como se puede observar, tanto la tarjeta STM32F769NI como el sintonziador FM y el reproductor MP3, han sido cedidos por profesorado de la asignatura, por lo que su precio no se ve reflejado en el presupuesto total.

Concepto	**Precio**
Sensor NFC	5.00
Componentes Analógicos	65.00
PCBs	40.00
Batería	8.00
Tarjeta STM32F769NI	Cedido
Sintonziador FM	Cedido
Reproductor MP3	Cedido
Total	118.00

Tabla 1: Presupuesto del proyecto

6. Equipo de trabajo

7. Acrónimos utilizados