Embedded Systems Project

A description of the idea of your system and its functionality

WAV player, giving some audio files in WAV format on a USB drive, we play the audio on earphone speakers with all possible sampling rates. The player can be controlled using Bluetooth class 4.

Detailed description of all the features that your system offers

Reading audio files in .wav format from a USB drive with FAT file system

When the user puts some audio files on a USB, then the microcontroller is able to read it and turn on a green led to inform the user that it could mount the file system successfully. The USB drive should contain only the audio files and no subdirectory is allowed. The audio files can have any possible sampling rate, but it should be stereo (two channels). Files names should be shorter than 13 characters.

Playing the audio files on earphones

When the user plugs the earphones into the audio jack and there's an audio being played, they can listen to that audio.

Control the Audio player with an android phone

Users can connect to microcontroller Bluetooth using a smart phone Bluetooth and an App to send terminal commands. The user can also connect any device but it should have Bluetooth class 4 and pair with the Bluetooth module.

The user can choose to:

1- List the audio files names (l)
2- Choose an audio file to be played (c [name with extension])
3- Pause the audio file. (p)
4- Resume the audio file. (r)
5- Stop the audio file. (s)
6- Mute the player (m)
7- Unmute the player (u)
8- Set the volume of the player (v [0-255])
9- Play the next audio file (>)
10- Play the previous audio file (<)

How each device is used to contribute to the system's functionality

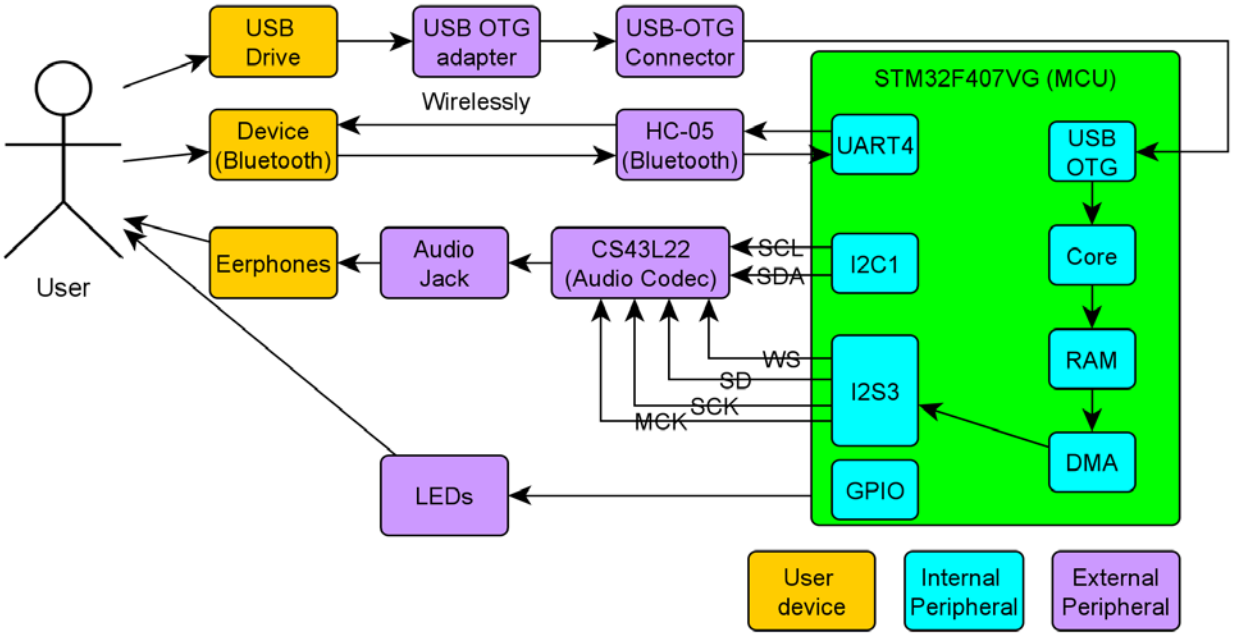


Figure 1 Block Diagram

USB Drive/USB OTG

Audio files on USB device are in .WAV format which contains 44 byte header, as well as the audio left and right samples in 16-bit format.

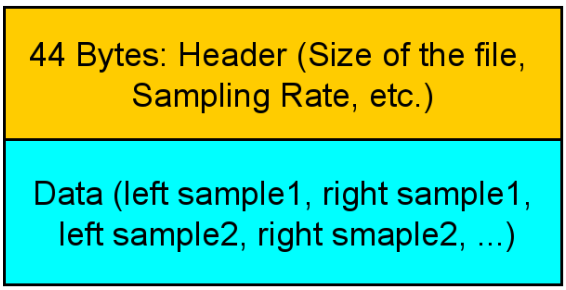


Figure 2 WAV format

The important information in the header file is the size of the file and the sampling rate. The size will help in deciding the end-of-file (EOF), so that the Audio player knows when to stop. Also, the sampling rate is important to know the required clock to be generated.

To deal with file I/O, we need a file system. We used FatFs file system for this purpose. Also to communicate with a storage device we need a communication (SPI, MMC, USB, etc.). We decided that our storage device to be a USB drive, so we used the middlewares provided by the ST company to interface with our USB.

I2S peripheral

The audio samples are transmitted to an Audio Codec (to be explained later) using I2S (Inter-IC Sound, eye-squared-ess) protocol. I2S protocol is made by Phillips like the I2C protocol. Its format is so close to I2C format.

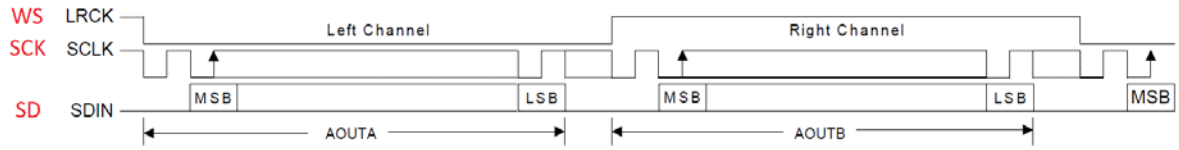


Figure 3 I2S Frame Format

From figure 2, WS determines which sample is sent right now; left or right. The SCK is equivalent to the I2C SCL which is used as a synchronous clock, SD is the data line and is equivalent to I2C SDA.

I2S is better than I2C when it comes to this problem, because there's no header (address) like in I2C, so the bandwidth is much higher. The data is sent directly.

MCLK	37	**Master Clock** (*Input*) **-** Clock source for the delta-sigma modulators.
SCLK	38	**Serial Clock** (*Input/Output*) - Serial clock for the serial audio interface.
SDIN	39	**Serial Audio Data Input** (*Input*) - Input for two's complement serial audio data.
LRCK	40	**Left Right Clock** (*Input/Output*) - Determines which channel, Left or Right, is currently active on the serial audio data line.

Figure 4 I2S pins source: https://datasheet.octopart.com/CS43L22-CNZ-Cirrus-Logic-datasheet-5397077.pdf

I2C peripheral

I2C is used to interface with the Audio codec, so that it can control its functionalities like setting volume, mute, unmute, etc.

UART peripheral

It's used to interface with the Bluetooth module.

HC-05 Bluetooth module

It's used as communication protocol between the user device and the embedded system, so that the user can send commands. The error messages (not found file, bad volume value, etc.) are sent through it.

CS43L22 Audio Codec

This device is responsible for the converting from digital to analog, so that the user can hear the audio file through the earphones. It uses I2C protocol for configuration and I2S protocol for receiving the audio samples. It uses DAC of type Sigma Delta which is very accurate type.

LEDs

It's used to inform users of any important notifications. Green Led : File system is successfully mounted.

STM32F407VG (MCU)

This microcontroller is made by ST company, it has ARM cortex-M4 CPU. It's responsible for handling the user inputs through Bluetooth, interfacing with all peripherals, handling all necessary state machines.

DMA (Direct Memory Access)

It's used to load the audio samples from memory to I2S data register using circular buffer. In this way, CPU can do any other job while the DMA takes care of transferring Audio samples to I2S. The CPU job is to just fill in the DMA buffer every time the buffer is fully/half fully read by DMA. The size of the DMA is chosen Arbitrary but we should care that's not so small otherwise the DMA is useless, or too large otherwise we take much memory space from RAM.

Test cases of each part of the system and each device showing the correctness of its operation, in addition to how to test in details.

Devices

USB Drive/USB-OTG/USB peripheral

- The user plugs in the USB into the USB-OTG port, there's should a green led turned on indicating the success of mounting the file system.

Bluetooth/UART

- The user pair with the Bluetooth module using a smart phone.
- To test UART, the user should send 'l' (list) command, it's expected to get the names of the audio files on the USB drive.

Audio codec CS43L22

- The user should send 'r' (resume) command using Bluetooth while there's at least one audio file on the USB drive, the user should listen to the audio file through the earphones.

Functionalities

Resume/Pause/Stop

- The user can send any of command (r, p, s) to resume/pause/stop the audio file.

Mute/Unmute

- The user should send (m, u) to mute/unmute the audio player. The audio files should not be paused if it's being played already.

List

- The user should send 'l' to list the audio file names on the flash drive.

Next/previous

- The user should send '>' to play the next audio file, or '<' to play the previous audio file.
- If it's the last/first file, the directory will rewind from beginning/end.

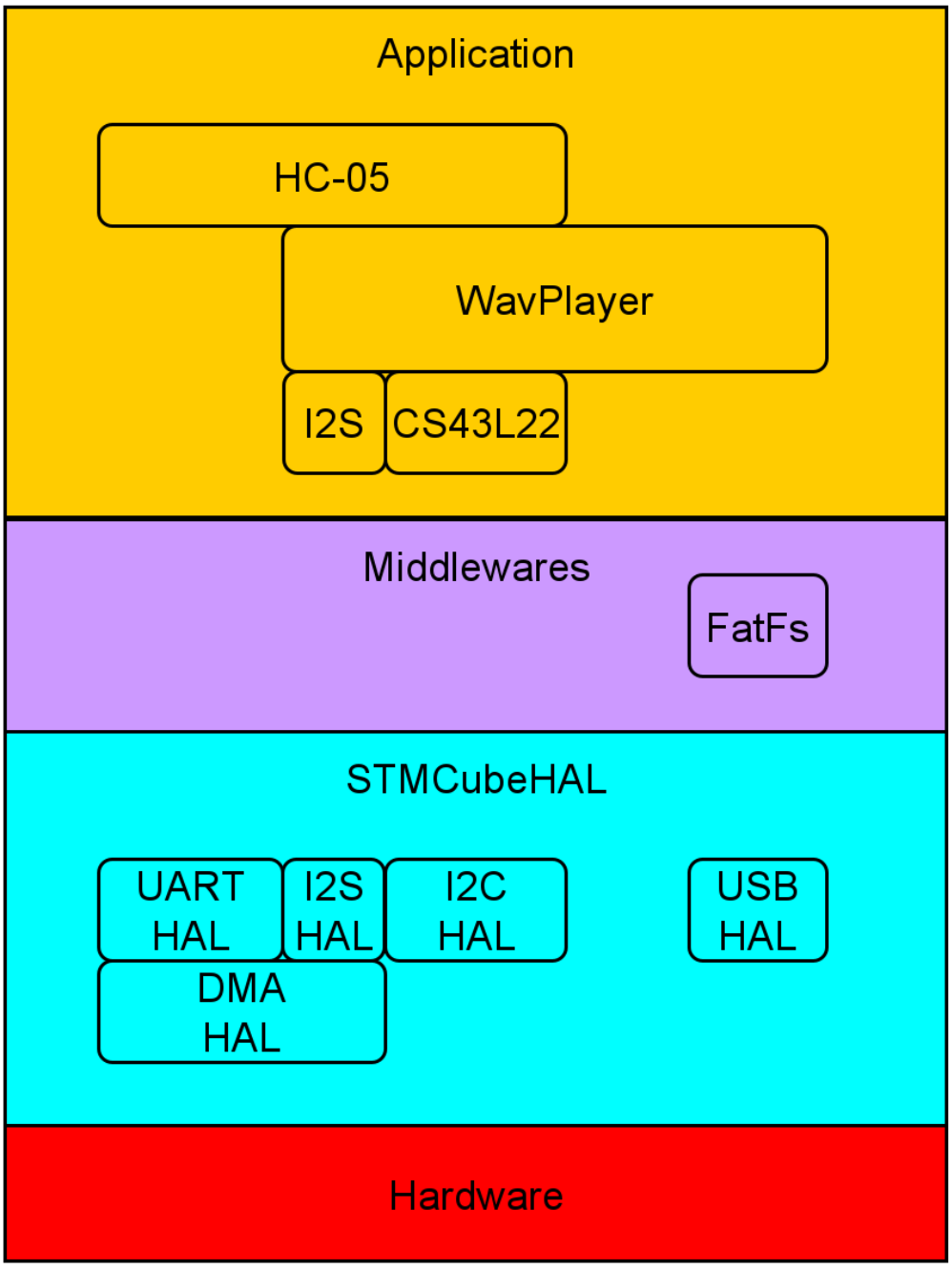
Choose a file to play

- The user should choose a file to play by sending 'c [name.wav]` where name.wav is the file name with the .wav extension.

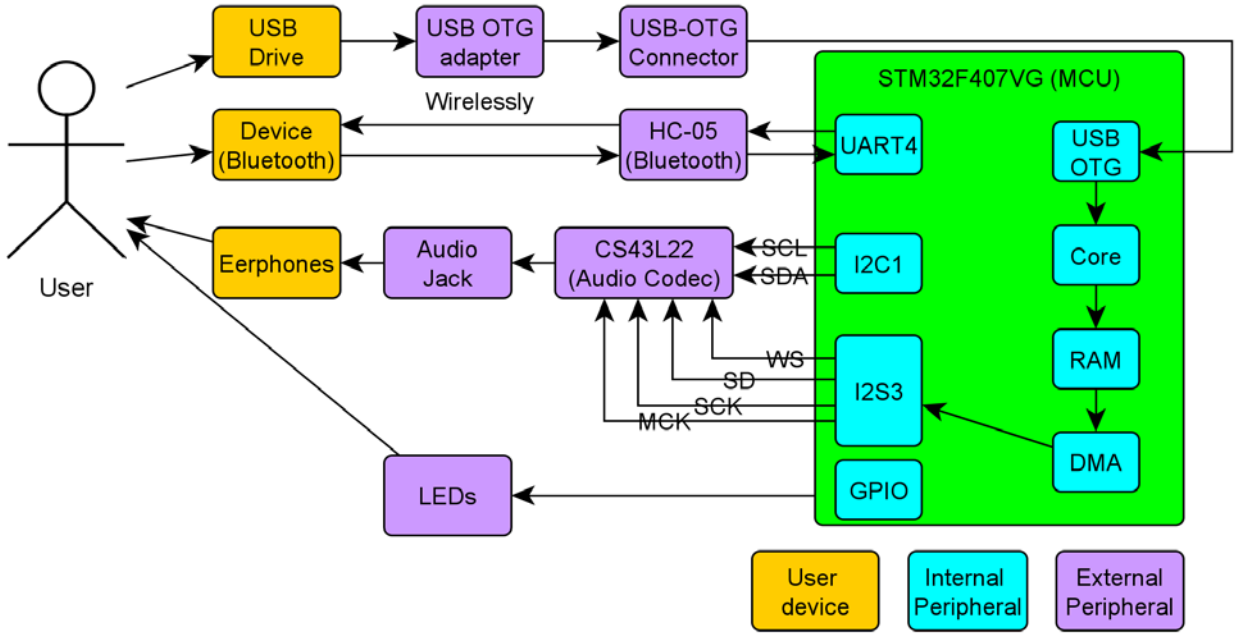
Change volume

- The user should change the volume in units of decibels (dB) from (0-255) by sending 'v [value]'

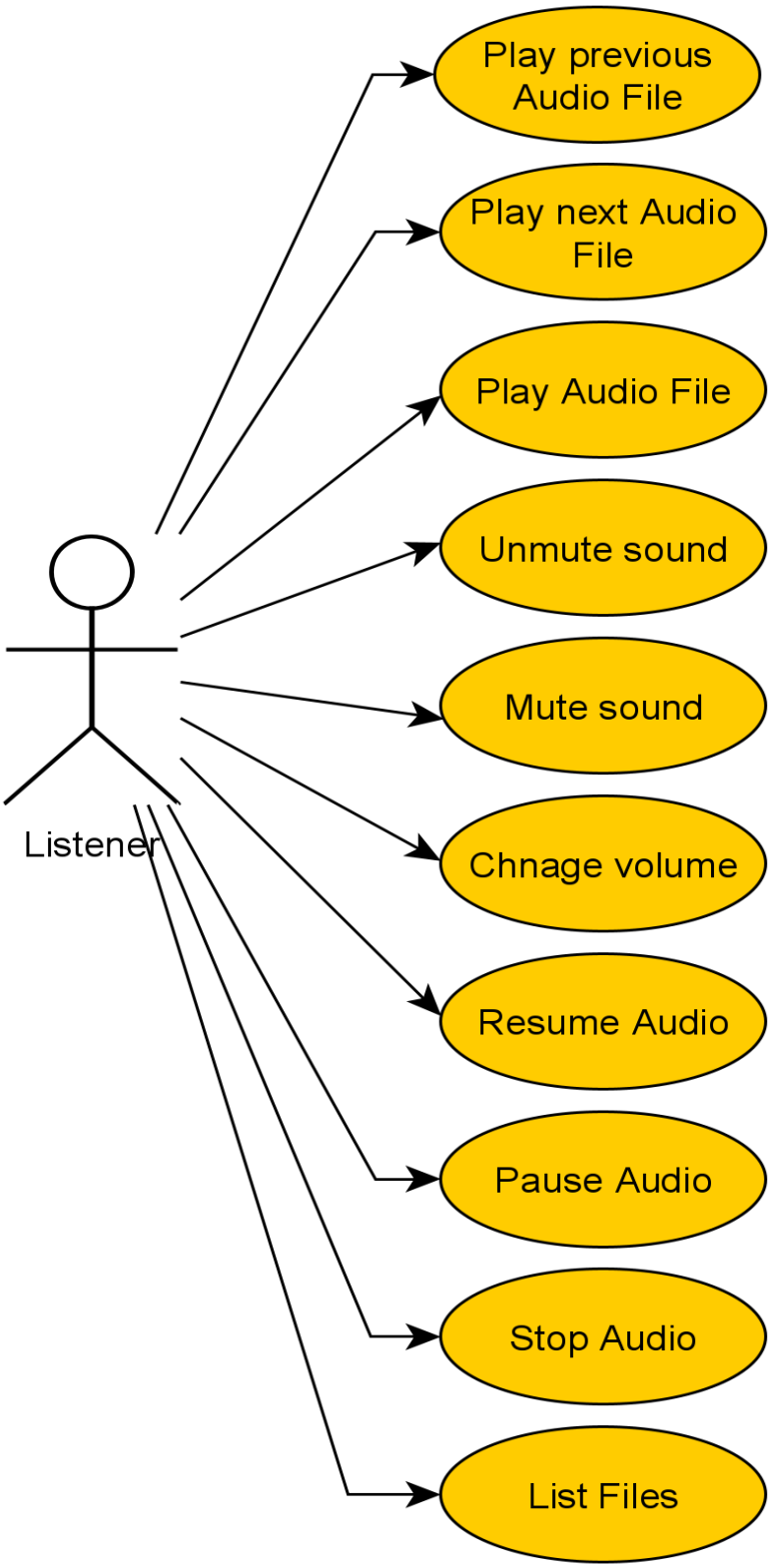
Software layered architecture



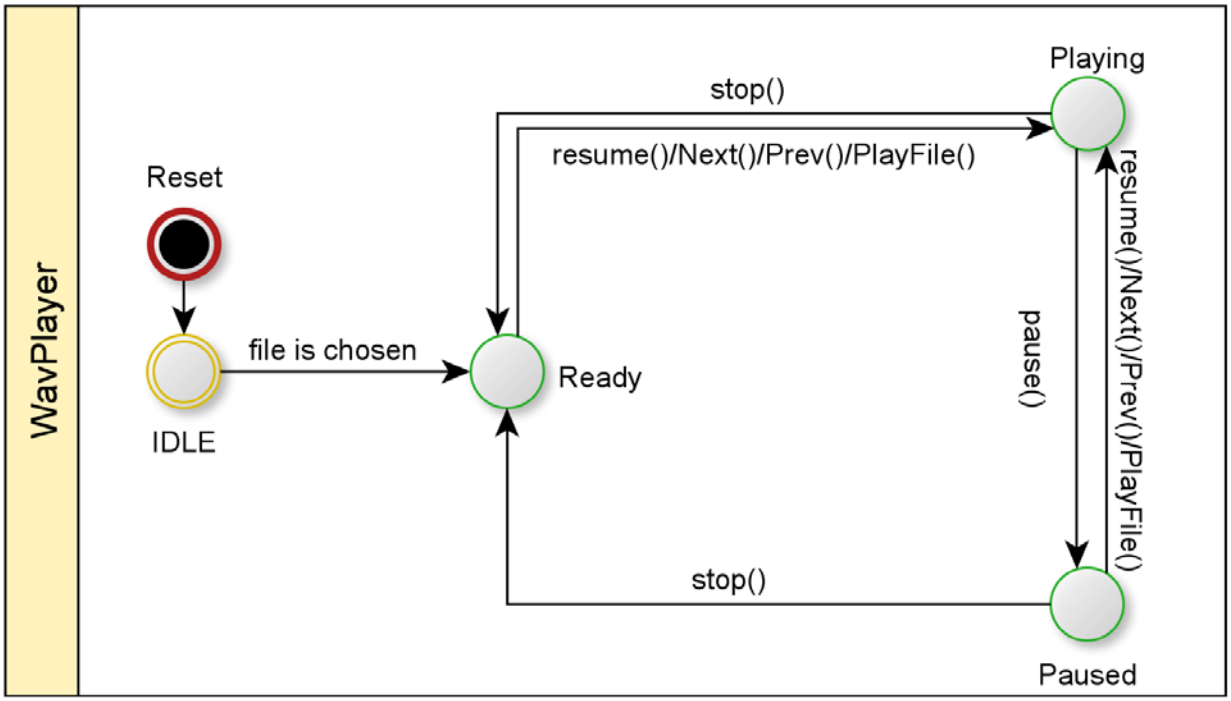
Functional diagram

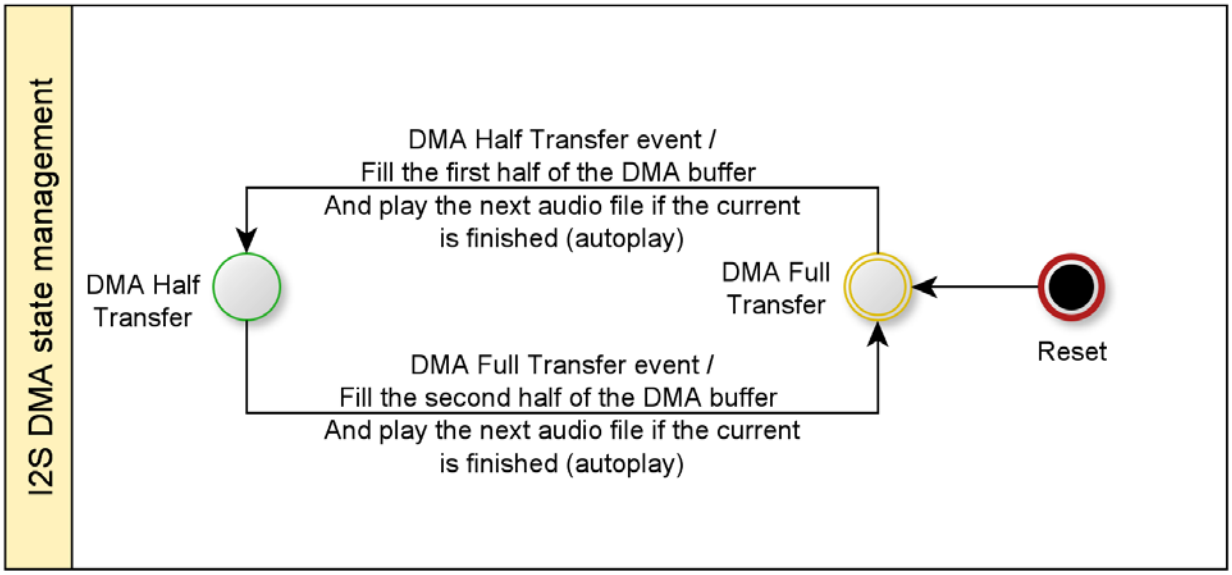


Use cases

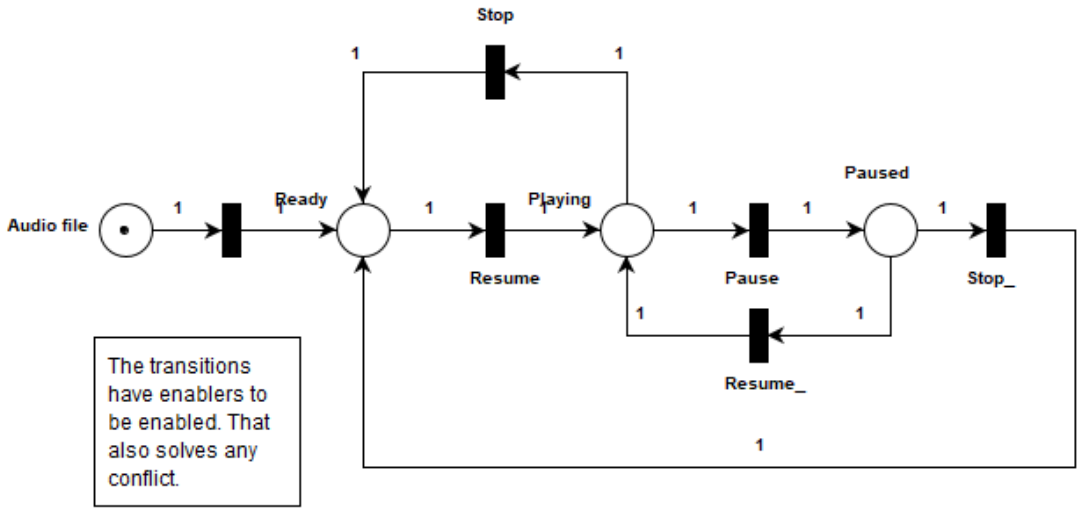

Play previous Audio File
Play next Audio File
Play Audio File
Unmute sound
Mute sound
Chnage volume
Resume Audio
Pause Audio
Stop Audio
List Files
Listener

State Machines





Petri net



The petri net is:

Live: whenever there's just one audio file on the flash drive, the petri net never has a dead end.

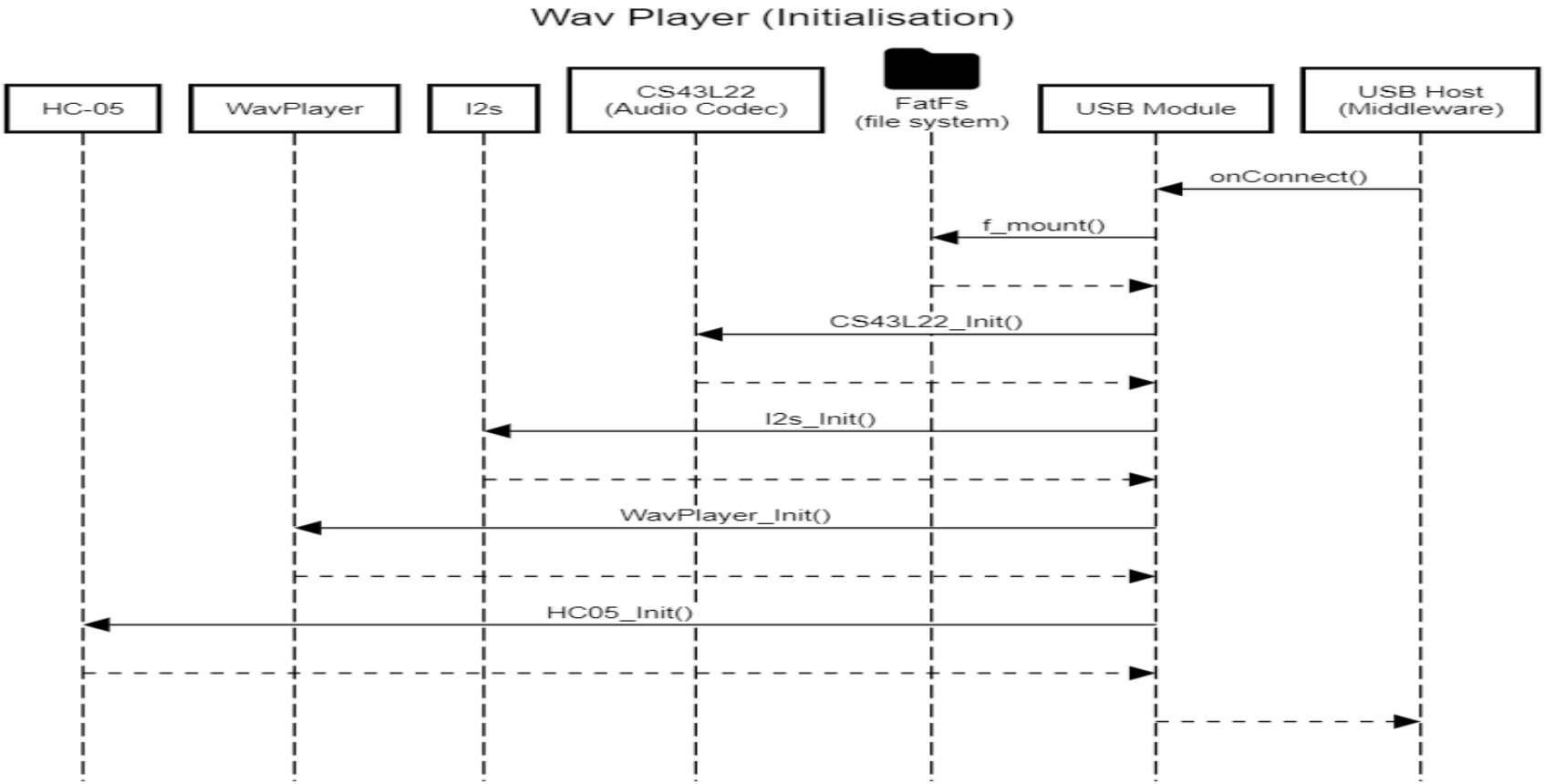
Conservative: in all cases the number of tokens in the net is one.

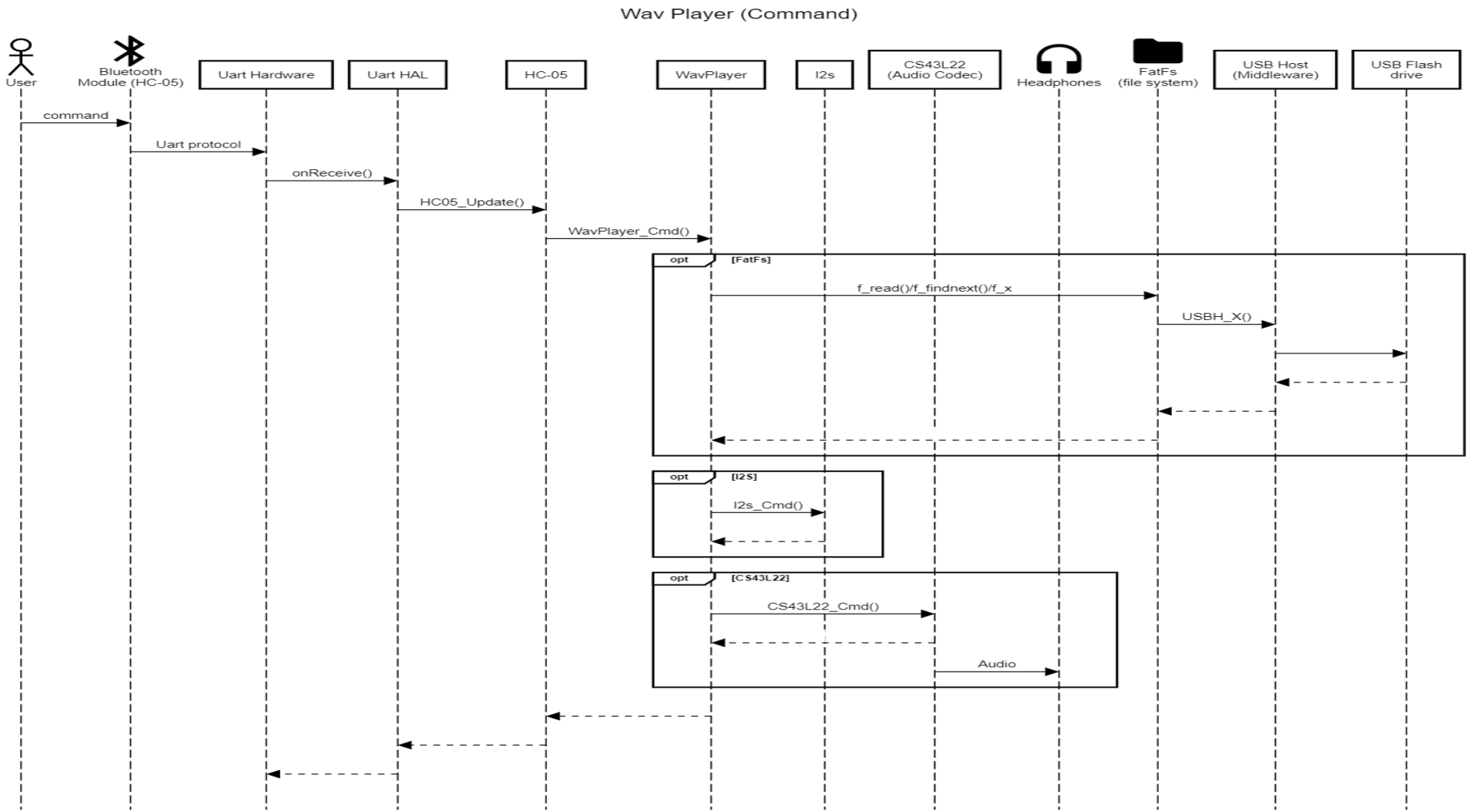
Safe: the number of tokens never exceed one.

Persistent: In case of transitions enablers, there'll be no conflicts resulting in a persistent net.

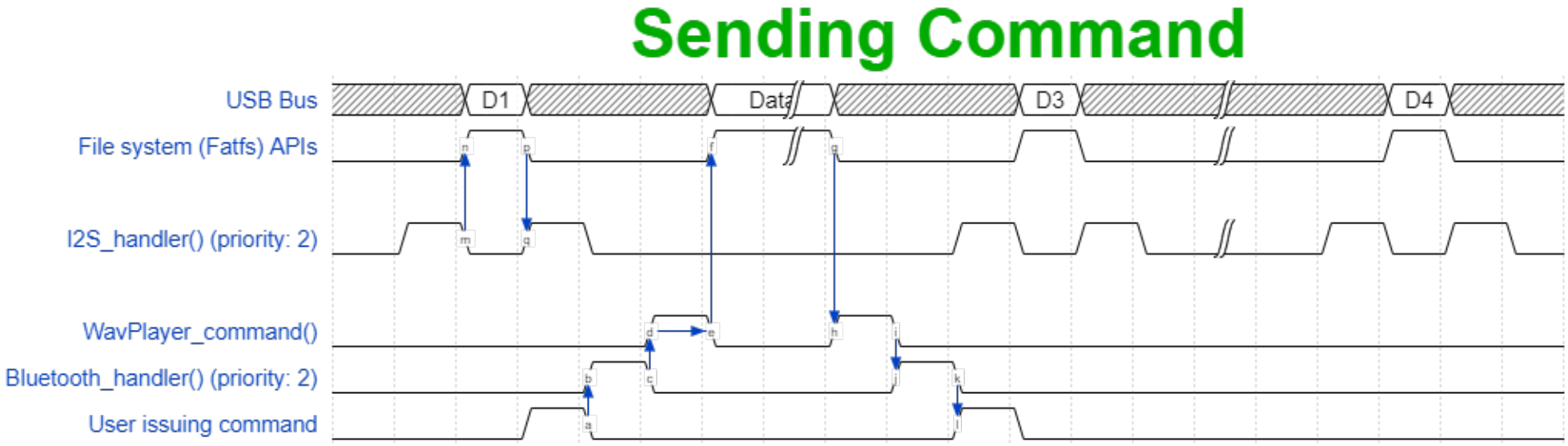
Bounded: the max number of tokens at the net is one.

Sequence Diagram




Wav Player (Command)
User
Bluetooth Module (HC-05)
Uart Hardware
Uart HAL
HC-05
WavPlayer
I2s
CS43L22 (Audio Codec)
Headphones
FatFs (file system)
USB Host (Middleware)
USB Flash drive
command
Uart protocol
onReceive()
HC05_Update()
WavPlayer_Cmd()
opt
[FatFs]
f_read()/f_findnext()/f_x
USBH_X()
opt
[I2S]
I2s_Cmd()
opt
[CS43L22]
CS43L22_Cmd()
Audio

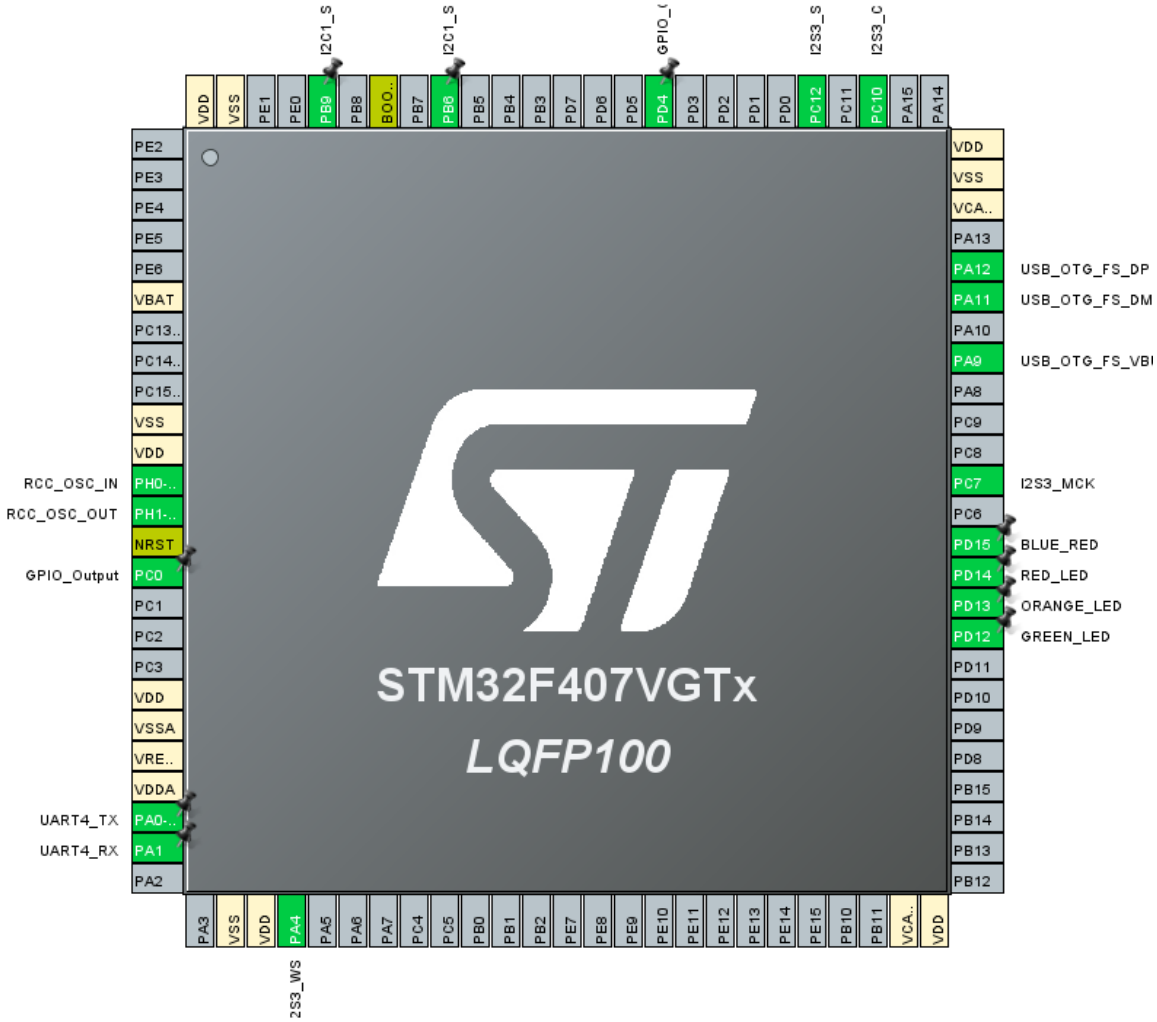
Timing Diagram



The user issues a command, then the blutooth handler (uart handler) excutes and parses the command. After that, according to the command it calls the right WavPlayer function. The WavPlayer command function can deal with the file system(Fatfs) APIs.

The I2S DMA handlers excutes in the background, their period depends solely on the DMA buffer size and the sampling rate.

Schematics and wiring diagram



Pins	Device
PH0(RCC_OSC_IN)/PH1(RCC_OSC_OUT)	External crystal oscillator
PA0(TX)/PA1(RX)	UART4
PA9(VBus)/PA11(D-)/PA12(D+)	USB OTG
PC7(MCK)/PC10(CK)/PC12(SD)/PA4(WS)	I2S
PB6(SCL)/PB9(SDA)	I2C
PD4	Audio Codec Reset Pin
PD12	Green Led

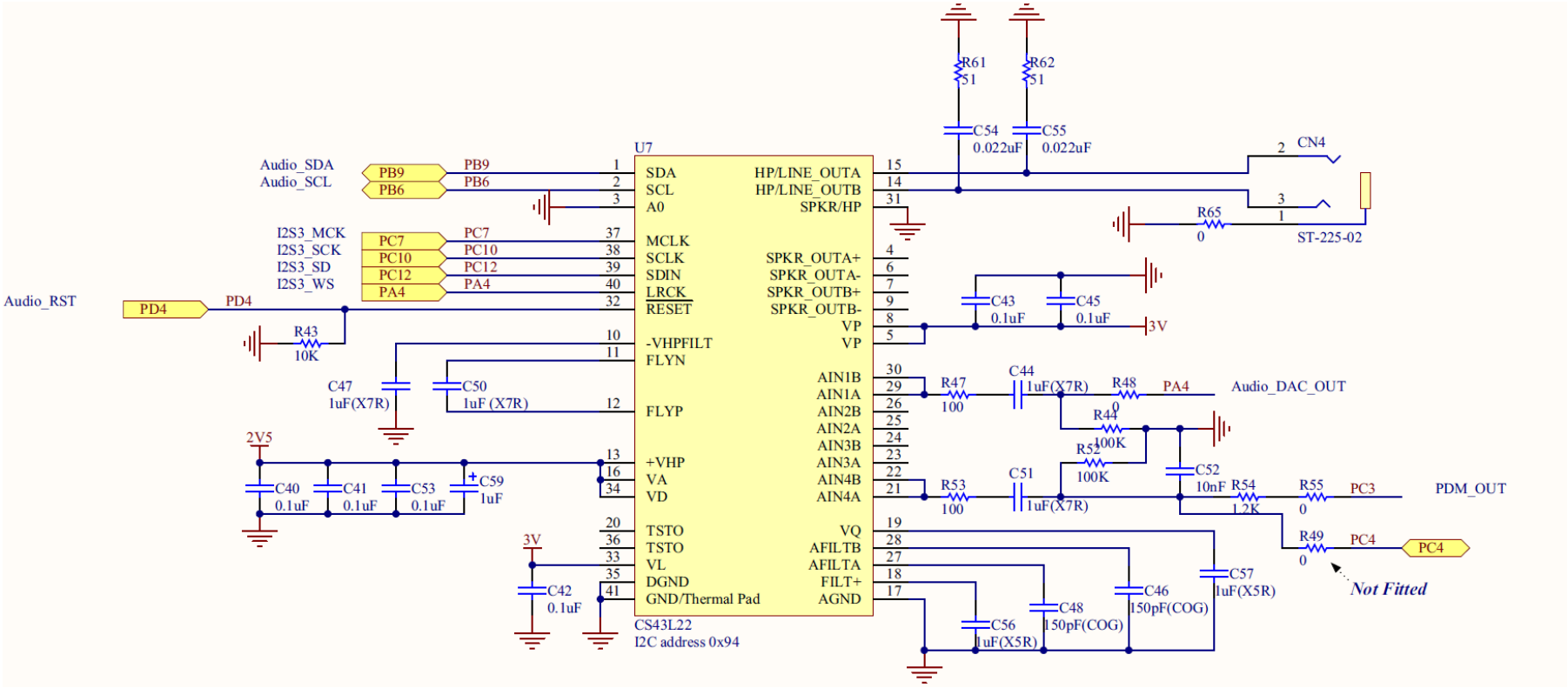


Figure 5 Audio Codec ref:https://www.digikey.at/htmldatasheets/production/1864260/0/0/1/stm32f407g-disc1-user-manual.html

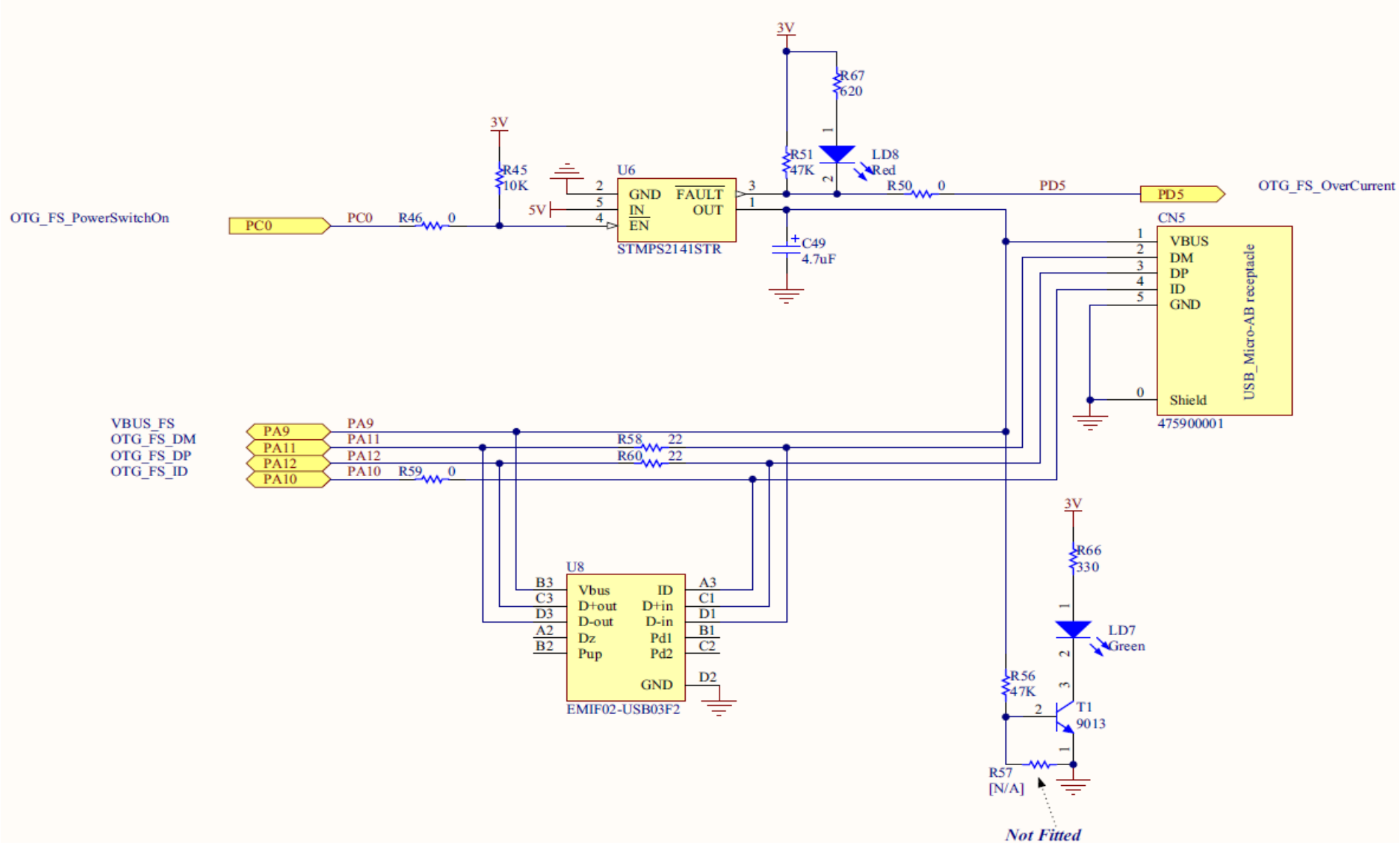


Figure 6 USB OTG ref: https://www.digikey.at/htmldatasheets/production/1864260/0/0/1/stm32f407g-disc1-user-manual.h

Criteria

Clean and Modular Code

The code is segmented into modules/layers. Application, middleware, HAL

Document and BB Deliverable

We did a similar document like the previous year’s ones.

System Functions

pause/play/stop next/previous list/chooseFile Mute/Unmute ChangeVolume

Scalable:

It's scalable as handheld sound players exactly like mp3 players evolution. It can involve other

communication protocols like Wifi.

Persistence:

The log of commands is stored in the user side.

Timing and Synchronization:

The synchronization is handled using designed software architecture that relies on event-based programming,

this avoids any race condition and force synchronization. Or system doesn't involve timing because it's event-based

Demo Quality:

A video showcasing the functionalities is attached.

Low Level:

CS43L22 (Audio codec) driver

No Delay:

Our while loop is empty :) and we don't have any timers used in the application layer.

HW/Other bonus:

We used STM32F407 industrial grade MCU that has "ARM Cortex-m4f" commercial processor. We used non-hobbyists hardware like Arduino, etc.

We used peripherals like USB and I2S.