

**软件频谱分析系统**

**详细设计说明书**

|  |  |
| --- | --- |
| **版本** | V1.0 |
| **状态** | 正式 |
| **生效日期** | 2019-09-18 |
| **作者** | 黄龙澎 |

[1. 引言 4](#_Toc1091049829)

[1.1. 编写目的 4](#_Toc1750482477)

[1.2. 背景 4](#_Toc600600945)

[1.3. 参考资料 4](#_Toc740945191)

[2. 设计概述 5](#_Toc1595156614)

[2.1. 任务和目标 5](#_Toc1671820519)

[2.1.1. 需求概述 5](#_Toc747015498)

[2.1.2. 运行环境概述 5](#_Toc676988952)

[2.1.3. 条件与限制 5](#_Toc894448646)

[2.1.4. 详细设计方法和工具 6](#_Toc1141628816)

[3. 总体方案确认 7](#_Toc1677570366)

[3.1. 系统总体结构确认 7](#_Toc1569330608)

[3.1.1. 底层程序流程图： 8](#_Toc1123292887)

[3.1.2. 上层程序流程图： 8](#_Toc126384335)

[4. 系统详细设计 9](#_Toc1941142042)

[4.1. 系统结构设计及子系统划分 9](#_Toc372950154)

[4.1.1. 底层结构设计 9](#_Toc980174937)

[4.1.2. 上层结构设计 9](#_Toc547658446)

[4.2. 系统功能模块详细设计 10](#_Toc259825814)

[4.2.1. 底层功能模块设计 10](#_Toc1895995308)

[4.2.2. 上层功能模块设计 15](#_Toc1802133764)

[4.3. 系统界面详细设计 16](#_Toc576180647)

[4.3.1. 用户界面设计 16](#_Toc1354636029)

[5. 非功能性设计 17](#_Toc1706236317)

[5.1. 可维护性 17](#_Toc637477532)

[5.2. 易用性 17](#_Toc1042850802)

[5.3. 可移植性 17](#_Toc1227071492)

[6. 环境配置 18](#_Toc42370725)

# 引言

## 编写目的

在本阶段中，确定应该如何具体地实现所要求的系统功能，从而在编码阶段可以把这个描述直接翻译成用具体程序语言书写的程序。主要工作有：根据《软件扫频功能需求分析文档》中所描述的数据、功能、运行、性能需求，详细设计软件系统的结构设计、模块设计。

解决了：

1. 输入：使用射频设备对空中射频信号采样，接收处理设备输出的基带信号
2. 输出：对基带数据进行FFT频谱分析，得到可视化FFT频谱图

## 背景

软件频谱分析系统主要由两部分组成：

1. 射频设备的初始化及处理基带信号数据的底层程序
2. 在网页上渲染及显示频谱分析结果图的上层程序

## 参考资料

1. USRP硬件驱动程序参考手册 。参考链接：<http://files.ettus.com/manual/>
2. 嘉兆科技官方网站。参考链接：<http://www.ettus.com.cn/>
3. Gqrx开源软件：Gqrx是一个基于gnuradio和qt架构，开发的一个开源的SDR接收机的应用。参考链接：<http://gqrx.dk/>

# 设计概述

## 任务和目标

### 需求概述

在自研基站启动前，对在当前区域将使用的频点进行扫频测试，提供频谱分析结果，供技术人员分析选择最合适的频点。要求系统能有效、可靠地完成扫频，并要求操作界面简要明了、易于操作、程序易于维护。

### 运行环境概述

* 硬件平台：
  + Intel(R) Core(TM) i7-8850H CPU @ 2.60GHz
* 软件平台：
  + 射频设备驱动版本：UHD\_3.14.0.0
  + 编译器版本：cmake version 3.5.1
* 操作系统：
  + Ubuntu 16.04 server版本、4.15.0-54.lowlatency内核

### 条件与限制

1. 条件：必须支持自研基站的软硬件平台及操作系统
2. 限制：可能由于基站使用的射频天线工作频率、双工器、功放等硬件限制，造成扫描天线工作频率外的频段效果不佳

### 详细设计方法和工具

1. 参考Gqrx开源软件程序代码的初始化射频设备过程，处理基带信号数据过程，FFT变换过程等
2. 将底层程序建立到qt工程中，利用qtgui显示出FFT频谱图，与Gqrx显示的FFT频谱图作对比，从而判断底层程序是否可行
3. 读取本地保存的FFT数据，将FFT频谱图输出到页面上显示
4. 前后端设计及页面设计使用的工具：IntelliJ IDEA集成编译器

# 总体方案确认

扫频的主要作用是：对当前环境下某频段的空中射频信号采样，提供频谱分析结果(频谱图)，供技术人员分析进而来选择最合适基站使用的频点。频谱图的显示并不需要对每一次的扫描结果实时显示，只需显示采样时间内(采样次数内)的数据峰值。

## 系统总体结构确认

系统总体结构可以划分为底层程序、上层程序。

底层程序用来支撑上层程序，最终会被上层程序调用，主要任务是：调用本地uhd驱动接口函数，实现对usrp射频设备的初始化，使设备能够正常采集空中射频信号，对信号做FFT频谱分析并保存峰值。底层程序开发用c++语言实现。

上层程序主要任务是：提供操作界面、调用底层程序、读取FFT数据并对数据进行渲染。上层程序开发java、h5、js语言实现。

本地文件(.dat)

调用

写入FFT数据

读取

上层程序

底层程序

用户操作界面、显示界面

### 底层程序流程图：

接收机初始化

基带信号数据FFT变换

开始

外部参数是否正确

结束

设置射频设备地址

设置时钟源及速率

设置采样速率及接收带宽

设置通道、接收中心频率、接收增益、接收天线

设置数据流格式

循环采样并作FFT变换、保存峰值

将峰值保存到本地(.dat)

否

是

底层流程图介绍：

程序开始前先判断外部传入的频率参数是否在规定范围内，不符合要求则结束程序，若符合要求，则程序继续运行，先初始化接收机，然后对采样的基带信号进行频谱分析，最终将峰值保存到本地。

### 上层程序流程图：

扫频功能页面

读取起始频率及终止频率

判断频率值是否符合要求

点击开始按键

否，重新输入

计算出中心频率并传给后端

调用本地底层扫频功能程序，并传入中心频率值

读取本地保存的.dat文件，并将数据传给前端

将数据以折线图显示

页面呈现

上层程序流程介绍：

扫频界面上有起始频率、终止频率输入栏，当按键被按下时，前端程序会先判断输入的频率值是否符合要求，不符合要求则提示错误信息，符合要求则计算出中心频率传给后端，后端将调用底层程序，若调用成功，则将fft数据返回到前端，否则不返回。

# 系统详细设计

## 系统结构设计及子系统划分

### 底层结构设计

主函数

接收机初始化函数

数据处理函数

设备地址

时钟源

时钟速率

采样速率

通道、天线

中心频率

接收带宽

接收增益

接收流格式

采样、FFT变换

数据文件保存

底层程序结构主要包括：

接收机的初始化函数、基带信号数据处理函数；

其中接收机初始化包括：

射频设备地址初始化、设备时钟源、时钟速率初始化、采样速率初始化、通道、天线、中心频率、接收带宽、接收增益的初始化；

基带信号处理包括：

设置接收流格式，循环采样并对数据做FFT频谱分析，最终保存峰值数据。

### 上层结构设计

ajax接口

扫频功能界面

起始频率输入框、终止频率输入框

启动扫频按键（判断输入值是否符合要求、计算出中心频率依次传给后端

）

频谱图显示画布

调用底层程序，传入中心频率值

读取FFT数值，将数组传给前端

后台

前端界面：

起始频率、终止频率输入框，启动按键，画布；

后端：

接收前端传来的中心频率参数，调用底层扫频程序，最后将FFT数据传给前端；

前后端数据交互接口：Ajax接口，数据格式：Json数据格式。

## 系统功能模块详细设计

### 底层功能模块设计

主要库函数介绍：

#include <uhd/usrp/multi\_usrp.hpp> //usrp驱动接口的c++头文件

#include <fftw3.h> //FFT变换使用的c头文件

1. mian函数接收外部传入的中心频率参数

* 定义接收中心频率变量：double frq;
* 判断：若频率范围在70MHz～6GHz，则程序继续运行，否则程序结束。

1. 接收机初始化函数

* 定义接收机类：

class receiver

{

public:

receiver(void);

~receiver();

Void init(receiver &init, double cent\_frq); /\* 初始化接收机函数 \*/

Void set\_clock\_source(std::string ref); /\* 设置时钟源 \*/

Void set\_rx\_subdev\_spec(std::string subdev); /\* 设置接收机通道 \*/

Void set\_master\_clock\_rate(double clock\_rate); /\* 设置主时钟速率 \*/

Void set\_antenna(const std::string &antenna); /\* 设置天线口 \*/

Void set\_input\_rate(double rate); /\* 设置采样速率 \*/

Void set\_rx\_bandwidth(double bw); /\* 设置接收带宽 \*/

Void set\_rf\_freq(double freq\_hz); /\* 设置接收中心频率 \*/

Void set\_rx\_gain(double value); /\* 设置接收增益 \*/

uhd::usrp::multi\_usrp::sptr usrp; /\* 定义设备源\*/

};

* 1. 设置设备地址函数
* 定义地址变量： string addr\_args=””; //地址可为空
* 调用函数：usrp = uhd::usrp::multi\_usrp::make(addr\_args);
  1. 设置时钟源函数
* 定义时钟源变量： string ref=”internal”; //使用内部时钟源
* 调用函数：usrp->set\_clock\_source(ref);
  1. 设置时钟速率函数
* 定义时钟速率变量： double clock\_rate=40e6; //时钟速率40MHz
* 调用函数：usrp->set\_master\_clock\_rate(clock\_rate);
  1. 设置采样速率函数
* 宏定义采样速率： #define SAMPLE\_RATE 20e6 //采样速率20MHz
* 调用函数：usrp->set\_rx\_rate(SAMPLE\_RATE);
  1. 设置接收通道函数
* 定义接收通道变量： string subdev("A:B"); //使用B通道
* 调用函数：usrp->set\_rx\_subdev\_spec(subdev);
  1. 设置接收天线函数
* 定义接收天线变量： string antenna=”RX2”; //使用RX2天线
* 调用函数：usrp->set\_rx\_antenna(antenna);
  1. 设置接收带宽函数
* 宏定义接收带宽： #define RX\_BW 20e6 //接收带宽 20MHz
* 调用函数：usrp->set\_rx\_bandwidth(RX\_BW);
  1. 设置接收中心频率函数
* 定义中心频率变量：double cent\_frq; //中心频率值由外部传入
* 调用函数：usrp->set\_rx\_freq(cent\_frq);
  1. 设置接收增益函数
* 宏定义接收增益：#define RX\_GAIN 40 //40dB的接收增益
* 调用函数：usrp->set\_rx\_gain(RX\_GAIN);

1. 数据处理

* 定义处理类：

class rx\_fft

{

public:

rx\_fft(void);

~rx\_fft();

float\* get\_fft\_data(uhd::usrp::multi\_usrp::sptr &usrp);

private:

fftw\_complex \*in; /\* 需要做fft变换的数据 \*/

fftw\_complex \*out; /\* fft变换后的数据 \*/

fftw\_plan p; /\* fft变换方法 \*/

std::complex<float> iq\_data\_buff[SAMPLE\_PER\_BUFF];/\*保存基带数据 \*/

float d\_realFftData[SAMPLE\_PER\_BUFF]; /\* 保存fft数据 \*/

float fftPeakHoldBuf[SAMPLE\_PER\_BUFF] ; /\* 保存fft峰值数据 \*/

double pt\_real,pt\_imag; /\* 基带信号的实部与虚部 \*/

float pwr; /\* 功率值 \*/

float pwr\_scale;

uhd::rx\_metadata\_t md;

};

* 1. 创建接收流
* 定义string cpu\_format = "fc32"; //fc32相当于complex<float>类型
* 定义string wire\_format = "sc16"; //sc16相当于complex<int16\_t>类型
* 设置接收流数据格式：
  + uhd::stream\_args\_t stream\_args(cpu\_format, wire\_format);
  + uhd::rx\_streamer::sptr rx\_stream = usrp->get\_rx\_stream(stream\_args);
  1. 设置接收流命令：
* 设置接收模式：
  + - uhd::stream\_cmd\_t stream\_cmd(uhd::stream\_cmd\_t::STREAM\_MODE\_START\_CONTINUOUS);
* 设置采样点数：stream\_cmd.num\_samps = SAMPLE\_PER\_BUFF; #define SAMPLE\_PER\_BUFF 4096
* 设置现在开始接收：
  + - stream\_cmd.stream\_now = true;
    - stream\_cmd.time\_spec = usrp->get\_time\_now();
* 配置rx\_stream参数：rx\_stream->issue\_stream\_cmd(stream\_cmd);
* 接收基带数据：
  + - 分配空间：memset(iq\_data\_buff, 0, SAMPLE\_PER\_BUFF \* sizeof(std::complex<float>));
    - 接收一次：rx\_stream->recv(iq\_data\_buff, SAMPLE\_PER\_BUFF, md, 3.0, false);
  1. 基带数据的FFT处理
* 给in、out分配空间：
  + in = (fftw\_complex \*)fftw\_malloc(sizeof(fftw\_complex) \* SAMPLE\_PER\_BUFF);
  + out = (fftw\_complex \*)fftw\_malloc(sizeof(fftw\_complex) \* SAMPLE\_PER\_BUFF);
* 提取基带数据实部、虚部：
  + for (uint32\_t i = 0; i < SAMPLE\_PER\_BUFF; i++){

in[i][0] = iq\_data\_buff[i].real();

in[i][1] = iq\_data\_buff[i].imag();

}

* 设置FFT变换方法：
  + p = fftw\_plan\_dft\_1d(SAMPLE\_PER\_BUFF,in,out,FFTW\_FORWARD,FFTW\_ESTIMATE);
* FFT变换：
  + fftw\_execute(p);
* 计算功率值、做fft shift

for (uint32\_t i = 0; i < SAMPLE\_PER\_BUFF; i++)

{

/\* fft shift \*/

if (i < SAMPLE\_PER\_BUFF/2) {

pt\_real = out[SAMPLE\_PER\_BUFF/2+i][0];

pt\_imag = out[SAMPLE\_PER\_BUFF/2+i][1];

}else{

pt\_real = out[i-SAMPLE\_PER\_BUFF/2][0];

pt\_imag = out[i-SAMPLE\_PER\_BUFF/2][1];

}

/\* 计算得到功率值 \*/

pwr = pwr\_scale \* (pt\_real \* pt\_real + pt\_imag \* pt\_imag);

d\_realFftData[i] = 10.0 \* log10f(pwr + 1.0e-20);

/\* 存储峰值 \*/

}

* 结束采样：
  + stream\_cmd.stream\_mode = uhd::stream\_cmd\_t::STREAM\_MODE\_STOP\_CONTINUOUS;
  + rx\_stream->issue\_stream\_cmd(stream\_cmd);
  1. 存储每一次采样的每个频率点的峰值数据，待采样结束后，将峰值数据保存到本地。

### 上层功能模块设计

1. 起始频率输入栏

* 类型type=”text”，id=”Start\_Frq”不允许为空，输入字段提示信息800，数据单位MHz

1. 终止频率输入栏

* 类型type=”text”，id=”Stop\_Frq”,不允许为空，输入字段提示信息900，数据单位MHz

1. 开始按键
   1. 类型type=”button”，id=”dsp\_button”，value=”开始”，onclick=”info()”
   2. 函数function Info()功能：

* 判断起始频率、终止频率不能为空
* 要求终止频率必须大于起始频率
* 终止频率与起始频率的差值必须在20～100MHz，且必须10MHz的整数倍
* 输入的频率范围在70MHz～6GHz
* 根据起始频率和终止频率计算出中心频率，并通过post提交到后端
* 按键在采样过程中失能，待采样结束后重新使能。目的是防止在扫频过程中，频繁操作按键

1. Canvas画布

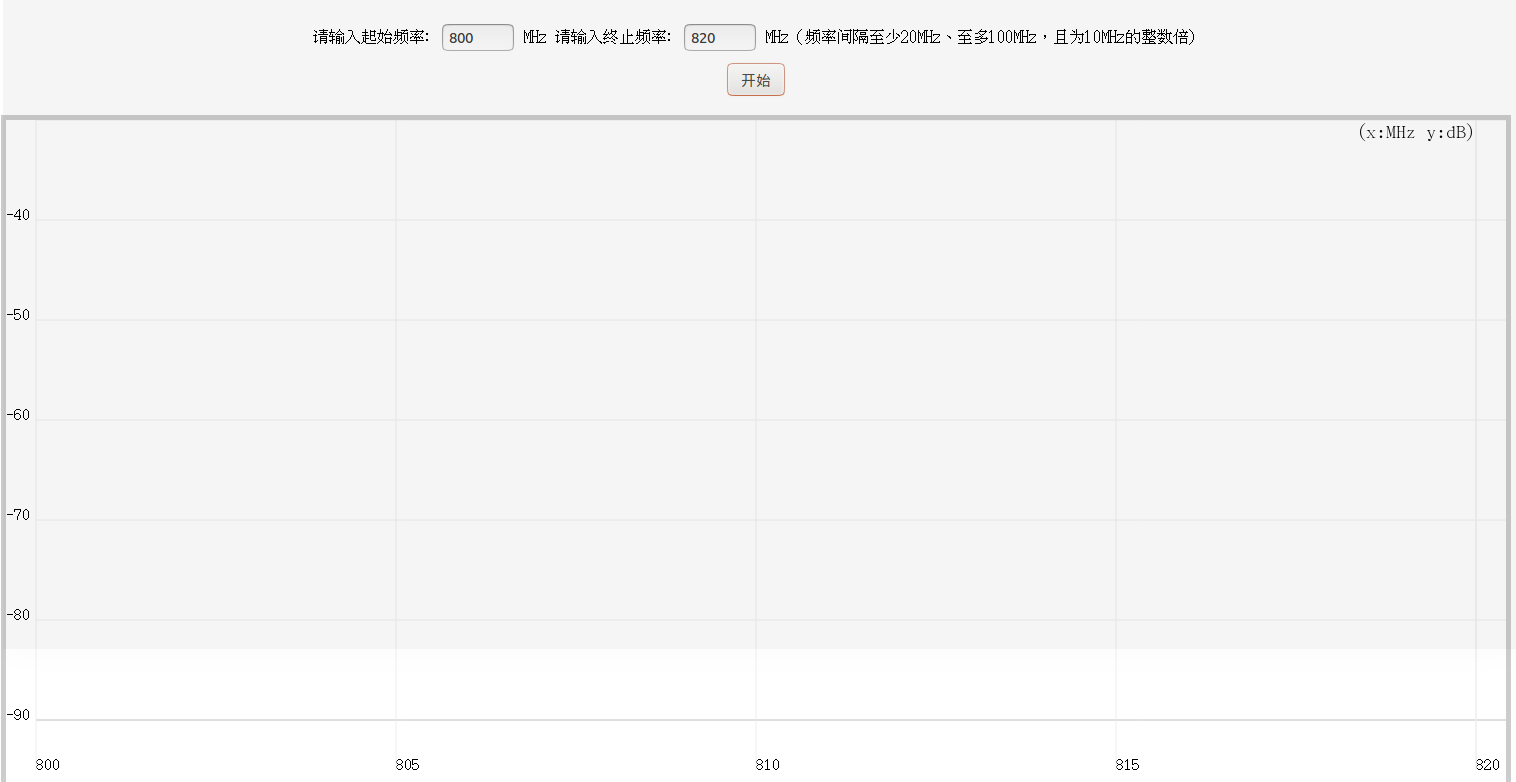
* 画布铺满整个页面
* 横坐标可以根据频率范围调整数值，最小刻度间隔5MHz
* 纵坐标固定，最大-30dB，最小-90dB（具体根据fft数据值调整）
* 画布右上角显示坐标单位

## 系统界面详细设计

### 用户界面设计

扫频界面简易明了，频谱图的显示画布占主体，两个频率输入栏和按键设计在同一个表单中，用户只需输入起始频率，与终止频率，点击开始按键，即可启动扫频功能。

界面效果大致如下图：



# 非功能性设计

## 可维护性

底层程序包结构清晰，程序包包含main.cpp、dsp包、app包、readme.txt，dsp包存放fft处理代码，app包存放接收机配置代码，readme.txt文档介绍了程序文件包结构，介绍编译方法，及调用方式。

## 易用性

操作页面功能简单，限制用户输入频率数据范围，设置按键状态，防止用户频繁操作，减少出错几率。

## 可移植性

底层程序配有自动安装部署的shell脚本，执行脚本即可自动安装部署底层扫频程序；

上层程序导出的war包，可以部署在任意tomcat服务器中

## 程序目录结构

### 底层程序目录

Src ##底层程序包

├── app ## app包，存放主要功能代码

│   ├── CMakeLists.txt

│   ├── receiver.cpp ## 接收机相关代码

│   └── receiver.h ## 接收机相关头文件

├── CMakeLists.txt

├── dsp ## dsp包，存放数据处理相关代码

│   ├── CMakeLists.txt

│   ├── rx\_fft.cpp ## fft处理代码

│   └── rx\_fft.h ##fft处理相关头文件

└── main.cpp ## 主函数

### 上层程序目录

Src ##后端程序文件包

   └── main

   └── Servlet.java ## 后端servlet代码

Web ## 前端程序文件包

├── index.html ## 扫频页面代码

├── js ## 存放js文件

│   ├── jquery-1.11.0.min.js ## jquery头文件

│   └── lineChart.js ## js绘图脚本

├── META-INF

│   └── MANIFEST.MF

└── WEB-INF

└── web.xml ## web配置文件

# 环境配置

* 环境要求：

底层编译环境：cmake version 3.5.1

上层使用的Jdk环境：jdk1.8.0\_211

服务器版本：tomcat8

* 环境自动部署包：
  + 包名称：web\_env

包含：apache-tomcat-8.5.43.tar.gz、jdk-8u221-linux-x64.tar.gz

* + 自动部署脚本：web\_env\_install.sh
    - 功能：为主机安装并配置jdk环境，安装tomcat8服务器