PROYECTO SISTEMAS ELECTRÓNICOS DIGITALES

CURSO 2021/2022

MEMORIA TRABAJO MICROS

REPRODUCTOR DE MUSICA CON PANTALLA LCD Y CONTROL REMOTO CON STM32F407

INTEGRANTES DEL GRUPO:

DAVID HERGUETA SOTO	Nº MAT: 54666
IGNACIO SÁNCHEZ APARICIO	Nº MAT: 54854
DEWEI ZHOU	Nº MAT: 54912

REPOSITORIO:

https://github.com/davidherguetasoto/Reproductor_musica_STM32.git

ÍNDICE

INTRODUCCIÓN

En este proyecto nos hemos aprovechado de las diferentes funcionalidades que incluyen nuestra placa de trabajo Discovery STM32F407 como son el procesador de audio CS43L22, el protocolo de comunicación síncrona I2S utilizada para trasmitir audio, el protocolo I2C y la capacidad de conexión de un USB gracias al middleware USB_HOST integrado. Todo esto unido a la configuración de los diferentes pines que detallaremos más adelante y el código de implantación organizado en 5 librerías que controlan tanto la conexión USB como la reproducción de audio la placa es capaz reproducir cualquier archivo en formato WAV que se encuentre almacenado en un pendrive. También se utiliza un LCD que mediante el protocolo I2C se comunica con la placa, el sensor de temperatura integrado en el microprocesador y un mando inalámbrico para controlar la reproducción que se comunica con la placa mediante el protocolo NEC. Tanto el LCD como el mando cuentan con su respectiva librería en la que recogemos las funciones que hacen funcionar a ambos dispositivos.

Los pasos a seguir a la hora de poner en funcionamiento el reproductor son muy sencillos, simplemente basta con insertar el USB aunque para ello es necesario contar con un adaptador USB OTG ya que la entrada que tenemos en la placa es de tipo micro A. Conectamos también una salida de audio como unos auriculares o un altavoz con cable jack macho y tras ejecutar el código en el espacio de trabajo la música guardada en el pendrive se empezará a escuchar, en el LCD se observará el nombre de la canción que se está reproduciendo junto a la temperatura de la sala y con el mando se podrá tanto pasar a la siguiente canción, volver a la anterior, pausar la reproducción, continuarla y subir o bajar el volumen.

MATERIAL UTILIZADO

PANTALLA LCD LCM1602A

Módulo de pantalla LCD con matriz de 16x2 caracteres en el que se le mostrará al usuario cierta información, como la música que se está reproduciendo, el estado de la reproducción (play/pause), y la temperatura en el exterior.

Se comunicará con el microcontrolador a través de puerto serie con protocolo I2C.

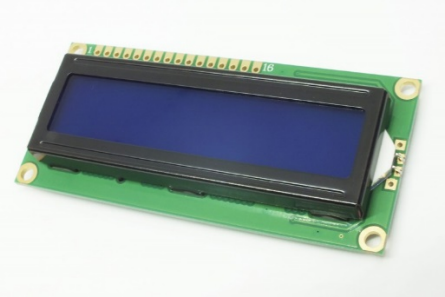

Ilustración 1. Módulo pantalla LCD 16x2 modelo LCM1602A.

DIODO RECEPTOR DE LUZ INFRARROJA AX-1838HS

Receptor de señales infrarrojas para controles remotos u otras aplicaciones, capaz de detectar señales de entre 850 y 1050 nm de longitud de onda.

Por defecto su salida se encuentra a nivel lógico alto, y cuando recibe una señal, pasa a nivel lógico bajo el tiempo que dure dicha señal.

Se empleará para la captación de las señales emitidas por el control remoto.

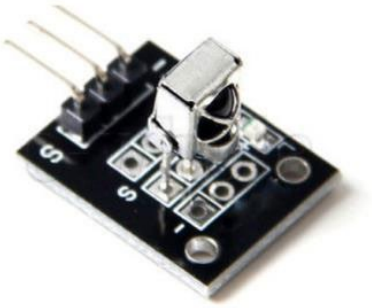

Ilustración 2. Diodo detector de señales infrarrojas modelo AX-1838HS.

CONTROL REMOTO UNIVERSAL

Control remoto desde el que se enviarán las órdenes al microcontrolador para manejar la reproducción de la música a través de señales infrarrojas.

El mando envía señales con una portadora de 38KHz codificadas aplicando formato NEC.

Ilustración 3. Control remoto universal utilizado para el control de la música reproducida.

MEMORIA USB

Se almacenará la música que se desee reproducir en una memoria USB en formato WAV a 44Khz. Se conectará al microcontrolador a través del puerto micro USB OTG.

PROTOCOLOS UTILIZADOS

Protocolo I2S

Una fuente de audio digital normalmente creará dos líneas de datos, una para el canal izquierdo y otra para el derecho por para cada intervalo de muestra. I2S proporciona una línea de selección de línea (también llamada a veces reloj L/R) para seleccionar muestras izquierda o derecha, y una línea de reloj de bits para mantener todo sincronizado, de esta manera se reduce el jitter producido por el reloj de la señal de datos.

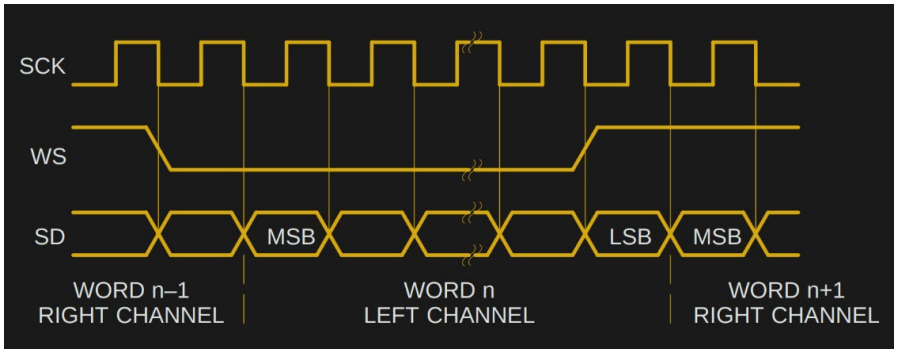

Ilustración 4 funcionamiento I2S

Protocolo I2C

Este protocolo lo hemos estudiado en el temario de clase, I2C es un puerto y protocolo de comunicación serie, define la trama de datos y las conexiones físicas para transferir bits entre 2 dispositivos digitales. El puerto incluye dos cables de comunicación, el de datos y el de reloj, SDA y SCL. En este proyecto se utiliza para la comunicación con dispositivos externos como serían los auriculares o el altavoz. También se utiliza para la comunicación del LCD.

Protocolo NEC

Protocolo empleado para el envío de datos con señales infrarrojas. Se basa en la detección de intervalos entre pulsos de la onda portadora. En función del tiempo que tarde, significará un 0 o un 1 lógico.

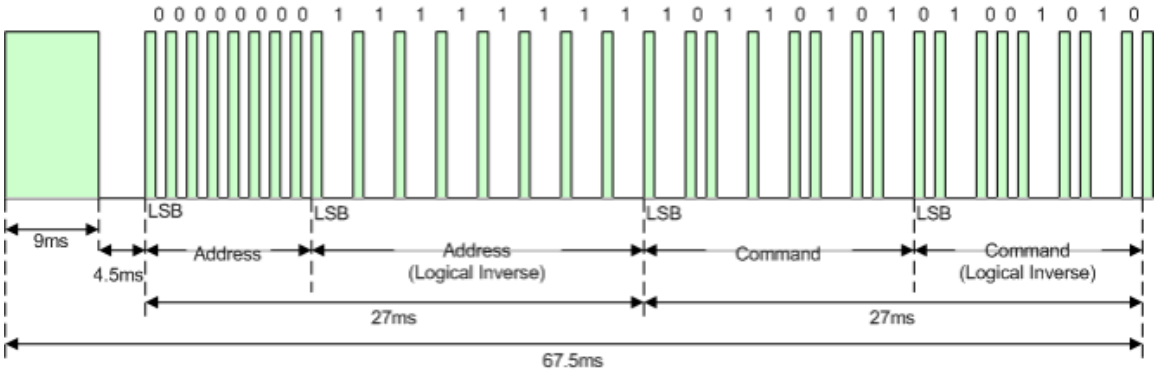

Ilustración 5. Señal codificada en formato NEC.

En primer lugar, se envía un pulso de 9ms de duración de la onda portadora. Tras un silencio de 4.5ms comenzará el envío del mensaje.

El mensaje enviado se compone de 32 bits ordenada de la siguiente manera:

- 8 bits con la dirección del dispositivo emisor.
- 8 bits con la dirección negada del dispositivo emisor.
- 8 bits con los datos que el dispositivo emisor quiera mandar.
- 8 bits con los mismos datos que el dispositivo ha enviado, negados.

Esta manera de codificar los mensajes con 8 bits de información, y 8 bits con la misma información negada, permiten la detección de errores.

Los 0 o 1 lógicos se codificarán también según la duración de los silencios tras los pulsos de la señal portadora.

- Si el tiempo entre pulsos es de 2.25ms será un 1 lógico.
- Si el tiempo entre pulsos es de 1.12 ms será un 0 lógico.

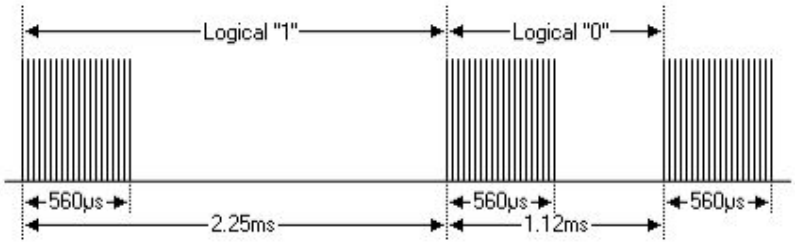

Ilustración 6. Codificación de 0 y 1 en formato NEC.

El receptor empleado generará una señal de salida inversa a la recibida, es decir, cuando se produzca un pulso en la portadora, el receptor producirá un cero a la salida de duración igual a la señal, y cuando se produzca un silencio, el receptor producirá un 1 lógico de igual duración.

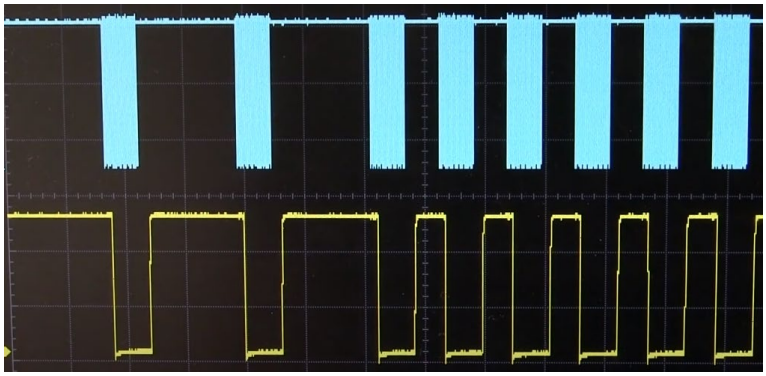

Ilustración 7. Comparativa seña infrarroja enviada en protocolo NEC (azul) y señal transcrita por el receptor infrarrojo (amarilla).

CONFIGURACIÓN PINES Y RELOJ

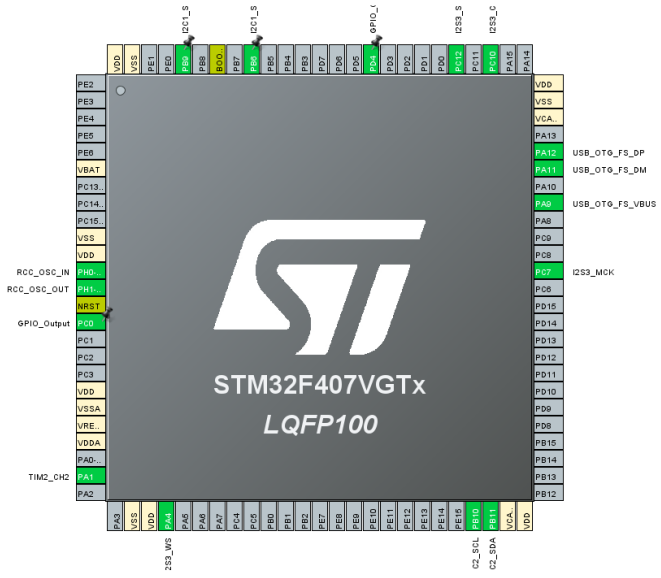

Ilustración 8. Esquemático de la configuración de los pines del microcontrolador.

Siguiendo la hoja de características proporcionada por STM Electronics configuramos los periféricos que necesitamos.

Configuración audio

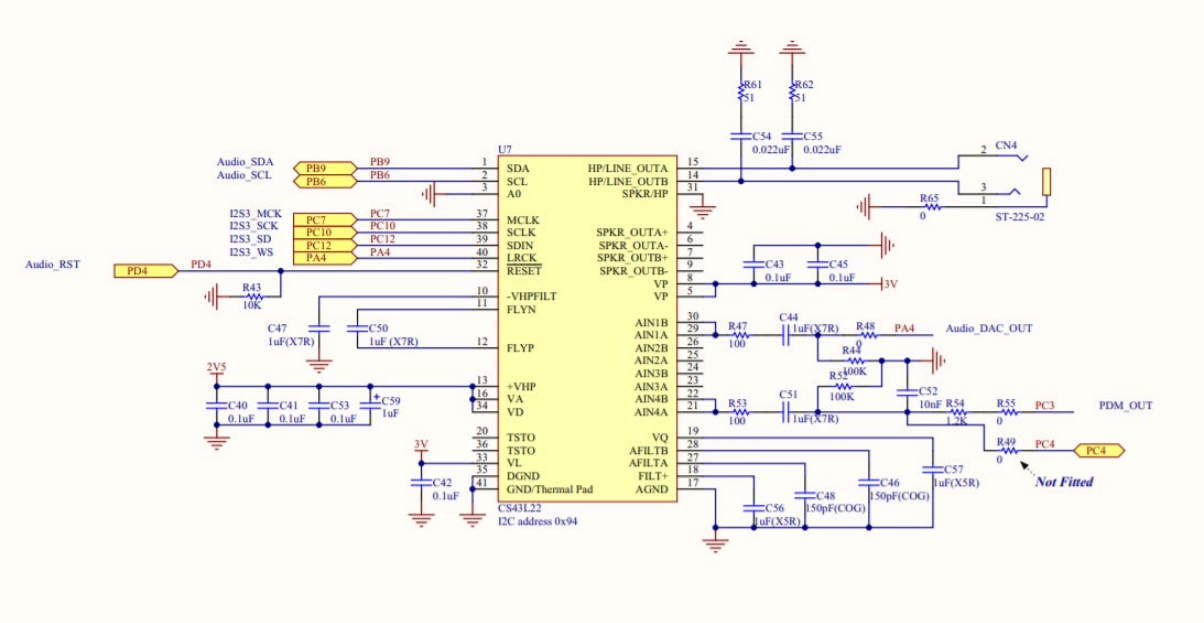

Ilustración 9: Esquema del bloque de audio

- Los pines PB6 y PB9 se configurarán en el modo I2C.
- Al configurar el I2S en modo half duplex los pines PC12 y PC4 son configurados.
- El master clock tal y como dice el esquema seguido será seleccionado en el PC7.
- El I2S clock lo colocaremos en el PC10.

Configuración USB

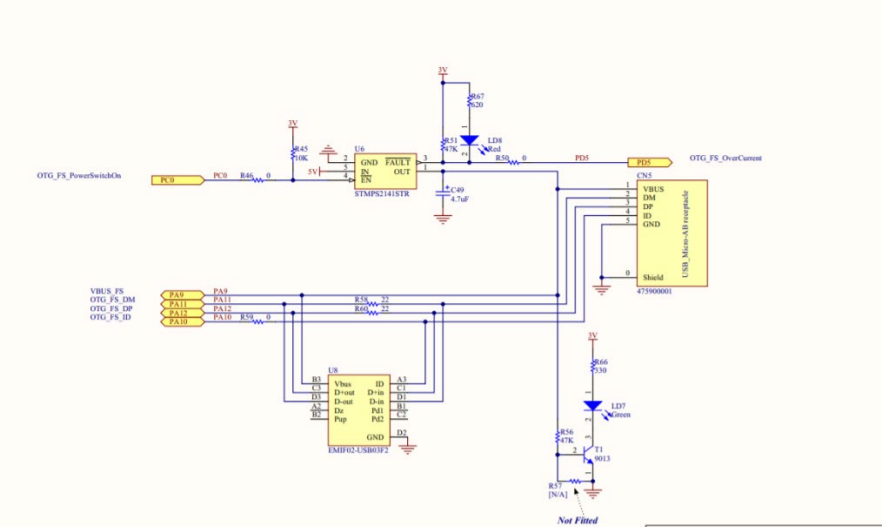

Ilustración 10: Esquema OTG

- Activamos el modo *Host*.
- PC0 se configura como salida y tendremos cuidado que se activa a nivel bajo.
- Activamos la opción *Mass Storage Host Class, de manera que así pueda tener acceso al almacenamiento del USB*

Configuración sensor de temperatura

Cada 5 segundos se realizarán tres lecturas del sensor de temperatura integrado en la placa para posteriormente hacer una media.

Para ello se configurará el conversor en mod DMA, y será disparado por el temporizador TIM3 (flanco ascendente).

- Configuración basic timer TIM3:

-Preescaler:7199.
-Period:49999.
El TIM3 se encuentra conectado al bus APB1 de la placa, funcionando a 72MHz. De esta manera se obtendrá un temporizador de 10KHz que cuente 5 segundos.

- Se utilizará el convertidor A/D: ADC1.
- Configuración de ADC1:

-Resolución: 10bits.
-Scan Conversion Mode: Enabled.
-DMA Continuous Requests: Enabled.

- Modo de Conversion de ADC1:

-Number Of Conversion: 3, a 480 ciclos y en el canal del sensor de temperatura cada una.
-External Trigger Conversion Source: Timer 3 Trigger Out evento.
-External Trigger Conversion Edge: Trigger detection on the rising Edge.

- Configuración del DMA2:

- Circular mode con un ancho de media palara (16 bits).

Configuración LCD 16x2

El LCD se comunicará a través del puerto I2C2.

- El SDA (Serial Data) lo colocaremos en el pin PB11.
- El SCL (Serial CLock) lo colocaremos en el pin PB10.

Configuración control remoto

Para la lectura de las señales recibidas por el recepto infrarrojo, se ha configurado el timer TIM2 en modo INPUT CAPTURE de manera que genere una interrupción y se pueda contar el tiempo entre flancos de la señal para la decodificación de los datos.

- Configuración del timer TIM2:
  Clock source: Internal clock
  Channel 2: Input Capture Mode
  Prescaler: 7199
  Counter period: Máximo permitido
  Polarity Selection: Falling Edge (se activa con flanco descendente de la señal del canal 2).

El TIM2 se encuentra conectado al puerto APB1 del microcontrolador, configurado con un reloj de 72MHz, por lo que con el prescaler generaremos un temporizador de 10Khz.

Se ha elegido el TIM2 porque tiene una resolución de 32bits, mayor que el resto, lo que nos permite que pueda contar más tiempo sin desbordarse. Perfecto para el modo de captura de entrada.

El canal 2 del TIM2 habilita el pin PA1 del microcontrolador, al que se le conectará la salida del receptor IR.

Configuración reloj

Se ha habilitado el reloj externo con una frecuencia de 8MHz para la utilización del USB OTG y el I2S.

Se han tratado de sincronizar todas las señales de relojes a frecuencias que permitan un funcionamiento correcto del puerto USB, el muestreo correcto de las canciones almacenadas, y la temporización llevada a cabo por los timers.

El HCLK funcionará a 72 MHz, y el I2S clock a 90MHz.

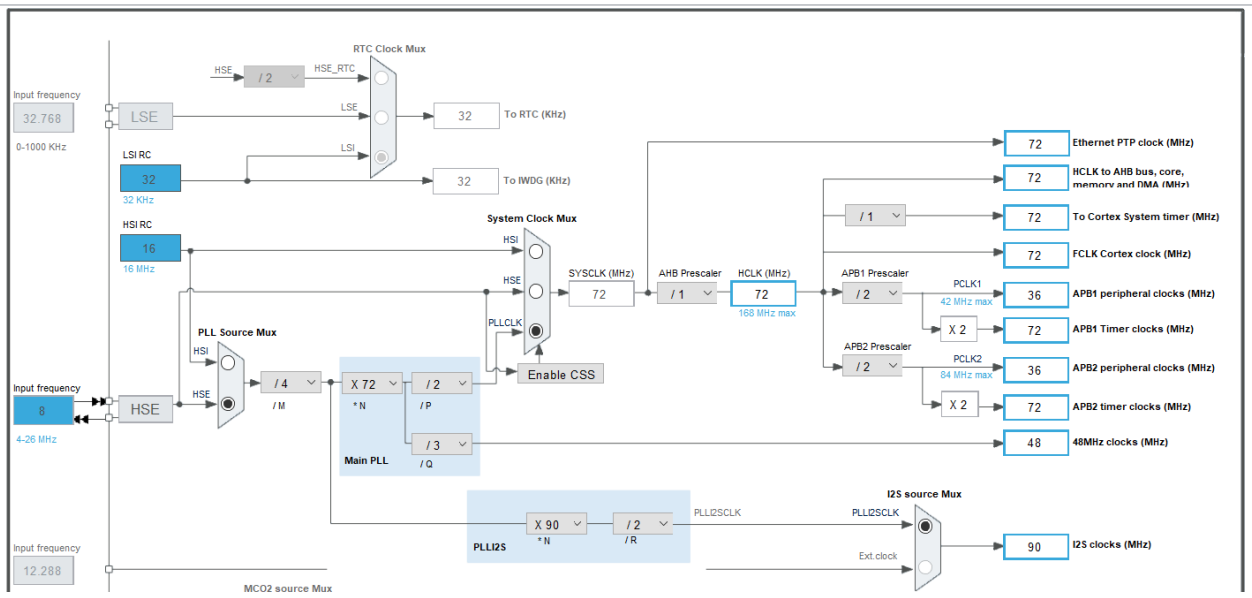

Ilustración 11: Clock tree

LIBRERÍAS

AUDIO Y USB

Se incluyen 5 librerías que nos permitirán la reproducción de audio en la placa.

-AUDIO_LINK: librería proporcionada por el fabricante, se encarga de hacer el link entre el audio y el protocolo I2C, utilizan las funciones **AUDIO_IO_WRITE y AUDIO_IO_READ** para leer y escribir los datos de audio.

-AUDIO: librería proporcionada también por STM, esta librería contiene las funciones relacionadas con el audio como puede ser la inicialización del driver: **AUDIO_OUT_Init** o las de control de reproducción como **AUDIO_OUT_Play**, **AUDIO_OUT_Pause,** **AUDIO_OUT_SetVolume** que respectivamente, activan, pausan o seleccionan el volumen.

- CS43L22: librería proporcionada por STM, es la encargada de todo lo relacionado con la codificación de audio de la placa.

-FILEHANDLING: esta librería es utilizada para ayudarnos a identificar los archivos WAV del USB conectado con la función **AUDIO_StorageParse,** gracias a esta función es posible conseguir también el nombre del archivo. En esta librería también encontramos la función **Mount_USB** con la que montamos el USB en el micro. Es importante que en esta librería está definida la estructura

_FILELIST_FileTypeDef que será el tipo utilizado para definir los archivos tipo WAV almacenados en el USB.

-WAVPLAYER: por último, tenemos esta librería proporcionada por el fabricante, que nos servirá también para trabajar con el audio con funciones como **AUDIO_PLAYER_Process**, que es la función encargada de reproducir en bucle las canciones guardadas y que incluye opcionalidades para pasar de canción, subir o bajar el volumen y reiniciar o pausar la reproducción. **AUDIO_PLAYER_Start** será la función encargada de abrir el archivo WAV y leerlo, obteniendo características del archivo como la frecuencia de muestreo. Esta librería se complementa con la librería de audio, mientras esta se enfoca más en obtener y procesar el archivo en formato WAV, la otra se encarga de configurar los periféricos de audio y configurar el codec.

LCD 16X2

lcd16x2_i2: esta librería es la encargada de proporcionar las funciones con la que podremos enviar la información al LCD, se emplea el protocolo de comunicación I2C, encontramos funciones para elegir la posición del cursor en el LCD como **lcd16x2_i2c_setCursor**, borrar el contenido de la pantalla como **lcd16x2_i2c_clear** y escribir en la pantalla como **lcd16x2_i2c_printf.**

CONTROL REMOTO

Para la decodificación y recepción de los datos recibidos del sensor infrarrojo, se ha creado la librería **nec_ir_stm32** con funciones que permiten este cometido en microcontroladores STM.

Contiene comentarios explicativos del código y de la librería en su fichero de cabecera.

- **void getDataIR(uint32_t* buffer, uint32_t time, uint8_t* END)** será la encargada de decodificar la información recibida en formato NEC. Habrá que pasarle un puntero a entero *buffer* en el que se almacenará la información recibida, un entero *time* que será el tiempo que ha pasado entre dos pulsos de la señal, y un puntero a entero *END* que se utilizará como bit de fin. Se le introducirá un 1 cuando la recepción del mensaje haya concluido.
  Los intervalos de tiempos empleados en la función para la decodificación de la señal están pensados para usarse con un temporizador de 10Khz.

- **void decodeIR(uint32_t* buffer, uint8_t* device_code, uint8_t* data)** se empleará para obtener los datos enviados y la dirección del dispositivo emisor almacenada previamente. Habrá que pasarle un puntero a entero *buffer* que deberá contener la información decodificada del receptor IR, y dos punteros a enteros *device_code* y *data* en los que se almacenará la dirección del dispositivo emisor y la información recibida respectivamente.
- **void getButton(uint8_t data, char* texto).** Esta función introducirá una cadena de caracteres en la variable *texto* con el nombre de la tecla pulsada en función del dato almacenado en *data*. (Esta función no ha sido necesario usarla en el proyecto).

DESARROLLO DEL CÓDIGO

VARIABLES GLOBALES

Las variables de tipo extern están definidas fuera del fichero de código fuente main.c, “Appli_state” de tipo extern ApplicaationTypeDef almacena el estado del USB y AudioState es una variable que almacena el estado de la reproducción del archivo Wav. Las variables de tipo volatile son variables que se emplean en interrupciones y pueden tomar otro valor en cualquier momento debido a esa interrupción.

```
//VARIABLES DATOS INFRARROJOS
volatile uint32_t buffer=0;     //Buffer para el almacenamiento de los datos recibidos por el IR
uint8_t device=0;               //Código del elemento emisor
uint8_t mensaje=0;              //Código del mensaje recibido
volatile uint8_t flag_end=0;    //Marca para indicar que se han recibido nuevos datos del sensor IR. 0 si no lo ha hecho, 1 si si

//VARIABLES FICHEROS Y CONTROL DE MUSICA
uint8_t IsPlaying=0;            //Marca para indicar si se esta reproduciendo musica. 1 si se esta reproduciendo, 0 si no
extern ApplicationTypeDef Appli_state;
extern AUDIO_PLAYBACK_StateTypeDef AudioState;
char nombre[40];                //Nombre de la canción que se está reproduciendo
uint16_t idx;                   //Indice de la canción que se está reproduciendo
extern FILELIST_FileTypeDef FileList;  //Lista de canciones generada a partir de los archivos del USB
uint8_t IsFinished=0;           //1 si ha terminado de reproducir la cancion

//VARIABLES SENSOR TEMPERATURA
volatile uint16_t ADC_value[3]={0,0,0}; //Valor leído del sensor de temperatura
float temperatura=0;            //Temperatura media de las tres medidas
float Vsense=0;                 //Tension medida con el sensor
uint8_t convCompleted=0;        //1 si se ha completado una lectura del sensor
```

CALLBACKS Y FUNCIONES EMPLEADAS

HAL_TIM_IC_CaptureCallback

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef* htim)
{
    if(htim->Instance==TIM2)
    {
        if(!HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1)) //Nos aseguramos de que esté la señal del receptor IR a nivel bajo
        {
            getDataIR(&buffer,__HAL_TIM_GET_COUNTER(htim),&flag_end);
        }
        __HAL_TIM_SET_COUNTER(htim,0); //Volvemos a poner el temporizador a cero para la siguiente cuenta

    }
}
```

Este callback asociado a las interrupciones de los temporizadores, es utilizado en el funcionamiento del control remoto. Su función es que cada vez que se envíe una señal del mando, se llame a la función getDataIR y se le pase el tiempo registrado en el TIM2, de manera que se pueda interpretar qué señal del mando se ha enviado. Esta función escribe en la variable flag_end, y la pone a 1 una vez se han leído y almacenado en buffer todos los bits de la señal.

HAL_ADC_ConvCpltCallback

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    convCompleted=1;
}
```

Este callback sencillo se emplea en la toma de temperatura. El temporizador tim3 está configurando para que cuente 5 segundos. Cuando termina la cuenta, se generará una señal que disparará el conversor, leyendo 3 medidas de temperatura. Una vez se han tomado, se llama a la callback que cambia el valor de convCompleted a 1. para que entre en la parte del código donde se calcula el valor a mostrar.

Aprovechando y para que se entienda mejor la función del callback anterior explicaremos en este apartado el funcionamiento del sensor de temperatura. Tal y como se observa en la imagen en la que se muestra el código del proceso que acabamos de comentar, podemos ver cómo, para obtener la temperatura a mostrar por el LCD, se realiza una media de tres valores captados por el sensor. Estos valores son voltajes que, mediante las operaciones mostradas, se transforman en un valor temperatura en grados centígrados. El valor medido en el ADC convertido a voltios es Vsense. Es de 10 bits que es multiplicado por el valor de referencia el cual hemos considerado 2910.0mV (2.91V)

tras medir el voltaje de referencia de la placa con un polímetro, ya que con el el valor de 3.3V, resultaba en una temperatura final alejada de la real. A dicho Vsense se la resta el valor de voltaje a 25 grados el cual es 0.76V tal y como indica la hoja de características y se divide entre la sensibilidad que es 0.0025, es decir 2.5mV/ºC.

```
if(convCompleted)
{
    temperatura=0.0f;
    Vsense=0.0f;
    for(int i=0;i<3;i++)
    {
        Vsense=((2910.0/0x3FF)*(float)ADC_value[i])/1000.0;
        temperatura+=((Vsense-0.76)/0.0025)+25;
    }
    temperatura/=3;
    imprimeLCD(1, IsPlaying, nombre, temperatura);
    convCompleted=0;
}
```

Ilustración 12 Obtención de temperatura y envío al LCD

En la ilustración, observamos la llamada a la función imprime LCD, dicha función es la siguiente:

```
void imprimeLCD(uint8_t USB_connected, uint8_t IsPlaying, char* nombre,float temperatura)
{
    if(USB_connected)
    {
        lcd16x2_i2c_clear();
        lcd16x2_i2c_1stLine();
        lcd16x2_i2c_printf("T:%dC",(int)temperatura);
        if(IsPlaying)
        {
            lcd16x2_i2c_setCursor(0, 9);
            lcd16x2_i2c_printf("PLAY");
            lcd16x2_i2c_setCursor(0, 0);
        }
        else
        {
            lcd16x2_i2c_setCursor(0, 9);
            lcd16x2_i2c_printf("PAUSE");
            lcd16x2_i2c_setCursor(0, 0);
        }
        lcd16x2_i2c_2ndLine();
        lcd16x2_i2c_printf(nombre);
        lcd16x2_i2c_1stLine();
    }
    else
    {
        strcpy(nombre,"Insert USB");
        lcd16x2_i2c_clear();
        lcd16x2_i2c_2ndLine();
        lcd16x2_i2c_printf(nombre);
        lcd16x2_i2c_1stLine();
    }
```

En primer lugar, esta función requiere de saber si el USB está conectado o no, ya que, de esta manera, mostrará o no en pantalla el mensaje de “Insert USB”. Si este está conectado, mostrará en pantalla la temperatura que ha sido enviada en el ejemplo anterior, y dependiendo de si la música esta activada o no gracias a la variable “IsPlaying” (activada cada vez se encuentra activa la reproducción), el LCD mostrará “PLAY” si vale 1, y en caso contrario, mostrará “PAUSE”.

Para realizar dichas escrituras en pantalla nos apoyamos en las funciones facilitadas en la librería que hemos comentado en el apartado anterior.

MAIN

En primer lugar, se inicializarán todos los periféricos empleados en el proyecto:

```
//Inicializacion interrupciones temporizador para receptor IR
HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_2);

//Inicializacion LCD
lcd16x2_i2c_init(&hi2c2);
lcd16x2_i2c_TwoLines();
lcd16x2_i2c_setCursor(0, 5);
lcd16x2_i2c_printf("Hola!!");
lcd16x2_i2c_setCursor(0, 0);
HAL_Delay(3000);

//Inicializacion sensor de temperatura y su temporizador
HAL_TIM_Base_Start(&htim3);
HAL_ADC_Start_DMA(&hadc1, (uint32_t*)ADC_value, 3);
/* USER CODE END 2 */
```

Ilustración 13. Inicialización de periféricos en el main.

Dentro del main se comprobará primero si el USB está listo para comunicar con la placa. Si está listo, se modifica el estado del USB (Appli_state), y pasará a ser APPLICATION_READY.

Una vez comprobado que el USB está listo para comunica,r se monta con Mount_USB().

Para ejecutar el archivo WAV se utiliza la función AUDIO_PLAYER_Start (de la librería waveplayer) que recibe el valor de idx.

Dentro de bucle while tenemos la función AUDIO_PLAYER_Process que indica que los archivos del pendrive se ejecutarán en bucle cuando el parámetro que recibe la función es true. Para poder mostrar el nombre del archivo WAV en la LCD se utilizará la función imprimeLCD, como se muestra en la siguiente ilustración:

```
if(strcmp(nombre,(char*)FileList.file[idx].name))
{
    strcpy(nombre,(char*)FileList.file[idx].name);
    imprimeLCD(1, IsPlaying, nombre, temperatura);
}
```

Ilustración 14: Mostrar nombre de archivo en LCD (código dentro de bucle while)

Dentro del bucle while también se podrá cambiar el estado de la música (AudioState) dependiendo del valor de *mensaje*, por ejemplo, si mensaje es 0xbf se cambia el estado a AUDIO_STATE_NEXT, este estado hará que se ejecute el siguiente archivo WAV que tenemos en el USB. Teniendo en cuenta el valor del AudioState podemos parar la música, aumentar o bajar el volumen, etc. Con la función decodeIR de la librería nec_ir_stm32 nos permite obtener los valores de device y mensaje, en device se guarda el valor de la dirección del dispositivo emisor y mensaje el valor del código enviado por el dispositivo:

```
if(flag_end)
{
    decodeIR(&buffer,&device, &mensaje);
    if(device==0xff)
    {
        switch(mensaje)
        {
        case 0xbf: //NEXT
            AudioState=AUDIO_STATE_NEXT;
            IsPlaying=1;
            break;
        case 0xbb: //PREV
            AudioState=AUDIO_STATE_PREVIOUS;
            IsPlaying=1;
            break;
        case 0xbc: //PLAY-PAUSE
            if(AudioState==AUDIO_STATE_PLAY)
            {
                AudioState=AUDIO_STATE_PAUSE;
                IsPlaying=0;
            }
            else if(AudioState==AUDIO_STATE_WAIT)
            {
                AudioState=AUDIO_STATE_RESUME;
                IsPlaying=1;
            }
            break;
        case 0xf8: //VOLUME DOWN
            AudioState=AUDIO_STATE_VOLUME_DOWN;
            break;
        case 0xea: //VOLUME UP
            AudioState=AUDIO_STATE_VOLUME_UP;
            break;
        }
        flag_end=0;
        imprimeLCD(1, IsPlaying, nombre, temperatura);
    }
}
```

Ilustración 15: Cambio de estado del Audio

Dentro del bucle while, también podemos mostrar la temperatura en la LCD (esto está explicado en el apartado anterior).

Cuando AudioState es igual a AUDIO_STATE_STOP, el valor de IsFinished pasa a 1, esto hará que salgamos del bucle while:

```
if(AudioState==AUDIO_STATE_STOP)
{
    IsFinished=1;
}
```

Ilustración 16: Salir del bucle while, AudioState=AUDIO_STATE_STOP

En caso de que no esté conectado el USB, cuando el estado Appli_state no es igual a APPLICATION_READY, se mostrará “Insert USB” en el LCD:

```
else
{
    if(strcmp(nombre,"Insert USB"))
    {
        imprimeLCD(0, IsPlaying, nombre, temperatura);
    }
}
```

Ilustración 17: Cuando USB no está conectado.