EJERCICIO PRÁCTICO 2

El objetivo de esta parte de la práctica era conseguir reproducir un archivo .wav desde una SD externa. Para eso he usado parte del código de la anterior práctica. Voy a explicar un poco el código usado para esta parte de la práctica.

Primero de todo mencionar que con esta práctica vamos a usar dos tipos de buses: *bus SPI, bus I2S*
Empezemos incluyendo las librerías necesarias:

```
#include "Audio.h"
#include "SD.h"
#include "FS.h"
// Digital I/O used
#define SD_CS 5
#define SPI_MOSI 23
#define SPI_MISO 19
#define SPI_SCK 18
#define I2S_DOUT 22
#define I2S_BCLK 26
#define I2S_LRC 25
```

Usaremos tres librerías: *Audio.h, SD.h, FS.h*.

Definimos los puertos que vamos a usar para la transmisión y recepción de datos. Podemos ver de forma rápida que puerto pertenece a cada componente mediante el nombre, SPI_MOSI,SPI_MISO,SPI_SCK,SPI_CS los vamos a usar para la tarjeta SD y las variables I2S_DOUT, I2S_BCLK, I2S_LRC las vamos a conectar al MAX98357A.

Que hacemos en el void setup?

void setup()

```
Audio audio;
void setup(){

Serial.begin(9600);

pinMode(SD_CS, OUTPUT);
digitalWrite(SD_CS, HIGH);
SPI.begin(SPI_SCK, SPI_MISO, SPI_MOSI);
SD.begin(SD_CS);
audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);
audio.setVolume(21); // 0...21
audio.connecttoFS(SD, "SOLDADITO MARINERO VOZ GUITAR.mp3");
Serial.println("Reproduciendo...");
}
```

Primero de todo creamos un nuevo objeto de la clase Audio, el cuál vamos a usar para enviar la configuración de pines, la configuración de volumen... Una vez entramos en el void setup(), primero inicializamos el puerto serie. Establecemos el pin SD_CS como salida. Configuramos los pines SPI para la recepción de datos desde la SD y incializamos esta comunicación mediante el SPI.begin(). Si nos fijamos en la función begin de la clase SPI vemos:

```
void SPIClass::begin(int8_t sck, int8_t miso, int8_t mosi, int8_t ss)
{
    if(_spi) {
        return;
    }

    if(!_div) {
        _div = spiFrequencyToClockDiv(_freq);
    }

    _spi = spiStartBus(_spi_num, _div, SPI_MODE0, SPI_MSBFIRST);
    if(!_spi) {
        return;
    }

    if(sck == -1 && miso == -1 && mosi == -1 && ss == -1) {
        _sck = (_spi_num == VSPI) ? SCK : 14;
        _miso = (_spi_num == VSPI) ? MISO : 12;
        _mosi = (_spi_num == VSPI) ? MOSI : 13;
        _ss = (_spi_num == VSPI) ? SS : 15;
    } else {
        _sck = sck;
        _miso = miso;
        _mosi = mosi;
        _ss = ss;
    }

    spiAttachSCK(_spi, _sck);
    spiAttachMISO(_spi, _miso);
    spiAttachMOSI(_spi, _mosi);

}
```

Vemos que después de comprovar dos cosas, comprueba si ha recibido parámetros que sean -1,de esa forma pondría los parámetros por defecto, pero si les pasamos nuestros propios pines usará esos mismos. Una vez establecidas las variables, llama a otras funciones spiAttach. He estado mirando estas funciones y básicamente lo que hacen son dos comprovaciones de si el PIN enviado es válido y lo compara con uno establecido por el framework de arduino que son normalmente usados. De esa forma se comprueba que no haya errores a la hora de declarar los pines que hemos pùesto y así lo sabríamos antes de mirar otros problemas de el código.

Ejemplo función spiAttachMISO()

```
void spiAttachMISO(spi_t * spi, int8_t miso)
{
    if(!spi) {
        return;
    }
    if(miso < 0) {
        if(spi->num == HSPI) {
            miso = 12;
        } else if(spi->num == VSPI) {
            miso = 19;
        } else {
            miso = 7;
        }
    }
    SPI_MUTEX_LOCK();
    pinMode(miso, INPUT);
    pinMatrixInAttach(miso, SPI_MISO_IDX(spi->num), false);
    SPI_MUTEX_UNLOCK();
}
```

Una vez inicializado el bus SPI, seguimos mediante la función SD.begin(). Esta configura la línea SS el cuál establecce el dispositivo con el que se va a llevar la comunicacion.

Por ahora hemos acabado con la configuración con el bus SPI, ahora empecemos con el I2S con el cuál vamos a enviar el audio leído de la SD. Primero seleccionamos el Pinout, es decir los pines de salida. La clase Audio tiene unas funciones establecidas para configuraciones muy básicas como seleccionar el SampleRate, el volumen y otras básicas de cvonfiguración como is_running... Una vez seleccionados los pines de salida, seleccionamos el volumen de audio. Mirando la función, vemos que comprueba si es mas pequeño que 21 y si lo es, estalece la variable m_vol de la clase como el número pasado que está guardado dentro de una tabla llamada volumetable[].

No he encontrado en que punto usa esta variable para configurar el volumen de la reproducción, supongo que va más allá de lo que pemsaba...

Finalmente ya usa una última función para saber que fichero leer de la SD. En esta función le pasamos el objeto SD y el nombre de la canción que deseamos reproducir. Esta función comprueba si el nombre de la función tiene un tamaño más grande de 255 y si lo es nos daría un error. Me ha precido curioso el número ya que es 2^8, no estoy seguro si tiene algo que ver... Una vez hecho esto establece el nombre del fichero de tal forma que se pueda leer de forma correcta por el programa. Mira si el fichero dicho existe y entonces lo abre... Para finalizar esta función ya mira que tipo de fichero es mediante una función para mirar el final de un string. Recordamos que acepta ficheros *.mp3, .wav, .flac* básicamente porque solo dispone de esos decodificadores de audio.

Una vez hecho todo esto entramos al void loop() en el cuál sólo usaremos una función de la librería Audio que nos va a hacer un loop del fichero.

```
void loop(){
audio.loop();
}
```

Todo lo demás que sigue en el código es solamente información que mostrarmos por el puerto serie.

CONCLUSIONES

He tenido varios problemas al ejecutar el código ya que me daba bastantes errores con la librería Audio.h y no estaba seguro si tenía que instalarla o pertenecía al framework de arduino. En el enunciado del ejercicio nos decía que la teníamos en el github por lo que la buscaba desde el mismo PlatformIO y encontraba algunas parecidas pero que no me servían con el código descrito. Por lo que finalmente pedí ayuda y me dijeron lo que debía hacer con esa librería.
Una vez hecho todo eso me encontré que no me leía el fichero de la tarjeta SD y tras mirar los pines que había puesto y ver que lo tenía correctamente conectado lo que hice fue formatear la tarjeta SD mediante Linux ya que con Windows tuve problemas y no me hacía un formateo correcto y finalmente pude abrir el fichero con la ESP32.
El siguiente problema fue el mismo descrito en el ejercicio anterior sobre el altavoz utilizado y el chip *MAX98357*.

SALIDA POR EL PUERTO SERIE

```
infoPSRAM not found, inputBufferSize: 6399 bytes
infobuffers freed, free Heap: 287056 bytes
infoReading file: "/SOLDADITO MARINERO VOZ GUITAR.mp3"
infoMP3Decoder has been initialized, free Heap: 263052 bytes
Reproduciendo...
infostream ready
infoContent-Length: 2587068
infofile has no mp3 tag, skip metadata
infoAudio-Length: 2587068
infosyncword found at pos 0
infoChannels: 2
infoSampleRate: 44100
infoBitsPerSample: 16
infoBitRate: 128000
```