A
PROJECT REPORT ON

Audio Player Using STM32F407G-DISC1

SUBMITTED IN
PARTIAL FULFILLMENT OF

DIPLOMA IN EMBEDDED SYSTEM DESIGN (PG-DESD)

BY

SHUBHANGI RAUT

AT

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY, HINJAWADI

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY, HINJAWADI.

CERTIFICATE

This is to certify that the project

Audio Player Using STM32F407G-DISC1

Has been submitted by

SHUBHANGI RAUT

In partial fulfillment of the requirement for the Course of **PG Diploma in Embedded System Design (PG-DESD SEP 2022)** as prescribed by The **CDAC** ACTS, PUNE.

Place: Hinjawadi Date:

Authorized Signature

ACKNOWLEDGEMENT

It is indeed a matter of great pleasure and proud privilege to be able to present this project on “Audio Player Using STM32f407DISC1”. The completion of this project work is great experience in student’s life and its execution is inevitable in hands of guide. We are highly indebted to the Project Guide Mr. Ajay Kumar for his valuable guidance and appreciation for giving form and substance to this project. It is due to his enduring efforts, patience and enthusiasm which has given sense of direction and purposefulness to this project and alternately made it a success.

Amongst the panorama of other people who supported us in this project, we are highly indebted to our technical staff of Sunbeam. It is their support, encouragement and guidance which motivated us to go ahead and make the project a successful venture.

Lastly, we would like to express our gratitude to all other helping hands which have been directly or indirectly associated with our project.

Date:

SHUBHANGI RAUT(69984)

ABSTRACT

With the advancement in semiconductor technology, scope for development of embedded systems has increased manifolds. New processors with improved computing capabilities and low power consumption have further accelerated the developments in embedded domain. Consumers are looking for affordable multimedia devices with high performance and durability making embedded developers to think creatively and use all resources at hand to meet the desired user specifications. This is one such attempt by designing an Audio Player to play the wave audio files from a USB flash drive in the same format.

INDEX

LIST OF FIGURES

1. INTRODUCTION

1.1. Introduction

STM32 microcontrollers offer a large number of serial and parallel communication peripherals which can be interfaced with all kinds of electronic components. Audio player using STM32F0 Discovery board interfaced to a USB device and using on chip DAC.

Music is stored on an USB device connected to STM32F0 board on I2S in the form of WAV files. Music is one of the best ways to relieve pressure in stressful modern society life.

An audio file is a type of file which stores different sounds which were recorded. These audio files store the digital form of these sounds and they are converted into Analog format to play them. there are 3 categories of audio files formats in real world.

The purpose of this project is to develop a player which can play the mainstream file format. Which can be controlled by mobile application. Meanwhile, this mobile application can play, pause and repeat with play button and next button according to sets requirement as well as set up songs.

1.2 Overview Of Project

An Audio Player which uses STM32F407DISC1 discovery board which takes Wave Audio files from the connected USB drive and play it with the help of DMA controller and I2S protocols. The audio is controlled (by having pause, resume, next, previous and stop features) with the help of a user switch which is on the board.

Playing wave audio files which are the basic high quality audio format because these are also uncompressed files, the quality of the output is nowhere compromised.

Thus, if one is seeking to play music with a single device, the device thus developed may come very handy. With the improvements in processor technology and development of software design environments for the same, designing of an embedded system has not remained a very tedious task as it was a decade ago. Processors such ARM which have very good computing performances are becoming more popular among embedded developers.

2. LITERATURE SURVEY

A. Statement of Purpose : -

The Audio Player is the only hardware which plays the audio from a recorded file. These provide excitement, comfort, well-being and reduce depression to its listeners. In general, these players should be designed with some specifications like :-

- The clarity of the audio output should be more with minimal noise so that humans can understand the sounds.
- The output shouldn't affect the medical conditions of a person at the maximum volume (especially for heart patients).It should cancel the external noise while playing an audio with minimal volume.
- There should be no data loss in the system.
- Sometimes high-quality audio is required to match the exact sound in the real world. This is mostly followed by audio players used for professional audio needs. Recently everyone is focusing on wireless hardware and mostly they want to control each and every hardware.
- To develop an application-based Audio player, which works according to the control given by command icons provided in application. With the help of Stm32F407 board, for communication between USB drive and speaker and use audio files in .wav format.

B. Research for Project :-

After viewing many papers and browsing many websites we made a decision to design an audio player using the advantages of the following :-

✦ **.WAV Format:** It's an uncompressed Audio file format, which was developed by IBM and Windows for storing audio bit streams on PC. The data is stored in a chunk in RIFF (Resource Interchange File Format) bit stream format. A file consists of single chunk which divided into three parts:" Header, Format sub-chunk, Data sub-chunk".

RIFF WAVE File Format

Endian	File Offset	Field Name	Field Size	Description
big	0	Chunk ID	4	**"RIFF" Chunk Descriptor** The format field here is "WAVE", and it requires two sub-chunks: "fmt" chunk and "data" chunk.
little	4	Chunk Size	4	
big	8	Format	4	
big	12	SubChunk1 ID	4	
little	16	SubChunk1 Size	4	
little	20	Audio Format	2	**"fmt" Sub-Chunk** This region contains all essential information we need to know about a WAVE file, such as sample rate, channel number etc.
little	22	Channel Number	2	
little	24	Sample Rate	4	
little	28	Byte Rate	4	
little	32	Block Align	2	
little	34	Bits Per-Sample	2	
big	36	SubChunk2 ID	4	**"data" Sub-Chunk** The field "SubChunk2 Size" indicates the size of RAW audio data.
little	40	SubChunk2 Size	4	
little	44	Data	SubChunk2 Size	

Fig 2.1 WAV file Chunk

- **FAT fs:** FatFs is a generic FAT/exFAT filesystem module for small embedded systems. The FatFs module is written in compliance with ANSI C (C89) and completely separated from the disk I/O layer. Therefore ,it is independent of the platform. It can be incorporated into small microcontrollers with limited resource, such as 8051, PIC, AVR, ARM, Z80, RX and etc. Also, Petit FatFs module for tiny microcontrollers is available.

  Features:
  - DOS/Windows Compatible FAT/exFAT Filesystem.
  - Platform Independent. Easy to port.
  - Very Small Footprint for Program Code and Work Area.
  - Various Configuration Options to Support for:
  - Long File Name in ANSI/OEM or Unicode.
  - exFAT Filesystem, 64-bit LBA and GPT for Huge Storages.
  - Thread Safe for RTOS.
  - Multiple Volumes. (Physical Drives and Partitions).
  - We have used FAT32 file system in this project.

- **I2S:** I2S is an electrical serial bus interface standard used for connecting digital audio devices together. It is used to communicate PCM audio data between integrated circuits in an electronic device. The I²S bus separates clock and serial data signals, resulting in simpler receivers than those required for asynchronous communications systems that need to recover the clock from the data stream.
  It transfers data through a simple, 3-line serial bus consisting of:
  - ***Continuous Serial Clock (SCK)***
  - ***Word Select (WS)***
  - ***Serial Data (SD)***

- **Features:**
- It supports only single master without clock stretching.
- The data doesn't have any start and stop bits.

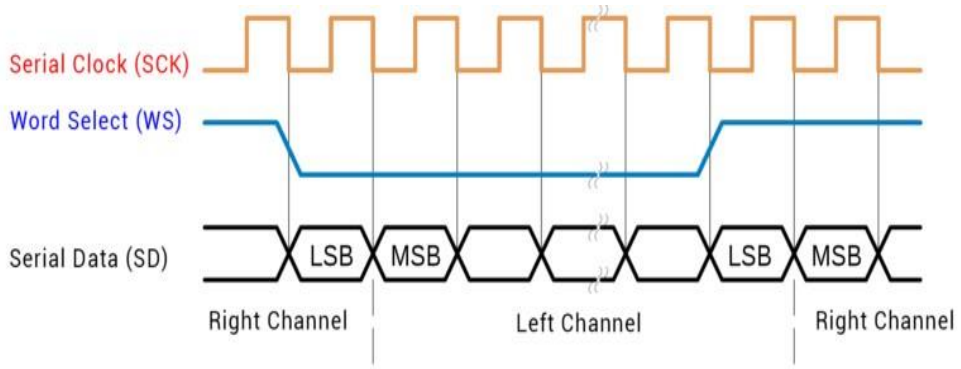

Fig 2.2 I2S Timing Diagram

✦ **I2C:** I2C stands for **Inter-Integrated Circuit.** It is a bus interface connection protocol incorporated into devices for serial communication. It was originally designed by Philips Semiconductor in 1982. Recently, it is a widely used protocol for short-distance communication. It is also known as Two Wired Interface (TWI).

Working: -

It uses only 2 bi-directional open-drain lines for data communication called SDA and SCL. Both these lines are pulled high.

- ***Serial Data (SDA)*** – Transfer of data takes place through this pin.
- ***Serial Clock (SCL)*** – It carries the clock signal.

❖ I2C operates in 2 modes –

- Master mode
- Slave mode

Each data bit transferred on SDA line is synchronized by a high to the low pulse of each clock on the SCL line.

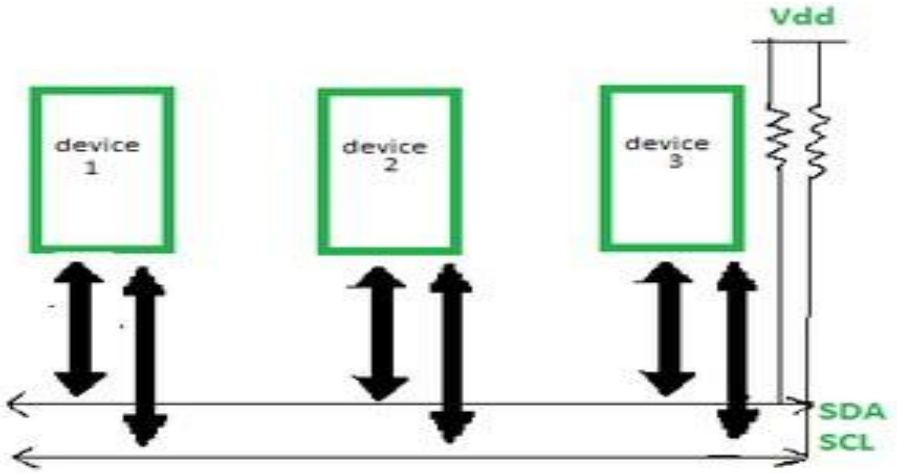

Fig 2.3 I2C connection

1. **Start Condition** – 1 bit
2. **Slave Address** – 8 bit
3. **Acknowledge** – 1 bit

Start and Stop Conditions :

START and STOP can be generated by keeping the SCL line high and changing the level of SDA. To generate START condition the SDA is changed from high to low while keeping the SCL high. To generate STOP condition SDA goes from low to high while keeping the SCL high, as shown in the figure below.

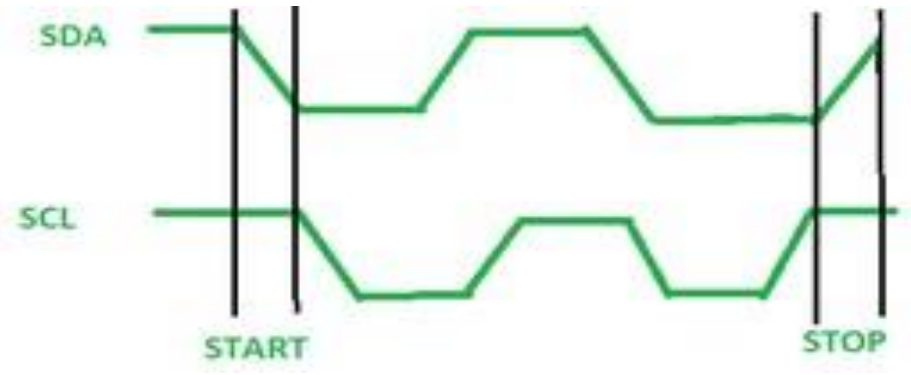

Fig 2.4 I2C Timing Diagram

3. PROJECT ARCHITECTURE

3.1. Block Diagram:

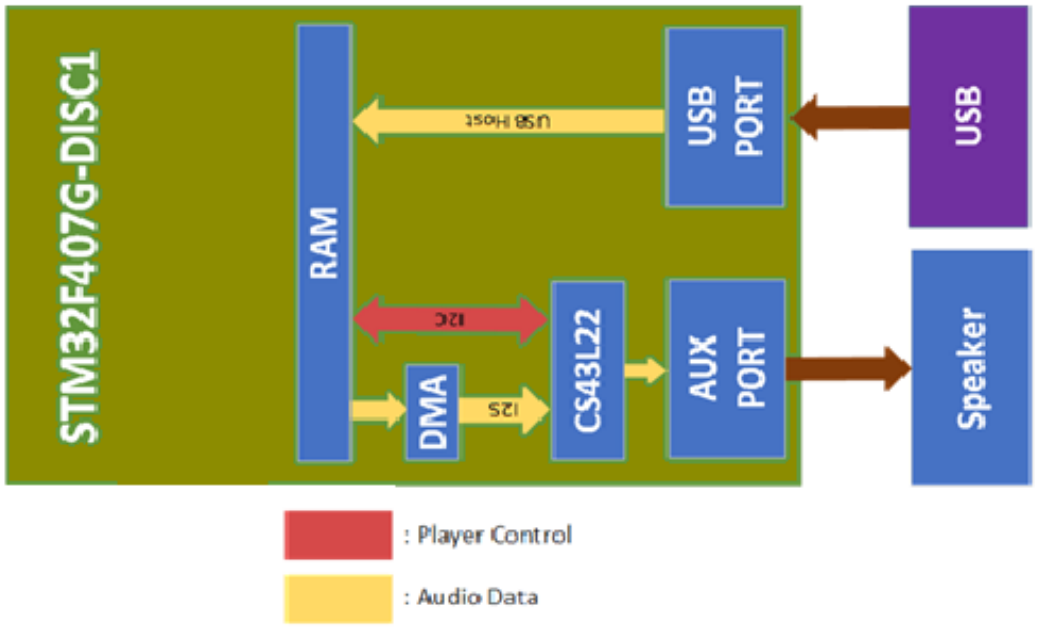

Fig 3.1. Block diagram

As you can see from the above figure. To design this system, we have used a 'STM32F407G-DISC1' discovery board which consists of a micro-USB port, AUX port and a DAC which are required mostly. Here USB port is used to connect the USB device which contains audio files with '.WAV' extension. AUX port is used to connect to a speaker using an AUX cable. CS43L22 DAC is used to convert the digital data of the audio to analog signal so that it can play continuously over the speaker.

To establish communication between these peripherals protocols like I2S(for transferring the audio data to DAC), I2C(for controlling the DAC as required), UART(for receiving commands from mobile), DMA(for accessing the memory without CPU interference), USB HOST(for retrieving the data from the connected USB storage device) are used.
We are connecting USB drive to STM32 Discovery Board (STM32F407G-DISC1) and the module is powered up by connecting it to the power supply of 5v. Connect the speaker to AUX port of the board using an AUX cable. Connect a Pen-drive to STM32 board's micro USB port directly (if the drive has a micro USB end) or use a Converter if it doesn't has it. Make sure there are audio files with .WAV format. STM32 has a CS43L22 chip which acts as a DAC and receives data either in Analog mode or I2S mode. In addition, this chip requires I2C protocol to control the chip operations.

3.2. Flowchart:

The workflow of this project is to complex to explain in one flowchart. So it is divided into four parts.

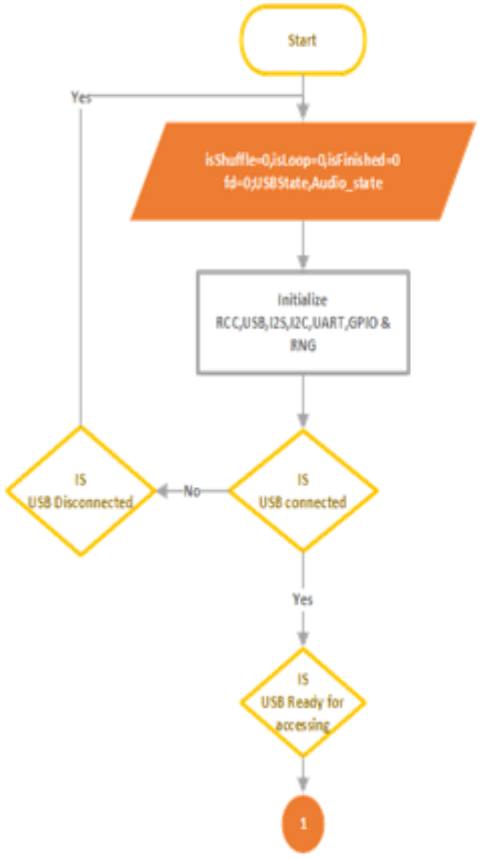

Fig 3.2. Flowchart part-1

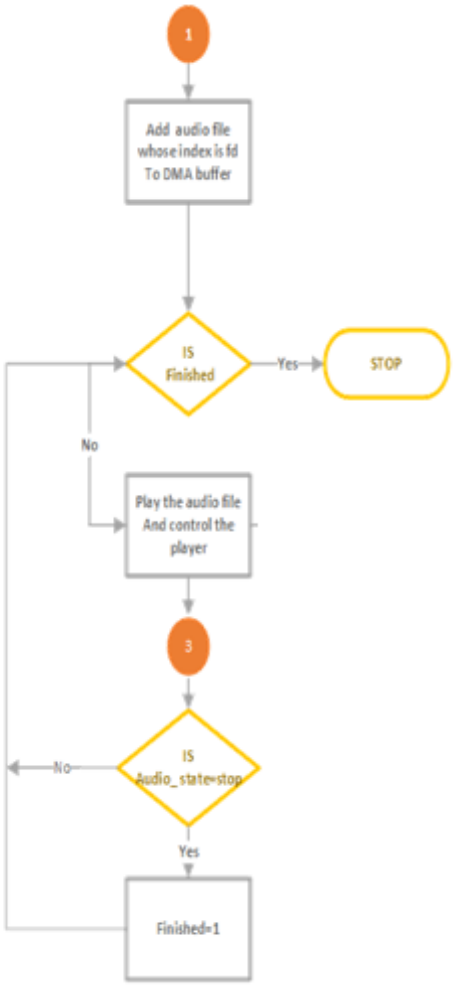

Fig 3.3. Flowchart part-2

- The first part of the flow chart describes the work on initializations by the circuit and it also deals with the USB Connections. If there is a USB Connected then it goes to the second part.
- The second Part of the flow chart tells how audio gets initialized if it has .WAV file and starts to play the audio.
- It is the last part of the execution where it runs the particular audio operation based on the Audio state variable and these operations(pause , resume , next , previous and stop). Each operation deals with the controlling of DAC using I2C protocol and Data transmissions using I2S with the help of DMA.

3.3. Working:

The STM32F4 Discovery Board has inbuilt USB port where an external USB drive is connected which stores .wav audio format. The DMA Controller gets data from USB drive to DMA buffer. The DMA & Timer to periodically trigger the DMA unit so that it moves a sample data point from the lookup table stored in memory to the DAC output. Each time the timer trigger event occurs, the DMA transfers a data point from the table to the DAC output and increments the memory pointer to move the next data point in the next trigger event. Further data get transfer to DAC using I2S protocol, where it converts digital data to Analog signal. Finally, the speaker receives this Analog signal from AUX port. For controlling part, the STM32F4 board is having I2C peripheral. As soon as we press a user switch on the STM32 board the .wave file transferred to the circuit based on this state of the Audio Player gets changed and then it performs the respective operation, after that we can listen the song.

4. HARDWARE DESCRIPTION

4.1. STM32F407 Discovery Board

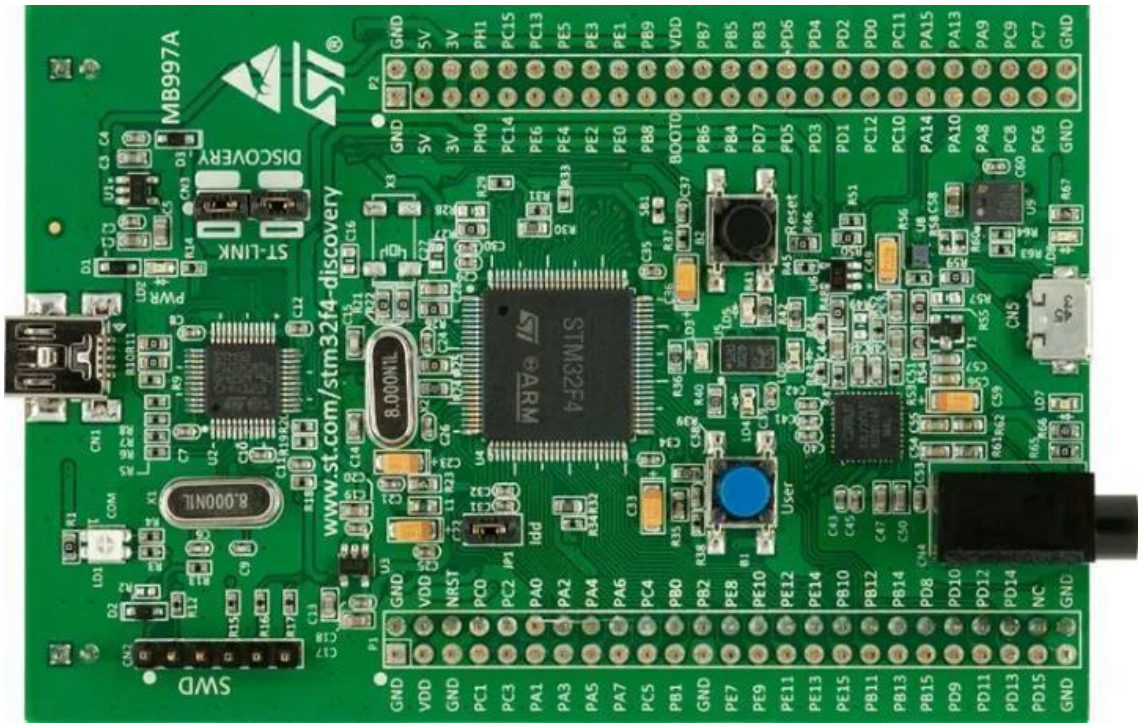

Fig 4.1 STM32F407 Discovery Board

The STM32F407lines are designed for medical, industrial and consumer applications where the high level of integration and performance, embedded memories and rich peripheral set inside packages as small as 10 x 10 mm are required.

The STM32F407 offers the performance of the Cortex™-M4 core (with floating point unit) running at 168 MHz.

Performance: At 168 MHz, the STM32F407 delivers 210 DMIPS/566 CoreMark performance executing from Flash memory, with 0-wait states using ST's ART Accelerator. The DSP instructions and the floating-point unit enlarge the range of addressable applications.

Power efficiency: ST's 90 nm process, ART Accelerator, and the dynamic power scaling enables the current consumption in run mode and execution from Flash memory to be as low as 238 µA/MHz at 168 MHz .

Rich connectivity: Superior and innovative peripherals: Compared to the STM32F4x5 series, the STM32F407 product lines feature Ethernet MAC10/100 with IEEE 1588 v2 support and a 8- to 14-bit parallel camera interface to connect a CMOS camera sensor.

- 2x USB OTG (one with HS support)
- ***Audio:*** dedicated audio PLL and 2 full-duplex I²S
- Up to 15 communication interfaces (including 6x USARTs running at up to 11.25 Mbit/s, 3x SPI running at up to 45 Mbit/s, 3x I²C, 2x CAN, SDIO)
- ***Analog:*** two 12-bit DACs, three 12-bit ADCs reaching 2.4 MSPS or 7.2 MSPS in interleaved mode
- ***Up to 17 timers***: 16- and 32-bit running at up to 168 MHz
- Easily extendable memory range using the flexible static memory controller supporting Compact Flash, SRAM, PSRAM, NOR and NAND memories
- Analog true random number generator

The STM32F407 product lines provide from 512 Kbytes to 1 MByte of Flash, 192 Kbytes of SRAM, and from 100 to 176 pins in packages as small as 10 x 10 mm.

4.2 CS43L22 DAC:

The CS43L22 is a highly integrated, low power stereo DAC which has 2 output channels for stereo and mono interface and Class D speaker amplifiers. This chip is internally available in STM32 discovery board. The CS43L22 offers many features suitable for low power, portable system applications. It has about 30 registers in which are used to controls the chip for audio transmission.

FEATURES:

- 98 dB Dynamic Range (A-wtd)
- 88 dB THD+N
- Headphone Amplifier - GND Centered
    - No DC-Blocking Capacitors Required
    - Integrated Negative Voltage Regulator
    - 2 x 23 mW into Stereo 16 Ω @ 1.8 V
    - 2 x 44 mW into Stereo 16 Ω @ 2.5V
- Stereo Analog Input Passthrough Architecture
    - Analog Input Mixing
    - Analog Passthrough with Volume Control
- Digital Signal Processing Engine
    - Bass & Treble Tone Control, De-Emphasis
    - PCM Input w/Independent Vol Control
    - Master Digital Volume Control and Limiter
    - Soft-Ramp & Zero-Cross Transitions ○ Programmable Peak-Detect and Limiter
- Beep Generator w/Full Tone Control
    - Tone Selections Across Two Octaves
    - Separate Volume Control
    - Programmable On and Off Time Intervals
    - Continuous, Periodic, One-Shot Beep Selections

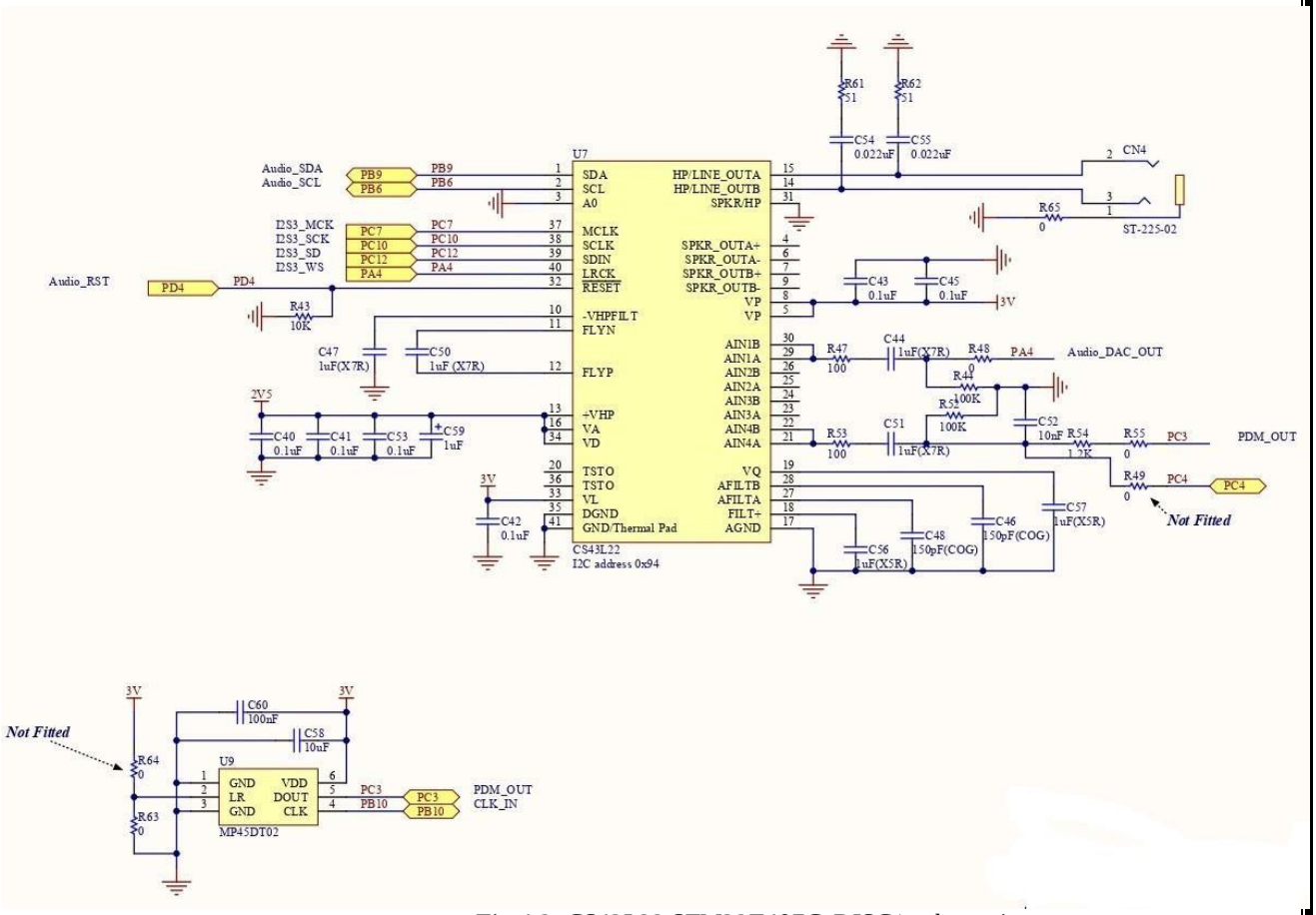

Fig 4.2. CS43L22 STM32F407G-DISC1 schematic

5 SOFTWARE REQUIREMENTS AND SPECIFICATIONS

5.1 STM32CudeIDE

STM32CubeIDE is an all-in-one multi-OS development tool, which is part of the STM32Cube software ecosystem. STM32CubeIDE is an advanced C/C++ development platform with peripheral configuration, code generation, code compilation, and debug features for STM32 microcontrollers and microprocessors.

STM32 Code IDE

6 Source Code Explanation

6.1 Functions used:

- **Main Function:**

It is the starting point of the program. It initializes all the necessary resources required and checks for USB connections and finds if there are any wav files. If there is a wav file then we it starts to play the first file which it found.

```
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_I2C1_Init();
  MX_I2S3_Init();
  MX_FATFS_Init();
  MX_USB_HOST_Init();
  MX_USART2_UART_Init();
  while (1)
  {
    MX_USB_HOST_Process();

    if(Appli_state == APPLICATION_START)
        {
          HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
        }
        else if(Appli_state == APPLICATION_DISCONNECT)
        {
          HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);
        }

    if (Appli_state == APPLICATION_READY)
        {
            Mount_USB();
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET);
            AUDIO_PLAYER_Start(0);
            while (!isfinished)
            {
                AUDIO_PLAYER_Process(repeat,shuffle);
                HAL_UARTEx_ReceiveToIdle_IT(&huart2,(uint8_t*)str,32);
                if (AudioState == AUDIO_STATE_STOP)
                {
                    isfinished = 1;
                }
            }
        }
  }
}
```

- **Audio Player Stop:**

This function internally calls the stop function of Audio codec to stop the player and closes the .wav file. The state of the player is changed to stop.

```
AUDIO_ErrorTypeDef AUDIO_PLAYER_Stop(void)
{
  AudioState = AUDIO_STATE_STOP;
  FilePos = 0;

  AUDIO_OUT_Stop(CODEC_PDWN_SW);
  f_close(&WavFile);
  return AUDIO_ERROR_NONE;
}
```

- **AUDIO_OUT_Play:**

This function actually transmits data to DAC using I2S. before sending data it sets the DAC to be in play state and checks it if it is ready to read the data.

```
uint8_t AUDIO_OUT_Play(uint16_t* pBuffer, uint32_t Size)
{
  if(pAudioDrv->Play(AUDIO_I2C_ADDRESS, pBuffer, Size) != 0)
    return AUDIO_ERROR;
  else
  {
    HAL_I2S_Transmit_DMA(&hAudioOutI2s, pBuffer, DMA_MAX(Size/AUDIODATA_SIZE));
    return AUDIO_OK;
  }
}
```

- **AUDIO_OUT_Pause:**

This function actually pauses the transmission of data to DAC from I2S. before pausing the data transmission it sets the DAC to be in pause state and checks it if it is ready to pause.

```
uint8_t AUDIO_OUT_Pause(void)
{
  if(pAudioDrv->Pause(AUDIO_I2C_ADDRESS) != 0)
    return AUDIO_ERROR;
  else
  {
    HAL_I2S_DMAPause(&hAudioOutI2s);
    return AUDIO_OK;
  }
}
```

- **AUDIO_OUT_Resume:**

This function actually resumes the transmission of data to DAC from I2S. before resuming the data transmission it sets the DAC to be in resume state and checks it if it is ready to resume.

```
uint8_t AUDIO_OUT_Resume(void)
{
  if(pAudioDrv->Resume(AUDIO_I2C_ADDRESS) != 0)
  {
    return AUDIO_ERROR;
  }
  else
  {
    HAL_I2S_DMAResume(&hAudioOutI2s);
    return AUDIO_OK;
  }
}
```

- **AUDIO_OUT_Stop:**

This function actually stops the transmission of data to DAC from I2S. first it stops the transmission and then it sets the DAC to be in stop state and checks it if it is ready to stop. If not, it means that there is an error in the system.

```
uint8_t AUDIO_OUT_Stop(uint32_t Option)
{
  HAL_I2S_DMAStop(&hAudioOutI2s);
  if(pAudioDrv->Stop(AUDIO_I2C_ADDRESS, Option) != 0)
    return AUDIO_ERROR;
  else
  {
    if(Option == CODEC_PDWN_HW)
    {
      HAL_Delay(1);
      HAL_GPIO_WritePin(AUDIO_RESET_GPIO, AUDIO_RESET_PIN, GPIO_PIN_RESET);
    }
    return AUDIO_OK;
  }
}
```

- **Mount_USB:**

This function is used to mount the USB to STM32 board.

```
void Mount_USB (void)
{
    fresult = f_mount(&USBHFatFS, USBHPath, 1);
}
```

- **Unmount_USB:**

This function is used to unmount the USB from STM32 board

```
void Unmount_USB (void)
{
    fresult = f_mount(NULL, USBHPath, 1);
}
```

7 Results

7.1. Hardware Circuit:

As you can see in the below figure, we have connected a HC-05 module, Pen-Drive, speaker to STM32 board. Since it is an audio player there is no change in the circuit that can be shown using image. But, we can show the changes in the app which we have designed.

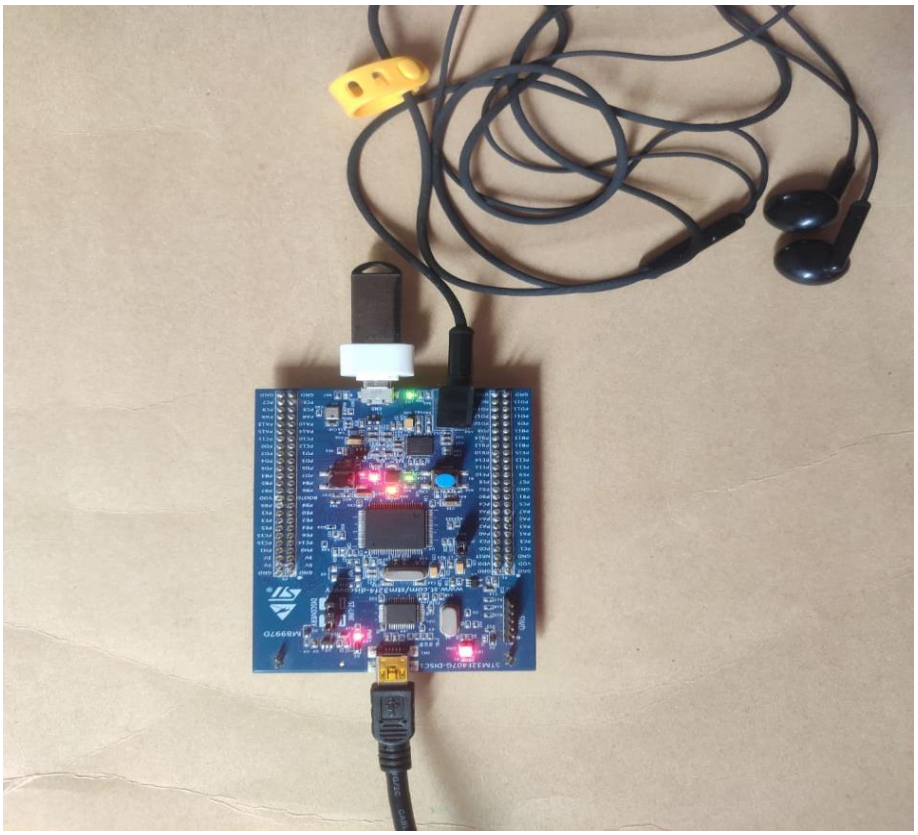

Figure 7.1 final output

8 FUTURE SCOPE

- Create a LIFI based audio transmission between audio player circuit and speaker using LASER lights.
- Upgrade the communication between mobile and hardware by replacing Bluetooth with WIFI and controlling it with the help of cloud.
- Add the volume feature too.
- Upgrade the system in such a way that it can accept multiple file formats.
- We can interface this by gesture control sensor and play audio by using using gestures, moving them right, left, up and down for next, previous, volume up and volume down.

9 CONCLUSION

Through the development of Audio Payer on STM32F4 and Android platform, we get a clear understanding of overall process of the system. The core part of the Audio Player is mainly composed of main interface, file browsing and song listing. This project performs the basic function of Audio Player: play, pause, next, previous, shuffle, etc. which is controlled through an Android System Itself.

As we are using Bluetooth device with the application the processing has become so handy that we can even perform the commands through voice as well.

10 REFERENCES

[1] stm32f407_user_manual

[2] stm32f407vg_disc1_user_manual.

[3] stm32f407vg-data-sheet

[4] stm32f407vg_disc1_schematic

[5] arm_cm4_tech_ref_manual

[6] arm_cm4_tech_ref_manual

[7] https://www.st.com/resource/en/technical_note/tn1199-stm32f4-audio-processing-stmicroelectronics.pdf

[8] https://www.st.com/en/embedded-software/stm32-audio100a.html