UNIVERSIDAD POLITÉCNICA DE MADRID

escuela técnica superior de ingeniería y diseño industrial

SISTEMAS ELECTRÓNICOS DIGITALES

TRABAJO DE MICROPROCESADORES

Javier González Corroto 55563

Costin Leonard Matulescu 54080

Javier Tauroni Bernaldo de Quirós 53439

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Tutor: Luis Dávila Gómez

ÍNDICE

1. Introducción

El objetivo de este trabajo ha sido el desarrollo de un reproductor de audio por USB utilizando la placa STM32F407VG. Entre todos los formatos de audio existentes se ha elegido que reproduzca archivos .wav, debido a su mayor calidad en comparación con otro formato más típico como es el .mp3

2. Funcionalidades

2.1. Estados del reproductor

El reproductor tiene tres estados fundamentales stop, play y pause, siempre y cuando haya un USB conectado.

Inicialmente se encuentra en estado de stop donde es posible seleccionar que archivo se quiere escuchar. Al pulsar el botón de usuario pasa al estado play.

En el estado play se reproduce por la salida de audio el archivo seleccionado, al volver a pulsar el botón pasa a estado pause si es una pulsación simple o a stop si la pulsación es continua, como ocurre para apagar muchos reproductores o móviles.

En el estado pause se detiene la reproducción del audio, pero al volver a pulsar sobre el botón de usuario retoma la reproducción. Si se permanece en el estado pause durante un tiempo, unos 10s en el caso de nuestro programa, aunque también es variable, pasa a estado de stop funcionando como apagado por ahorro de energía.

Cada estado tiene asociado un LED que se enciende únicamente al estar en ese estado. El LED verde se enciende en play, el naranja en pause y el rojo en stop.

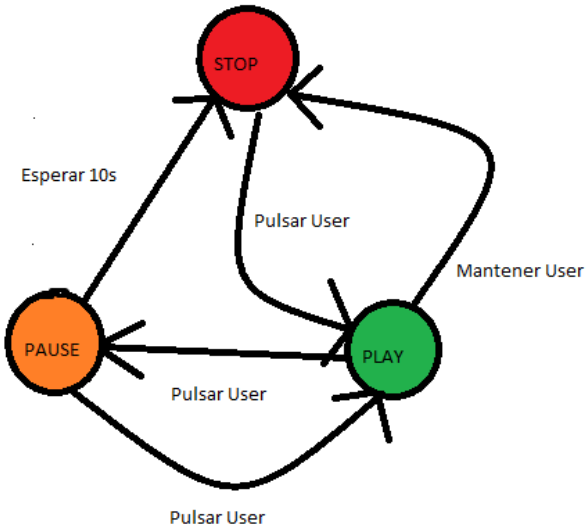

Figura 1. Diagrama de estados

```
148     /* USER CODE BEGIN 3 */
149     if(Appli_state == APPLICATION_START)
150     {
151       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
152
153     }
154     else if(Appli_state == APPLICATION_DISCONNECT)
155     {
156       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
157       f_mount(NULL, (TCHAR const*)"", 0);
158       isSdCardMounted = 0;
159     }
160
161     if(Appli_state == APPLICATION_READY)
162     {
163
164
165       if(!isSdCardMounted)
166       {
167         f_mount(&USBHFatFS, (const TCHAR*)USBHPath, 0);
168         isSdCardMounted = 1;
169       }
170       if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0))
171       {
172         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
173         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
174         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
175         HAL_Delay(500);
176         if(HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_8))
```

```
209         if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0))
210         {
211           pauseResumeToggle^=1;
212           if(pauseResumeToggle)
213           {
214             HAL_TIM_Base_Start_IT(&htim2);
215             interrupcion=0;
216             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
217             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
218             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);
219             wavPlayer_pause();
220             HAL_Delay(200);
221           }
222           else
223           {
224             HAL_TIM_Base_Stop_IT(&htim2);
225             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
226             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
227             HAL_Delay(1000);
228             if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0))
229             {
230               HAL_TIM_Base_Stop_IT(&htim2);
231               wavPlayer_stop();
232             }
233             {
234               wavPlayer_resume();
235             }
236           }
237         }
```

2.2. Control de volumen

Se ha conectado un potenciómetro a la placa que permite variar el volumen de reproducción en un rango de 0 a 255. El control de volumen funciona de manera independiente, pudiéndose realizar en cualquiera de los tres estados.

```
198         while(!wavPlayer_isFinished())
199         {
200
201           wavPlayer_process();
202           HAL_ADC_Start(&hadc1);
203             if(HAL_ADC_PollForConversion(&hadc1, 100)==HAL_OK)
204             {
205                 volumen=HAL_ADC_GetValue(&hadc1);
206             }
207           HAL_ADC_Stop(&hadc1);
208           CS43_SetVolume(volumen);
209           if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0))
210           {
211             pauseResumeToggle^=1;
```

Si se usan cascos no es recomendable llegar a valores demasiado altos.

2.3. Selector de archivo

Es posible seleccionar que archivo se desea reproducir. Se puede elegir entre cinco archivos distintos conectando un cable a los pines PD8, PD9, PD10, PD11 o dejando el cable sin conectar.

```
165       if(!isSdCardMounted)
166       {
167         f_mount(&USBHFatFS, (const TCHAR*)USBHPath, 0);
168         isSdCardMounted = 1;
169       }
170       if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0))
171       {
172         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
173         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
174         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
175         HAL_Delay(500);
176         if(HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_8))
177         {
178             wavPlayer_fileSelect(WAV_FILE0);
179         }
180         else if(HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_9))
181         {
182             wavPlayer_fileSelect(WAV_FILE1);
183         }
184         else if(HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_10))
185         {
186             wavPlayer_fileSelect(WAV_FILE2);
187         }
188         else if(HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_11))
189         {
190             wavPlayer_fileSelect(WAV_FILE3);
191         }
192         else
193         {
194             wavPlayer_fileSelect(WAV_FILED);
195         }
```

Es posible cambiar el cable en cualquier momento sin que ello afecte a la reproducción, sin embargo, para hacerse efectivo el cambio el programa tendrá que pasar por el estado stop. Estos archivos deberán llamarse 0.wav, 1.wav, 2.wav, 3.wav y D.wav, en caso contrario la selección del archivo no será posible.

3. Funcionamiento

Además del main.c, el programa incluye otros archivos para poder realizar correctamente el programa, desde la comunicación del USB con la placa hasta la reproducción del audio.

A la hora de establecer la comunicación entre el USB y la placa se ha optado por el protocolo I^2C. Se ha elegido este protocolo por la posibilidad de conectar otros dispositivos utilizando

el mismo bus serie en caso de ampliar las funcionalidades. Para la transmisión del audio también se ha empleado DMA.

Para mejorar la transmisión del audio se ha modificado la frecuencia de reloj a 8 MHz, por esta razón los valores usados para configurar el temporizador básico TIM2 empleado para el paso del estado pause a stop, tal y como se ha descrito en el apartado 2.1, han sido modificados en consecuencia.

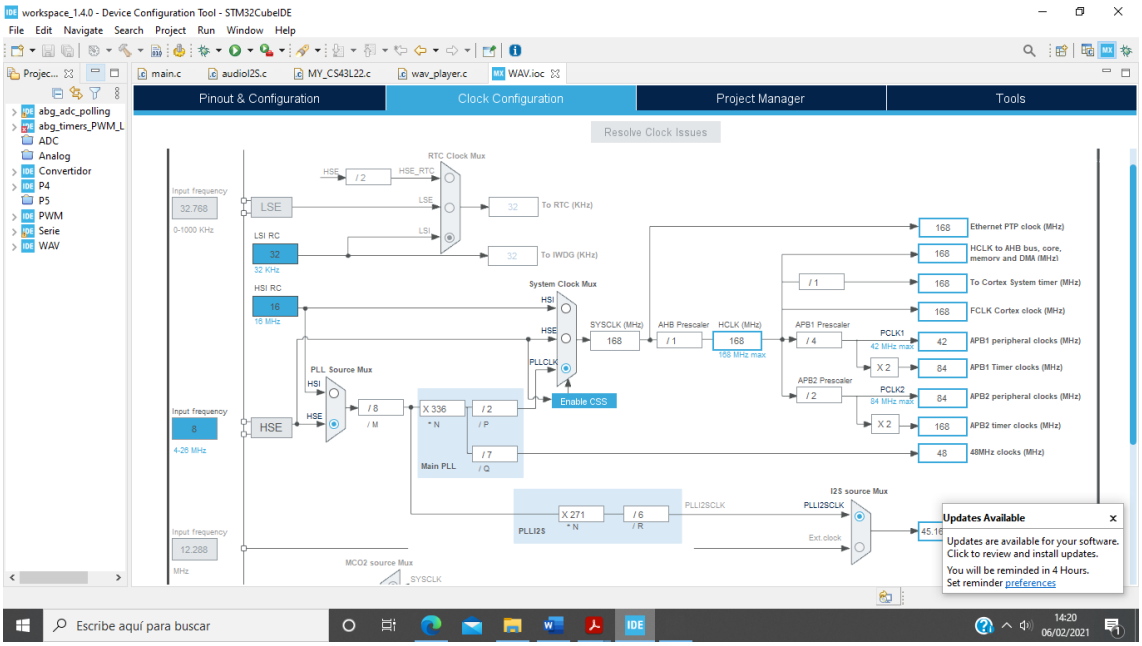

Figura 2. Cambios realizados en la frecuencia del reloj

Para evitar que el temporizador generase una interrupción instantáneamente, se optó por implementar un algoritmo para ignorar esa primera interrupción instantánea.

Para que el micro lea el USB se utiliza un adaptador USB a microUSB, también se podría introducir una tarjeta SD o microSD siempre y cuando se disponga del adaptador correspondiente y sea compatible con el tipo de archivo FATFS.

Para la gestión del CS43L22, el convertidor DAC de audio que incorpora la placa se han utilizado una librería creada por M.Yaqoob.

También existe la posibilidad de conectar la salida de audio del micro a la FPGA, para usar el audio como parte del proyecto de FPGA

4. Posibles mejoras

Si bien estamos muy satisfechos con el trabajo realizado es cierto que estos proyectos siempre tienen margen de mejora, pudiéndose añadir funciones o mejorar las ya implementadas. A continuación, se presentan algunas posibles mejoras que por falta de tiempo no se han desarrollado.

4.1. Pantalla LCD

Como ya se ha comentado anteriormente, la elección del protocolo I^2C para las comunicaciones en serie vino motivado por la posibilidad de añadir varios periféricos compartiendo el mismo bus de datos. Uno de estos periféricos podría ser una pantalla LCD en la que se puedan visualizar datos como el nombre del archivo o el volumen.
Se intentó implementar esta pantalla en el proyecto, logrando que registrase e indicase el valor de la señal analógica "volumen" procedente del potenciómetro. Sin embargo, al incorporar dicho código al proyecto completo dejaba de funcionar el resto.

Pantalla LCD

4.2. Control remoto por IR

Otro posible añadido hubiera sido un mando a distancia que se comunicase a través de infrarrojos, gestionando así el paso de un estado a otro del reproductor. También se podría haber controlado la selección de archivo o el volumen, aunque esto último hubiera provocado que la eliminación del potenciómetro.

5. Conclusiones

Creemos que hemos realizado un buen trabajo que además nos ha servido para ampliar los conocimientos de la asignatura. El aspecto más complicado del trabajo ha sido también el más novedoso, entender el funcionamiento del audio y adaptarlo a nuestras necesidades.

Aunque nos hubiera gustado disponer de algo más de tiempo para añadirle algunas de las funciones anteriormente descritas, en resumen, estamos satisfechos con los resultados del trabajo.

6. Enlaces de interés

Repositorio del que se han obtenido las librerías para la gestión del audio:
MYaqoobEmbedded (github.com)
Repositorio del proyecto: Micro/STM32 at main · Jtauroni/Micro (github.com)