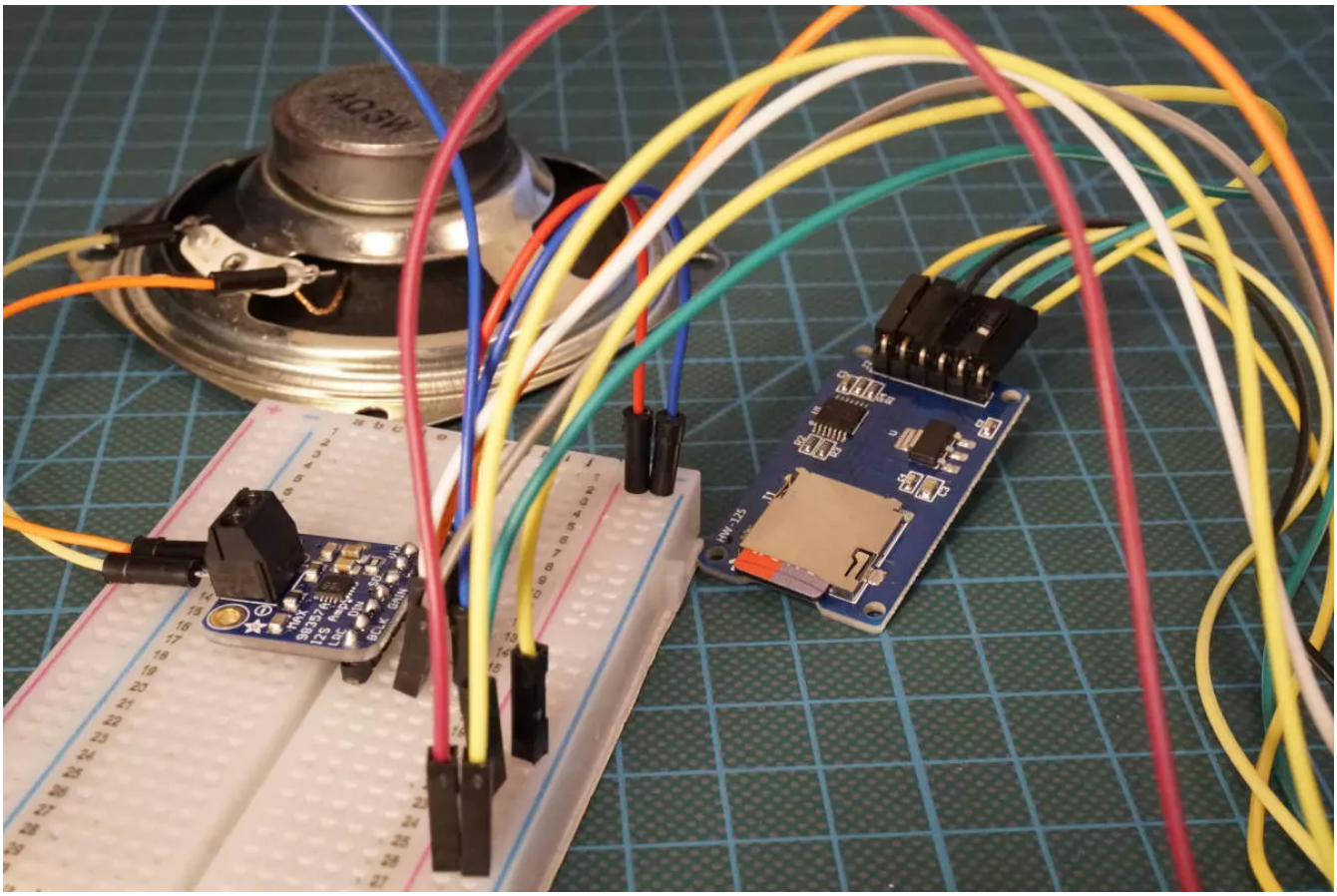
PRACTICA 7 : Buses de comunicación III (I2S)

El objetivo de la practica actual es describir el funcionamiento del bus I2S y realizar una practica para comprender su funcionamiento

Introducción teórica

El protocolo de comunicación I2S se usa para transferir señales de sonido digitales Nosotros realizaremos una practica para reproducir música desde la memoria interna, así como desde una tarjeta SD externa.

También comparamos diferentes microcontroladores y vemos por qué preferimos un microcontrolador ESP32 para nuestros proyectos I2S.

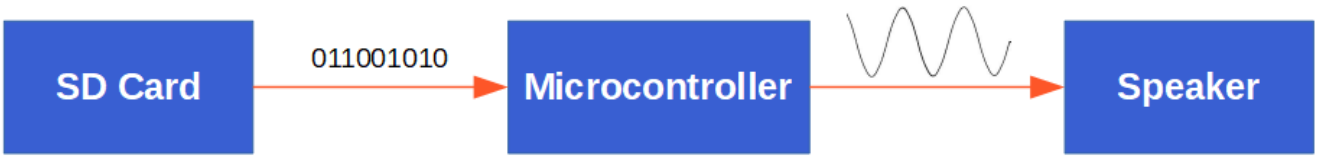


El protocolo de comunicación I2S

El protocolo de comunicación I2S fue desarrollado por Philips Semiconductors en 1986. I2S son las siglas de Inter-Integrated Circuit Sound y, como interfaz de bus serie eléctrica, I2S es el estándar para conectar diferentes dispositivos de audio digital.

Los microcontroladores ESP32 y ESP8266 admiten el protocolo I2S, donde solo algunos microcontroladores Arduino especiales admiten el protocolo de comunicación.

¿Por qué necesitamos el protocolo I2S? Si queremos reproducir un archivo de audio digital con la ayuda de una placa de microcontrolador, tenemos que considerar toda la cadena de audio digital. El siguiente esquema muestra cómo se almacena un archivo de audio en una tarjeta SD y se lee desde la placa del microcontrolador. Luego, la placa se conecta al altavoz a través de un pin digital y tierra.



En mi caso, tengo un archivo de audio de muestra de muestras de ondas libres con una frecuencia de muestreo de 44,1 kHz, formato estéreo y una profundidad de bits de 16 bits. En nuestro lado de entrada, donde queremos leer el archivo de música, no tenemos ningún problema porque la conexión SPI es lo suficientemente rápida como para que la calidad no se reduzca durante la transmisión.

Pero en el lado de la salida tenemos que transferir la señal digital a una señal analógica. Esto se realiza mediante un convertidor de digital a analógico (DAC). Dependiendo del microcontrolador utilizado existen diferentes problemas:

- Arduino y ESP8266 : Las placas Arduino, así como el ESP8266 en general, no tienen un DAC interno y, por lo tanto, tendrías que construir un DAC con componentes externos.
- ESP32 : El ESP32 tiene un DAC interno para crear una señal de salida analógica, sin embargo, el DAC solo tiene una resolución de 8 bits. Debido a que tenemos una señal de entrada de 16 bits, perderíamos algo de calidad.

Pero, ¿cómo podemos convertir los datos digitales del archivo WAVE al altavoz? La solución a esto es el protocolo de comunicación I2S, que admite entre 4 y 32 bits de datos por muestra. Para hacer nuestra vida aún más fácil, utilizamos una placa de conexión de audio MAX98357 I2S. Pero primero profundizamos en el protocolo de comunicación I2S.

Protocolo de comunicación I2S en detalle

queremos echar un vistazo más de cerca al protocolo de comunicación I2S. Por lo tanto, cubrimos tres temas importantes.

- Conexión de 3 cables I2S
- Componentes de la red I2S
- Diagrama de temporización I2S

La siguiente tabla muestra qué tarjetas tienen una interfaz I2S y cuáles no.

Procesador	Tener una interfaz I2S	No tiene una interfaz I2S
Arduino	Arduino Due, Arduino MKR Zero, Arduino MKR1000 WiFi	Placas con microcontrolador ATmega328P, como Arduino Uno o Arduino Nano.
ESP	ESP8266 ESP32	

En la tabla puede ver que solo algunas placas Arduino especiales tienen la interfaz I2S pero no las placas más utilizadas como Arduino Uno. Además, todas las placas ESP8266 y ESP32 admiten la interfaz I2S y, por lo tanto, vemos que la elección del ESP32 es una buena elección para el procesado de audio .

Conexión de 3 cables I2S

El protocolo I2S utiliza tres cables para la comunicación.

El reloj serial (SCK), también llamado línea de reloj de bits (BCLK), se usa para obtener todos los componentes en el mismo ciclo. La frecuencia del reloj serial se define por: Frecuencia = Frecuencia de muestreo * Bits por canal * Número de canales.

Para el archivo WAVE que utilizaremos en esta practica tenemos las siguientes variables:

- Frecuencia de muestreo: 44,1 kHz
- Bits por canal: 16
- Cantidad de canales: 2

Por lo tanto, el reloj en serie tiene una frecuencia de 44,1 kHz * 16 * 2 = 1,411 MHz.

La segunda línea del protocolo de comunicación I2S es el cable **Word Select (WS)** o Frame Select (FS) que diferencia entre el canal izquierdo o el derecho.

- Si WS = 0 → se utiliza el canal 1 (canal izquierdo)
- Si WS = 1 → se utiliza el canal 2 (canal derecho)

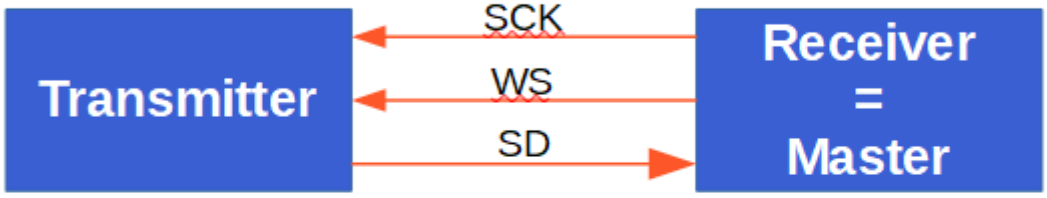
El último cable es la línea **Serial Data (SD)** donde la carga útil se transmite en 2 complementos. Es importante que el bit más significativo se transfiera primero (MSB primero), porque el transmisor y el receptor pueden tener diferentes longitudes de palabra. Por lo tanto, el transmisor ni el receptor deben saber cuántos bits se transfieren. Pero, ¿qué pasa si la longitud de la palabra entre el transmisor y el receptor no coincide?

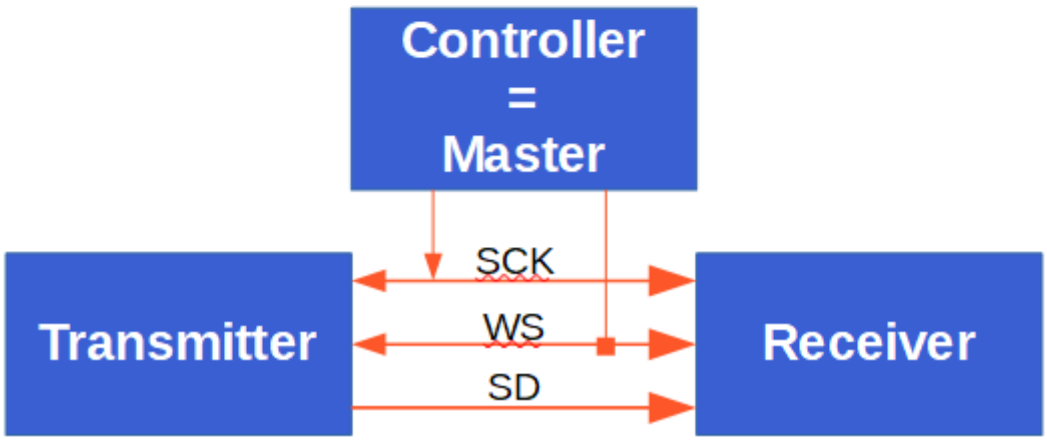
- Si WS del receptor> transmisor WS → la palabra está truncada (los bits de datos menos significativos se establecen en 0)
- Si WS del receptor <WS transmisor → bits después del LSB se ignoran

Componentes de la red I2S

Si hay varios componentes I2S conectados entre sí, llamo a esto una red I2S. Los componentes de la red tienen diferentes nombres y también diferentes funciones. La siguiente imagen muestra tres redes diferentes, que describo en la siguiente sección.

En la primera imagen tenemos un transmisor y también un receptor. El transmisor podría ser una placa ESP NodeMCU y el receptor una placa de salida de audio I2S, que describimos en la siguiente sección. También tenemos los tres cables para conectar los dispositivos I2S.





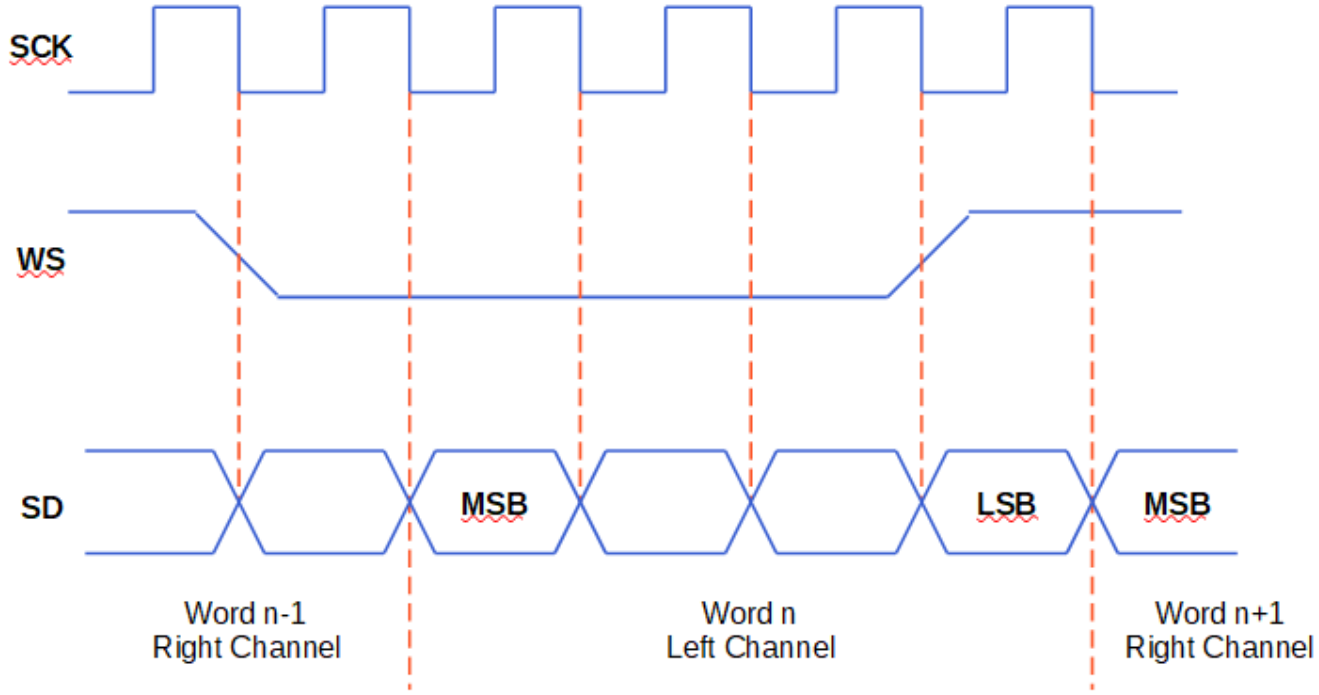
En este primer caso, el transmisor es el maestro porque el maestro controla el reloj en serie (SCK) y las líneas de selección de palabras (WS). En la segunda imagen vemos lo contrario porque también el receptor de los mensajes I2S puede ser el maestro. Por lo tanto, las líneas SCK y WS comienzan en el receptor y terminan en el transmisor.

La tercera imagen muestra que también un controlador externo puede ser el dispositivo maestro que genera SCK y WS. El controlador está conectado a los nodos de la red.

En todas las redes I2S solo hay un dispositivo maestro. Puede haber muchos otros componentes que reciban o transmitan datos de sonido.

Diagrama de temporización I2S

Para comprender mejor el comportamiento y también la funcionalidad del protocolo de comunicación I2S, echamos un vistazo al siguiente diagrama de tiempos I2S.



En el diagrama de tiempo, verá las tres líneas: SCK, WS y SD. Primero tenemos nuestro reloj serial que tiene la frecuencia de Frecuencia de muestreo * Bits por canal * Número de canales, en nuestro ejemplo 1.411 MHz. El segundo canal es la línea de selección de palabras que cambia entre 1 para el canal de sonido derecho y 0 para el canal izquierdo.

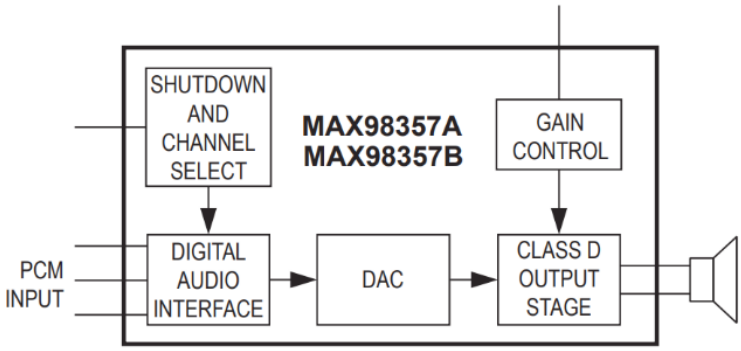
Desde la línea de datos en serie, vemos que los datos se envían en cada ciclo de reloj en el borde descendente (línea de puntos roja) → HIGH a LOW. Para la comunicación I2S también es posible enviar datos en un cambio de BAJO a ALTO.

También vemos que la línea WS cambia un ciclo de reloj antes de que se transmita el bit más significativo (MSB). Eso le da al receptor tiempo para almacenar la palabra anterior y borrar el registro de entrada para la siguiente palabra. El MSB se envía cuando SCK cambia después de que WS cambia.

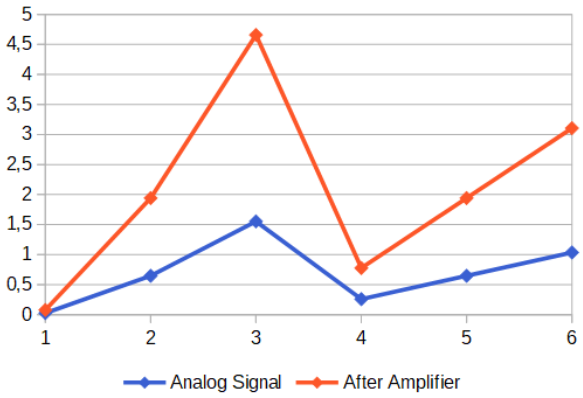
La placa de conexión de audio MAX98357 I2S

Una vez que sabemos que podemos usar el protocolo de comunicación I2S para obtener los datos de sonido del microcontrolador sin ninguna reducción en la calidad, el siguiente problema es que tenemos que decodificar las señales I2S en señales analógicas y también necesitamos un amplificador para usar un altavoz.

Decodificador de señal I2S a señal analógica, porque los altavoces solo funcionan con señales analógicas. El amplificador aumenta la potencia de la señal analógica para aumentar la intensidad del sonido. El MAX98357 es un amplificador de entrada de modulación de código de pulso digital (PCM) que decodifica la señal I2S en una señal analógica con un convertidor de digital a analógico (DAC) y también tiene un amplificador incorporado. La siguiente imagen muestra el diagrama de bloques simplificado de la hoja de datos MAX98357.

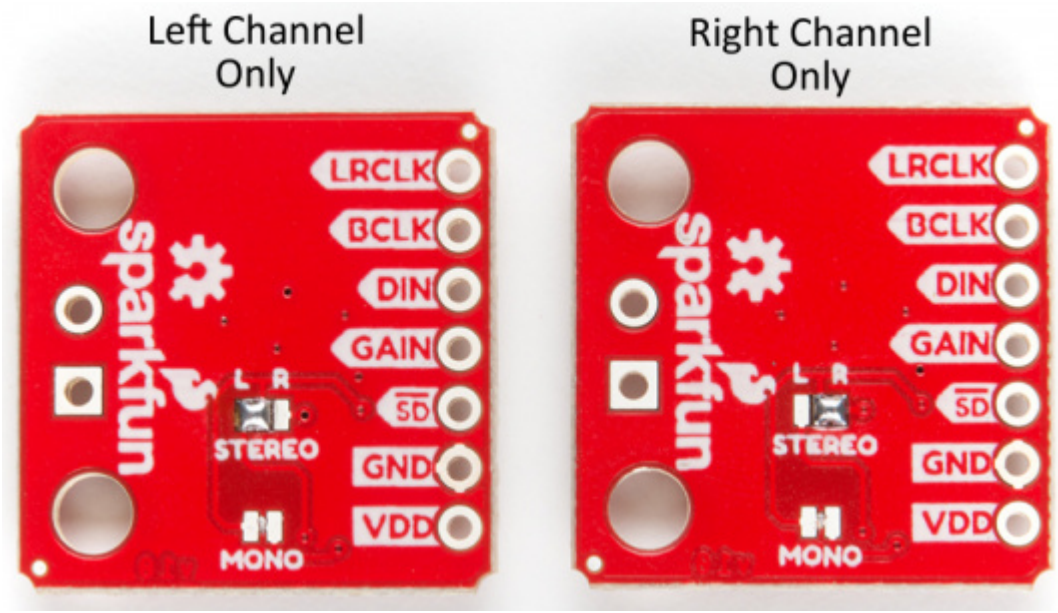


Digital Bit	Digital DEC	Analog Voltage
0 0 0 0 1 0 1 0	2	0.026V
0 0 1 1 0 0 1 0	50	0.647V
0 1 1 1 1 0 0 0	120	1.553V
0 0 0 1 0 1 0 0	20	0.259V
0 0 1 1 0 0 1 0	50	0.647V
0 1 0 1 0 0 0 0	80	1.035V



El voltaje de operación del MAX98357 está entre 2.7V y 5.5V. Por lo tanto, puede alimentar el microcontrolador con una placa basada en Arduino (5V) o ESP (3.3V). La potencia de salida es de 3,2 W para un altavoz de 4 Ω y de 1,8 W para un altavoz de 8 Ω.

La configuración predeterminada de la placa es la operación "mono", lo que significa que las señales izquierda y derecha se combinan para impulsar un solo altavoz. Si desea cambiar a sonido estéreo, debe cortar el puente mono en la PCB y soldar la conexión estéreo para el canal izquierdo o derecho.



La frecuencia de muestreo del MAX98357 está entre 8 kHz y 96 kHz y, por lo tanto, nuestra música de ejemplo con 44,1 kHz encaja perfectamente en la frecuencia de muestreo. La resolución de la muestra es de 16 bits o 32 bits y la corriente de reposo es muy baja con 2,4 mA.

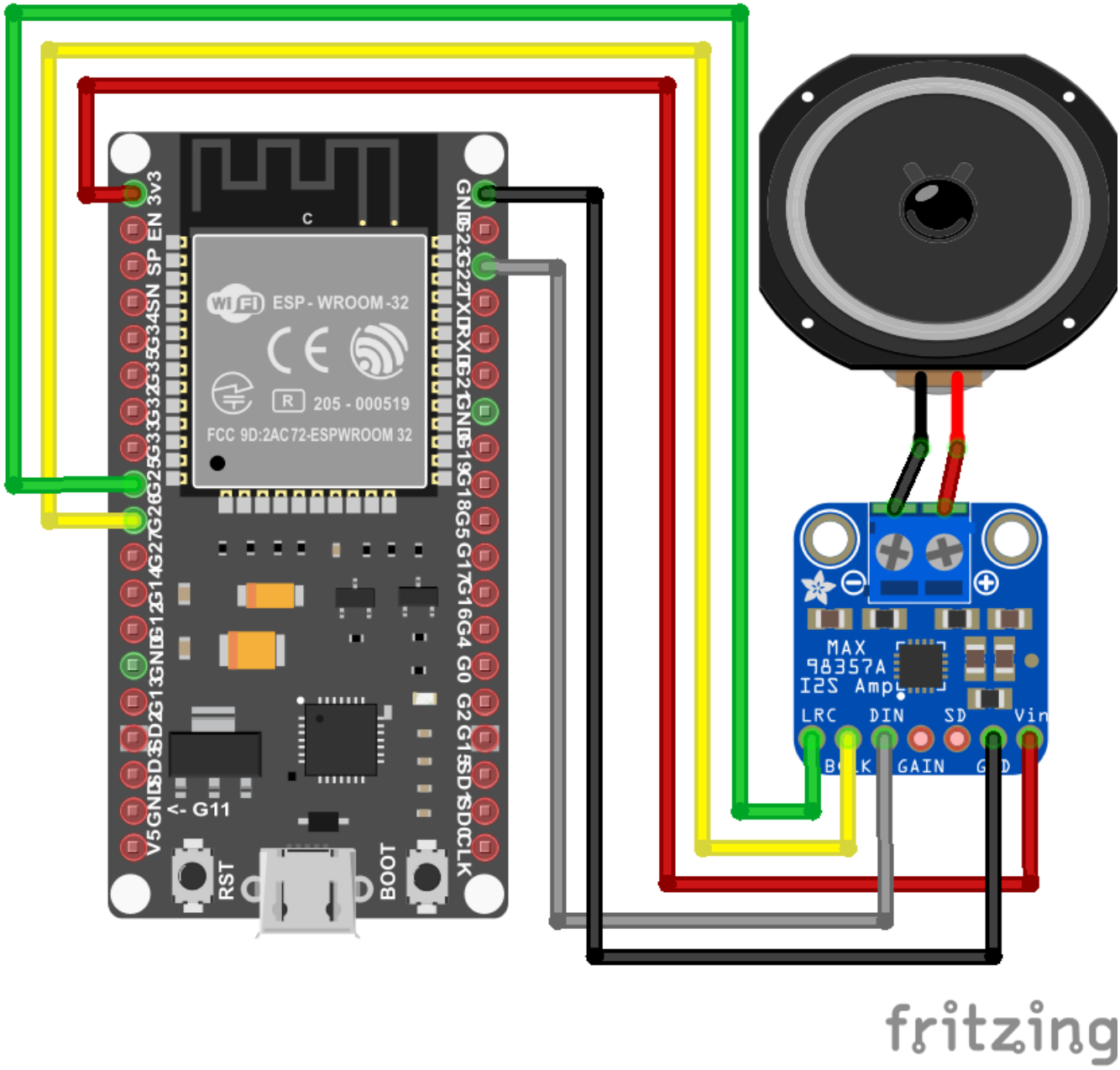
Debido a que el amplificador usa modulación de ancho de pulso para controlar los dispositivos de salida, es un amplificador de clase D. La tasa de ganancia del amplificador está entre 3dB y 15dB con una tasa de ganancia predeterminada de 9dB. La siguiente tabla muestra cómo cambiar la tasa de ganancia. La clave es que el pin de ganancia debe estar conectado a otros pines para cambiar la tasa de ganancia.

Tasa de ganancia Conexión de pin de ganancia 15 dB Conectado a GND a través de una resistencia de 100 kΩ 12 dB Conectado a GND 9 dB Desconectado (predeterminado) 6 dB Conectado a VDD / Vin 3 dB Conectado a VDD / Vin a través de una resistencia de 100 kΩ

Ejercicio Practico 1 reproducción desde memoria interna

Los datos de sonido se almacenan como una matriz en la RAM interna del ESP32. Usamos la placa de conexión de audio MAX98357 I2S para decodificar la señal digital en una señal analógica. Por lo tanto, utilizamos el protocolo I2S para generar los datos de sonido digital sin pérdidas de calidad.

La siguiente imagen muestra el cableado entre el ESP32 NodeMCU, la placa de conexión de audio MAX98357 I2S y el altavoz.



Para el código Arduino usamos la biblioteca de audio ESP8266 de Earle F. Philhower

https://github.com/earlephilhower/ESP8266Audio

```
 #include "AudioGeneratorAAC.h"
#include "AudioOutputI2S.h"
#include "AudioFileSourcePROGMEM.h"
#include "sampleaac.h"

AudioFileSourcePROGMEM *in;
AudioGeneratorAAC *aac;
AudioOutputI2S *out;

void setup(){
  Serial.begin(115200);

  in = new AudioFileSourcePROGMEM(sampleaac, sizeof(sampleaac));
  aac = new AudioGeneratorAAC();
  out = new AudioOutputI2S();
  out -> SetGain(0.125);
  out -> SetPinout(26,25,22);
```

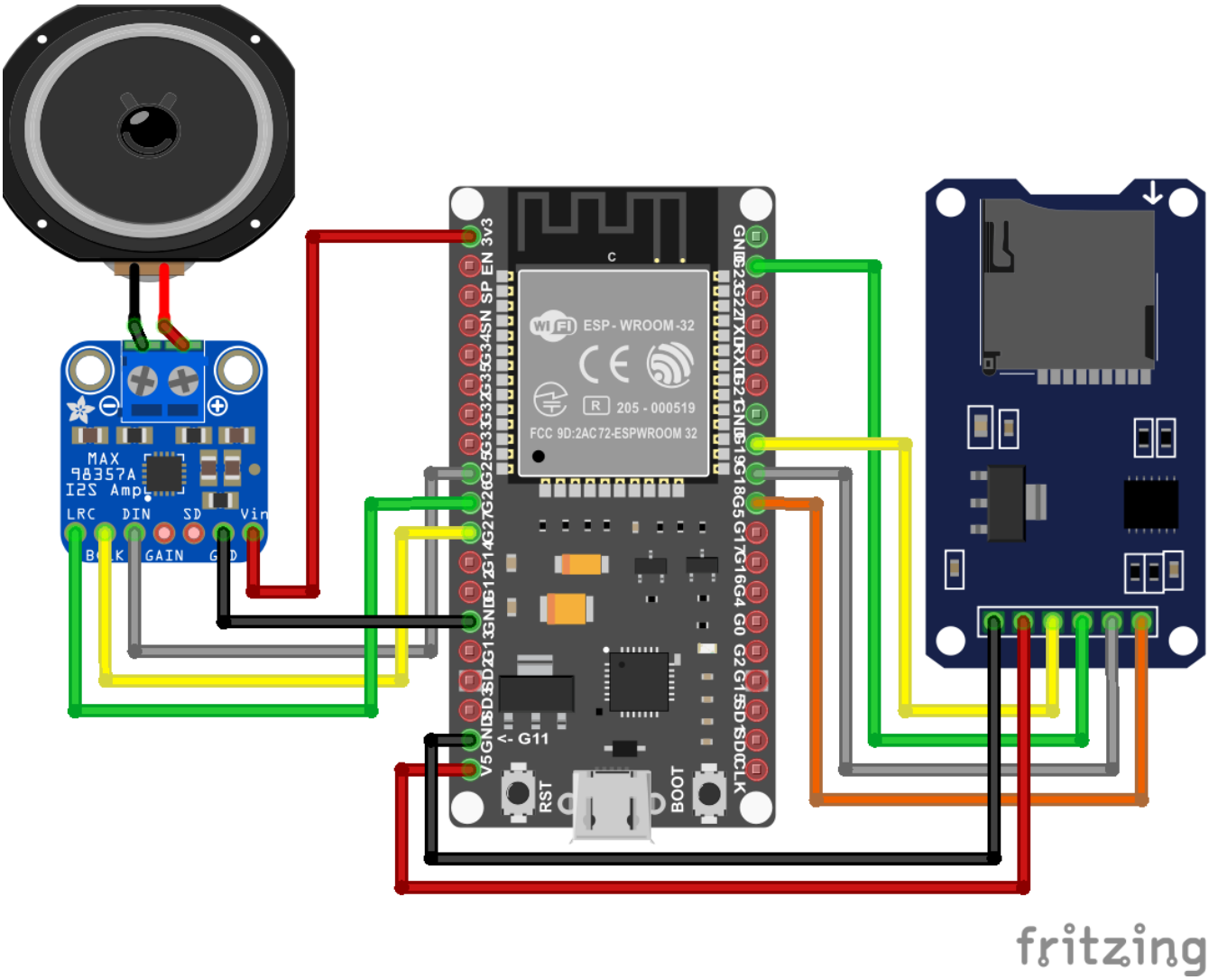
```
  aac->begin(in, out);
}

void loop(){
  if (aac->isRunning()) {
    aac->loop();
  } else {
    aac -> stop();
    Serial.printf("Sound Generator\n");
    delay(1000);
  }
}
```

1. Descibir la salida por el puerto serie
2. Explicar el funcionamiento

Ejercicio Practico 2 reproducir un archivo WAVE en ESP32 desde una tarjeta SD externa

queremos reproducir el archivo WAVE que mencioné al principio de este tutorial a través del ESP32 NodeMCU y el altavoz. Debido a que el ESP32 tiene que leer el archivo WAVE y reenviar la señal de audio digital al MAX98357A, tenemos que usar una tarjeta SD con el archivo WAVE. También puede utilizar un archivo MP3 en lugar del archivo WAVE.

La siguiente imagen muestra el cableado del ESP32 NodeMCU con el módulo de tarjeta (Micro) SD, el MAX98357A y el altavoz. En la imagen que ve, tiene que cambiar el pin DIN del MAX98357A, en comparación con el segundo proyecto.



para este proyecto utilizaremos la libreria

https://github.com/schreibfaul1/ESP32-audioI2S

```
#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      25
#define I2S_BCLK      27
#define I2S_LRC       26

Audio audio;

void setup(){
    pinMode(SD_CS, OUTPUT);
    digitalWrite(SD_CS, HIGH);
    SPI.begin(SPI_SCK, SPI_MISO, SPI_MOSI);
    Serial.begin(115200);
    SD.begin(SD_CS);
    audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);
    audio.setVolume(10); // 0...21
```

```
    audio.connecttoFS(SD, "Ensoniq-ZR-76-01-Dope-77.wav");
}

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

// optional
void audio_info(const char *info){
    Serial.print("info        "); Serial.println(info);
}
void audio_id3data(const char *info){  //id3 metadata
    Serial.print("id3data     ");Serial.println(info);
}
void audio_eof_mp3(const char *info){  //end of file
    Serial.print("eof_mp3     ");Serial.println(info);
}
void audio_showstation(const char *info){
    Serial.print("station     ");Serial.println(info);
}
void audio_showstreaminfo(const char *info){
    Serial.print("streaminfo  ");Serial.println(info);
}
void audio_showstreamtitle(const char *info){
    Serial.print("streamtitle ");Serial.println(info);
}
void audio_bitrate(const char *info){
    Serial.print("bitrate     ");Serial.println(info);
}
void audio_commercial(const char *info){  //duration in sec
    Serial.print("commercial  ");Serial.println(info);
}
void audio_icyurl(const char *info){  //homepage
    Serial.print("icyurl      ");Serial.println(info);
}
void audio_lasthost(const char *info){  //stream URL played
    Serial.print("lasthost    ");Serial.println(info);
}
void audio_eof_speech(const char *info){
    Serial.print("eof_speech  ");Serial.println(info);
}
```

La última parte de la función de configuración es conectar las entradas y salidas de este ejemplo. . Por lo tanto, conectamos el objeto de audio con el objeto de la tarjeta SD y definimos la ruta al archivo WAVE. Si coloca el archivo de sonido en una carpeta, debe copiar la ruta completa al archivo de sonido con barras diagonales (“/”).

audio.connecttoFS(SD, "Ensoniq-ZR-76-01-Dope-77.wav")

ahi se tiene que escribir el fichero .wav que dispongamos

1. Descibir la salida por el puerto serie
2. Explicar el funcionamiento