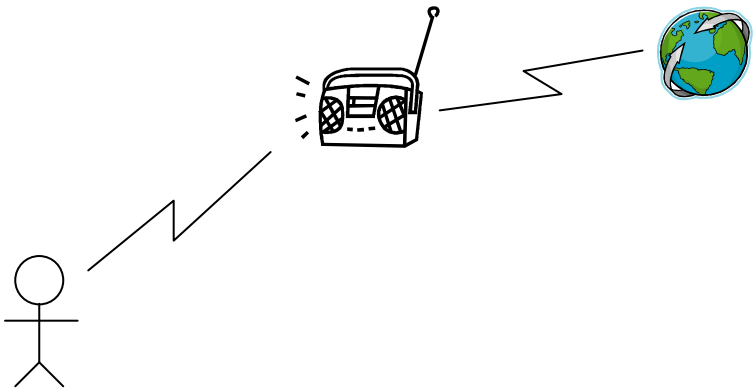
网络播放器需求说明

说明：*对于一个软件系统的设计，最开始的步骤就是需求说明，所以网络播放器的开篇还是从最正规的设计方法开始。另外，所有的设计文档都是从实用角度出发，所以在文档格式上不做过多规范，或许最后会有一个总结格式。*

嵌入式系统是一个完整的系统，包含了软件、硬件两个方面。所以嵌入式系统之软件系统需求说明是从完整的系统需求中截取和软件相关的部分。当然有的时候也会存在，需要的部分功能应该采用软件还是硬件方式来实现。例如，STM32 网络播放器中，mp3 部分的解码部分，到底是采用硬解码还是软解码。在通常的实施过程中，系统的需求说明会包括一部分系统可行性研究工作，以避免设计虽然做出来了，但到最后却不可能实现。

网络播放器的上下文环境

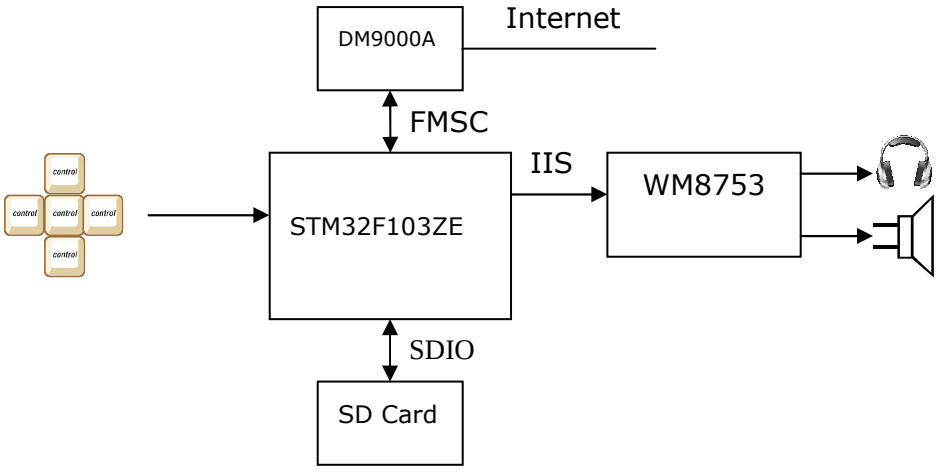


如上图所示的网络播放器操作环境，

1、 网络播放器通过网线连接到路由器，通过路由器连接到互联网中。
2、 网络播放器采用 5V 电源供电。
3、 用户可通过网络播放器收听网络上的音频流。
4、 用户可通过网络播放器播放用户提供的 SD 卡上的数据。
5、 用户可通过网络播放器上的按钮进行网络播放器的操作。
6、 用户可通过网络播放器自带的喇叭或耳机收听网络电台；

系统硬件框图

软件的设计离不开硬件的实现，所以在设计时很有必要知道硬件框图是如何的。



和软件相关的需求说明

1、 网络播放器通过网线连接到路由器，通过路由器连接到互联网中。

网络播放器支持 DHCP 方式从路由器获取 IP v4 地址。

2、

3、用户可通过网络播放器收听网络上的音频流：

用户可播放网络上的 mp3、wma 音频，能够支持 http 音频流和 mms 音频流协议。为了更好地支持网络音频流的播放，网络播放器支持音频流的缓冲播放。

4、用户可通过网络播放器播放用户提供的 SD 卡上的数据。

用户可播放 SD 卡上的 mp3、wav、wma 歌曲。当用户插上 USB 线后，用户通过电脑操作能够操作 SD 卡上的文件（做为 U 盘操作文件）。

5、用户可通过网络播放器上的按钮进行网络播放器的操作。

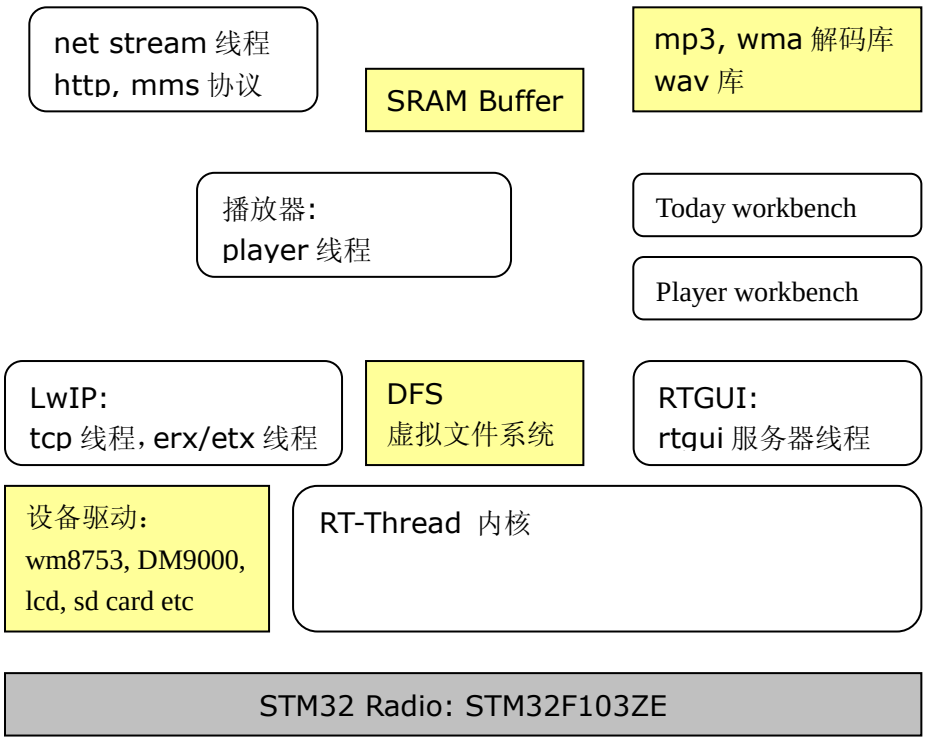
网络播放器能够接收按键进行播放器的操作，进行下一电台，上一电台，播放开始，播放停止，声音增大，声音减小的操作。网络播放器也能够支持 EQ 过滤操作。网络播放器能够通过自带的液晶显示屏给出播放的状态。

6、 用户可通过网络播放器自带的喇叭或耳机收听网络电台；

网络播放器体系结构设计

说明：*体系结构设计是需求说明后，软件系统划分的第一步。它会把一个潘多拉黑盒打开，当然一般是大黑盒套小黑盒的，也即，把一个大的黑盒，按照它的功能特点进行小一些的模块划分。*

网络播放器组件分解如下图所示（一些小型组件，例如键盘未给出）



基础组件部分：

基础组件部分尽量能够采用一些成熟的平台，以避免当问题出现时，不知道到底是上层的问题还是应用的问题。RT-Thread，一套国内主导开发的开源实时系统，它已被数家国内公司所采用，并且还提供相配套的附属组件（而这些也是这个网络播放器需要用到的），选择 RT-Thread 组件有一定风险，也有一些非常有利的地方。

RTOS Kernel：使用国内开源实时操作系统 RT-Thread。RT-Thread 做为一套完善、稳定的实时操作系统，对于这个应用完全能够适用。通过使用 RT-Thread 来作为网络收音机的操作系统，也能够了解、学习 RT-Thread 实时操作系统。

文件系统：使用 RT-Thread 内置的 DFS 虚拟文件系统，目前这个 DFS 虚拟文件系统包装的是 EFSL。采用虚拟文件系统的好就是，底层的具体文件系统实现换掉，但上层的应用可以保持不变，而且 DFS 向上提供的接口是 POSIX 兼容的，这对代码的可移植性更加有好处。

TCP/IP 协议栈：LwIP 轻型 TCP/IP 协议栈，与 RT-Thread 的整合也几乎是无缝的，当前的 RT-Thread/STM32 也已经支持了 ENC28j60 接口的驱动，只需要把它改成 DM9000A 即可。LwIP 协议栈对应用层提供了标准的 socket 接口，并且 LwIP 自身支持多种协议，例如 ICMP, IGMP, DHCP, PPP 等。

图形用户界面：RTGUI，这个在 RT-Thread 那边还未露面 (网络收音机项目包含一个 320x240 的 TFT 屏，5 向导航键，无触摸屏，主要采用键盘操作)，当网络播放器硬件做出来时，RTGUI 也应该发布或能够拿到 Beta 版本。

USB Mass Storage：U 盘，采用 STM32 USB 固件库来实现一个完整的 U 盘，使得能够直接从电脑上操作播放器上的文件。在通过 U 盘操作文件时，系统自动停止播放，关闭打开的文件，不再操作文件，直到断开 USB 连接。

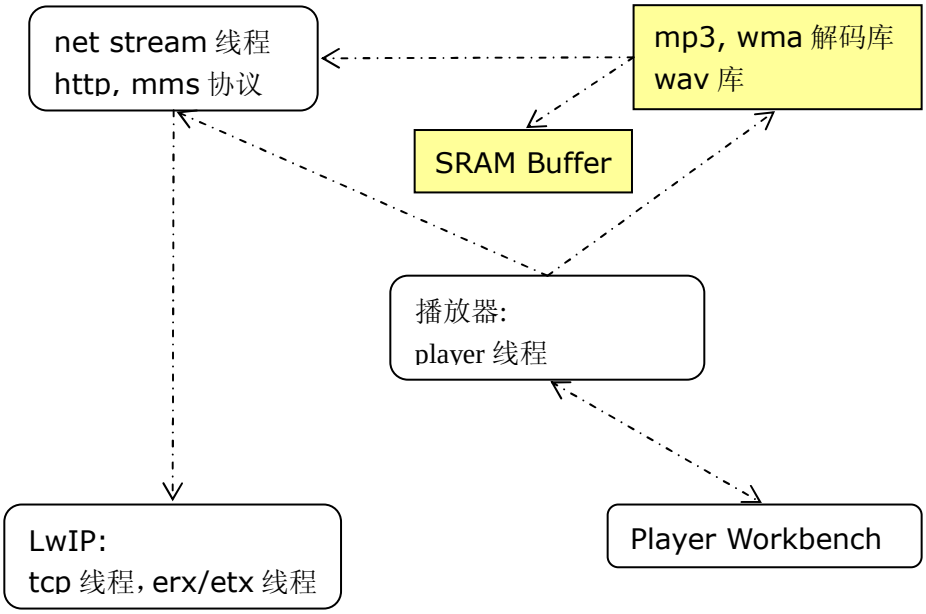
网络播放应用部分：
UI: Information Workbench，用于显示系统的一些信息，例如 USB 连接状态，电池状态，当前日期、时间等。
UI: Player Workbench，用于显示播放器界面。界面不会做得太过复杂，毕竟软解已经耗费了很多资源，想做视频动画效果估计芯片的能力已经跟不上了。需要包括：播放视图；网络电台列表更新视图；收藏列表视图；当前播放列表视图；
UI: Home Workbench，这个实际上是 RTGUI 的应用管理器，用于启动其他的应用，当前活动应用的列表等，一个 workbench 等于一个应用。

player 模块，播放控制的核心模块。用于控制播放何种格式，何种来源。以及一些通常都有的控制操作，下一个、上一个、暂停、停止等。
mp3 codec 模块，也就是 mp3 软解库，ID3TAG 是需要的。
wav 模块，提供 wav 音频格式的信息解析，及提供音频数据。
wma 模块，wma 软解库，必须是针对 ARM 定点优化过的解码库算法。。
netstream 模块，网络音频流模块，它会包含几个子模块：http 流，mms 流等。

为了管理 STM32F103ZE 宝贵的片内 SRAM 资源(片内 SRAM 存取速度要比外扩的 FSMC SRAM 快很多)，还需要一个 sram buffer 管理模块，形成一个类似 cache 的模块，部分的缓冲将直接从它上面分配出来。

设备驱动部分：
codec 驱动，wm8753 驱动，向上层应用提供设备名为 snd 的抽象设备接口。
keyboard 驱动，五向导航键驱动，只用于 RTGUI 中。
LCD 驱动，FSMC 接口的 LCD 驱动，只用于 RTGUI 中。
网卡驱动，DM9000 驱动

各模块依赖关系



此处主要给出了播放器核心模块间的依赖关系，而对于 RTOS 底层服务间的关系没给出（因为基本都会有关联）。

播放器模块直接使用解码库、wav 库模块以获取 PCM 音频数据，同时如果是网络音频流，它也应使用 net stream 模块以获取相应信息；解码库为了读取网络音频流，它依赖于 net stream 模块提供网络数据。同时，对于解码时要求较高的数据存取，它需要使用 sram buffer 以申请片内 SRAM 内存块。播放器与 Player Workbench 应用是相互依赖的，用户通过 UI 执行播放音频操作需要播放器模块来实施，而播放器也需要把播放过程中的一些信息反馈给 Player Workbench 应用。

Player 模块设计

说明：*一个嵌入式软件系统的设计和通常意义的软件系统设计是不相同的。*
在通用系统上，通常考虑的是如何顺序的完成一个任务，所以大多数关注点在于如何完成这个任务，如何把一个复杂的任务进行层层分解，由繁到简的划分模块，实现模块。

在嵌入式系统上，当引入了实时操作系统时，编程模式大多转换成并行任务的方式解决一个问题，即包括了线程、任务的模式。所以在进行嵌入式软件系统设计时，不仅需要考虑通常意义上的对象划分，对象与对象之间的关系，更需要考虑系统的动态行为：线程的划分，线程的状态图跃迁（如果复杂的话），线程间的消息序列等。

在下面的设计中，按照上面的原则，将从静态和动态的角度来分析系统模块的实现。

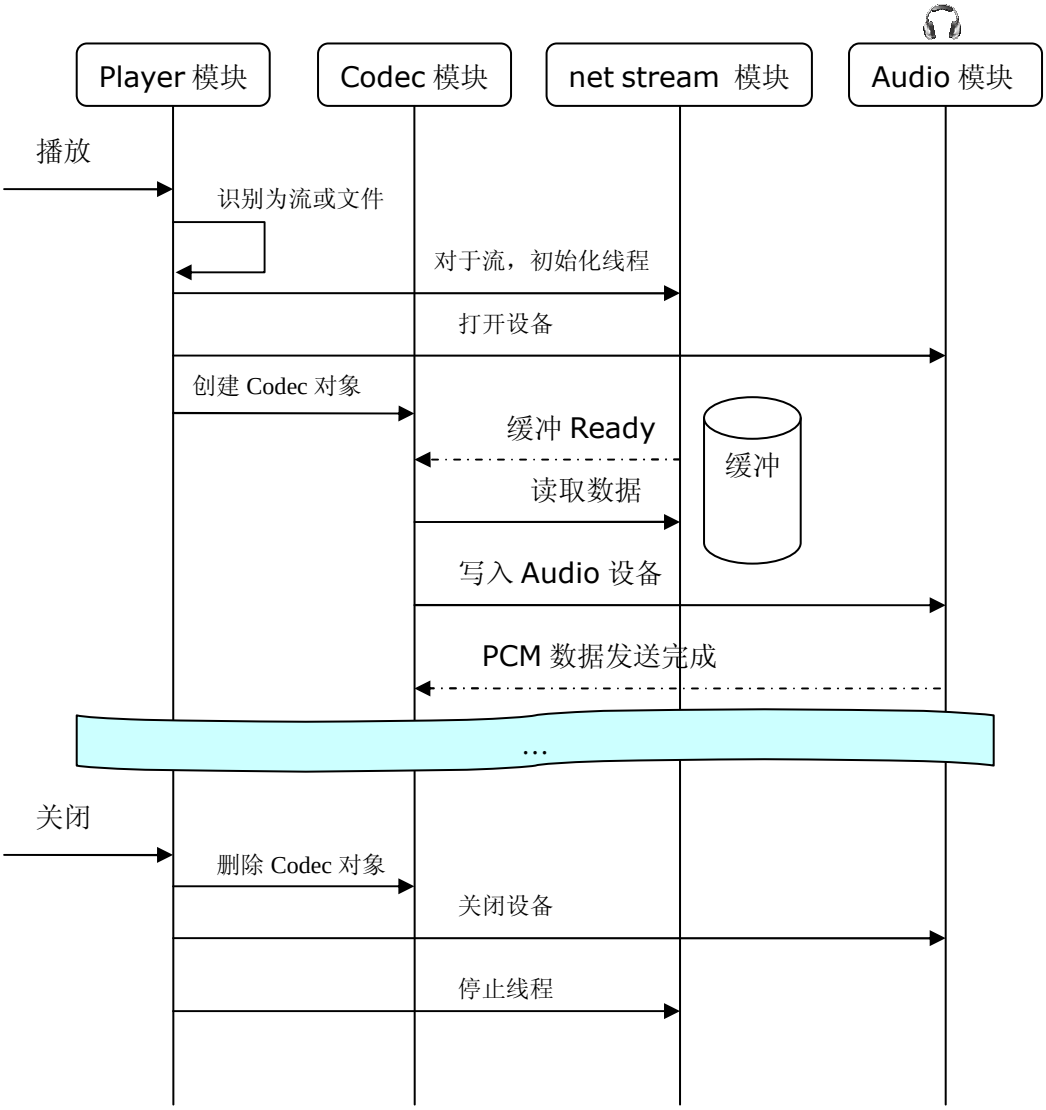
播放模块是网络收音机中的核心模块，它包括了
player 模块：用于控制播放何种格式，何种来源。以及一些通常都有的控制操作，下一个、上一个、暂停、停止等操作。

音频库
包括提供 wav 文件，进行 mp3 音频的软解码，进行 wma 音频文件的软解码。除了解码外，它也需要包括一些文件格式信息的分析。

Net stream 模块设计
流仅指音频流，流的传输协议可能不同，流的内容格式也可能不同。按照需求，当前流传输协议应支持 http 音频流和 mms 音频流。

Audio 输出
声频数据通常指的是 PCM 数据，它由音频库提供，然后交给下层驱动进行播放。PCM 数据分单、双声道，采样率也不完全相同。但一次 PCM 数据播放，其采样率应相同。Audio 输出采用直接控制 snd 抽象设备的方式进行，并不直接操作底层 wm8753 驱动。

播放/关闭操作的模块与模块间的消息序列图



模块线程视图

播放器部分主要由两个线程构成

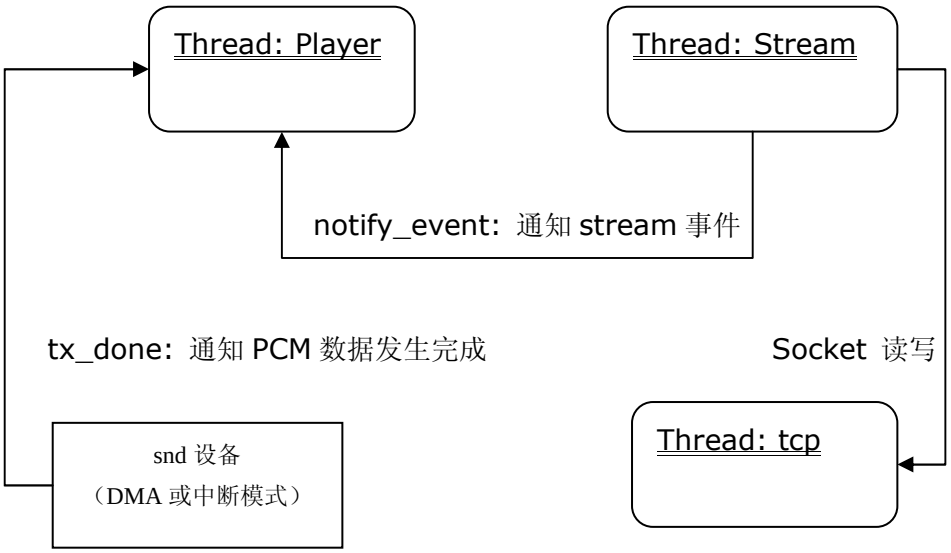
- 网络流线程
- 播放器线程

网络流线程

网络流线程用于从互联网获取音频流，并进行缓冲，当缓冲到一定门限时，向播放器线程提供数据。同时它也需要提供一些事件通知给播放器线程，例如连接打开，关闭等。

播放器线程

播放器线程接收来自用户的命令，进行播放文件或播放音频流操作。播放器在接收到播放命令时，它将先识别出是文件或流，如果是网络音频流将初始化网络流线程进行内容读取。



从并发视图可以看出，线程间并没有共享的数据，因此也就没有线程间的数据区保护问题。并发视图中的几个点也就是线程状态跃迁的几个点：

- 播放线程，当它获取音频缓冲时，如果内存池为空，它将进入阻塞状态。而当音频缓冲数据在 snd 设备发送成功时，snd 设备将回调 tx_done 函数以释放音频缓冲资源。此时播放线程将由于能够申请到音频缓冲而变为就绪状态。
- 流线程，当它进行 socket 读写时，由于网络数据未达到而进入 Block 状态。当然 LwIP::tcp 线程接收到网络数据时，将唤醒流线程进行数据读取操作。
- 当流线程缓冲网络数据到达一定门限时，它将通过 notify_event 函数唤醒播放线程以读取网络流数据。

模块静态视图

播放器模块的接口

从消息序列图可以看出，播放器模块需要提供：

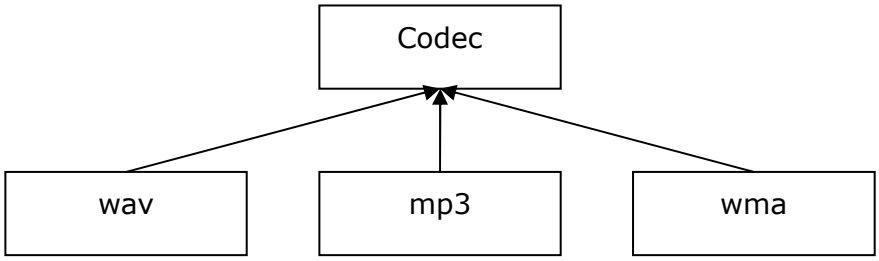
- play，播放
- stop，停止

等接口。但这两个接口还没法满足用户需求中定义的功能（设计文档中应包含用户需求在模块设计中的映射，以避免丢失用户需求），还应包括：

- pause，暂停
- get_info，获取播放信息（时间进度、ID3Tag、采样率、立体声等）

Codec 模块的接口

Codec 模块会包含数种音频格式的支持：

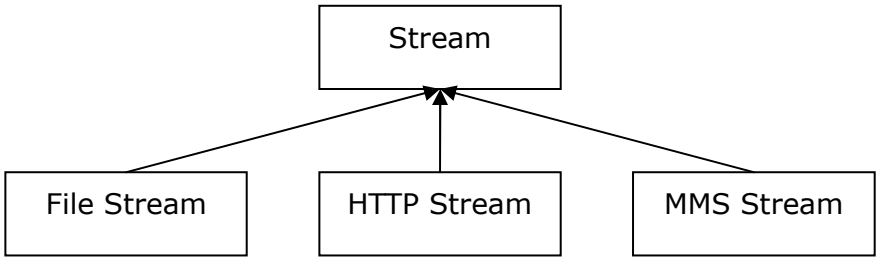


- wav 格式，未压缩的 PCM 数据，但有文件头信息；
- mp3 格式，压缩格式，包含 ID3Tag 信息；
- wma 格式，压缩格式

Codec 模块需要提供的接口包括：
- get_type，对于一段数据流，给出可能的音频格式（WAV，MP3 或 WMA）；
- create/delete，创建和删除接口；
- run，进行数据流的解析

Stream 模块的接口
Stream 模块的接口包括：
- open，根据给定的 URL，尝试获取流数据；
- read，读取流数据；
- close，关闭连接

根据文件操作的情况，也完全可以把文件作为一种特殊流来处理，因此将包含三种流的支持：



按照消息序列图的情况，Stream 模块也应该提供回调函数以通知 Player 模块，流数据读取的情况，因此需要提供如下回调函数接口：
- notify_event，通知 Player 模块一些事件产生，这些事件包括：
  * open successful
  * connection closed
  * ready