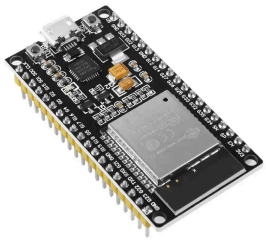
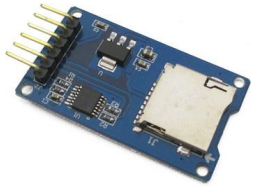
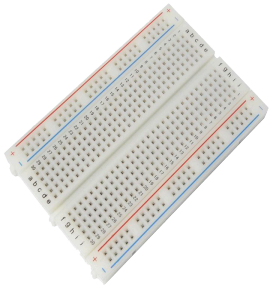
PRACTICA 7 : Buses de comunicación III (I2S)

MATERIAL

Para esta practica necesitaremos el microcontrolador ESP32, Tarjeta microSD y módulo lector, altavoz, cables y una protoboard.




OBJETIVO Y FUNCIONALIDAD DE LA PRACTICA

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.

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.

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

https://github.com/earlephilhower/ESP8266Audio

Código comentado

```
// Incluye las librerías necesarias para manejo de audio AAC e interfaz I2S
#include "AudioGeneratorAAC.h"         // Decodificador de audio AAC
#include "AudioOutputI2S.h"            // Salida de audio usando I2S
#include "AudioFileSourcePROGMEM.h"    // Fuente de archivo desde PROGMEM
#include "sampleaac.h"                 // Archivo de audio AAC embebido en PROGMEM

// Declaración de punteros a objetos de audio
AudioFileSourcePROGMEM *in;  // Fuente del archivo desde la memoria
AudioGeneratorAAC *aac;      // Decodificador de audio AAC
AudioOutputI2S *out;         // Salida de audio a través del bus I2S

void setup() {
  Serial.begin(115200);  // Inicia la comunicación serial para depuración

  // Inicializa la fuente de audio desde la memoria flash
  in = new AudioFileSourcePROGMEM(sampleaac, sizeof(sampleaac));

  // Crea una instancia del decodificador AAC y de la salida I2S
  aac = new AudioGeneratorAAC();
  out = new AudioOutputI2S();

  // Configura el volumen de salida (0.125 = volumen bajo)
  out->SetGain(0.125);

  // Define los pines usados para la comunicación I2S: BCLK, LRCLK, y DATA
  out->SetPinout(12, 13, 14);

  // Inicia la reproducción de audio
  aac->begin(in, out);
}

void loop() {
  // Mientras el generador de audio esté en ejecución, continúa el bucle de reproducción
  if (aac->isRunning()) {
    aac->loop();
  } else {
    // Si ya no está activo, detiene la reproducción y muestra un mensaje
    aac->stop();
    Serial.printf("Sound Generator\n");
```

```
    delay(1000);  // Espera antes de revisar nuevamente
  }
}
```

Funcionamiento

Primero de todo, **para que el código funcione correctamente deberás haberte descargado algún archivo sampleaac.h**.
Adjunto link del que hemos utilizado en esta práctica (deberás guardarlo en la carpeta src):
https://drive.google.com/file/d/18Mik7QvDORN9yfqkLEqSE5Yf69e2D4O4/view?usp=share_link

El funcionamiento del código esta dividido en 2 partes:

1. Inicialización (setup):
    - Se configura la comunicación serial para depuración.
    - Se crea una fuente de audio desde la memoria (sampleaac).
    - Se inicializa un decodificador AAC y una salida I2S.
    - Se ajusta el volumen y se configuran los pines del bus I2S.
    - Se inicia la reproducción del audio.
2. Reproducción continua (loop):
    - Mientras el archivo se está reproduciendo, el programa mantiene activo el bucle de decodificación.
    - Cuando termina, se detiene el generador y se muestra un mensaje en el monitor serial.

Salida Monitor Serial

Este es el único mensaje que aparecerá por el monitor serial, cada segundo una vez que el audio ha terminado de reproducirse.

```
Sound Generator
Sound Generator
Sound Generator
...
```

Video demostración

https://drive.google.com/file/d/1FEKngKEQThw_zCvuI9-KuqanQDOpBi9-/view?usp=share_link

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.

Para este proyecto utilizaremos la libreria

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

Código comentado

```
// Librerías necesarias para reproducción de audio desde SD con I2S
#include "Audio.h"    // Biblioteca principal para manejo de audio
#include "SD.h"       // Biblioteca para la tarjeta SD
#include "FS.h"       // Sistema de archivos

// Definición de pines digitales utilizados
#define SD_CS     10  // Pin chip select de la tarjeta SD
#define SPI_MOSI  11  // Pin MOSI para comunicación SPI
#define SPI_MISO  13  // Pin MISO para comunicación SPI
#define SPI_SCK   12  // Pin de reloj SPI
#define I2S_DOUT   1  // Pin de salida de datos I2S
#define I2S_BCLK  42  // Pin de reloj de bits I2S
#define I2S_LRC   21  // Pin de reloj de canal (LRCLK) I2S

// Se crea un objeto de tipo Audio para gestionar la reproducción
Audio audio;

void setup(){
    // Configura el pin CS de la SD como salida y lo pone en estado alto
    pinMode(SD_CS, OUTPUT);
    digitalWrite(SD_CS, HIGH);

    // Inicializa la comunicación SPI con los pines definidos
    SPI.begin(SPI_SCK, SPI_MISO, SPI_MOSI);

    // Inicia la comunicación serial a 115200 baudios
    Serial.begin(115200);

    // Inicializa la tarjeta SD con el pin CS
    SD.begin(SD_CS);

    // Establece los pines usados para la salida I2S
    audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);

    // Ajusta el volumen de salida (rango de 0 a 21)
    audio.setVolume(10);

    // Conecta el archivo "prova.wav" en la tarjeta SD como fuente de audio
    audio.connecttoFS(SD, "prova.wav");
```

```
}

void loop(){
    // Esta función mantiene la reproducción en funcionamiento
    audio.loop();
}

// Las siguientes funciones son opcionales y se activan automáticamente
// por la librería si hay metadatos o eventos relevantes durante la reproducción

void audio_info(const char *info){
    Serial.print("info        "); Serial.println(info);
}

void audio_id3data(const char *info){  // Muestra metadatos ID3 (como título, artista)
    Serial.print("id3data     "); Serial.println(info);
}

void audio_eof_mp3(const char *info){  // Indica que terminó la reproducción del archivo M
    Serial.print("eof_mp3     "); Serial.println(info);
}

void audio_showstation(const char *info){  // Nombre de la estación (para streaming)
    Serial.print("station     "); Serial.println(info);
}

void audio_showstreaminfo(const char *info){  // Información general del stream
    Serial.print("streaminfo  "); Serial.println(info);
}

void audio_showstreamtitle(const char *info){  // Título actual del stream
    Serial.print("streamtitle "); Serial.println(info);
}

void audio_bitrate(const char *info){  // Bitrate del archivo o stream
    Serial.print("bitrate     "); Serial.println(info);
}

void audio_commercial(const char *info){  // Información de anuncios (streaming)
    Serial.print("commercial  "); Serial.println(info);
```

```
}

void audio_icyurl(const char *info){  // URL del sitio web de la estación
    Serial.print("icyurl      "); Serial.println(info);
}

void audio_lasthost(const char *info){  // Último host de stream utilizado
    Serial.print("lasthost    "); Serial.println(info);
}

void audio_eof_speech(const char *info){  // Fin de un segmento de voz (speech)
    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, "nombre_archivo.wav") En nuestro caso, como hemos guardado el archivo en la sd sin carpetas ni nada, hemos puesto el nombre directamente. (prova.wav). Si tienens un archivo .mp3, necesitarás alguna aplicación de edición musical para poder exportarlo a .wav. **También es necesario que la micro sd este formateada en fat32.**

Funcionamiento

1. Inicialización (setup):
    - Se configuran los pines SPI e I2S.
    - Se inicia la tarjeta SD y se carga el archivo "prova.wav".
    - Se ajusta el volumen y se prepara la salida de audio por I2S.
2. Reproducción (loop):
    - La función audio.loop() mantiene la reproducción activa.

Salida Monitor Serial

En este caso los mensajes no siempre aparecen. Solo lo hacen si el archivo contiene alguna información en concreto.
Para archivos .wav simples sin metadatos, puede que solo veas info y eof_mp3.

Explico detalladamente cada información que puede aparecer:

1. info WAV format detected
    - Significa: El archivo cargado es de tipo WAV.
    - Cuándo aparece: Al comenzar la reproducción, cuando la librería identifica el formato del archivo.
2. streaminfo Sample rate: 44100 Hz
    - Significa: La tasa de muestreo del audio es 44,100 Hz (estándar en calidad de CD).
    - Cuándo aparece: Justo después de detectar el formato, cuando se extrae la información técnica del archivo.
3. bitrate 1411 kbps
    - Significa: El archivo tiene una tasa de bits de 1411 kilobits por segundo (propio de WAV sin compresión).
    - Cuándo aparece: Al inicio, junto con los datos del stream o archivo.
4. id3data Artist: Example Artist
    - Significa: Se leyó una etiqueta ID3 con el nombre del artista.
    - Cuándo aparece: Si el archivo tiene metadatos ID3 (comunes en archivos MP3, no tanto en WAV).
5. id3data Title: Example Title
    - Significa: Se leyó una etiqueta ID3 con el título de la canción.
    - Cuándo aparece: Igual que el anterior, si los metadatos están presentes.
6. eof_mp3 File finished
    - Significa: Se ha llegado al final del archivo de audio.
    - Cuándo aparece: Al terminar completamente la reproducción del archivo.

Ejemplo:

```
info        WAV format detected
streaminfo  Sample rate: 44100 Hz
bitrate     1411 kbps
id3data     Artist: Example Artist
id3data     Title: Example Title
eof_mp3     File finished
```

Video demostración

https://drive.google.com/file/d/1jT4zRnVOT6snaNY74205sHUvjpfgM6SU/

view?usp=share_link