Trabajo práctico obligatorio

**Titulo del Proyecto**: *“UTN Wav Player”*

**Universidad**: Universidad Tecnológica Nacional

**Catedra**: Informática II

**Integrantes de la Catedra**: Marcelo Trujillo y Hernan Goin

**Curso**: R2002

**Grupo**: 5

**Integrantes**: Juan Pablo Peveri

**Año**: 2014

***Descripción breve del proyecto***

***Introducción:***

Decidí hacer un reproductor de sonido porque me interesó el aprender como hacen las computadoras y artefactos electrónicos para reproducir música o sonidos a traves de un parlante y poder lograrlo yo mismo en la placa Infotronic.

***Resumen del proyecto:***

El proyecto consiste en un reproductor de música de archivos WAV,el cual esta compuesto por la placa de Infotronic,la placa de expansión 3, el microcontrolador LPC1769, un parlante y una memoria SD. El reproductor lee desde la memoria SD un archivo de música WAV y lo reproduce a través del parlante conectado a la placa. A su vez, mientras la canción es reproducida el LCD de Infotronic muestra el nombre y autor de la canción en cuestión, y el tiempo actual de la misma es mostrado por los leds de 7 segmentos de la expansión 3.

En cuanto a la interfaz, el usuario puede, utilizando los botones de la expansión 3, reproducir, pausar y/o frenar una canción, adelantar o rebobinar y también ir a la próxima o anterior canción que se encuentre en la lista de reproducción.

La interfaz gráfica de la PC (imagen 3) incluye los elementos típicos de un reproductor de música (nombre,autor,tiempo total y actual de la canción en curso, volumen, botones de play, stop, pausa, próxima y anterior canción),una lista de reproducción y una ventana que muestra todas las canciones que hay en la tarjeta SD.

***Periféricos utilizados:***

-UART : Para la conexión via puerto serie entre la PC y el kit.

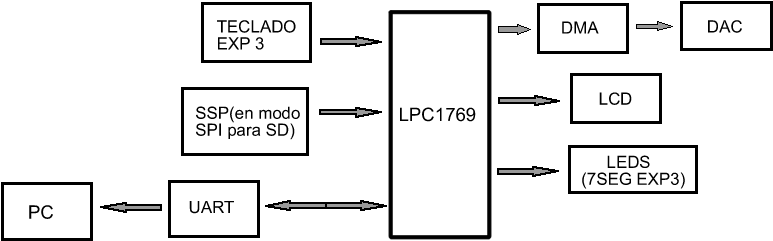
-SSP : Para la lectura de datos de la tarjeta SD.

- DMA: Para la transferencia de datos entre la memoria del LPC1769 y el DAC.

-DAC: Para la reproducción del sonido.

***Desafíos adoptados:***

El desafío principal que decidí realizar fue el lograr leer y reproducir archivos wav a traves del amplificador del kit de Infotronic. Para conseguirlo,necesité interiorizarme sobre dos módulos que no se enseñan en la catedra: el SSP para leer los datos de la tarjeta SD y el DMA para hacer las transferencias de datos con el DAC.También es necesario el comprender como se lee un archivo wav,teniendo en cuenta el sample rate,bits per sample,etc

***Esquemático del Proyecto:***

***Descripción detallada del proyecto***

***Especificaciones:***

* Para la lectura de la tarjeta SD se utiliza la librería realizada por Chan, la cual contiene los drivers para inicializar la SD y luego montarla con un sistema de archivo FAT para poder leer los archivos y directorios de la misma.
* El reproductor esta configurado para reproducir canciones con estas características unicamente:

- En formato crudo .wav, es decir, sin estar codificadas.

- Frecuencia de muestreo de 8khz

- 8 bits por muestra

- Mono

* Si se retira la tarjeta SD a la fuerza durante la ejecución del programa se interrumpe todo lo que se estaba haciendo y se emite un mensaje de error, y hasta que no se resetee manualmente la placa, el reproductor deja de funcionar.
* El reproductor tiene dos modos, uno esta activo si la placa esta conectada a la PC (modo PC) y el otro si no lo esta (modo placa). Cada vez que se cambia de modo,el buzzer emite un pitido para avisarle al usuario. En el modo placa los botones de la expansión 3 tienen funciones dependiendo del botón (Imagen 1) , y en el modo PC los botones son deshabilitados para que la placa solo responda a los comandos enviados desde el programa de Qt.
* Para poder conectar un mini jack stereo(imagen 2) a la salida del amplificador de la placa se utiliza un conector mini jack stereo hembra. Este tiene tres cables, uno va a masa y los otros dos son uno para el canal izquierdo y otro para el derecho. Como en la placa hay una sola salida para el audio solo existe un canal, entonces conectando el cable de masa del mini jack al de masa de la bornera del DAC y los otros dos cables de los canales juntos en el vivo de la bornera se puede escuchar el sonido en un auricular o parlante estéreo.
* Si se desconecta el cable RS232 desde la PC, la aplicación de Qt interrumpe lo que estaba haciendo, vuelve todo a su estado inicial y deshabilita todo hasta que se vuelva a conectar la PC con la placa. Si se reconecta el cable en el mismo puerto usb del cual se habia desconectado, la aplicación se encarga sola de conectarse nuevamente con la placa.
* Si deja de haber conexión entre la PC y la placa, el micro pasa automaticamente al modo placa hasta que se vuelva a establecer la conexion entre ambas y ahi pasa al modo PC.
* La luz roja encendida significa que la canción en reproducción esta pausada.

***Problemas encontrados a lo largo del desarrollo del TPO:***

Los dos principales problemas que surgieron al realizar el proyecto fueron el lograr acceder a un directorio dentro de la SD y que se escuche la canción por el parlante. En cuanto al primer problema, lo difícil fue el entender como se manejaba un sistema de archivo FAT. Si bien las funciones de acceso a la SD que se utilizaron fueron tomadas de las que hizo Chan, fue un desafío el entender como funcionaban y como leer archivos y directorios.

Por otro lado, una vez logrado el acceso a la SD y a la lectura de los archivos, la nueva complicación que aparecío fue el como hacer para que, teniendo la información para enviar al DAC, suene la canción y no un sonido desordenado e inentendible. Para lograr esto tuve que aprender conceptualmente como esta compuesto un archivo de música y con que frecuencia se debe enviar informacion al DAC para que esta se pueda escuchar. También necesité aprender a utilizar el DMA, el cual se encargó de enviar los datos de la canción al DAC. Al ser un periférico que no se enseñó a usar en clase, costó un poco entender como funciona.

Una vez superados esos dos problemas, no fue muy difícil encontrarle solución al resto de los inconvenientes que fueron surgiendo. Al realizar la interfaz gráfica costó un poco el entender como funcionan las señales,los slots y como comunicar las ventanas entre si, pero al haber en la web tantos soportes y tutoriales de Qt tanto por la empresa como por los usuarios, todo lo que tuvo que ver con la interfaz gráfica fue menos complicado que la parte del micro.

***Beneficios encontrados a lo largo del desarrollo del TPO:***

Realizar este proyecto me aportó mucho en cuanto a conocimiento principalmente, ya que no solo aprendí a aplicar lo que me enseñaron durante la catedra sino que ademas aprendí cosas que no forman parte de la materia, como el manejo del DMA, de la SD o el saber como leer archivos de audio. Tambien el desafío de partir de cero, con solo las herramientas que me enseñaron durante el año, y terminar haciendo un reproductor de musica fue algo que me enseño mucho a desenvolverme solo y buscar por mi cuenta la información y los recursos que necesité para realizar este TPO.

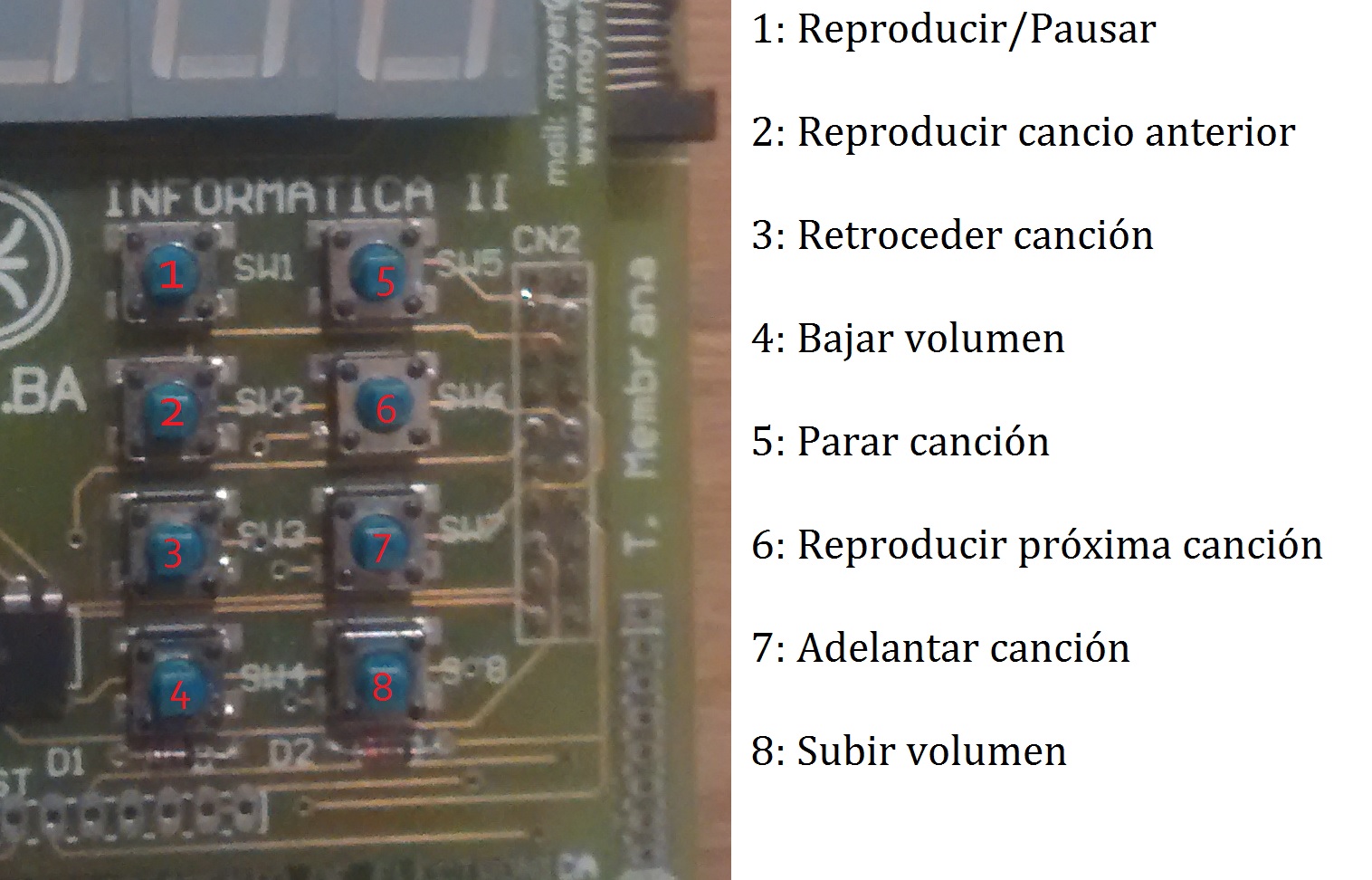
Si bien el reproductor funciona bien y tiene lo suficiente como para que un usuario lo pueda usar, quedaron ideas pendientes que podrian mejorar aun mas el proyecto pero por cuestiones de tiempo no se pudieron plasmar. Entre ellas estaba la idea de poder explorar las carpetas de la tarjeta SD desde la interfaz gráfica y no necesariamente tener que usar solo las canciones que aparecen en una carpeta seleccionada arbitrariamente. Tambien se podría mejorar el reproductor al agregarle un reconocimiento de voz, conectando un micrófono de ambiente a la entrada de ADC y grabando previamente comandos de voz como "pausar" o "próxima canción" para que luego durante su ejecución con solo decirlo el micrófono lo capte y haga la acción correspondiente.

***Conclusión:***

El proyecto del reproductor de musica resultó ser un desafío que me llevó varios meses el poder realizarlo. Aprendí mucho tanto en conocimiento sobre archivos de audio o en el uso del Qt como en aprender a desenvolverme solo, a investigar y a organizar y documentar codigo como hacen los profesionales.

***Imágenes:***

Imagen 1:

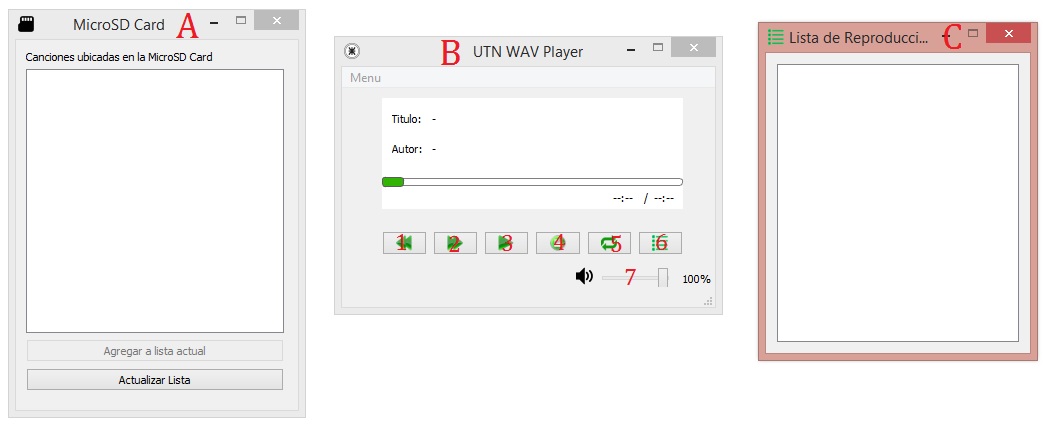


Im

Imagen 2:



Imagen 3:



A: Ventana que muestra todas las canciones encontradas en la tarjeta SD. Con el botón agregar a lista actual se agrega la canción seleccionada a la lista de reproducción

B: Ventana principal. Muestra la información de la canción en reproducción (titulo,autor y duración).

Botones: 1: Ir a canción anterior en lista de reproducción

2: Ir a próxima canción en lista de reproducción

3: Reproducir/Pausar

4: Parar

5: Habilitar/Deshabilitar modo de repetición (cuando termina la ultima canción vuelve a empezar la primera)

6: Abrir/Cerrar ventana de lista de reproducción

7: Control de volumen

C: Ventana de lista de reproducción. Muestra las canciones a reproducir.Se puede reproducir la canción que se quiera escuchar dando doble click sobre la canción deseada.

Apendice:

Como reanudar la conexion via puerto serie entre la pc y el micro automaticamente:

La libreria QextSerialPort contiene una clase llamada QextSerialEnumerator. Esta clase contiene, entre otras cosas, dos señales asociadas a dos slots llamadas deviceDiscovered() y deviceRemoved(). La primer señal se emite cuando se conecta un cable o dispositivo a alguno de los puertos de la PC y la ultima se emite cuando se desconecta algo de un puerto de la computadora. Para que ,si se desconecta el cable usb-rs232 desde la pc y se lo vuelve a conectar mientras la aplicacion este corriendo, esta reanude la conexion, se asocia mediante la funcion connect() las señales mencionadas anteriormente a dos slots hechos por uno mismo.

QextSerialEnumerator \*enumerator = new QextSerialEnumerator();

connect(enumerator, SIGNAL(deviceDiscovered(const QextPortInfo&)),

this, SLOT(onDeviceConnected(const QextPortInfo &)));

connect(enumerator, SIGNAL(deviceRemoved(const QextPortInfo &)),

this, SLOT(onDeviceRemoved(const QextPortInfo &)));

El slot conectado a la señal de deviceDiscovered() chequea si el nuevo puerto conectado es el mismo que se estaba usando para volver a conectar la aplicacion automaticamente, y el slot conectado a la señal deviceRemoved() chequea si el puerto desconectado es el que se estaba usando para avisarle a la aplicacion que ya no hay conexion con el microcontrolador.

/\*Slot que se ejecuta cuando se desconecta un dispositivo de la PC\*/

void pc\_rs232\_handler::onDeviceRemoved(const QextPortInfo &info)

{

if(Timer\_devices->remainingTime()==0 || Timer\_devices- >isActive()==false)

{

if (info.portName==puerto) //si el cable que se desconecto es el que estaba usandose,se avisa que se corto la conexion

emit SinConexion();

}

}

/\*Slot que se ejecuta cuando se conecta un dispositivo a la PC\*/

void pc\_rs232\_handler::onDeviceConnected(const QextPortInfo &info)

{

if (info.portName==puerto && flag\_connected==false) //si el cable que se conecto es el que estaba usandose y estaba desconectado,se conecta automaticamente

this->Connect();

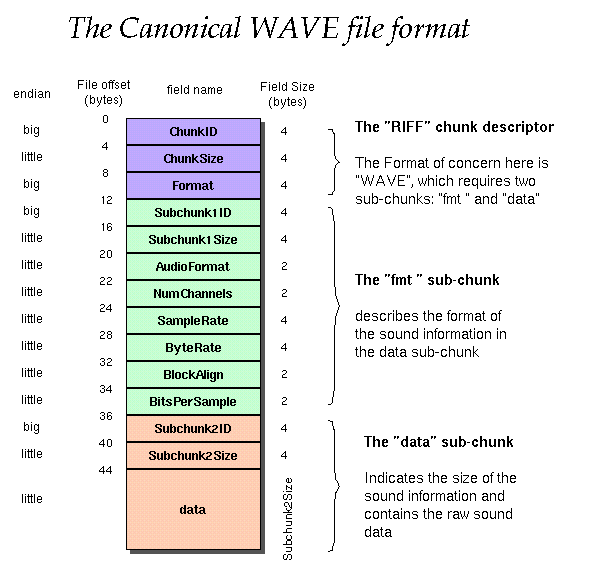
Timer\_devices->start(2000); //Pongo un delay porque hay un error en la señal,ni bien conecto el cable lo toma como que se desconecto al instante,por eso esperando ese tiempo no llega a desconectarse ni bien lo conecto

}

NOTA: Para que se reconecte automaticamente el usuario debe volver a conectar el usb en el mismo puerto de donde lo habia desconectado, de lo contrario habra que volverlo a conectar manualmente

Manejo de archivos de formato WAV:

El tipo de audio WAV, es un tipo no codificado realizado por Windows. Su manera de manejarlo es facil, ya que solo los primeros 44 bytes del archivo contienen informacion del mismo y luego el resto son conjuntos de bytes "crudos" que se envian directamente al DAC para reproducir el sonido. Pasemos al analisis de los primeros 44 bytes.



De estos 44 bytes, los mas importantes para leer al momento de tener que reproducir un archivo WAV son:

* Byte 22-23 NumChannels: Indica cuantos canales tiene (1= Mono y 2=Stereo)
* Byte 24-27 SampleRate: Frecuencia de muestro en Hertz
* Byte 28-31 ByteRate: Cantidad de bytes por segundo que se deben leer
* Byte 34-35 BitsPerSample: Cantidad de bits por muestra
* Byte 40-43 SubChunk2Size: Indica el tamaño del archivo en bytes( sacando los 44 primeros).

Teniendo esta informacion, se puede saber como configurar el DMA y el DAC para enviar cierta cantidad de paquetes de bytes a una frecuencia determinada y lograr que por el parlante se escuche el sonido.

Funcionamiento del DMA

El DMA significa Direct Memory Access (acceso directo a memoria). Este periferico se encarga de enviar datos desde una fuente(memoria o periferico) a un receptor(memoria o periferico). De esta manera, el DMA da la posiblidad de realizar cuatro tipos de transferencias de datos: Memoria a Periferico, Memoria a Memoria, Periferico a Memoria y Periferico a Periferico. Esto es de mucha utilidad en casos donde se tiene que estar constantemente enviando datos de un lugar a otro ya que si de esto se encargase el procesador, tendria menos capacidad para ocuparse de otros asuntos.

Al DMA se lo inicializa con los siguientes pasos:

* Se enciende el periferico
* Se configura la logica de sincronismo (se habilita o deshabilita)
* Se habilita el DMA con logica little-endian o big-endian
* Se configura la interrupcion del DMA en el NVIC

**void** **InitDMA** ( **void** )

{

PCONP |= 1<<29; //Energizo el periferico

DMACSync = 0xFFFF;//Deshabilito la lógica de sincronismo

DMACIntTCClear = 0x01; //Deshabilito interrupciones pendientes

DMACConfig = 0x00000001; //Habilito el DMA, con logica little endian

ISER0 |= (1<<26); //Habilito interrupciones del DMA en NVIC

}

Luego de inicializar el periferico, se debe configurar el canal a utilizar,ya que el DMA del lpc1769 tiene ocho canales. Una vez decidido que canal se va a utilizar se procede a inicializar el canal:

* Selecciono la direccion de memoria del origen de la transmision
* Selecciono la direccion de memoria del destino de la transmision
* Configuro el registro DMACCxControl para seleccionar el tamaño total de la transferencia, configurar el modo BURST, seleccionar el tamaño de los datos de origen y de destino,indicar si se debe incrementar la direccion de origen y/o destino al realizar una transferencia y seleccionar si al terminar la transferencia se debe interrumpir o no.
* Configuro el registro DMACCxConfig para habilitar el canal, indicar el origen , destino(memoria o algun periferico en particular) y el tipo de transferencia y seleccionar si al terminar la transferencia se debe interrumpir o no.

Nota: Si se desea que el DMA interrumpa al finalizar una transferencia se debe habilitar la interrumpcion en el NVIC, en el registro DMACCxControl y en el registro DMACCxConfig. Si alguno de estos registros no tiene la interrupcion seteada no se producira la misma.

Este es un ejemplo de la inicializacion del canal 0 del DMA para transferir datos de un buffer de memoria a el periferico DAC,y que el DMA interrumpa al finalizar la transferencia.

**void** **DMAChannelInit**(**unsigned** **int**\* buffer)

{

//Canal 0 del DMA

//Borro datos anteriores

DMACC0Control = 0;

DMACC0Config= 0;

DMACIntTCClear = 0x01;

//Source: buffer

DMACC0SrcAddr = (uint32\_t)buffer;

//Dest:DAC (registro del DAC)

DMACC0DestAddr = (uint32\_t((uint32\_t\*)0x4008C000UL ); // DACR

/\*

Control:

Tamanio de la transferencia: 1024

Cantidad de transferencias por burst en SRC: 1 (0x00)

Cantidad de transferencias por burst en DST: 1 (0x00)

Source width (tamanio de los datos en origen): 32 bits (0x02)

Dest width (tamanio de los datos en destino) : 32 bits (0x02)

Source increment (se debe incrementar la direccion de source: SI (1)

Dest increment (se debe incrementar la direccion de destino: NO (0)

Activacion de la interrupcion al finalizar la transferencia: SI (1)

\*/

DMACC0Control = 1024 | (0<<12) | (0<<15) | (2<<18) | (2<<21) | (1<<26) | (0<<27) | (1<<31);

DACCTRL |= 1<<3; //Habilito el pedido de transferencia desde el DAC

/\*Configuro y disparo el canal 0 de DMA:

Habilito el canal (1)

Source: Memory (nada)

Dest: DAC (7)

Tipo de transferencia (memory to peripheral): (1)

Interrupcion por finalizacion habilitada: SI (1)

\*/

DMACC0Config = 0x01 | (0x00<<1) | (7<<6) | (1<<11) | (1<<15);

}

Por ultimo, cuando interrumpe el DMA, lo unico que se debe hacer es limpiar el flag de interrupcion y luego realizar la accion que uno quiera.

**void** **DMA\_IRQHandler** ( **void** )

{

DMACIntTCClear = 0x01;

//Realizo las acciones que quiera

}

Funcionamiento de la SD

Para leer la tarjeta SD se debe inicializar el periferico SSP en modo SPI y enviar una serie de comandos a la SD card necesarios para inicializarla. Luego de esto, se debe montar la tarjeta con el filesystem FAT para poder acceder a los archivos y directorios. De todo esto se encargan las funciones disk\_initialize() y f\_mount(), las cuales se encuentran en la libreria realizada por Chan e incluida en este proyecto.

La funcion disk\_initialize() tiene como parametro un char para indicar si se va a inicializar una SD o un USB (para SD siempre poner "0") y devuelve 0 al menos que haya un error en la inicializacion.

La funcion f\_mount() usa dos parametros, el primero es el mismo que el de disk\_initialize() y el segundo es para indicar la direccion de memoria de la variable de tipo FATFS donde se guardara la informacion de la SD montada. Esta variable debe permanecer durante todo el tiempo que se use la SD ya que si se borra su informacion no se podra acceder mas a la tarjeta durante la aplicacion.

Un ejemplo de la inicializacion de ambas funciones:

**int** **main**(**void**)

{

FATFS G\_fatfs;

disk\_initialize(0)); //Inicializa la SD

f\_mount(0, &G\_fatfs); //Monta la SD y guarda la informacion en G\_fatfs

}

Una vez inicializada ya se puede acceder a los directorios y archivos de la tarjeta SD. Para eso se utilizan las funciones f\_opendir() y f\_readdir().

La funcion f\_opendir() se encarga de abrir un directorio dado por una direccion y guardar su informacion en una variable de tipo DIR.Esta funcion utiliza como parametros la variable de tipo DIR que es donde se guarda la informacion del directorio abierto y un string que indica la ruta del directorio a ser abierto. Si se lo pudo abrir la funcion devuelve 0.

Ejemplo:

DIR direccion;

f\_opendir(&direccion,"0:/Musica");

La funcion f\_readdir() toma un tipo de dato DIR pasado como parametro para abrir un directorio y guardar en una variable de tipo FILINFO,la cual tambien es pasada como parametro, el nombre del archivo o directorio leido dentro del directorio abierto. La funcion va leyendo uno por uno las carpetas(directorios) o archivos dentro del directorio dado hasta que llega al final y guarda en la variable de tipo FILINFO un 0, indicando que ya no hay nada para leer. f\_readdir() devuelve 0 si no hubo ningun error al leer.

Ejemplo:

DIR direccion;

FILINFO fno;

f\_opendir(&direccion,"0:/Musica");//Abre el directorio de la ruta indicada

res = f\_readdir(&direccion, &fno); //Lee un elemento del directorio

Nota: para saber si lo que se leyo con f\_readdir() es una carpeta o un archivo se hace esta operacion:

fno.fattrib & AM\_DIR

Si lo que devuelve esta operacion es 0, lo que se leyo fue un archivo, y en caso contrario, lo que se leyo fue un directorio.

Por ultimo, ya pudiendo acceder a la SD y sabiendo todos los archivos y carpetas que esta contiene con sus respectivas rutas, solo queda escribir o leer los archivos. Esto funciona practicamente igual que cuando se manejan archivos en C en la PC. Las funciones utilizadas son f\_open(), f\_close(), f\_read() y f\_write().

La funcion f\_open() toma como parametros el fichero donde se guardara la informacion del archivo abierto(una variable de tipo FIL), la ruta del archivo a abrir y el tipo de lectura/escritura con el que se manejara al archivo.Si no hubo errores al abrir el archivo la funcion devuelve 0.

La funcion f\_close() cierra el archivo abierto por f\_open() tomando como parametro la variable FIL que guarda la informacion del archivo en cuestion.

Ejemplo:

FIL G\_fp;

f\_open(&G\_fp,"0:/Musica/archivo.wav",FA\_OPEN\_EXISTING | FA\_READ);

f\_close(&G\_fp);

La funcion f\_read() lee una cantidad de bytes desde un fichero FIL , lo guarda en una variable e indica cuantos bytes se leyeron realmente de los que se pidieron. f\_read() toma como parametros el fichero de tipo FIL, la variable donde se guardara la informacion leida, la cantidad de bytes que se leeran, y la variable donde se guarda la cantidad de bytes que realmente se leyeron.

Ejemplo:

FIL G\_fp;

uint32\_t bytes;

**char**\* buffer;

f\_open(&G\_fp,"0:/Musica/archivo.wav",FA\_OPEN\_EXISTING | FA\_READ);

f\_read(&G\_fp,buffer,1024,(UINT\*)(&bytes));

f\_close(&G\_fp);