|  |
| --- |
| **学士学位论文** |
| **基于USRP的飞机轨迹信息捕捉系统的设计与实现** |
| |  |  | | --- | --- | | 姓　 名： | 王子通 | | 学 科 专 业： | 网络工程 | | 指 导 教 师： | 曾德泽 副教授 | | 培 养 单 位： | 计算机学院 | |
| 二○一七年六月 |

中国地质大学（武汉）学士学位论文原始声明

本人郑重声明：本人所呈交的学士学位论文《基于USRP的飞机轨迹信息捕捉系统的设计与实现》，是本人在我的导师的指导下，在中国地质大学（武汉）攻读学士学位期间进行的独立研究工作所取得的成果。除了该论文的已注明部分外不包含他人已发表或撰写过的研究成果，对论文的完成提供过帮助的有关人员已在文中说明并致以谢意。

我所呈交的学士学位论文没有违反学术道德和学术规范，没有侵权行为，我愿意承担由此而产生的法律责任和法律后果。

学位论文作者签名：

日 期： 年 月 日

摘要

如今，为了提高空中交通监视技术和飞行器间自主防撞功能，一种新型的ADS-B技术应运而生。飞行在飞行过程中通过广播ADS-B信号来告知周围飞行器和地面接收设备，ADS-B信号中包含位置和机型等信息，我们可以通过获取这些信息来模拟飞机飞行的航线轨迹和动态地展示。

本系统硬件基于USRP B210平台来模拟地面ADS-B的信号接收机，USRP搭载GNU Radio开源软件将信号进行捕获和解析，并将数据保存至数据库中，在Google Earth和Web端分别进行实时航班飞行轨迹的展示和历史飞行轨迹的重演。

在Google Earth端进行展示时，将数据库中保存的飞机轨迹信息读取并通过调用Google Earth服务接口写入KML格式的文件中，以便Google Earth能够读取并动态展示。在实时展示飞机飞行轨迹功能中，Google Earth定时读取实时生成的KML文件便可以动态展示飞机飞行的轨迹。在历史飞行轨迹重演功能中，在配置文件中更改想要查看飞机的航班号便可以在数据库通过多表查询的方式中获取与该航班号相关的所有信息如经纬度和海拔等，将该信息封装成数组并按照时间顺序进行排序，循环该数组，定时重写新的KML文件，让Google Earth定时读取同名的KML文件，便可以实现该航班的历史飞行轨迹的重演。

在Web端，启动Flask服务器，前端界面使用百度地图Web服务的API接口，呈现当前所在位置的地图区域。前端回向后端发送请求，后端对请求进行处理，将前端所需要的轨迹信息通过jQuery ajax方法返回给前端，Html文件中内嵌的Javascript处理函数该功能能够进行动态展示。在实时展示飞机飞行轨迹的功能中，执行USRP的启动程序并启动Flask服务器。USRP对上空飞行的飞机广播数据进行解析插入数据库的同时，生成此时飞机轨迹信息的KML文件。前端定时对后端发送请求，后端对当前KML文件进行解析，获取数据保存成数组返回给前端进行动态展示。在历史飞行轨迹重演功能中，输入想要查看飞机的航班号，前端把此航班号发送给后台，后台在数据库中获取与该航班有关的所有信息封装成数组，将通过jQuery ajax方法其返回给前端，前端循环该数组进行动态展示，便可以实现该航班的历史飞行轨迹的重演。

**关键词：ADS-B信号**；USRP；飞机轨迹信息；KML文件；Flask

**Abstract**

Today, a new type of ADS-B technology has emerged to improve air traffic surveillance technology and autonomous collision avoidance between aircraft. During the flight, the ADS-B signal is broadcasted to inform the surrounding aircraft and ground receivers. The ADS-B signal contains information such as location and aircraft type. We can obtain this information to simulate the route trajectory of the aircraft and dynamically display it. .  
The system hardware is based on the USRP B210 platform to simulate the signal receiver of the ground ADS-B. The USRP carries GNU Radio open source software to capture and analyze the signals, and save the data to the database, and perform real-time flight flight on Google Earth and Web sites respectively. The display of trajectories and the recurrence of historical flight trajectories.  
When displaying on Google Earth, the aircraft trajectory information saved in the database is read and written into the KML format file by calling the Google Earth service interface so that Google Earth can read and display it dynamically. In real-time display of the aircraft flight path function, Google Earth can read the real-time generated KML file periodically to display the flight path of the aircraft. In the historical flight path replay function, changing the flight number of the aircraft to be viewed in the configuration file can acquire all the information related to the flight number such as latitude, longitude, and altitude in the database through a multi-table query and package the information. Array and sort in chronological order, loop through the array, rewrite the new KML file at regular intervals, and let Google Earth periodically read the KML file of the same name, then we can replay the historical flight path of the flight.  
On the Web side, start the Flask server. The front-end interface uses the API interface of the Baidu map Web service to present the map area of ​​the current location. The front end sends back the request to the back end, and the back end processes the request, and the track information needed by the front end is returned to the front end through the jQuery ajax method. This function can be dynamically displayed by the embedded Javascript processing function in the Html file. In the function of displaying flight trajectories in real time, the USRP launch program is executed and the Flask server is started. The USRP parses aircraft broadcast data over the sky and inserts it into the database. At the same time, the KML file for generating aircraft trajectory information is generated. The front end periodically sends a request to the back end, and the back end parses the current KML file, and the acquired data is saved into an array and returned to the front end for dynamic display. In the historical flight path replay function, input the flight number of the aircraft you want to view, the front end sends the flight number to the background, and all the information related to the flight is retrieved in the database and encapsulated into an array, which will be returned to the jQuery ajax method. Front-end, front-end loop through the array for dynamic display, you can achieve a replay of the historical flight trajectory of the flight.  
**Keywords**: ADS-B signal; USRP; aircraft trajectory information; KML file; Flask

**目录**

[第一章 绪论 4](#_Toc514595175)

[1.1课题研究的背景和意义 4](#_Toc514595176)

[1.2 ADS-B技术 5](#_Toc514595177)

[第二章USRP相关原理和技术 5](#_Toc514595178)

[2.1 USRP设备介绍 6](#_Toc514595179)

[2.2 USRP设备安装 6](#_Toc514595180)

[2.2.1选择操作系统 6](#_Toc514595181)

[2.2.2更新和安装依赖库 6](#_Toc514595182)

[2.2.3编译安装UHD 7](#_Toc514595183)

[2.2.4编译安装GNU Radio 8](#_Toc514595184)

[2.2.5连接USRP 9](#_Toc514595185)

[2.2.6本章小结 9](#_Toc514595186)

[第三章 Google Earth端飞机轨迹展示 9](#_Toc514595187)

[3.1数据库格式设计 10](#_Toc514595188)

[3.1.1需求分析 10](#_Toc514595189)

[3.1.2概念设计 10](#_Toc514595190)

[3.1.3逻辑设计 11](#_Toc514595191)

[3.1.4详细设计 11](#_Toc514595192)

[3.2 KML文件数据结构设计 13](#_Toc514595193)

[3.2.1 KML介绍 13](#_Toc514595194)

[3.2.2概要设计 13](#_Toc514595195)

[3.2.3详细设计 14](#_Toc514595196)

[3.3实时显示飞机运行轨迹 15](#_Toc514595197)

[3.3.1需求分析 15](#_Toc514595198)

[3.3.2概要设计 15](#_Toc514595199)

[3.3.3详细设计 16](#_Toc514595200)

[3.4历史飞行轨迹重演 17](#_Toc514595201)

[3.4.1需求分析 17](#_Toc514595202)

[3.4.2概要设计 17](#_Toc514595203)

[3.4.3详细设计 17](#_Toc514595204)

[3.5本章小结 19](#_Toc514595205)

[第四章 Web端飞机轨迹信息展示 20](#_Toc514595206)

[4.1 Flask框架 20](#_Toc514595207)

[4.1.1 Flask简介 20](#_Toc514595208)

[4.1.2概要设计 20](#_Toc514595209)

[4.1.3详细设计 21](#_Toc514595210)

[4.2实时显示飞机运行轨迹 21](#_Toc514595211)

[4.2.1概要设计 21](#_Toc514595212)

[4.2.2详细设计 21](#_Toc514595213)

[4.3历史飞行轨迹重演 21](#_Toc514595214)

[4.4本章小结 21](#_Toc514595215)

[第五章 总结和展望 21](#_Toc514595216)

[5.1全文总结 21](#_Toc514595217)

[5.2后续研究展望 21](#_Toc514595218)

[5.2.1分布式系统 22](#_Toc514595219)

[5.2.2数据分析 22](#_Toc514595220)

[致谢 22](#_Toc514595221)

[参考文献 23](#_Toc514595222)

1. 绪论

1.1课题研究的背景和意义

进入到二十一世纪以来,随着民航航空运输事业的持续发展,构建全新的民航航行系统已经成为各个国家民航航空的重要发展趋势[1]。中国航空工业第一集团公司预测，到2026年我国 客运周转量接近1.2万亿人千米[2]。根据“国家发展纲要”，民航运输行业将继续保持高速增长，空中交通流量将会继续增加，对提升空域容量能力和提高空中交通管制的综合保障能力提出了更高要求。

自动相关监视（ADS）技术是新型导航系统开发最重要的成果之一，也是解决国际空中交通管制最有效的方法。广播式自动相关监视ADS-B是一种基于GPS全球卫星定位系统和空-空、地-空数据链通信的航空器运行监视技术。[3]根据冲突检测和交通情况图，实现飞机相互监视和指挥能力，提高航行安全性。通过有效降低飞机在空域的最小间隔来增加空域容量，并提高空域资源的利用率。 ADS-B技术的良好通信功能和监测方法可以更准确，实时和持续地了解飞机动力学和有效控制。

根据统一规划和全面推进的原则和国家五年发展规划，运输航空ADS-B的建设和运营将分两个阶段实施。 到第十二个五年计划结束时，ADS-B OUT（地对地监测）的初步运行实现了，到第十三个五年计划结束时，ADS-B OUT全面投入运行。

路线，终端和机场塔都采用基于ADS-B的新型监视技术作为主要的空中交通监控工具，以构建完整的ADS-B运营保障和信息服务系统。在运输航空和通用航空领域，对 ADS- B IN技术进行了全面测试，推广和应用， 实现了空对空监视，搭建了天空协作系统，为实现空天提供了强大的技术支持。

抓住机遇，加快ADS-B等新技术的建设和应用，对于促进民航业持续增长，提高民航安全水平，空域容量，运行效率和服务能力具有重要意义。

1.2 ADS-B技术

ADS-B主要实施空对空监视，一般情况下，只需机载电子设备(GPS 接收机、数据链收发机及其天线、驾驶舱冲突信息显示器CDTI)，不需要任何地面辅助设备即可完成相关功 能，装备了ADS-B的飞机可通过数据链广播其自身的精确位置和其它数据(如速度、高度及飞机是否转弯、爬 升或下降等)[4]。ADS-B接收机与空管系统、其它飞机的机载ADS-B结合 起来，在空地都能提供精确、实时的 冲突信息[5]。

ADS-B系统由多地面站和机载站构成，以网状、多点对多点方式完成数据双向通信[6]。 机载 ADS-B 通信设备广播式发出来自机载信息 处理单元收集到的导航信息，接收其他飞机和地面的广播信息后经过 处理送给机舱综合信息显示器[7]。机舱综合信息显示器根据收集的其他飞机和地面的ADS-B信息、机载雷达信息、导航信息后给飞行员提供飞机周围的态势信息和其他附加信息(如：冲突告警信息，避碰策略，气象信息)[8]。

ADS-B技术能提高飞行中的航空器之间的相互监视能力[9]。与ACAS / TCAS相比，ADS-B的报告是自发广播式的，飞机可以接收和处理飞机位置报告，而无需发送询问。因此，它可以有效地提高机载协调能力，提高机载防撞系统的性能，并实现这一目标能够保持最小安全间隔并避免和解决冲突的空对空协调目标。ADS—B系统的这一能力，使保持飞行安全间隔的责任更多地向空中转移，这是实现“自由飞行”不可或缺的技术基础[10]。

1.3本文的工作

本文旨在模拟ADS-B地面监控系统，通过USRP设备捕捉武汉市上空所有机载ADS-B设备的航班广播式发出的信息，经过处理后，航班的航行轨迹能实时地显示在本地的Google Earth上，还提供了该航班的海拔，经纬度，机头朝向，速度，垂直速度，捕捉时间，以及该航班的飞机照片等等。同时在Web端也进行了相关的部署，可以通过浏览器进行远程访问。

利用GNU Radio开源的平台，结合通用软件外设USRP，搭建无线电通信系统的开源软件系统以模拟ADS-B信号地面接收站。通过Python编写程序处理无线电信号转化为图像文字信息在Google Earth进行展示。完成并实现了航班飞机信息实时捕捉，航班历史轨迹信息重演和航班信息分析等功能。

在Web端，重写了以上的功能，利用百度地图提供的API将飞机航行轨迹也显示在百度地图上，可进行飞机航行信息的实时捕捉和航班历史轨迹重演。

第二章 系统设计

2.1系统架构

2.1.1系统架构图

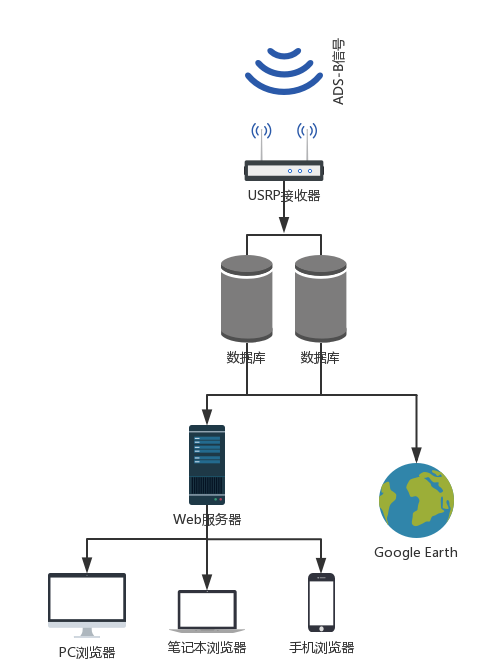


图 1

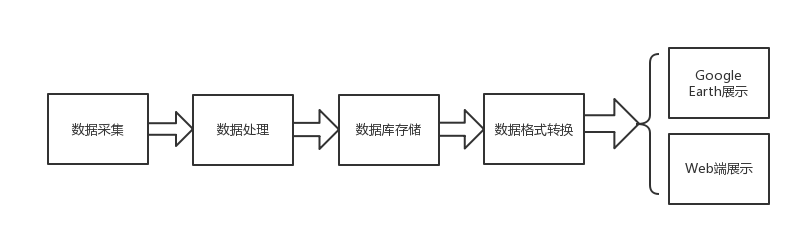
2.1.2系统流程图 

图 2

2.2系统原理

2.2.1 USRP设备介绍

USRP（Universal Software Radio Peripheral）中文名为通用软件无线电外设。大多数USRP通过高速链路连接到主机，主机软件用它来控制USRP硬件和发送和接收数据。USRP系列产品专为无障碍设计而设计，其中的产品都是[开源硬件](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com.hk&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Open_source_hardware&xid=17259,15700019,15700124,15700149,15700168,15700173,15700186,15700189,15700201&usg=ALkJrhiAV3bxk_yLQKz5pzF0jEpJ2LmMKw" \o "开源硬件) ，并且所有的USRP产品都使用开源UHD驱动程序进行控制，这是[免费的开源软件](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com.hk&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Free_and_open_source_software&xid=17259,15700019,15700124,15700149,15700168,15700173,15700186,15700189,15700201&usg=ALkJrhiUgXhnuEWuehI-k5DhXy8WXX_Y-A" \o "免费和开源软件) 。 USRP通常与[GNU Radio](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com.hk&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/GNU_Radio&xid=17259,15700019,15700124,15700149,15700168,15700173,15700186,15700189,15700201&usg=ALkJrhhEe30VsJ62IO0TAzI50qwECUEQeQ" \o "GNU广播)软件套件一起使用，以创建复杂的软件定义无线电系统。

本系统将使用USRP硬件平台，利用GNU Radio进行无线电的开发，来捕捉飞机广播的ADS-B信号并加以利用。本系统所用USRP型号为B210，只需要USB端口与电脑进行连接。

USRP硬件平台的安装主要分为四部分，首先选择合适的操作系统，其次更新和安装依赖库，然后编译安装UHD（USRP Hardware Driver）和GNU Radio，最后对环境变量进行配置。[11]

2.2.2 Google Earth端介绍

USRP（Universal Software Radio Peripheral）中文名为通用软件无线电外设。大多数USRP通过高速链路连接到主机，主机软件用它来控制USRP硬件和发送和接收数据。USRP系列产品专为无障碍设计而设计，其中的产品都是开源硬件，并且所有的USRP产品都使用开源UHD驱动程序进行控制，这是免费的开源软件。 USRP通常与[GNU Radio](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com.hk&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/GNU_Radio&xid=17259,15700019,15700124,15700149,15700168,15700173,15700186,15700189,15700201&usg=ALkJrhhEe30VsJ62IO0TAzI50qwECUEQeQ" \o "GNU广播)软件套件一起使用，以创建复杂的软件定义无线电系统。

本系统将使用USRP硬件平台，利用GNU Radio进行无线电的开发，来捕捉飞机广播的ADS-B信号并加以利用。本系统所用USRP型号为B210，只需要USB端口与电脑进行连接。

USRP硬件平台的安装主要分为四部分，首先选择合适的操作系统，其次更新和安装依赖库，然后编译安装UHD（USRP Hardware Driver）和GNU Radio，最后对环境变量进行配置。[11]

2.3

2.4

第三章 概要设计

第四章 详细设计

第三章 USRP相关原理和技术

3.1 USRP设备介绍

本文采用的系统平台：

* OS Ubuntu 16.04
* Python Version：python 2.7
* USRP B210

3.2 USRP设备安装

3.2.1选择操作系统

本系统使用的操作系统是Ubuntu 16.04，同时也在MacOS 10.12.6上进行测试，均未发现异常，本文以Ubuntu 16.04作为样本进行详述。不同操作系统的安装USRP过程不一，但是大致相同，亦可尝试Ubuntu其他版本和Fedora系统。

3.2.2更新和安装依赖库

在编译安装UHD和GNU Radio之前，需要首先安装一系列的依赖库，并且要保证这些安装包已经更新到最新版本，否则会出现系统不兼容等情况，以致整个过程安装失败。

在Ubuntu系统的命令行中运行一下命令：

$ sudo apt-get -y install git swig cmake doxygen build-essential libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev libncurses5-dev libfftw3-bin libfftw3-dev libfftw3-doc libcppunit-1.13-0v5 libcppunit-dev libcppunit-doc ncurses-bin cpufrequtils python-numpy python-numpy-doc python-numpy-dbg python-scipy python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev python-qt4-doc python-qt4-doc libqwt6abi1 libfftw3-bin libfftw3-dev libfftw3-doc ncurses-bin libncurses5 libncurses5-dev libncurses5-dbg libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake autoconf libtool python-dev libfftw3-dev libcppunit-dev libboost-all-dev libusb-dev libusb-1.0-0-dev fort77 libsdl1.2-dev python-wxgtk3.0 git-core libqt4-dev python-numpy ccache python-opengl libgsl-dev python-cheetah python-mako python-lxml doxygen qt4-default qt4-dev-tools libusb-1.0-0-dev libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4 cmake git-core wget libxi-dev gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0 liborc-0.4-dev libasound2-dev python-gtk2 libzmq-dev libzmq1 python-requests python-sphinx libcomedi-dev python-zmq

如果在安装到完成在过程中没有出现任何错误，然后就可以进行下一步安装UHD和Radio的操作了。

3.2.3编译安装UHD

UHD即USRP的硬件驱动程序，安装完UHD才能自由地操控USRP硬件系统。安装UHD的过程并不复杂，并且所有代码都已开源，需要从GitHub上下载源码进行编译安装。UHD代码仓库地址<https://github.com/EttusResearch/uhd/tree/release_003_010_001_000>

首先Clone代码仓库到本地。

$ git clone <https://github.com/EttusResearch/uhd>

$ cd uhd

检查UHD的版本，确保是最新的版本。

$ git tag -l

在本地的代码仓库的host目录下新建一个文件夹命名为build，用于编译。

$ cd host

$ mkdir build

$ cd build

采用CMake的方式创建Makefile进行编译。

$ cmake ../

运行make命令对源码进行编译

$ make

接下来安装UHD，使用特权模式。

$ sudo make install

更新系统共享库的缓存，使用特权模式。

$ sudo ldconfig

配置LD\_LIBRARY\_PATH环境变量。

$ vi ~/.bashrc

$ export LD\_LIBRARY\_PATH=/usr/local/lib

自此，编译安装UHD过程结束，USRP的驱动安装完成。

3.2.4编译安装GNU Radio

GNU Radio 的软件编程是基于 python 脚本语言和 C++ 的混合方式[11]。GNU Radio 中称这种模块为 block。Python是一种新型的脚本语言，不需要编译，语法和全面的对象定位。GNU Radio具有出色的多功能性和开放性，可以支持多种操作系统，如Linux，MacOS X，NetBSD和Windows。

和UHD相同，GNU Radio的源代码也在GitHub上开源，代码仓库的地址如下：<https://github.com/gnuradio/gnuradio/tree/v3.7.10.1>。同样的，GNU Radio也是通过Clone源代码进行编译安装的，当然还要确定版本是否兼容。

首先也是将GitHub上的代码Clone到本地。

$ git clone --recursive <https://github.com/gnuradio/gnuradio>

进入本地代码仓库所在的文件夹内查看版本号，以确保是最新的版本。

$ cd gnuradio

$ git checkout v3.7.10.1

$ git submodule update --init --recursive

在本地代码仓库中创建一个build文件夹，用于存储编译生成的文件。

$ mkdir build

$ cd build

调用CMake方法来创建MakeFiles文件

$ cmake ../

利用Make命令进行编译GNU Radio的源代码。

$ make

安装GNU Radio，采用特权模式。

$ sudo make install

更新系统共享库的缓存，使用特权模式。

$ sudo ldconfig

3.2.5连接USRP

在命令行运行以下命令，来找到外接的USRP设备。

$ uhd\_find\_devices

若发现设备为空，可以尝试将USB数据线重新将USRP与电脑相连，重新连接后便可发现设备。

加载更新USRP的固件，对USRP与其驱动UHD进行匹配。

$ uhd\_usrp\_probe

当连接设备后，如果发现烧录失败等情况，说明该型号USRP与该版本的UHD不匹配，则需要另外下载固件对USRP进行烧录。

3.2.6本章小结

本章主要任务是硬件平台的准备和硬件软化。首先简述了什么是USRP以及USRP设备的软件平台GNU Radio，然后详细阐述了USRP外设在Ubuntu操作系统的详细安装步骤。

在编译安装过程中可以使用-j后缀命令开启多个进程共同执行，可降低等待时间。

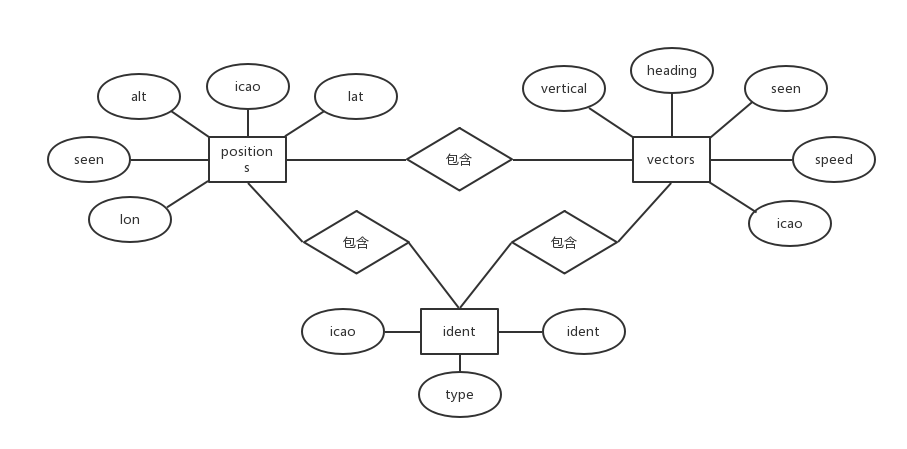
本过程较为复杂，需要外部依赖库较多，而且囿于软件版本的影响，存在不兼容的现象。但只要一步步来，将每一步的做法和目的熟稔于心，安装过程也绝非难事。

第四章 Google Earth端飞机轨迹展示

4.1数据库格式设计

4.1.1需求分析

4.1.2概要设计

E-R图

4.1.3逻辑设计

ident (icao, ident, type) icao是主码和其他表的外码

positions (icao, seen, alt, lat, lon) icao是主码和其他表的外码

vectors (icao, seen, speed, heading, vertical) icao是主码和其他表的外码

icao为三个表的外码，通过icao进行多表关联。

4.1.4详细设计

首先导入SQLite驱动.

import sqlite3

连接数据库来操作关系运算符，若文件不存在，会自动在当前目录下创建该数据库，如果连接成功就会返回一个连接对象。

db = sqlite3.connect(filename)

连接好数据库后，需要打开游标Cursor，通过Cursor来执行SQL语句，返回执行结果。

c = db.cursor()

创建第一个表ident：

query = """CREATE TABLE IF NOT EXISTS "ident" (

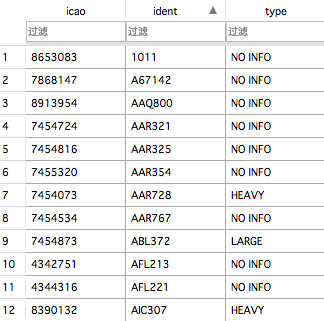
"icao" INTEGER PRIMARY KEY NOT NULL,

"ident" TEXT NOT NULL,

"type" TEXT NOT NULL

);"""

效果图如下：



创建第二个表positions：

query = """CREATE TABLE IF NOT EXISTS "positions" (

"icao" INTEGER KEY NOT NULL,

"seen" TEXT NOT NULL,

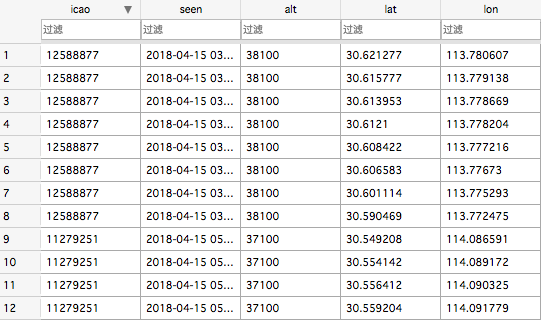
"alt" INTEGER,

"lat" REAL,

"lon" REAL

);"""

效果图如下：



创建第三个表vectors：

query = """CREATE TABLE IF NOT EXISTS "vectors" (

"icao" INTEGER KEY NOT NULL,

"seen" TEXT NOT NULL,

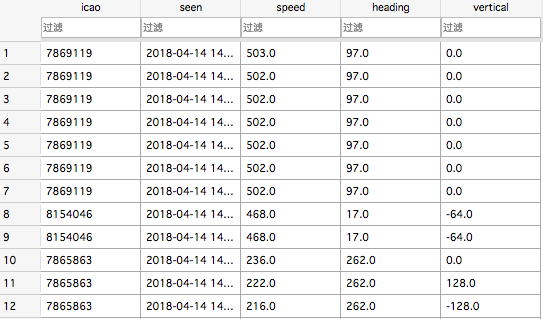
"speed" REAL，

"heading" REAL，

"vertical" REAL

);"""

效果图如下：



通过游标Cursor来执行SQL语句，实现创建或插入等操作：

c.execute(query)

4.2 KML文件数据结构设计

4.2.1 KML介绍

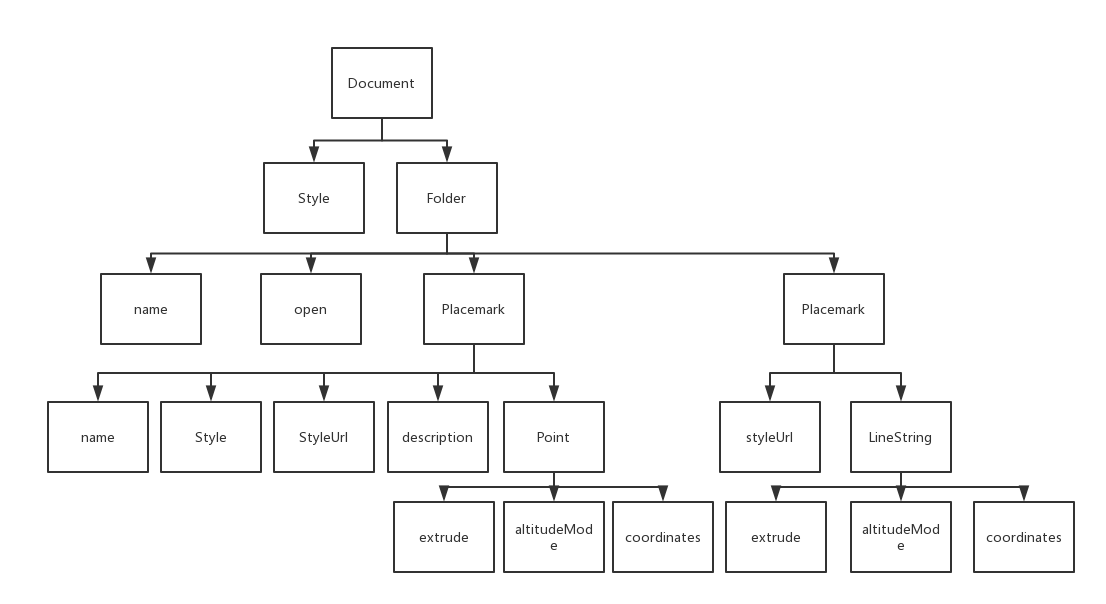
KML(Keyhole Markup Language, Keyhole 标记语言)最初是由Google 旗下的Keyhole 公司开发和维护的一种基于XML 的标记语言， 使用 XML语法格式描述地理空间数据（例如点，线，多边形，多边形，模型等）它适用于在网络环境中协作和共享地理信息。 2008年4月，OGC宣布KML 2.2的最新版本为开放式地理信息编码标准，由OGC维护和开发。

Google Earth可以直接读取KML文件中地理数据信息，并进行可视化展示。KML实用包含嵌套的元素和属性的结构，在KML标签中插入数据库中存储的飞机飞行轨迹信息。

4.2.2概要设计

本单元旨在Google Earth中显示飞机所处的位置以及飞行轨迹，故在地点标记（Placemark）中设计两个标签：点标签（Point）和线标签（LineString）。在点标签中叠放图片标签和经纬度以及各种描述信息的标签；在线标签中叠放颜色和经纬度等信息。

4.2.3详细设计

通过参考Google Earth API中生成KML的文档，设计了如下的数据结构:

在Folder标签中，加入四个子标签，分别为name，open，点的地点标签Placemark和线的地点标签Placemark。

在点的地点标签Placemark中的Points标签，存放当前飞机所在的位置信息。

<Point>

<altitudeMode>absolute</altitudeMode>

<extrude>1</extrude>

<coordinates>113.98233,30.668707,7193</coordinates>

</Point>

extrude标签表示该坐标点在Google Earth中延伸，即叠层是否突出。

coordinates标签表示经纬度和海拔高度信息。

Python代码的实现：

KML.Point(

KML.altitudeMode("absolute"),

KML.extrude(1),

KML.coordinates("{},{},{}".format(lon, lat, alt))

)

在线的地点标签Placemark中的LineString标签，存放当前飞机所在的位置和历史轨迹信息。

<LineString>

<extrude>0</extrude>

<altitudeMode>absolute</altitudeMode>

<coordinates>113.982330,30.668707,7193.280000 113.984198,30.668661,7193.280000 114.142025,30.664398,7193.280000 </coordinates>

</LineString>

extrude标签表示该坐标点在Google Earth中延伸，即叠层是否突出。

coordinates标签表示当前经纬度和海拔高度信息和历史经纬度和海拔高度信息，通过空格进行分割。

KML.LineString(

KML.extrude(0)，

KML.altitudeMode("absolute"),

KML.coordinates("{}".format(history))

)

在KML文件中插入矢量图片，并根据飞机当前的飞行方向调整图片的角度。

<Icon><href>airports.png</href></Icon>

<Style><IconStyle><heading>237</heading></IconStyle></Style>

heading标签是飞机的飞行方向。

Python代码的实现：

KML.Icon(

KML.href("gr-air-modes/apps/airports.png")

)

KML.IconStyle(

KML.heading(heading)

)

4.3实时显示飞机运行轨迹

4.3.1需求分析

这一功能能够实时地准确捕捉到在USRP ADS-B信号接收器250-450公里内的飞机广播信号，将该信号进行处理，保存在数据库中，并在Google上实时展示出来。

4.3.2概要设计

开启USRP信号接收程序，把捕获的信号进行解析并保存到数据库的相应数据表中，同时把刚刚插入数据表中的数据写入KML文件中，每个0.2秒进行KML文件的重写，若是同一航班号，在更新新坐标的同时，保存历史轨迹，从而形成一条飞行轨迹。在Google Earth端能够每隔极短的时间读取该KML文件，进行飞机动态实时展示，并显示飞行轨迹。

4.3.3详细设计

首先电脑连接USRP外设，加载更新最新的固件，启动实时接收和处理脚本：

./run\_current.sh

该执行脚本是启动modes\_rx程序，将捕捉解析后的位置等信息插入到数据库中，并将最新数据临时保存到KML文件中，以便Google Earth能够读取进行实时展示。

modes\_rx程序中包含，ADS-B无线电信号的捕捉，ADS-B信号的解析，将数据写入KML文件和将数据插入到SQLite数据库。

对无线信号的捕捉需要无线信号处理模块air\_modes和消息处理队列模块zmq进行数据传输。

import air\_modes

import zmq

无线电信号捕捉和传输代码如下：

context = zmq.Context(1)

tb = air\_modes.modes\_radio(options, context)

servers = ["inproc://modes-radio-pub"]

if options.remote is not None:

servers += options.remote.split(",")

relay = air\_modes.zmq\_pubsub\_iface(context, subaddr=servers, pubaddr=None)

对捕捉到的数据进行解析处理，将解析后的原始数据保存在数组中，代码如下：

publisher = pubsub()

relay.subscribe("dl\_data", air\_modes.make\_parser(publisher))

my\_position = [float(n) for n in options.location.split(",")]

cpr\_dec = air\_modes.cpr\_decoder(my\_position)

将保有原始数据的输入，通过多线程的方式插入进数据库的不同数据表中，该过程用到了锁，以防在一次插入多组数据的情况下造成死锁，引发异常。

dbname = 'adsb.db'

lock = threading.Lock()

sqldb = air\_modes.output\_sql(cpr\_dec, dbname, lock, publisher) #input into the db

同时也将该数组中的数据写入到KML文件中，同时当捕获解析出下一组数据的时候，判断该航班是否出现过：如果出现过，则更新当前的坐标点，并将原有的坐标点加入到轨迹中；如果未出现过，则新增该航班的位置；此外如果新的数据中缺少已在KML文件中存在过的航班，则将KML中关于该组航班的信息全部删除。

kmlgen = air\_modes.output\_kml(options.kml, dbname, my\_position, lock)

4.4历史飞行轨迹重演

4.4.1需求分析

这一功能能够将以往捕捉到的飞机轨迹信息进行重演，在Google Earth中进行动态展示。所有数据信息是基于数据库中保存过的信息。

该功能能够重现特定航班在特定时间段内的飞行状态，模拟航行过程中可以预测该航班在空中是否晚点，是否发生事故以致航线变化等等。

4.4.2概要设计

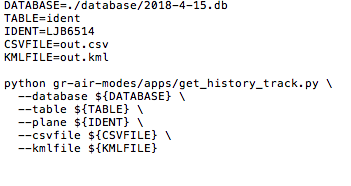
根据数据库中存储的信息，可以动态模拟飞机的飞行状态，进行历史飞行轨迹的重演。在配置文件中输入想查询飞行轨迹的航班号，执行脚本文件，随即在数据库中查询与该航班号有关的所有信息，在Google Earth中观察该航班的飞行轨迹和其他飞行信息。

4.4.3详细设计

首先在数据库中查询到我们想要查看的飞机航班号，在配置文件中修改修改想查询的数据库，航班号，即可执行脚本文件

./run\_history.sh

脚本文件如下：



在get\_history\_track.py文件中主要分成四个函数连接数据库函数connect\_database()，get\_icao() ，get\_message() ，write\_to\_csv() ，update\_kml()，功能分别为连接数据库，根据航班号获取航空识别码ICAO值，获取该航班所有的信息，将该航班所有数据信息写入CSV文件，将该航班所有数据信息按时写入KML文件以在Google Earth上进行展示。

在获取该航班信息的函数get\_message()中，通过选定的航班号在数据库的ident表中进行模糊查询获取icao值：

sql = "select icao from " + self.table +" where ident like '%"+self.ident+"%'"

再通过获取的icao值在另外表中positions和vectors获取seen， speed，heading，vertical和alt，lon，alt等元素。

sql1 = "SELECT seen, speed, heading, vertical from vectors where icao="+str(self.icao)

sql2 = "SELECT seen, alt, lat, lon from positions where icao="+str(self.icao)

将以上所有元素保存在一个二维数组中，再进行下一步利用。

将该航班所有数据信息写入CSV文件，通过write\_to\_csv()进行实现:

csvFile = open(self.csvfile, 'wb')

writer = csv.writer(csvFile)

for msg in self.message:

writer.writerow(msg)

csvFile.close()

效果图如下:

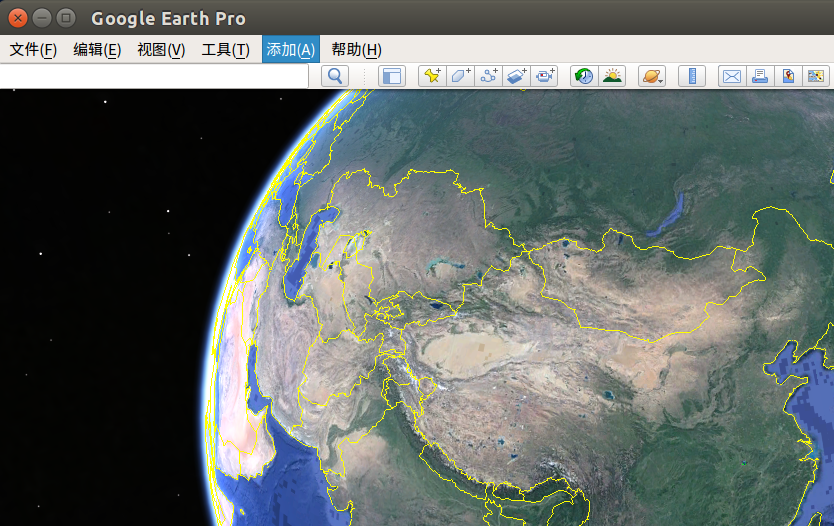


将该航班所有数据信息写入KML文件，将该航班所有信息的二维数组作为参数传入update\_kml()函数，每隔0.2秒对KML进行一次重写，同时设置Google对KML文件的读取周期为1秒，即KML文件每次重写更新，便可以在Google Earth中展示出来。

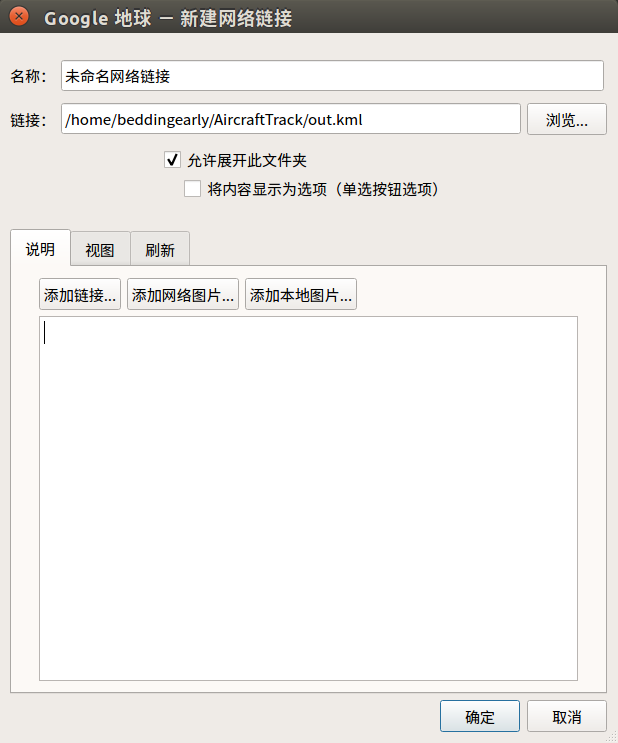
在KML文件中，为了展示该航班的历史轨迹，要将历史的经纬度和海拔高度信息在每次更新重写时，一起添加到StringLine的标签中每次，并用空格进行分割:

history = str(lon) + ',' + str(lat) + ',' + str(alt) + " " + history

在执行run\_curent.sh脚本文件后，在Google Earth中进行设置，上方菜单点击“添加—网络连接”。



在“浏览”中添加KML文件路径，并新的网络连接进行命名。



在“刷新” 中设置定期刷新，刷新时间为1秒。



4.5本章小结

本章主要介绍了在Google Earth端对飞机运行轨迹的模拟方法，分成四个部分进行展开，分别为数据库的设计，KML文件的设计，以及以下两个功能：实时显示飞机运行轨迹和历史飞行轨迹重演的实现。

首先是数据库部分，在数据库部分中先是介绍了本系统根据实际需求所采用的数据库引擎SQLite，然后分析了本系统对于存储的设计需求，按照实际情况设计多张数据表，每张表存储与每张表的相关内容，通过航空识别码icao将多张表关联起来，实现多表查询。

其次详细介绍KML文件以及KML文件的结构和生成方式，由于KML文件与Google Earth交互十分友好，故采用数据库中的飞机轨迹等内容生成KML文件的形式在Google Earth中进行展示。通过参考Google Earth给出的API文档详述，自行编写代码对数据库中存储的信息按照KML文件的数据形式进行组织。最后在Google Earth中添加KML文件路径，并定时对文件进行读取以达到动态演示的功能。

在实时显示飞机运行轨迹这个功能中，执行启动脚本，USRP开始接收飞行中的飞机广播的ADS-B信号，信号解析成数据信息，将该数据信息通过多线程的方式插入到数据库中和写入KML文件。Google Earth定时对KML文件进行读取并动态展示。

在飞机历史飞行轨迹重演这个功能中，执行启动脚本，程序在数据表ident查询所需要的航班号所对应的icao值，得到后在其他两个表positions和vectors中获取经纬度，海拔，飞机型号，飞行速度等信息，并按照一定的时间间隔写入KML文件中， Google Earth定时对KML文件进行读取并动态重演。

第五章 Web端飞机轨迹信息展示

5.1 框架和API

5.1.1 Flask简介

Flask是通过Python编写的Web应用框架，结构轻巧，只需要Pip安装即可使用，在Flask文件的同级目录下创建两个固定的文件集static和template，static文件夹下放有js文件，template文件夹下放有html文件。

使用过程中手动引入Flask类，Flask通过路由route()装饰器传入URL来触发函数。通过run()函数来让该服务运行在本服务器上。run()函数定义在主函数中，确保服务器在Python解释器直接执行的时候才会运行。

本系统的Web端展示采用基于Flask的Web框架实现，简洁轻量，源码也是短小精悍，这样在编写Web端的应用时，只需要处理前端与后台的交互逻辑即可。

在本系统中，Flask本身带有Web服务器功能，故通过启动Flask来运行服务器，前端作为客户端通过Html和CSS以及Javascript进行编写，Javascript向后端Flask发送请求，Flask后台收到请求后向SQLite数据库进行查询操作，将得到的数据返回给前端进行动态展示。

5.1.2百度地图API

在百度地图API中，使用了如下几个功能：百度地图的展示，在地图中添加经纬度点的矢量标注点（Marker），在Marker上添加图片覆盖物，在两个Marker中添加轨迹线，在Marker中创建点击时间的信息窗口等等。

以上功能均实用Javascript实现，在使用的过程中根据飞机信息的经纬度信息，动态地

5.1.3详细设计

5.2实时显示飞机运行轨迹

5.2.1概要设计

启动Flask服务器，在网页上选择实时显示飞机运行轨迹。在后台执行USRP启动脚本，当USRP开始接收和解析数据时，会将实时的飞行航班信息写入KML文件中。前端定时向Flask后台发送请求，后台会对KML文件进行解析得到数据，并将数据返回给前端进行展示，从而实现实时显示飞机的运行轨迹。

5.2.2详细设计

5.3历史飞行轨迹重演

5.3.1概要设计

启动Flask服务器，在网页上选择历史飞行轨迹重演，选择想要查看飞机的航班号，前端将此航班号返回给后台Flask服务器，后台通过该航班号在数据库中的ident表进行搜索的到对应的icao值，通过icao值在其他两个表positions和vectors中查询经纬度和海拔等其他轨迹信息。将这些轨迹信息封装成数组返回给前端，前端定时进行飞行轨迹进行展示，从而实现历史飞行轨迹的重演。

5.3.2详细设计

本功能设计主要是前端后台的交互逻辑和前端的展示，在历史轨迹查询界面，设置两个按钮控件和一个文本输入框控件。在id为ident的文本输入框中输入想要查询的航班信息，并通过getJson()方法发送一个查询请求，直接跳转到请求后端Flask路由（/ddd）的函数中，将输入到文本框中的内容保存到变量keyvalue中发送给后端消息处理函数。

$SCRIPT\_ROOT = {{request.script\_root|tojson|safe}}

var keyvalue=document.getElementById('ident').value;

$.getJSON($SCRIPT\_ROOT+'/ddd', {

ddd: keyvalue

}, function (data) {/\*\*/}

后端通过request.args.get()方法接收到前端发送的keyvalue变量，即航班号信息。随即通过此航班号在数据库中进行查询，得到该航班号下的所有飞机信息。

@app.route(‘/ddd’)

def ShowHistory():

keyvalue = str(request.args.get('ddd'))

a = SqliteOperator('../database/2018-4-15.db', 'ident', keyvalue, '', '')

aa = a.get\_message()

return jsonify(aa)

后端把飞机数据通过jsonify函数进行封装，再返回给前端。前端也是通过上述getJson()方法中的第三个参数function(data)进行获取。前端对data进行处理。

for(var i = 0; i < data.length; i++){

var utc = data[i][0]

var ident = data[i][1]

var icao = data[i][2]

var alt = parseFloat(data[i][3])

var lat = parseFloat(data[i][4])

var lon = parseFloat(data[i][5])

var speed = data[i][6]

var heading = parseInt(data[i][7])

var vertical = data[i][8]

5.4本章小结

本章旨在将第三章的功能在Web端进行展示，实现方法大致相同，但是用到了前端后台的交互逻辑以及前端的动态展示。用Javascript和Html编写前端，Python和Flask编写后端，通过jQuery ajax方法实现前端和后端的交互。

第六章 总结和展望

6.1全文总结

本系统是基于USRP硬件平台，通过利用GNU Radio进行无线电的开发，对飞机广播的ADS-B信号进行捕捉和解析成原始轨迹信息的数据，将数据信息保存至数据库并加以应用。本系统功能有实时飞机飞行轨迹的演示和历史飞机飞行轨迹的重演。并在Google Earth端和Web端进行分别进行了实现。

本系统所做的工作有数据库的基本操作，GNU Radio对信号进行捕获和解析，USRP硬件平台的搭建，Flask Web服务器的搭建，使用Python语言对整个系统的编写和重构，前端与后台的交互逻辑和jQuery ajax方法，使用Html和Javascript对前端界面进行视觉呈现，对百度地图API的使用和对Google Earth API的使用。

6.2后续研究展望

6.2.1分布式系统

在本系统中，由于USRP接收飞行的ADS-B信号范围有限，大约300公里范围，所以只能接收到在武汉上空飞过的飞机。而如果想搜集全国乃至全世界范围的飞机飞行轨迹，最好的方法就是采用分布式的方式。

分布在各个地区的USRP设备等能够ADS-B信号的设备将捕获到的信息上传到云服务器上，进行信息共享，从而组织成一张全国乃至全世界的飞机航线信息图。

6.2.2数据分析

在乘坐飞机过程中，我们会担心晚点问题，而航空公司官网只能给出起飞时候和预计的降落时刻，这些对于惜时如金的我们显然是不合理。当飞机飞向高空不能等同于进入了黑箱，我们需要时刻了解航班的动向。

对于近期频发的劫机和事故等事件，能够根据飞行轨迹等信息进行预判，当飞机飞行轨迹出现严重偏差时，我们可以推断出该航班可能遇到了麻烦。

对于飞跃海峡或者大洋的飞机，由于没有地面雷达设备捕捉，可以根据分布式USRP接收设备最后一次捕捉到的轨迹信息，并参考其他对该区域覆盖重叠的USRP设备捕捉到的信息对未知航线轨迹进行预测。

致谢

时光如白驹过隙，仿佛是昨天刚进中国地质大学的校门。回想过去的这四年大学生活，最大的感触就是感激二字。首先我要感谢我的论文老师，中国地质大学（武汉）计算机学院的网络工程系主任曾德泽老师，正是在曾德泽老师的指导和帮助下，我才对USRP硬件平台有所理解，在系统设计的同时也提高了我的代码编程能力，同时对前端和后台的交互的理解也才能更加深入、更加彻底，从而才能够顺利地完成这篇论文。曾老师也是我自2014年入校以来的班主任，曾经也给过我很多锻炼自己的机会，而且他作为一名海归，有着与国内教授不同的眼光和间接，对我来说曾老师是亦师亦友的存在。因此，在本科毕业设计选题的时候才会义无反顾的选了曾德泽老师，当然，事实也证明我的选择是非常明智的，在论文撰写过程中对我遇到的障碍和不懂的地方给予了耐心的帮助，提出了许多建设性的意见，也付出了大量的心血和精力。与此同时，我也选择了曾老师作为未来三年研究生的导师，有曾老师的帮助和带领，我认为研究生时光将会非常美好。

此外同时还要感谢193142班的同学们，四年来，我们相互学习，相互帮助，感谢你们在这四年来的陪伴，感谢你们在学习和生活上对我的包容和支持，如果没有了你们，大学四年我就不会收获这么多幸福和感动。

参考文献

1. 靳龑, 孙志浩. ADS-B技术的发展情况分析及应用前景讨论[J]. 中国新通信, 2015(3):95-95.
2. 新浪.军民两用技术与产品[J].国防，2007(10):7
3. 范军. 浅析广播式自动相关监视技术的应用[J]. 江苏科技信息, 2014(19):35-37.
4. 赵健. 长海机场航务现有管理模式及合理化建议[J]. 科技创新与应用, 2013(15):279-280
5. 时宏伟.ADS-B数据链应用技术的综合评述[J].空中交通 管理，2007(6):13-16.
6. 张天平, 郝建华, 许斌,等. ADS-B技术及其在空管中的发展与应用[J]. 电子产品世界, 2009, 16(6):34-37.
7. 陈宝玉. ADS-B技术在空管自动化系统中的应用[J]. 科技视界, 2014(17):95-96.
8. 国徽. ADS-B引导下的目视进近规范及查询软件设计[D]. 电子科技大学, 2011.
9. 戴超成. 广播式自动相关监视(ADS-B)关键技术及仿真研究[D]. 上海交通大学, 2011.
10. 李自俊. ADS-B技术在通用航空飞行中的应用[J]. 国际航空, 2008(4):62-64.
11. 郭志刚, 付毅. 基于USRP的无线电收发系统研究[J]. 电子质量, 2016(11):5-10.