Reproductor De Música

Sebastian Perez Capitano

Irania Aguinaga Muñoz

PROYECTO (REPRODUCTOR DE MÚSICA)

Introducción

El proyecto tiene como objetivo el desarrollo de un sistema de reproducción musical interactivo basado en el microcontrolador ESP32-S3-WROOM-1. Utilizando tecnologías accesibles y componentes de bajo coste, buscamos crear una plataforma eficiente que permita a los usuarios reproducir música en formato digital mediante una interfaz sencilla, a través de un display OLED, un sensor de proximidad y un pulsador para el control de volumen y selección de canciones.

El sistema está diseñado para ser utilizado sin conexión a internet, lo que lo hace ideal para aplicaciones de bajo consumo y portátiles. El ESP32-S3-WROOM-1 es la pieza central, dado su potente procesador, sus capacidades de conectividad y su versatilidad para gestionar múltiples dispositivos a través de interfaces como I2C, SPI y I2S.

Objetivo Principal

El objetivo principal del proyecto es implementar un sistema que permita al usuario:

1. **Reproducir música desde una tarjeta microSD o almacenamiento local.**

2. **Controlar la reproducción mediante un pulsador para subir/bajar el volumen y un sensor de proximidad para cambiar de canción sin necesidad de contacto físico.**

3. **Visualizar la información sobre la canción actual (nombre, artista, duración) en un display OLED.**

4. **Garantizar un consumo eficiente de energía, utilizando la potencia de procesamiento del ESP32-S3 para gestionar audio y otros sensores en tiempo real.**

Componentes del Sistema

El sistema consta de los siguientes elementos:

- **ESP32-S3-WROOM-1: Microcontrolador principal encargado de la gestión del sistema.**
- **Display OLED 0.96" (I2C): Para mostrar información sobre la canción que se está reproduciendo.**
- **Sensor de proximidad: Para detectar el movimiento y cambiar automáticamente la canción sin necesidad de pulsar botones.**
- **Pulsador: Para controlar manualmente el volumen y otras funciones de la reproducción.**
- **Módulo microSD: Para almacenar las canciones en formato WAV o MP3.**
- **DAC I2S y altavoz: Para la salida de audio.**

Metodología

El desarrollo de este proyecto se realizará utilizando el framework Arduino IDE para programar el ESP32, facilitando el acceso a las bibliotecas necesarias para interactuar con los periféricos como el display OLED, el sensor de proximidad y la tarjeta microSD. El sistema de audio se gestionará utilizando la librería Audio para ESP32, que es capaz de manejar archivos de audio y controlar la salida a través del puerto I2S.

Proceso De Montaje

Paso 1 Comprobación de Componentes Individuales

Para asegurarnos de que cada componente funcione correctamente por separado, realizamos las siguientes pruebas iniciales:

A. Verificación del Display OLED (I2C)

1. **Conectar el OLED al ESP32 utilizando los pines SDA (GPIO 21) y SCL (GPIO 22).**

2. **Utilizar el código de prueba de la librería Adafruit_SSD1306 para verificar que el display muestra texto correctamente.**

B. Verificación del Sensor de Proximidad

1. **Conectar el sensor de proximidad al GPIO 15 del ESP32.**

2. **Utilizar un código básico para leer el estado del sensor y verificar su funcionamiento.**

C. Verificación del Pulsador

1. **Conectar el pulsador al GPIO 4, con una resistencia pull-up interna configurada en el ESP32.**

2. **Probar la lectura del estado del pulsador (presionado/no presionado).**

D. Verificación de la Tarjeta microSD

1. **Conectar la tarjeta microSD al ESP32 utilizando los pines SPI: CS, MOSI , MISO y SCK..**

2. **Utilizar un código de prueba para asegurarse de que los archivos pueden leerse y escribirse correctamente en la tarjeta.**

E. WW

1. **Conectar el DAC a los pines BCLK, LRC y DOUT del ESP32.**

2. **Realizar una prueba básica para verificar la salida de audio a través de un altavoz conectado al DAC.**

Paso 2: Integración de los Componentes

Paso 2: Integración de los Componentes

Conexión de los Componentes

Una vez que se ha verificado el funcionamiento individual de cada componente (pantalla OLED, sensor de proximidad, pulsador, microSD y DAC), se procede a integrar todos los módulos al microcontrolador ESP32-S3-WROOM-1. A continuación, se detallan las conexiones necesarias para cada uno de los periféricos:

1. **Conexión de la Pantalla OLED (I2C)**
    - **VCC → 3.3V (del ESP32)**
    - **GND → GND (del ESP32)**
    - **SDA → GPIO 21 (del ESP32)**
    - **SCL → GPIO 22 (del ESP32)**
2. **La pantalla OLED se conecta utilizando el protocolo I2C, lo cual implica que solo necesitamos los pines SDA y SCL para la comunicación de datos, además de la alimentación VCC y GND.**
3. **Conexión del Sensor de Proximidad**
    - **VCC → 3.3V o 5V (dependiendo del sensor, pero el ESP32 es compatible con 3.3V)**
    - **GND → GND (del ESP32)**

- **OUT → GPIO 15 (del ESP32)**

4. **El sensor de proximidad se conecta al ESP32 para detectar cambios en la proximidad. El OUT del sensor envía una señal digital (HIGH o LOW) al ESP32, indicando si la proximidad ha sido detectada.**

5. **Conexión del Pulsador**

   - **Un extremo → (del ESP32)**

   - **Otro extremo → GND (del ESP32)**

6. **El pulsador está configurado como una entrada con resistencia pull-up interna del ESP32. Cuando el pulsador se presiona, el pin GPIO 4 pasa a LOW, lo que indica una acción como el ajuste de volumen o el cambio de canción.**

7. **Conexión de la Tarjeta microSD (SPI)**

   - **VCC → 3.3V (del ESP32)**

   - **GND → GND (del ESP32)**

   - **CS → GPIO 5 (del ESP32)**

   - **MOSI → GPIO 23 (del ESP32)**

   - **MISO → GPIO 19 (del ESP32)**

   - **SCK → GPIO 18 (del ESP32)**

8. **La tarjeta microSD se conecta al ESP32 utilizando el protocolo SPI. Los pines CS, MOSI, MISO y SCK permiten la comunicación con la tarjeta microSD para leer y escribir archivos de audio.**

9. **Conexión del DAC I2S y Altavoz**

   - **BCLK (Bit Clock) → GPIO 26 (del ESP32)**

- **LRC (Left-Right Clock) → GPIO 25 (del ESP32)**
- **DOUT (Data Out) → GPIO 22 (del ESP32)**
- **VCC → 3.3V (del ESP32)**
- **GND → GND (del ESP32)**
- **Altavoz → Conectado al módulo DAC (según el modelo de DAC utilizado)**

El DAC I2S recibe los datos de audio en formato digital a través del bus I2S, y luego convierte esta señal digital en una señal analógica que se envía al altavoz para su reproducción.

IMPLEMENTACION DEL CODIGO

```
#include <Arduino.h>

#include <Wire.h>

#include <Adafruit_SSD1306.h>

#include <Adafruit_GFX.h>

#include <Adafruit_VL53L0X.h>

#include <Audio.h>   // ESP32-audio I2S

#include <SD.h>

#include <FS.h>

#include <SPI.h>
```

Arduino.h: Incluye funciones básicas de Arduino.
Wire.h: Librería para comunicación I2C, usada para el OLED.
Adafruit_SSD1306.h: Librería específica para controlar pantallas OLED de Adafruit.
Adafruit_GFX.h: Librería gráfica de Adafruit para trabajar con pantallas.
Adafruit_VL53L0X.h: Librería para interactuar con el sensor VL53L0X de proximidad.
Audio.h: Librería para manejar la reproducción de audio en el ESP32 utilizando I2S.

SD.h y FS.h: Librerías para gestionar la lectura/escritura de archivos en una tarjeta microSD.
SPI.h: Librería para la comunicación SPI, que es usada por la tarjeta microSD.

```
// ---------- OLED ----------

#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 32

#define OLED_SDA 4

#define OLED_SCL 5

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);


// ---------- VL53L0X ----------

Adafruit_VL53L0X lox = Adafruit_VL53L0X();


// ---------- SD ----------

#define SD_CS 15

#define SD_MOSI 18

#define SD_MISO 17
```

```
#define SD_SCK  16


// ---------- BOTONES ----------

#define BTN_VOL_UP   12

#define BTN_VOL_DOWN 14
```

Pantalla OLED (I2C): Configura la pantalla OLED con 128x32 píxeles usando los pines SDA (4) y SCL (5).

Sensor VL53L0X: Se define el sensor de proximidad VL53L0X para medir distancias.

Tarjeta SD: Define los pines de comunicación SPI (para la tarjeta microSD) usando CS, MOSI, MISO, y SCK.

Botones de Volumen: Los botones para subir y bajar el volumen están conectados a los pines 12 y 14.

Audio: Se inicializa la librería Audio para gestionar la salida de audio a través de I2S.

```
// ---------- AUDIO ----------

Audio audio;


// ---------- VARIABLES ----------

#define MAX_CANCIONES 20

String canciones[MAX_CANCIONES];

int totalCanciones = 0;

int currentSong = 0;

float volume = 0.5; // 0.0 a 1.0
```

canciones[]: Array que almacenará las rutas de las canciones detectadas en la tarjeta SD.

totalCanciones: Contador del número de canciones encontradas en la tarjeta SD.

currentSong: Índice de la canción actual en reproducción.

volume: Control del volumen, con valores entre 0.0 (silencio) y 1.0 (máximo).

```
// ---------- FUNCIONES ----------
void mostrarInfo(String titulo, String artista) {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println("Reproductor MP3");
  display.println("Cancion:");
  display.println(titulo);
  display.println("Artista:");
  display.println(artista);
  display.display();
}
```

Objetivo: Muestra en el OLED el título de la canción y el artista.

Flujo: Se limpia la pantalla (clearDisplay()), se define el tamaño del texto, y luego se imprime la información en la pantalla.

```
void reproducirCancion(String path) {
  audio.stopSong();
  audio.connecttoFS(SD, path.c_str());
  audio.setVolume(volume * 21);
  mostrarInfo(path, "Artista X");
}
```

Objetivo: Reproduce la canción ubicada en la ruta path.

Flujo:

1- Se detiene cualquier canción en curso (audio.stopSong()).

2- Se conecta al archivo de audio en la tarjeta SD usando la ruta proporcionada (audio.connecttoFS(SD, path)).

3-Se ajusta el volumen con audio.setVolume(volume * 21), ya que el volumen se escala a un rango adecuado para el ESP32.

4-Se actualiza la información del OLED con el título de la canción.

```
void escanearCanciones(fs::FS &fs, const char *dirname) {
  File root = fs.open(dirname);
  if (!root || !root.isDirectory()) {
    Serial.println("Directorio inválido");
    return;
  }
  while (true) {
    File file = root.openNextFile();
    if (!file) break;
```

```
    String nombre = file.name();
    if (!file.isDirectory() && nombre.endsWith(".wav")) {
      if (totalCanciones < MAX_CANCIONES) {
        canciones[totalCanciones++] = "/" + nombre;
        Serial.println("Encontrado: " + canciones[totalCanciones - 1]);
      }
    }
    file.close();
  }
}
```

Objetivo: Escanea la tarjeta SD y encuentra todos los archivos .wav.

Flujo:

Abre la raíz del directorio de la tarjeta SD

Recorre todos los archivos en el directorio (openNextFile()).

Si el archivo es un archivo .wav, se agrega a la lista de canciones.

Imprime en el monitor serie el nombre de cada canción encontrada.

```
// ---------- SETUP ----------
void setup() {
  Wire.begin(OLED_SDA, OLED_SCL);
  Serial.begin(115200);

  // OLED
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("Fallo al iniciar el OLED");
    while (1);
  }

  // VL53L0X
  if (!lox.begin()) {
    Serial.println("Fallo al iniciar el VL53L0X");
    while (1);
  }

  // BOTONES
  pinMode(BTN_VOL_UP, INPUT_PULLUP);
  pinMode(BTN_VOL_DOWN, INPUT_PULLUP);


  // SD
  if (!SD.begin(SD_CS)) {
```

```
  Serial.println("No se pudo montar la tarjeta SD");
  while (1);
 }

 escanearCanciones(SD, "/");
 if (totalCanciones == 0) {
  Serial.println("No hay canciones .wav en la SD");
  while (1);
 }

 // AUDIO
 audio.setPinout(46, 3, 29); // Ajusta a tu conexión I2S
 audio.setVolume(volume * 21);
 reproducirCancion(canciones[currentSong]);
}
```

- Objetivo: Inicializa todos los componentes (OLED, sensor, botones, SD, y audio) y comienza a reproducir la primera canción encontrada en la tarjeta SD.
- Flujo:
    1. Inicializa la pantalla OLED.
    2. Inicializa el sensor VL53L0X para medición de distancia.
    3. Configura los pines de los botones volumen como entradas con resistencias pull-up.

4. Inicializa la tarjeta microSD y escanea las canciones disponibles.
5. Configura la salida de audio I2S.
6. Reproduce la primera canción disponible.

```
// ---------- LOOP ----------
void loop() {
  audio.loop();


  // Volumen
  if (digitalRead(BTN_VOL_UP) == LOW) {
    volume = min(1.0, volume + 0.1);
    audio.setVolume(volume * 21);
    Serial.println("Volumen: " + String(volume, 1));
    delay(200);
  }
  if (digitalRead(BTN_VOL_DOWN) == LOW) {
    volume = max(0.0, volume - 0.1);
    audio.setVolume(volume * 21);
    Serial.println("Volumen: " + String(volume, 1));
    delay(200);
  }
```

```
 // Sensor de proximidad: cambio de canción
  VL53L0X_RangingMeasurementData_t measure;
  lox.rangingTest(&measure, false);
  if (measure.RangeStatus == 0 && measure.RangeMilliMeter < 100) {
    currentSong = (currentSong + 1) % totalCanciones;
    reproducirCancion(canciones[currentSong]);
    delay(1000); // evitar doble lectura por un solo gesto
  }
}
```

- Objetivo: Gestiona la reproducción en bucle y los controles de volumen y cambio de canción mediante los botones y el sensor de proximidad.
- Flujo:
    1. Llama a audio.loop() para gestionar la reproducción de audio.
    2. Si el botón de volumen arriba es presionado, incrementa el volumen.
    3. Si el botón de volumen abajo es presionado, decrementa el volumen.
    4. El sensor de proximidad se utiliza para detectar cuando el usuario está cerca (menos de 100 mm) y, si es así, cambia a la siguiente canción.

DIFICULTADES

Problema de lectura SD

Durante el desarrollo del proyecto, se presentaron varios problemas de conectividad relacionados con la tarjeta microSD. Específicamente, el ESP32 no detectaba la tarjeta, lo que impedía la correcta lectura de los archivos almacenados en la misma. Esta falta de detección por parte del microcontrolador resultó en la incapacidad de reproducir las canciones que habíamos cargado en la tarjeta SD, lo que afectó el funcionamiento esperado del sistema.

Problema de Conectividad con el Altavoz:

Durante la fase de integración de los componentes, surgió un problema de conectividad relacionado con la conexión del altavoz al sistema basado en el ESP32-S3-WROOM-1. El principal desafío fue la interfaz de salida de audio, ya que no estábamos seguros de cómo conectar el altavoz correctamente al microcontrolador para que la señal de audio se reprodujera correctamente.