

摘要

随着技术的进步和工业化的普及，无人机和普通人的距离越来越小。消费级无人机越来越普及，植保无人机等商用无人机也得到大量应用。对于无人机群的需求也越来越大，无人机群可以广泛应用于灯光表演和战术攻击等多种场景。

本设计立足于解决无人机群应用时对于内部网络连接状态的监控，并可以实现实时监视无人机位置和姿态，同时还可以对单个无人机的网络信息进行设置。

本文开篇先介绍了关于无人机管理系统以及无人机网络的国内外研究现状，分析了相关发展趋势。然后简单介绍了无人机群网络管理系统开发时所涉及的相关技术和开发环境，本设计主要基于 WPF（Windows 呈现基础），包括用于界面设计的 XAML 语言，用于实现后台逻辑的 C# 语言。数据库存储使用了 SQLite。

文章的重点在于各个功能的设计实现上，主要包括：通过 GMap.NET 技术加载电子地图，展现无人机群内各无人机的实时位置，进一步实现了无人机历史飞行轨迹绘制；通过在 WPF 的 Canvas 绘制二次贝塞尔曲线展示集群内无人机之间的网络拓扑；通过 Material Design In XAML 控件库，使得无人机网络设置信息展示更加美观，当用户更改设置后，信息将借由 Socket 类发送给后台服务器以更新无人机设置；无人机飞行姿态的展示是通过虚拟仪表盘实现的，虚拟仪表盘展示了无人机的偏航角、俯仰角、滚动角和海拔；无人机历史飞行轨迹和姿态的展示，调用了数据库内存储的历史飞行信息，同时展示电子地图上的轨迹和实时姿态。

最后，通过对系统各模块的单独测试和系统整体调试，验证了系统设计的可行性和可靠性，满足了系统设计的需求，实现了预期效果。

关键词：无人机群；网络；数据库；WPF；GMap.NET；网络拓扑；飞行姿态

ABSTRACT

With the advancement of technology and the popularization of industrialization, the distance between UAVs and ordinary people's lives is getting smaller and smaller. Consumer UAVs are becoming more and more popular, and commercial UAVs such as plant protection UAVs have also been widely used. The demand for UAVs group is also increasing, and UAVs group can be widely used in various scenarios such as light shows and tactical attacks.

This design is based on the solution of the monitoring of the internal network connection status when the UAVs group is applied, and can realize real-time monitoring of the position and attitude of the UAV, and can also set the network information of a single UAV.

This article begins with an introduction to the current research status of UAV management systems and UAV networks at home and abroad, and analyzes related development trends. Then it briefly introduces the related technologies and development environment involved in the development of the UAVs group network management system. This design is mainly based on WPF (Windows Presentation Foundation), including XAML language for interface design, and C# language for implementing background logic. SQLite is used for database storage.

The focus of the article lies in the design and implementation of each function, including: loading electronic maps through GMap.NET technology to display the real-time position of each UAV in the UAVs group, and further realize the historical flight trajectory drawing of the UAV; Canvas draws a quadratic Bezier curve to display the network topology between UAVs in the cluster; through the Material Design In XAML control library, the display of UAV network settings information is more beautiful. When the user changes the settings, the information will be sent to the background server through the Socket class to update the drone settings; the display of the UAV's flight attitude is achieved through the virtual dashboard, which displays the yaw angle, pitch angle, roll angle and altitude of the UAV; The display of the historical flight trajectory and attitude of the UAV calls the historical flight information stored in the database, and displays the trajectory and real-time attitude on the electronic map at the same time.

Finally, the feasibility and reliability of the system design were verified through the individual testing of each module of the system and the overall debugging of the system, which met the requirements of system design and achieved the expected results.

Key words: UAVs group; Internet; Database; WPF; GMap.NET; Flying attitude; Network topology

目 录

1 緒論	1
1.1 研究背景和意义	1
1.2 国内外研究现状及发展趋势	2
1.2.1 国内外研究现状	2
1.2.2 发展趋势	4
1.3 本文主要研究内容和章节安排	6
1.3.1 主要研究内容	6
1.3.2 章节安排	7
2 系统开发平台及相关技术	8
2.1 引言	8
2.2 WPF 技术	8
2.3 开发环境	9
2.4 开发语言	10
2.4.1 C# 语言	10
2.4.2 XAML 语言	10
2.5 SQLite 数据库	11
2.6 Material Design In XAML 控件库	11
2.7 本章小结	12
3 无人机群网络管理系统总体设计	13
3.1 引言	13
3.2 软件功能设计	13
3.3 软件界面设计	14
3.4 本章小结	16
4 主要功能模块设计	17

4.1	引言	17
4.2	基于 GMap.NET 的电子地图实现.....	17
4.2.1	电子地图技术.....	17
4.2.2	GMap.NET 简介	19
4.2.3	电子地图加载.....	20
4.3	无人机群网络连接拓扑	22
4.3.1	无人机相对位置绘制	22
4.3.2	网络拓扑展示.....	24
4.4	单无人机网络节点设置	25
4.4.1	Socket 网络通信	25
4.4.2	单无人机网络节点信息.....	26
4.5	无人机虚拟仪表盘	27
4.5.1	虚拟仪表盘绘制	28
4.5.2	虚拟仪表盘展示	29
4.6	飞行记录保存	29
4.6.1	飞行轨迹和姿态信息保存.....	29
4.6.2	飞行轨迹和姿态信息展示	30
4.7	本章小结	31
5	系统实现和测试	32
5.1	引言	32
5.2	实时位置模块实现和测试.....	32
5.2.1	实时位置模块实现	32
5.2.2	实时位置模块测试	33
5.3	网络拓扑模块实现和测试.....	33
5.3.1	网络拓扑模块实现	33
5.3.2	网络拓扑模块测试	33
5.4	网络节点模块实现和测试.....	35

5.4.1	网络节点模块实现	35
5.4.2	网络节点模块测试	35
5.5	飞行姿态模块实现和测试.....	36
5.5.1	飞行姿态模块实现	36
5.5.2	飞行姿态模块测试	36
5.6	飞行记录模块实现和测试.....	37
5.6.1	飞行记录模块实现	37
5.6.2	飞行记录模块测试	37
5.7	本章小结	39
6	总结与展望	40
6.1	本文工作总结	40
6.2	本文不足与工作展望	41
	参考文献	42
	致谢	46
	附录 A 测试数据	47
A.1	无人机节点信息.....	47
A.2	无人机连接信息.....	48
A.3	无人机网络节点信息	49
A.4	无人机历史飞行任务索引表.....	50
A.5	无人机历史飞行任务示例.....	51

1 绪论

1.1 研究背景和意义

无人机 (Unmanned Aerial Vehicle, UAV)，是一种近年来正在快速发展的新型技术，其广泛应用于军事、公共和民生领域。随着科技的发展，无人机成本的下降和相关行业的发展，使更多普通人得以接触使用无人机，无人机摄影、表演，越来越多出现在人们的视野中。许多无人机应用场景包括但不限于公共安全、国土安全、环境监控、通信中继、农业、建筑、监视、互联网交付和货物运输（如京东和邮政正在试验的无人机货物运输）。^[1] 在当前新冠疫情流行的背景下，无人机更是在疫情防控、监测和治理中发挥了重要作用。^[2]

在新冠疫情中，可以看到众多无人机应用的场景。无人机被用来代替警察进行日常巡逻，提醒人们保持社交距离，出行携带口罩；如图 1.1 所示，无人机被用来进行环境消毒，喷洒消毒药剂；在出行被严格限制的区域，无人机还可以用来传递药品和生活用品；在人群聚集区域，无人机还可以从空中进行人群测温，及时报告疑似患病者的实时位置。在日常生活中，无人机成为众多航拍爱好者必备的摄影工具，拍出美丽的图片和视频；如图 1.2 所示，无人机还可以用于节目表演，无人机群可以在夜空中组合成各种图形，带来极佳的视觉体验。



图 1.1 无人机消毒



图 1.2 无人机表演

在工农业中，无人机也有很多应用场景。在电力行业中，无人机主要有以下几个应用：如图 1.3 所示，电网巡检过程中人机配合，通过相机和精细成像装置检查设备缺陷；通过无人机搭载的定位和成像模块，检测电网周围风险隐患；协助进行电网规划设计；通过无人机协助进行线路抢修和维护。随着智能电网、数字电网、信息通信技术的飞速发展以及无人机技术的不断突破，无人机技术在智能电网中的作用愈加重大，将成为不可或缺的任务载荷平台和自动巡检作业平台。^[3] 在农

业领域，如图 1.4 所示，同传统技术方式比较，无人机植保具有效率高、作业质量好、适应环境、节约水资源等优点，在当前农业现代化的背景下，有着广阔的应用空间。



图 1.3 无人机巡视电网



图 1.4 无人机植保

根据获得升力的机翼与机身是否产生相对运动，无人机可以分为固定翼和旋翼类型的无人机，按照旋翼机的螺旋桨数量，旋翼机可以进一步区分为单旋翼、双旋翼与多旋翼。^[4] 无人机系统主要包括无人机机体、飞行控制系统、数据通信链路以及管理系统等几个组成部分。^[5] 其中，无人机管理系统是无人机系统的重要组成部分，它担负着监控无人机飞行状态和位置网络等重要信息，同时还可以向无人机发送相关指令，调整无人机系统的状态，可以视为无人机系统的“大脑”。

在当前无人机技术飞速发展的情况下，消费级无人机日渐普及，商业场景下的无人机应用更加细化，无人机的研究成为一大热点，吸引着国内外众多科研院所和商业公司进行开发。本文参考了众多国外厂家和科研院校对无人机地面控制站的相关研究，根据实际应用中的具体需求，自行设计了一个网络管理系统，重点在于监控无人机系统的网络拓扑和网络设置信息，控制无人机网络相关设置，同时拓展了无人机飞行状态监控，无人机飞行任务回放功能。

1.2 国内外研究现状及发展趋势

1.2.1 国内外研究现状

无人机管理系统研究现状

无人机管理系统作为无人机系统的控制中心，可以对无人机的飞行姿态进行监控，接收飞行数据、展示无人机的实时位置信息、规划无人机航迹、任务回放和错误信息处置等操作。

日本千叶大学的无人机管理系统，如图 1.5 所示。该团队自主设计研发了微型自主旋翼飞机，系统实现了航点导航、轨迹跟踪、视觉导航、精确悬停以及自动起

飞和着陆。^[6] 开源地面站 Misson Planner 如图 1.6所示。它可以实时监控无人机飞行状态，以第一人称视角操作无人机，规划航点任务，使用多种地图，下载和分析飞行日志。



图 1.5 千叶大学无人机管理系统



图 1.6 开源 Misson Planner 地面站

无人机管理系统的电子地图界面可以实时展示无人机的位置信息、规划飞行路线、展示历史航迹。当前大多数无人机都携带 GPS 模块，可以实现全球定位。通过无线通讯链路，实时传输自身位置信息。无线通信方面，主流的解决方案包含数传电台、TCP 通信、UDP 通信等。使用较多的地图软件主要为 Google Maps，在中国大陆地区，Google Maps 使用体验较差，可以考虑 Bing Maps、高德地图、百度地图、腾讯地图等。为了实现航迹储存，可以使用 SQL Server、MySQL 等数据库软件。

当前国内较为知名的无人机厂商有深圳的大疆创新科技有限公司、北京的零度智控智能科技有限公司与广州的极飞科技股份有限公司。大疆创新在消费级无人机方面有着非常大的市场占有率，个人无人机摄影的主力品牌主要是大疆。零度智控主要定位于智能飞行器产品和智能无人机整体解决方案供应商。极飞科技在农业植保用无人机方向有较强的技术竞争力。

无人机网络研究现状

机载网络是一种网络物理系统，其中物理组件和网络组件之间存在激烈的相互作用。^[7] Naser Hossein Motlagh 在其论文^[8] 中讨论了五种无人机通信网络：A. Node-to-Node Communication, B. Mesh Networking, C. Delay Tolerant Network, D. MANET/VANET Data Routing, E. Flying Ad-Hoc Network. 其中 FANET(飞行自组织网络)可以看作是 MANET (移动自组织网络) 和 VANET (车载自组织网络) 的一种特殊形式。^[9]

论文^[10] 主要探究了通过研究无人机的部署和移动提高基于地面的无线自组织网络的连通性，提出了一些自适应方案以改进不同网络连接的连通性。

论文^[9]研究了FANET(飞行自组织网络)的应用场景,设计特点,设计考虑,物理层设计,访问控制层设计,网络层和传输层设计,并讨论了相关的FANET实验模拟。

论文^[8]展望了无人机在物联网方面多样的应用场景,揭示了无人机在物联网的巨大潜力。论文^[1]设计实现了一个无人机物联网平台,研究了无人机应用时可能遇到的各种问题。论文^[11]研究了多无人机作为空中基站从相关物联网设备收集数据的高效部署和移动性,提出了一种新框架,可以用于优化无人机3D放置和移动性,控制上行链路发射功率和设备—无人机关联,节约了能耗。

论文^[12]提出了一种通过调整慢启动阈值来提高无人机网络传输性能的算法,可以恢复网络传输中不必要的速度下降。

论文^[13]设计了相关的蜂群无人机数据链自组网协议,包括MAC协议,路由协议等,模拟仿真了集群数据链的数据收发。

论文^[14]介绍了两类常见的大规模无人机自组网分层架构——分簇与联盟,分析了各类算法的应用场景,对比研究了两类分层架构。论文^[15]研究了基于粒子群算法的无人机网络拓扑控制机制,设计实现了基于DQN的无人机网络分簇算法,并讨论了一种无人机中继选择机制。

1.2.2 发展趋势

无人机管理系统发展趋势

结合众多商业和开源无人机管理系统,可以发现无人机管理系统普遍具有以下基本功能:

1. 都可以展示机载传感器的参数
2. 都展示了电子地图
3. 都可以定位无人机
4. 都可以设置参数

在当前的应用背景下,无人机管理系统还有很多可以继续发展的方向和目标。如何更好地发挥出无人机的硬件潜力,降低开发成本,完善丰富无人机的使用体验和应用场景,成为无人机管理系统软件研究的重点,对于无人机的普及和应用具有重要意义。无人机管理系统未来的发展趋势主要有:

1. 发展多功能通用型管理系统。目前做的较好的无人机管理系统都是商业公司

针对某型号的无人机专门设计开发的，大都无法适用于其他型号的无人机。无人机管理系统不同商家之间也各自为战。针对新型号的无人机，其管理系统可能需要重新开发，这大大增加了软件开发成本，消耗人力。一款多功能通用型无人机将可以有效提高商家和个人开发者的工作效率。

2. 发展具备高效通信功能的管理系统。随着无线网络技术的发展，5G 成为工业界最新的应用技术。如果可以将 5G 通信应用于无人机管理系统和无人机机体之间的通信，将大大缩短通信延迟，提高无人机群之间的网络通信效率。在某些极端应用环境下，管理系统还应具备同无人机进行卫星通信的能力，以拓展无人机应用的场景和距离。
3. 发展无人机群式的管理系统。随着无人机的普及，工业制造成本大大降低。无人机群编组飞行在军用和民用领域都得到了广泛应用，这就给无人机管理系统提出了新的要求，希望可以在一台终端管理多台设备，提升无人机群的协作能力，增强无人机的作业效果。
4. 发展可扩展性高的管理系统。根据用户使用无人机的场景不同，对于无人机的功能要求也不尽相同。这就需要我们在原有管理系统的基础，可以模块化地增加新的功能，这样当不同用户对管理系统进行二次开发时，可以节省时间，提升开发效率。
5. 发展结合 AR 的管理系统。通过 AR(增强现实) 技术，可以让无人机操作人员具有身临其境的“飞行体验”，实时了解周围环境，帮助操作人员做出正确决策。还可以让操作人员足不出户体验飞行的感觉。
6. 发展结合 AI 的管理系统。当前，人工智能在无人车应用中已经进入实际测试，如果可以将人工智能拓展到无人机方面，那么无人机就可以达到真正的“无人”，工业化制造的无人机将大量应用，将人类从相关工作中解脱出来，无人机将按照预设的任务自动决策。甚至有朝一日，将可以实现无人驾驶的客机飞翔在天空中。

无人机网络发展趋势

有关无人机网络的研究是当前无人机研究的另一个热点问题，既有基于单个无人机的网络优化，也有基于大规模无人机集群的组网通信问题。

无人机的网络通信有以下几个趋势^[16]:

1. 安全性和可靠性。无人机作为嵌入式设备，其安全性仍为通信最主要的发展

方向，如何实现无人机同有人终端之间的可靠通信，如何在无人机集群内部构建可信的网络，如何在多种电磁干扰的环境下仍然维持通信的质量。这些都需要进一步研究。

2. 小型化和低功耗。无人机在大多情况下不需要很大的体积，从小型设备的角度出发，其通信模块应向着小型化和低功耗方向发展，节省的空间和能量可以用于加载其他飞行任务模块和增强飞行半径，有助于提高无人机的任务拓展性。
3. 通用化和标准化。通信模块的通用化有助于实现多种无人机之间的相互通信，将拓展无人机群的快速组网能力。标准化的网络通信将有助于相关飞行协议的建构，极大节约了无人机网络通信二次开发的工作量，提高了网络通信设计的适用范围。
4. 智能化趋势。无人机网络单机通信和组网通信对于智能化都提出了各自的要求，单个无人机需要适应多种通信环境，及时根据本机的通信状态和群体的通信质量智能化地自动改进通信相关设置。集群组网智能化将有助于集群网络通信质量的提升，动态根据设备的增减调整内部通信链路。

1.3 本文主要研究内容和章节安排

1.3.1 主要研究内容

在无人机网络管理系统开发过程中，本文完成的相关工作包括：

1. 设计实现对用户友好的操作界面；
2. 设计通信协议，实现网络管理系统与后台服务器之间的通信；
3. 处理无人机数据，存储飞行日志，相关无人机群网络信息；
4. 实时展示无人机群网络拓扑和位置信息；
5. 展示无人机群节点网络设置信息并实现设置修改；
6. 实时展示无人机姿态信息，构建虚拟仪表盘；
7. 读取飞行日志，展示历史航迹和飞行姿态。

本设计基于实际工程需求，专门用于无人机群的网络管理，开发基于 Windows 平台下，使用 WPF 实现相关设计，用户界面使用 XAML 布局，为了提高系统的友好度，使用了 MaterialDesign 控件库，后台使用 C# 关联程序逻辑。数据库使用 SQLite 暂存网络拓扑和飞行数据，系统从数据库加载相关内容展示给用户。使用

GMap.NET 展示电子地图，通过加载高德地图展示无人机位置信息。构造无人机群网络拓扑时，绘制了带文本框的二次贝塞尔曲线，以展示不同节点之间的链接情况。管理系统内对网络节点的设置可以同步回传到后台，更新无人机设置。利用 GDI+ 绘制虚拟仪表盘，以实时展示无人机姿态信息。通过 SQLite 中的飞行数据，回放历史飞行任务。

1.3.2 章节安排

第一章为绪论。绪论部分主要阐述了无人机群网络管理系统的研究背景和意义，介绍了国内外无人机管理系统的发展现状，并对无人机管理系统的发展趋势进行了分析。

第二章介绍了无人机群网络管理系统的开发平台和相关技术，包括 WPF 框架，visual studio 开发环境，C# 后台开发语言，XAML 界面布局，SQLite 数据库，以及 MaterialDesign 控件库。

第三章介绍了无人机系统的总体设计方案，包含功能模块的划分与设计以及软件界面的设计。

第四章重点介绍了无人机群网络管理系统关键功能的设计实现。主要内容包括：利用 GMap.NET 实现电子地图展示、通过绘制带文本框的二次贝塞尔曲线展示无人机群网络拓扑、单个无人机节点网络信息的展示和设置、GDI+ 绘制虚拟仪表盘实现飞行姿态展示、历史飞行任务的保存和回放、系统内外数据交互。

第五章系统功能的应用和测试。对无人机群网络管理系统进行了整体综合测试，对不同的模块进行单独测试，验证本设计的可行性，验收无人机群网络管理程序。

第六章对本文所做的工作进行了总结，并对本网络管理系统进一步的设计研究提出了展望。

2 系统开发平台及相关技术

2.1 引言

关于无人机地面管理系统，目前看来存在多种开发路线，既有基于 MFC^[17]、Labview^[18] 开发的，也有基于 Qt^[19]、Winform^[20] 等技术开发的，同时还有针对移动平台 Android^[21] 开发地无人机管理系统，其中有些软件在不同平台下因为分辨率不同导致模糊，也有些软件界面风格简单朴素，风格过于早期，对用户不够友好。所以本文采用了 WPF 框架进行开发，将前端 UI 的不同元素和后端控制逻辑分离；使用 C# 编写后台逻辑，XAML 实现页面布局；为了保存相关数据，使用了轻型关系数据库 SQLite；同时为了实现更加友好人性化的界面，引入了 MaterialDesign 控件库，让整个程序看起来更美观。

2.2 WPF 技术

WPF 为 Windows Presentation Foundation 的首字母缩写，WPF 统一了 Windows 创建、显示以及操作文档、媒体和用户界面 (UI) 的方式，使开发人员和设计人员可以创造出更好的应用视觉效果、丰富的用户体验。WPF 作为微软的新一代图形系统，其运行在.NET Framework 架构下，为 UI、图形、媒体和文档提供了统一的设计和应用方法，基于 DirectX 引擎技术，WPF 可以带来全新的 2D/3D 界面。WPF 相对于 Windows 客户端的开发来说，做出了巨大的跨越，它提供了丰富多彩的.NET UI 框架，集成了矢量图形，丰富的流动文字支持 flow text support，3D 视觉效果和强大无比的控件模型框架。^[22]

WPF 技术相对于传统 Windows 应用程序开发有以下优势：

1. WPF 可以将软件界面设计和逻辑控制分开，程序前台界面设计使用 XAML 进行布局、设计，后台通过 C# 高级语言进行面向对象的逻辑控制，使得设计程序时可以分开考虑页面布局和数据的处理，有利于模块化设计整个程序。
2. WPF 可以使界面设计更加美观、更加丰富多彩，WPF 可以使页面布局更加合理，控件的过渡更加自然，而且 WPF 还有丰富的第三方控件库可供使用，开发人员可以在控件库的基础上获得更好地设计效果。
3. WPF 可以利用.NET 平台上的网络通信功能实现收发数据。.NET 平台提供了

丰富的功能类库，无人机网络管理系统需要和后台进行网络通信时，可以直接调用相关的网络通信类库，节约了开发时间，提高了开发效率。

4. WPF 可以充分利用 GPU 资源。传统的开发技术偏向于使用 CPU 进行数据图像处理，占用了过多的 CPU 资源，影响其运算速度。利用 WPF 技术，可以充分利用计算机的 GPU 处理图形界面，节省出的 CPU 资源可进行数据处理，有效提高了 CPU 运算速度，保证了系统的流畅性。^[23]

2.3 开发环境

本文所设计的无人机群网络管理系统运行在 Windows 10 操作系统平台，集成开发环境为 Visual Studio Community 2019，主要使用 Visual C# 语言进行开发，本文的网络管理系统的目地框架为.NET Framework 4.7.2。开发环境如图 2.1 所示。.NET Framework，简称.NET 框架，它是一个多语言组件开发和执行环境，以公共语言运行库 (CLR: Common Language Runtime) 为基础，提供了一个跨语言的统一编程环境。使用.NET 框架具有以下优势：

1. .NET 框架支持生成和运行下一代应用程序，具有标准的面向对象开发环境、代码安全性以及强大的开发工具等特点，成为主流的 Windows 软件开发平台。
^[24]
2. .NET 框架支持多种编程语言，为开发人员提供了一套统一的、面向对象的、层次化的、可拓展的类库，减少了开发者的编程任务，增强了开发的灵活性。
^[25]
3. .NET 框架使得开发人员可以同时进行 Windows 应用软件开发和 Web 服务的建构、配置和运行，大量的类库减少了开发人员编写底层代码的时间，提高了开发效率。

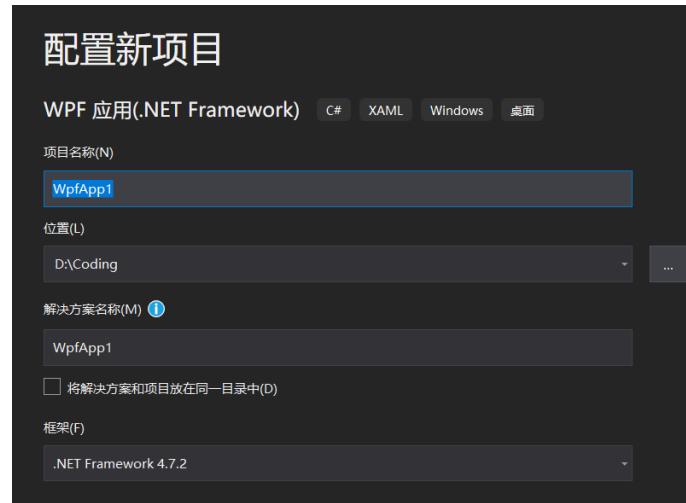


图 2.1 软件开发环境配置

2.4 开发语言

2.4.1 C# 语言

C# 是由微软公司发布，从 C 和 C++ 衍生而来的一种面向对象编程语言，是一种运行于.NET Framework 和.NET Core 上面的高级程序设计语言。C# 和 Java 有相似之处也有不同之处，它类似 Java 具有单一继承、接口、语法也基本相同，先编译成中间代码再运行的过程更有异曲同工之妙；不同点在于它是与 COM（组件对象模型）直接集成的。

C# 具有强大的操作能力、创新的语言特性、对面向对象组件编程的便捷支持以及优雅的语法风格等特性，使得它成为.NET 开发的首选语言。因为其源自 C 和 C++，对于 C/C++ 程序员来说，学习压力较小，可以很快从 C/C++ 转向 C# 开发。

2.4.2 XAML 语言

XAML 是 eXtensible Application Markup Language(中文名：可扩展应用程序标记语言) 的缩写。这是一种 Microsoft 公司为构建应用程序用户界面 (UI) 而创造的新的描述性语言。XAML 是一种解析性语言，但也可以被编译，优点在于简化编程上的用户创建过程。

WPF 使用 XAML 来构建美观的用户界面，使用 XAML 可以定义控件、文本、图像、形状、动画等各种元素。如果需要向应用程序添加运行时逻辑，就需要使用 C# 来添加相应代码。这种 UI 和逻辑代码分离的设计方式，使得开发人员和设计人员可以分开完成各自的工作，不会拖累项目进度。

2.5 SQLite 数据库

SQLite，是一种开源轻型关系型数据库管理系统。数据库是基于文件的，便于开发人员迁移，无需安装和配置。SQLite 还有很好的跨平台特性，可以在多种常见的操作系统之上运行，而且也可以用于嵌入式操作系统。SQLite 是一种无服务器的数据库，不需要服务器进程或者操作系统来进行操作。SQLite 为大多数编程语言提供了 API(应用程序编程接口)，对于使用 WPF 编程的开发人员来说，使用方便。

SQLite 的数据库是储存在单一磁盘文件的完整数据库，可以在不同字节顺序的机器间自由共享，比一些流行数据的大部分操作都要快一些。缺点在于一般处理小到中型的数据存储，并不适用高并发高流量的应用。

2.6 Material Design In XAML 控件库

Material Design In XAML 是一个开源的，受欢迎的 WPF GUI(图形用户界面)库，该控件库提供了大多数 WPF 控件的样式和变体，还有很多拓展控件用于实现”Material Design”风格。在设计 WPF 应用程序时，使用该控件库，仅需要 XAML 即可构建现代化的、用户友好型的界面，让应用的界面更加美观和人性化。该控件库的使用如图 2.2 所示。

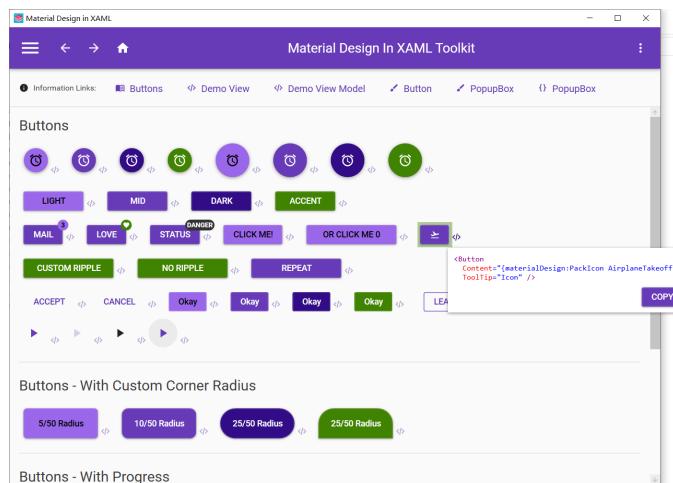


图 2.2 Button 样例

2.7 本章小结

本章首先介绍了 WPF 技术，说明了其作为新的 Windows 开发技术的优势。然后介绍了.NET 平台及其特点，接着说明了在该平台上使用的两种开发语言：C# 语言和 XAML 语言，并简单介绍了它们的特性。然后说明了本文设计的无人机群网络管理系统的两个特色：SQLite 数据库和 Material Design In XAML 控件库。

3 无人机群网络管理系统总体设计

3.1 引言

本设计主要包含前端软件界面设计和后端软件功能设计，前端通过 XAML 语言进行界面布局和 UI 绘制，后端通过 C# 实现相关的操作逻辑和数据操作。

本系统主要与后台服务器进行数据交流，后台服务器将直接和无人机进行交流，数据链路如下图 3.1 所示：

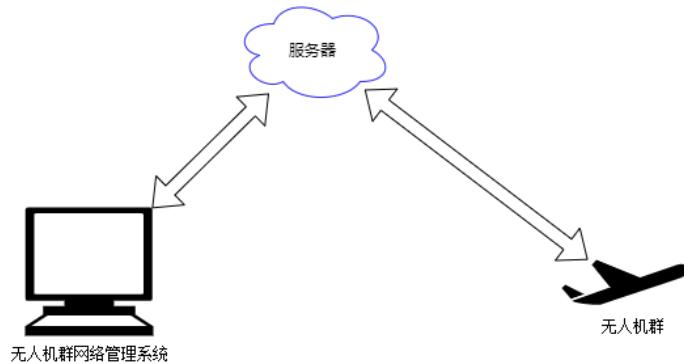


图 3.1 无人机群网络管理系统数据链路

3.2 软件功能设计

本文所设计的无人机群网络管理系统的主要功能模块如下图 3.2 所示：

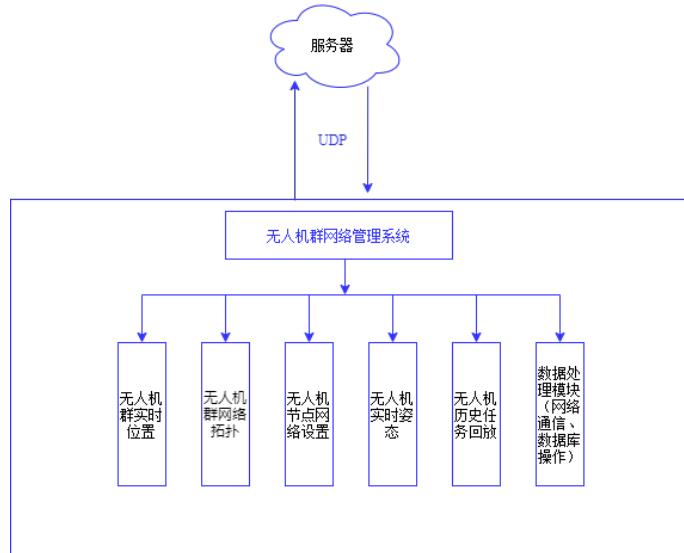


图 3.2 无人机群网络管理系统主要功能模块

软件的具体功能如下所示：

1. 无人机群实时位置。通过定时刷新数据库获取无人机群内各无人机的经纬度信息，同时刷新电子地图，在电子地图上添加无人机。
2. 无人机群网络拓扑。本功能展示了无人机群内个无人机之间的连接状态，根据连接质量不同使用不同颜色的线连接无人机，无人机之间的相对位置由实际经纬度信息转化而来。
3. 无人机节点网络设置。通过本功能可以方便用户查看机群内任一无人机的网络节点信息，可以方便用户进行无线设置和网络设置，设置的结果可以通过网络通信传值服务器。
4. 无人机实施姿态。通过绘制虚拟仪表盘，展示无人机的三个姿态角，动态展示无人机的实时飞行姿态。
5. 无人机历史任务回放。通过调取系统内无人机的历史飞行数据，回放过往飞行任务中无人机的实时位置和实时姿态。
6. 数据处理。主要包含网络通信功能和数据库操作功能，通过 Socket 通信从服务器获取无人机群的相关信息，维护系统数据库表。

本系统还设计了软件登陆功能，通过对比系统数据库表内存储的用户信息，向用户提供登录注册功能。

3.3 软件界面设计

软件界面布局主要依赖了 WPF 项目中基于 XAML 的页面布局，使用 Grid 将页面划分为不同网格进行初始布局，在具体控件的排列上使用了 StackPanel，当需要向页面动态添加控件时，使用了 Canvas(画布)。

软件登陆界面如图 3.3 所示。

该界面主要包含输入文本区和操作按钮。输入区可以获取用户输入的账户和密码，按钮区则根据用户点击不同操作按钮实现不同的逻辑。

该程序的主界面如图 3.4 所示。

该界面主要包含 A、B 区域，A 区实现不同功能模块之间的跳转，B 区则可以根据不同的功能模块设计实现不同的控件布局。在章节 5 我们将进行详细的介绍。

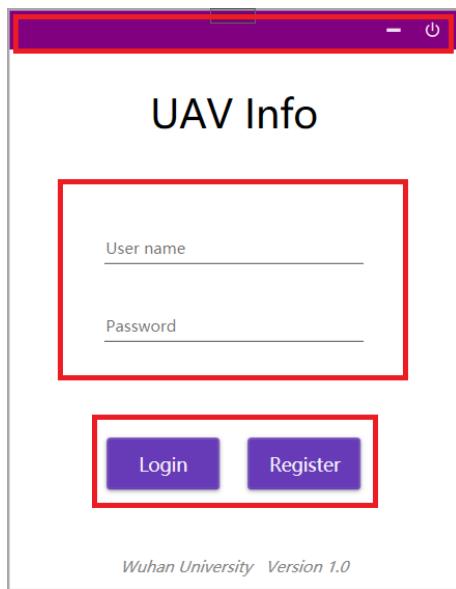


图 3.3 登陆界面展示

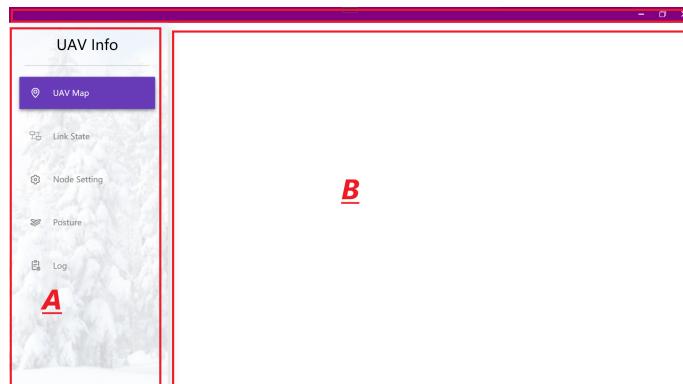


图 3.4 主界面展示

3.4 本章小结

本章节简要介绍了系统的功能特性和界面的整体设计，下一章我们将专注于功能设计的实现细节，解决开发过程中可能遇到的技术难题，在章节 5我们将展示各界面实现后的效果。

4 主要功能模块设计

4.1 引言

前面的内容我们介绍了本无人机群网络管理系统的整体功能设计和界面设计，本章我们将专注于相关功能的具体实现细节，包含电子地图加载，机群网络拓扑的绘制，无人机节点网络信息设置，虚拟仪表盘绘制和历史飞行任务回放。

4.2 基于 GMap.NET 的电子地图实现

无人机管理系统中常常涉及电子地图的应用，本无人机群网络管理系统在无人机位置展示时也使用了电子地图。电子地图是一种 GIS(Geographical Information System, 地理信息系统)技术，常见的 GIS 组件有 ArcGIS、MapX、SuperMap 等。MapX 是 MapInfo 公司推出的，具有强大地图分析功能的 ActiveX 控件^[26]。SuperMap 控件功能有限，开发资料少^[27]。本文中的电子地图采用 GMap.NET 控件实现，它是一个强大、跨平台、免费、开源的.NET 控件，支持基于瓦片技术加载 Google 地图、Bing Maps、百度、高德等地图。

4.2.1 电子地图技术

当前，网络地图服务和应用已经成为日常生活中必不可少的一部分，极大方便了人们的出行和生活。日常经常见到的谷歌地图 (Google Maps)、百度地图、高德地图、必应地图等，人们只需要打开网站或应用，就可以浏览全球各地任何地点，了解周围的建筑、街道，为人们规划出行提供了便利。地球作为一个实际存在的空间体，是不可展的，无法将其直接展开成一个连续的、没有裂痕和褶皱的平面，需要使用地图投影的方法，将地球面上的经纬线网、多种底物和现象转换到连续的地图平面上去^[28]。网络地图服务所使用的地图投影被称为 Web 墨卡托坐标系 (Web Mercator, 见图 4.1)。Web Mercator 最早是由谷歌在其网络地图服务 Google Maps 中应用的，所以也被叫做“Google Web Mercator”。赵永辉认为 Web 墨卡托投影是基于墨卡托投影的近似坐标系统，均是采用正轴切圆柱进行投影，主要区别在于 Web 墨卡托投影是将地球视作球体而非椭球体^[29]。

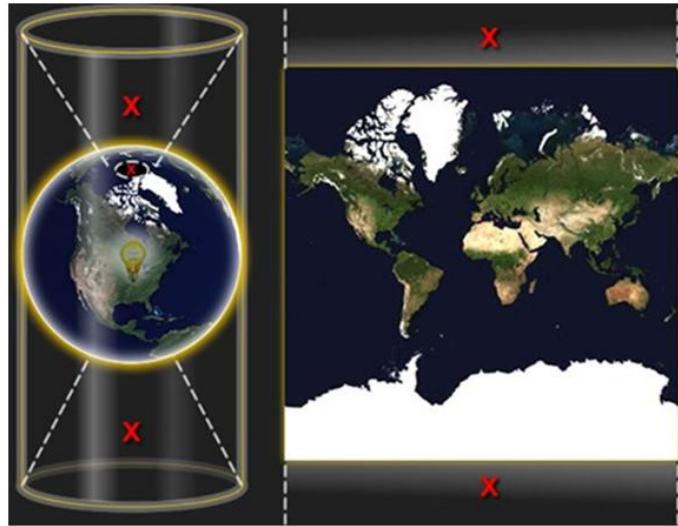


图 4.1 Web 墨卡托示意图

Web 墨卡托投影的基准面 WGS84，故又称为”WGS 84 Web Mercator”。在进行 Web 墨卡托投影时，将表示地球的参考椭球体近似的作为正球体处理，以赤道作为标准纬线（横坐标轴），本初子午线作为中央经线（纵坐标轴），两者交点为坐标原点，向北向东为正，向南向西为负，建立直角坐标系，利用公式（4.1）可以将经纬度坐标转换为直角平面坐标^[30]：

$$\begin{cases} x = R_e \times \theta \\ y = R_e \times \ln(\tan(\frac{\pi}{4} + \frac{\xi}{2})) \end{cases} \quad (4.1)$$

其中： R_e 为地球赤道的平均半径，赤道的周长为 $2\pi R_e$ ， θ 为经度， ξ 为纬度。投影坐标系中横坐标轴的取值范围为： $[-\pi R_e, \pi R_e]$ ，当 ξ 趋近于 90° 时，纵坐标趋于无穷大，即 Web 墨卡托投影在地球两极处投影至无穷远处，考虑到两极地区大多都是无人居住的区域，故可以将纵坐标的取值范围也限定为 $[-\pi R_e, \pi R_e]$ ，对应的地理坐标为南北纬 85.05° 。这样处理以后，整幅地图为正方形，便于建立地图影像金字塔系统，降低了坐标之间互相转换的算法复杂度，同时还去掉了大多数的无人区域，将注意力集中在有人居住区域。

以 Google Maps 为例，该网络地图服务就采用了影像金字塔缓存技术，采用四叉树剖分方式，将全球地图分层、分块进行切分和重采样，分级存储不同分辨率下的切片文件及和。分层级别以 0 为首层编号，不同分层级别以 1 递增，最大的分层数为 22 层。地图的缓存采用地图瓦片技术，地图瓦片技术可以将设置好坐标范围和比例尺的地图，切成若干行和列的指定尺寸的正方形图片，保存为图像文件，

存储到目录系统中或数据库系统中，这些地图缓存将形成影像金字塔，地图切片所获得的地图切片也叫瓦片 (Tile)^[31]。Google Maps 中，层级为 0 时，整个地图投影在一个 256px×256px 的影像上，缩放级别为 1 时，将原来的 256px×256px 切分成为 4 张 256px×256px 的影像，新影像的比例尺为原来的 2 倍，分辨率是原来的 4 倍。以此类推，当缩放级别为 n 时，整个地球应该由 4^n 张瓦片地图构成，瓦片数量呈几何级增长，形成影像金字塔（如图 4.2）。金字塔从顶层向底层运动的过程中，分辨率越来越高，从用户的角度看来，地图越来越清晰，可以展示更多的地图细节。

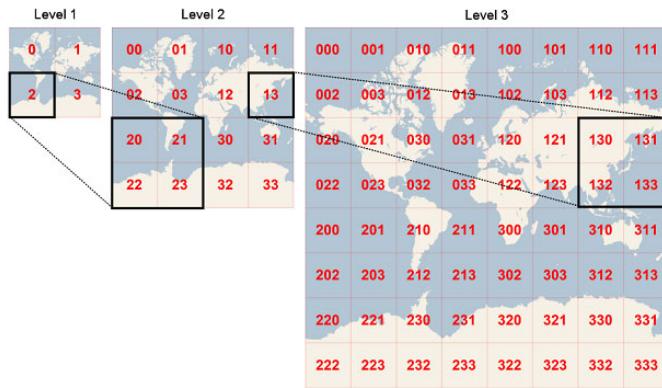


图 4.2 瓦片数据组织模型

4.2.2 GMap.NET 简介

GMap.NET 是一款优秀的开源.NET 地图控件，可以在 WindowsForms 和 WPF 应用程序中加载各种地图，提供多样的地图服务。Gmap.NET 的代码结构遵循了低耦合-高内聚的原则，各模块之间的联系是基于接口编程的^[32]。该控件基于 HTTP 协议加载各个在线地图服务商的电子地图，通过解析各个电子地图服务提供商关于地图服务的 URL(Uniform Resource Locator, 统一资源定位器)，传入相对应的参数从地图服务器获得地图瓦片，然后在用户端将瓦片拼接形成电子地图。在地图服务器中，每个地图瓦片都有唯一的 URL，不同的在线地图服务供应商的地图瓦片的 URL 是不同的^[33]。鉴于在国内无法使用 Google Maps，故本文采用的是高德地图。高德地图的某个瓦片地址如下：

```
http://webrd01.is.autonavi.com/appmaptile?lang=zh_cn&size=1&scale=8&style=8&x=855&y=417&z=10
```

表 4.1 地图瓦片 URL 参数表

参数	含义
webrd01	服务器名, 取值范围 webrd01~webrd04
lang	语言类型, zh_cn 为中文, en 为英文
style	地图类型, 6 是卫星地图, 7 为简图, 8 为详图
x	瓦片的 x 坐标
y	瓦片的 y 坐标
z	缩放等级 zoom, PC 端取值范围为 3~18

其中最主要的参数为 x,y,z, 根据这三个参数可以唯一确定瓦片的 URL^[34]。GMap.NET 中不同地图图源接口位于 GMap.NET.Core 文件夹下, 支持 ArcGIS、BingMap、HereMap、GoogleMap、OpenStreetMap、YahooMap 等, 但是因为服务原因, 这些地图在国内地区的服务质量不如国内地图服务提供商, 故改用国内高德地图。图源接口的实现主要是由 GMapProvider 和 GMapProviderBase 来完成的^[20], 通过改写相关类可以实现加载高德地图的服务。

4.2.3 电子地图加载

WPF 工程使用 GMap.NET 首先需要安装相关的 DLL(Dynamic Link Library, 动态链接库文件), 通过 Visual Studio 2019 的 NuGet, 可以轻松简化安装过程。使用 GMap 需要安装 GMap.NET.Core 以及 GMap.NET.WinPresentation 两个引用。通过自定义类 MapControl 继承自 GMapControl, 以实例化地图控件。通过继承类 GMapProvider 来编写所需的高德地图应用接口。关键点在于生成高德地图的瓦片 URL, 以从地图服务器获取相关瓦片。

在相关的地图界面, 使用 XAML 语言将 MapControl 控件引入界面, 将控件的相关性质留给 C# 语言从后台进行定义。地图界面如图 4.3 所示。

```
<map:MapControl x:Name="mapControl1" MinZoom="3"
    MaxZoom="18" />
```

GMap.NET 中常用的类主要包含:

PointLatLng: 以经度和纬度表示的点, 该类主要包含两个属性, Lat 表示纬度 (Latitude), 是一个 double 类型的数据, Lng 表示经度 (Longitude), 也是一个 double 类型的数据^[35]。

GMapMarker: 地图标志, 主要包含一个位置信息和一个形状, 位置类型为 PointLatLng 类型, 本设计中使用无人机的位置, 形状可以通过 BitmapImage 类来

定义。

关于控件 mapControl 主要定义了以下几个性质：

- mapControl.MapProvider = Map.AMapProvider.Instance; //AMapProvider 为自定义的高德地图接口
- GMaps.Instance.Mode = AccessMode.ServerAndCache; //地图加载方式为服务器和缓存加载并存
- mapControl.Zoom = 16; //缩放等级定为 16
- mapControl.ShowCenter = false; //不显示地图中心十字点
- mapControl.DragButton = MouseButton.Left; //左键拖拽地图
- mapControl.Position = new PointLatLng(30.527783115600307, 114.3613009682781); //将地图中心点定位于武汉大学信息学部操场
- mapControl.MouseRightButtonDown += new MouseButtonEventHandler(Map_MouseDown); //检测地图中的无人机图标是否被按下



图 4.3 地图加载展示

当地图加载模式为 ServerAndCache 时，GMap.NET 在加载地图瓦片的同时会缓存到本地文件夹内，以供离线状态下使用。通过 SQLiteStudio 打开缓存文件

Data.gmdb 可以查看全部缓存瓦片的信息(如图 4.4 所示)以及瓦片图片(如图 4.5 所示)。

	id	X	Y	Zoom	Type	CacheTime
1	1	53585	26927	16	1907254936	2021-03-29 15:45:34.1891299
2	2	53587	26927	16	1907254936	2021-03-29 15:45:34.5576408
3	3	53586	26928	16	1907254936	2021-03-29 15:45:34.9156894
4	4	53586	26927	16	1907254936	2021-03-29 15:45:35.2705282
5	5	53586	26926	16	1907254936	2021-03-29 15:45:35.6217844
6	6	53585	26928	16	1907254936	2021-03-29 15:45:35.987081
7	7	53587	26928	16	1907254936	2021-03-29 15:45:36.3367002
8	8	53585	26926	16	1907254936	2021-03-29 15:45:36.7080033
9	9	53586	26925	16	1907254936	2021-03-29 15:45:37.0718079
10	10	53588	26927	16	1907254936	2021-03-29 15:45:37.4607953
11	11	53587	26926	16	1907254936	2021-03-29 15:45:37.840219
12	12	53584	26927	16	1907254936	2021-03-29 15:45:38.2223733
13	13	53584	26928	16	1907254936	2021-03-29 15:45:38.5876803
14	14	53586	26929	16	1907254936	2021-03-29 15:45:38.9415582
15	15	53588	26928	16	1907254936	2021-03-29 15:45:39.2888848
16	16	53587	26925	16	1907254936	2021-03-29 15:45:39.6451964
17	17	53587	26929	16	1907254936	2021-03-29 15:45:40.00004039
18	18	53584	26926	16	1907254936	2021-03-29 15:45:40.3552434
19	19	53585	26929	16	1907254936	2021-03-29 15:45:40.7047446
20	20	53588	26926	16	1907254936	2021-03-29 15:45:41.05927
21	21	53585	26925	16	1907254936	2021-03-29 15:45:41.4000345
22	22	53584	26929	16	1907254936	2021-03-29 15:45:41.7469401
23	23	53588	26929	16	1907254936	2021-03-29 15:45:42.0976158
24	24	53584	26925	16	1907254936	2021-03-29 15:45:42.4586527
25	25	53588	26925	16	1907254936	2021-03-29 15:45:42.8165185
26	26	53589	26929	16	1907254936	2021-03-29 15:45:43.1761591

图 4.4 缓存瓦片信息



图 4.5 瓦片样例

判断地图中的无人机是否被点击使用了 WPF 特有的碰撞检测。主要代码为：

```
PointHitTestParameters parameters = new
    PointHitTestParameters(pt);           // (1)
VisualTreeHelper.HitTest(mapControl, null,
    HitTestCallback, parameters);       // (2)
```

语句(1)用于将 mapControl 控件被点击的位置转化为进行命中测试的参数。语句(2) VisualTreeHelper.HitTest 是主要的检测方法, HitTestCallback 为命中后的回调函数, 在这个函数中, 我们将 GMapMarker 的参数作为不同页面之间传递的参数。

4.3 无人机群网络连接拓扑

本站的创新点在于将机群网络的连接拓扑形象地展现给管理人员, 以便用户了解机群内各个无人机之间网络连接地状态和质量。

4.3.1 无人机相对位置绘制

本设计在从服务器端获取无人机群内各无人机的信息后, 会将位置相关信息和网络连接相关信息存入数据库, 当用户查询机群网络连接拓扑信息时, 将从数据库读取相关的数据, 进而可以绘制直观的连接拓扑图。

从数据库读取时, 我们可以得到无人机的经纬度这一绝对位置, 为了在画布(Canvas) 中绘制无人机群内各无人机的相对位置, 需要先将无人机绝对坐标转换为相对坐标。为了实现这一功能, 设计了以下算法, 坐标转换公式如(4.2)所示:

$$\begin{cases} X = Mid_Width + \lfloor \frac{Lon-Mid_Lon}{Lon_Len} \times Width + \frac{1}{2} \rfloor \\ Y = Mid_Width - \lfloor \frac{Lat-Mid_Lat}{Lat_Len} \times Width + \frac{1}{2} \rfloor \end{cases} \quad (4.2)$$

在本设计中，Canvas 被设为 800px×800px，Mid_Width 为 Canvas 中间点，此处为 400，Lon 为无人机的经度 (Longitude)，Mid_Lon 为无人机群中经度最大值和最小值之和的 $\frac{1}{2}$ ，Lon_Len 为无人机群中经度的最大值与最小值之差，Width 为无人机群网络拓扑图的宽度，为了留出空白部分方便绘制无人机图标，故本设计将此值设为 600。纬度转换公式类似，两个公式的不同之处在于：纬度越高，对应的 Y 坐标值越小；经度越大，对应的 X 坐标值越大。

此处的无人机展示通过向 Canvas 添加 Button(按钮) 实现，Button 的形状和性质通过样式 (Styles) 和模板 (Templates) 实现，主要 XAML 语言如下所示：

```
<Window.Resources>
    <Style x:Key="UavBtn" TargetType="Button">
        <Setter Property="Width" Value="40"/>
        <Setter Property="Height" Value="40"/>
        <EventSetter Event="Click" Handler="UAV_Button_Click"/>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="Button">
                    <Border
                        Width="{TemplateBinding Width}"
                        Height="{TemplateBinding Height}">
                        <Image x:Name="Img"
                            VerticalAlignment="Center"
                            HorizontalAlignment="Center"
                            Source="uav.png"
                            Stretch="UniformToFill"/>
                    </Border>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
</Window.Resources>
```

通过样式设置了按钮的长宽属性，通过模板，设置 Button 的背景图片为无人机。设置后的 Button 按钮如图 4.6 所示。

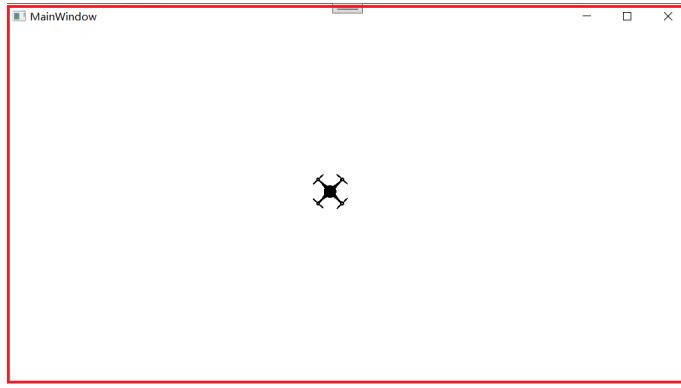


图 4.6 模板改写后的 Button

4.3.2 网络拓扑展示

无人机群内两两之间的连接质量和连通性根据具体场景会产生差异，所以将无人机之间的网络连接设计为单向。不同的网络连接质量使用不同颜色来表示，可以直观展现无人机群内的网络拓扑情况。

单条网络连接的绘制通过在 Canvas(画布) 上添加二次贝塞尔曲线 (Quadratic Bézier curve) 实现。System.Windows.Media.QuadraticBezierSegment 是.NET 中包含的类，它可以在两点之间创建二次贝塞尔曲线，这个类的实例有两个参数：控制点和终点。通过自定义类，调用 QuadraticBezierSegment 来实现绘制连接的目的。

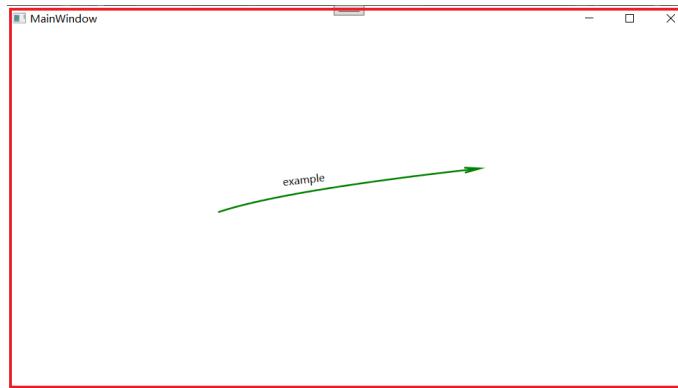


图 4.7 带文字的二次贝塞尔曲线示例

本文所设计的带文字的二次贝塞尔曲线如图 4.7 所示，该曲线的主要参数包含以下几个：(1)StartPoint: 出发点，(2)EndPoint: 终点，(3)ControlPoint: 控制点，(4)Stroke: 画笔。在小节 4.3.1 中，我们已经获取了无人机在画布上的相对位置，此时只需要将起点和终点对应网络连接的起终点即可，二次贝塞尔曲线的关键控制点可以根据起终点的无人机坐标计算得出。

4.4 单无人机网络节点设置

为了让用户可以更准确地了解无人机群内各无人机的网络信息，本功能的设计是为了可以将任意一个无人机的网络节点信息展现给用户，用户可以从本页面了解无人机的网络相关信息，同时还可以人为加以设置相关信息，修改后的无人机设置将经由网络通信传递给服务器完成修改。

4.4.1 Socket 网络通信

TCP/IP 的运输层主要包含两个不同性质的运输协议，即面向连接的 TCP 协议 (Transmission Control Protocol, 传输控制协议) 和无连接的 UDP 协议 (User Datagram Protocol, 用户数据报协议)。TCP 是面向连接的，进行网络通信前需要先进行连接，通信接收后要释放连接。UDP 不需要建立连接，是一种不提供可靠交付的网络服务。网络通信中，为了准确识别通信终点，运输层使用协议端口号 (protocol port number) 来指定 IP 层的数据交付终点，当报文交给目的主机的某个合适的目的端口后，TCP 或 UDP 将完成最后交付给目的进程的工作^[36]。

Socket 可以分为流式 Socket 和数据报式 Socket。流式 Socket 所提供的数据流服务是一种双向的、有序的、不重复的，但是这种服务没有记录边界，例如 TCP 协议就是属于流式 Socket 协议。数据报式 Socket 所提供的服务虽然也是双向的，但是无法保证可靠、有序和不重复特点，可能会出现重复发送或者乱序现象，例如 UDP 协议，但是这种协议的优点在于它是有记录边界的^[37]。

C# 应用程序通过 Socket(套接字) 进行网络通信，在 WPF 工程中使用 Socket，首先需要引入相应的命名空间 **System.Net** 和 **System.Net.Sockets**^[38]，通过实例化 Socket 类就可以直接进行网络通信。本设计尝试分别使用 TCP 和 UDP 进行通信测试，测试结果表明使用 UDP 开发更简单易行。使用 TCP 连接服务器时，因为报文数据部分的长度是变长的，所以设置了足够大的缓冲区，但是因为缓冲区过大就容易出现报文粘连的现象，给数据处理增加了很多麻烦。使用 UDP 连接时则不会出现粘连报文的情况，故最后选择使用 UDP 为本设计与服务器通信的数据协议。

在使用时，要创建新的 Socket 类实例，实例化 Socket 后，使用该实例的 Bind 方法可以将该连接绑定为指定的 IP 和端口的形式。为了实现监听报文的功能，需要新建一个监听线程，在该线程中需要指定远端服务器的 IP 和端口。需要发送数据时，可以使用该实例 SendTo() 方法将编码后的数据内容发送给远端服务器，在

程序运行期间，设计中程序会不停监听从服务器发给本程序的 UDP 报文，接收数据时可以使用 Socket 类的 ReceiveFrom() 方法，然后就可以对报文进行相对应的的处理。

为了实现编码和解码的统一，我们规定服务器端和本程序均使用 UTF-8 编码。UTF-8(8-bit Unicode Transformation Format) 是万维网应用最广泛的编码方式，而且可以支持中文字符，还可以兼容 ASCII 码，使用 UTF-8 对字符进行编码和解码，可以扩大应用场景，方便程序进一步扩展。当需要发送数据时，先将字符串使用 UTF-8 编码为字符数组，接收数据后要先将字符数组通过 UTF-8 解码为字符串再处理，字符串的编码和解码都可以使用 **System.Text.Encoding** 类的方法来实现。

本设计是基于 WPF 开发的桌面端软件，对服务端程序无具体规定，为了和不同开发语言所开发的服务器进行通信，可以使用 JSON 来进行数据交换。JSON(JavaScript Object Notation) 是一种独立于语言的轻量数据交换格式。使用 JSON，可以方便地将类实例序列化，转化成需要发送的字符串，同样也很方便将字符串反序列化为目标类实例。在 WPF 中使用 JSON 可以通过 NuGet 安装 **Newtonsoft.Json** 拓展，安装以后需要引入相关命名空间 **Newtonsoft.Json** 和 **Newtonsoft.Json.Linq**。

需要序列化类实例为字符串时可以使用以下方法^[39]:

```
string str = JsonConvert.SerializeObject(obj);
```

反序列化为指定的类时方法如下：

```
Object obj = JsonConvert.DeserializeObject<Object>(str);
```

如果 JSON 中包含多个类或者 JSON 格式不确定时，可以先将其转换为 JObject 类，然后读取需要的键值对，进而可以进行不同的操作：

```
JObject jobj = JObject.Parse(str);
```

4.4.2 单无人机网络节点信息

本设计中除了可以实现无人机群网络拓扑这种整体的网络信息展示，还可以实现查看任意无人机节点的网络信息，用户可以根据自己的实际需要更改无人机的相关网络设置，用户保存后将更新相关数据。

本部分设计在数据库中维护了一张无人机节点信息的表格，当加载无人机节点信息时，将从数据库读取相关数据，这些数据被用来初始化类实例列表。这一

列表将通过数据绑定的方式，成为前台 XAML 所定义的多种控件的数据源。

WPF 的数据绑定方式包含 OneWay、TwoWay 和 OneTime 等^[40]，本文所用到的主要时 TwoWay 方式，这种绑定是一种双向的绑定，当绑定完成后，任意一端的更改都将影响另一方，通过这种绑定，我们可以实现更改某个无人机的网络相关的设置。WPF 指定数据源时可以采用 Source 标记、ElementName 标记或者使用 RelativeSource 标记，本文用到了 Source 标记和 ElementName 标记。绑定数据源后，使用 Path 标记还可以实现绑定数据源的某些属性值，简化了数据操作流程^[41]。在读取数据库节点信息后，我们将使用 Source 标记将数据指定为控件的数据源，如图 4.8 所示；当需要实现不同控件之间的互相绑定时，就可以使用 ElementName 方法，并且使用 Path 指定数据源的某些属性值，如图 4.9 所示。

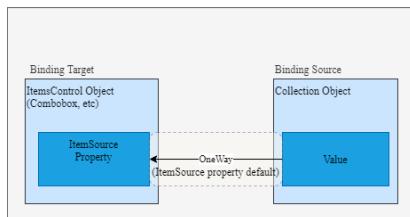


图 4.8 单向绑定数据源

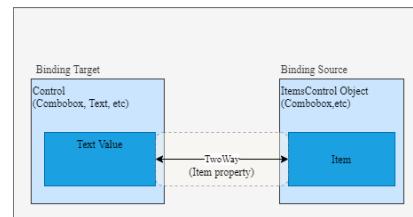


图 4.9 双向绑定数据源

4.5 无人机虚拟仪表盘

无人机的飞行姿态主要包含三个数据：偏航角 (yaw)、俯仰角 (pitch)、滚动角 (roll)，如图 4.10 所示。在无人机的飞行过程中，这三个数据是动态变化的，为了方便用户了解无人机的实时飞行姿态，故本部分设计实现了一个虚拟仪表盘，除了展示三个姿态信息外，还展示了无人机所处的高度 (altitude)。

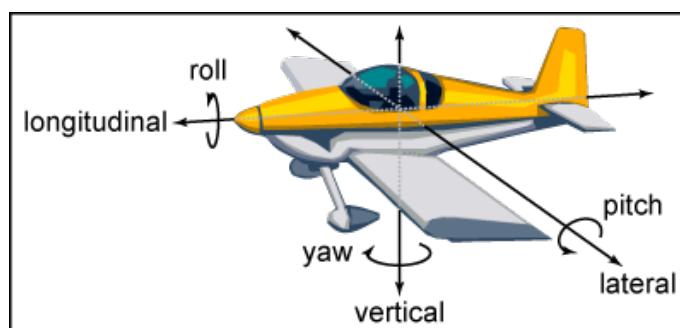


图 4.10 无人机姿态信息

4.5.1 虚拟仪表盘绘制

无人机的虚拟仪表盘通过自定义用户控件的方式来定义。自定义的用户控件可以像 Window(视窗)一样使用 XAML 定义前台页面，后台利用 C# 来实现逻辑。虚拟仪表盘实现时，前台进行分区的功能，定义了多个 Canvas(画布)，后台通过向这些画布添加文本和直线来实现构建。

无人机虚拟仪表盘如图 4.11 所示。其中显示了偏航角(A 区)、滚动角(B 区)、俯仰角(C 区)、海拔(D 区)。

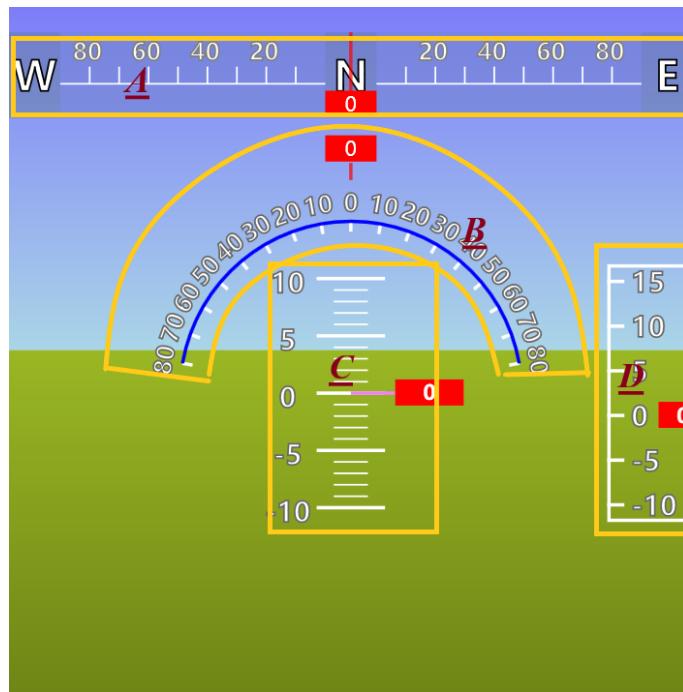


图 4.11 虚拟仪表盘

以海拔的绘制为例，虚拟仪表盘控件的 cs 文件中使用了下面的代码：

```
// 绘制右边海拔（高度）矩形
private void RedrawHeight()
{
    Rectangle rectangle = new Rectangle();
    ... // 设置 rectangle 的属性值和位置
    Canvas_ViewPortRight.Children.Add(rectangle);
    // 向 Canvas 添加矩形框
    for (int i = 0; i < 6; i++)
    {
        Line li = new Line();
        ... // 设置刻度线的属性值和位置
        Canvas_ViewPortRight.Children.Add(li);
        // 添加刻度线
        BorderTextLabel texti = new BorderTextLabel();
```

```

// BorderTextLabel 为自定义文本标签格式
...
    // 设置标签属性值和位置
    Canvas_ViewPortRight.Children.Add(texti);
    // 添加刻度
}
TextBlock textblock = new TextBlock();
...
    // 设置标尺的属性值和位置
    Canvas_ViewPortRight.Children.Add(textblock);
    // 添加标尺
}

```

4.5.2 虚拟仪表盘展示

为了可以实现实时展现无人机的姿态信息，本设计中添加了一个定时器 **DispatcherTimer**，在定时器间隔内，将会重新读取数据库，获取最新姿态信息。虚拟仪表盘的相关控件的属性值绑定到无人机的姿态信息上面，当无人机姿态信息改变时，虚拟仪表盘也会随之改变，就可以实现展示实时姿态的功能。

DispatcherTimer 可以用来调整 UI 线程的优先级，在本设计中，当定时器经历一个时间间隔后，将会重新绘制虚拟仪表盘，实现根据刷新频率更新姿态的功能。

虚拟仪表盘相关的绑定代码如下所示，相关的数据绑定到无人机的属性值上面：

```

<local:HUD x:Name="hud" VerticalAlignment="Center"
    HorizontalAlignment="Center" MaxRollAngle="80"
    Altitude="{ Binding ElementName=FilledComboBox2,
        Path=SelectedItem . Altitude , Mode=OneWay}"
    YawAngle="{ Binding ElementName=FilledComboBox2 ,
        Path=SelectedItem . Yaw , Mode=OneWay}"
    PitchAngle="{ Binding ElementName=FilledComboBox2 ,
        Path=SelectedItem . Pitch , Mode=OneWay}"
    RollAngle="{ Binding ElementName=FilledComboBox2 ,
        Path=SelectedItem . Roll , Mode=OneWay}" />

```

4.6 飞行记录保存

4.6.1 飞行轨迹和姿态信息保存

本设计设定系统运行期间将不间断与服务器通信，获取最新的无人机位置和姿态信息，这些实时信息将通过数据库传递给程序的窗口 (Window)。为了可以将无人机飞行数据保存下来，以备用户查看分析，系统运行期间将以某个设定的时

间间隔将飞行数据保存到数据库中，当需要查阅历史飞行任务时，只需读取数据库中的相关信息。

数据库使用 SQLite，系统内会维护一个飞行任务总表，记录每个飞行任务的无人机名称，飞行任务名称和飞行时间。根据飞行任务名称可以索引到该飞行任务的表格，其中记录了无人机的经纬度信息和姿态信息。

使用 SQLite 前需要先引入命名空间 **System.Data.SQLite**，SQLite 创建表格的代码如下所示：

```
string createtable = "create table Tasks(Id INTEGER, Devname TEXT, TaskId TEXT, FlyTime TEXT);  
//string为SQLite命令的文本  
SQLiteCommand cmdcreate = new SQLiteCommand(createtable, connect);  
//connect为打开状态下的SQLiteConnection  
cmdcreate.ExecuteNonQuery();
```

读取数据库的代码如下所示：

```
SQLiteCommand cmd = connect.CreateCommand();  
///connect为打开状态下的SQLiteConnection  
cmd.CommandText = "select * from Tasks";  
SQLiteDataReader reader = cmd.ExecuteReader();
```

4.6.2 飞行轨迹和姿态信息展示

飞行任务列表存放于数据库中，为了方便用户调取指定的飞行任务数据库，需要将该列表展示给用户做参考，故此处采用 WPF 的 **DataGridView**(数据网格) 控件，该控件可以采用类似表格的形式展示数据，本设计中使用 XAML 语言定义 **DataGridView**，在后台中将该控件的数据源绑定为从数据库读取的任务实例列表。

DataGridView 的相关定义代码如下所示，此处定义了一个四列的 **DataGridView**，并将其数据绑定到后台：

```
<DataGrid Name="mydatagrid" AutoGenerateColumns="False"  
VerticalScrollBarVisibility="Visible"  
ItemsSource="{Binding}" SelectionMode="Extended"  
SelectionUnit="FullRow" IsReadOnly="True" >  
<DataGrid.Columns>  
    <DataGridColumn Header="Id" Width="150"  
    Binding="{Binding Id}"></DataGridColumn>
```

```

<DataGridTextColumn Header="Devname" Width="150"
    Binding="{Binding Devname}"></DataGridTextColumn>
<DataGridTextColumn Header="TaskId" Width="200"
    Binding="{Binding TaskId}"></DataGridTextColumn>
<DataGridTextColumn Header="FlyTime" Width="200"
    Binding="{Binding FlyTime}"></DataGridTextColumn>
</DataGrid.Columns>
</DataGrid>

```

.DataBind 绑定数据后的展示效果如图 4.12 所示：

Id	Devname	Taskname	FlyTime
1	UAV1	Task1U1	2021/3/24
1	UAV1	Task2U1	2021/3/25
2	UAV2	Task1U2	2021/3/24
2	UAV2	Task2U2	2021/3/25
3	UAV3	Task1U3	2021/3/24

图 4.12 绑定数据后的 DataGrid

历史任务回放过程中的轨迹绘制设计电子地图的应用参考小节 4.2.3。

回放过程中的实时姿态展示相关设计参考小节 4.5。

4.7 本章小结

本章主要介绍了系统设计过程中所遇到的比较重要的技术点，包含了电子地图的应用，网络拓扑的绘制，节点信息设置，虚拟仪表盘绘制，历史任务回放，同时还介绍了相关的 Socket 通信，WPF 中 SQLite 操作，数据绑定，DataGrid 控件应用等。

通过本章的介绍，解决了系统开发过程中遇到的知识性的难点，下一章我们将专注于各功能模块的实现的测试，也将在下一章完成本文的所有设计工作。

5 系统实现和测试

5.1 引言

通过前一章的介绍，学习并掌握了系统开发过程中相关的技术难点和各功能模块的初步设计，本章将专注于实现各个功能模块，并且对各模块数据进行测试，以验证系统的可行性。

5.2 实时位置模块实现和测试

5.2.1 实时位置模块实现

实时位置模块的实现主要包含两步：

1. 加载电子地图。从 GMap.NET 加载电子地图，本设计中使用的是高德地图的电子地图服务，可以实现加载无人机群附近地图瓦片。
2. 添加无人机。向地图中添加标志，展示无人机的实时位置。

在系统运行过程中，始终保持一个定时器，按照一定的频率从数据库读取最新的无人机群各无人机位置，加载电子地图。

本模块的流程图如图 5.1 所示。

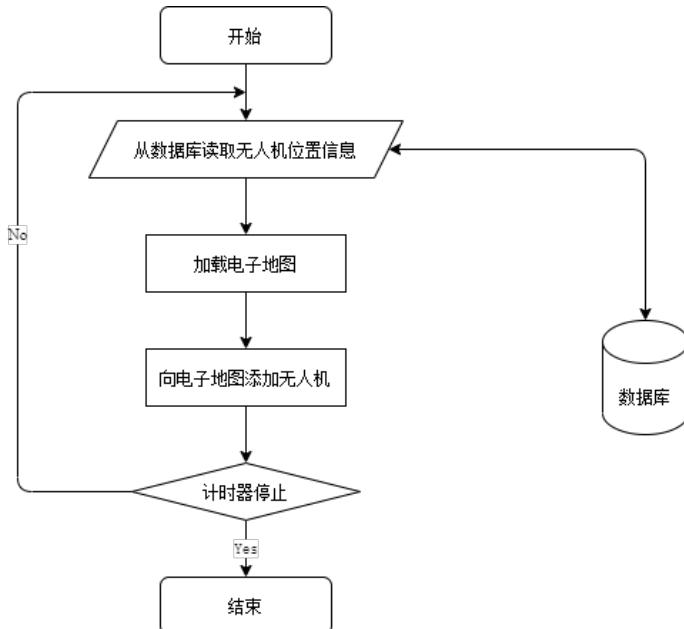


图 5.1 实时位置模块流程图

5.2.2 实时位置模块测试

测试目的：验证实时位置模块是否可以正常加载电子地图和无人机。

测试数据：测试所需的无人机位置信息数据来自附录 A.1

测试结果：测试最终效果如图 5.2 所示，电子地图加载正常，无人机标志加载正常。

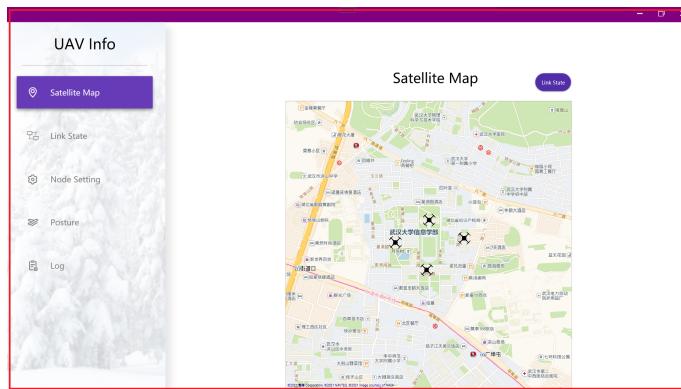


图 5.2 实时位置模块运行效果

5.3 网络拓扑模块实现和测试

5.3.1 网络拓扑模块实现

网络拓扑模块的实现主要包括以下步骤：

1. 将无人机的绝对位置转为相对位置，并添加相关控件。
2. 将连接定义到相应曲线，然后添加到画布。

本模块的流程图如图 5.3 所示。

5.3.2 网络拓扑模块测试

测试目的：验证网络拓扑是否可以争取绘制。

测试数据：无人机位置信息的数据来自附录 A.1，无人机群内连接数据来自附录 A.2。

测试结果：无人机群网络连接拓扑加载正常，无人机标志和连接曲线绘制正常。运行效果如图 5.4 所示。

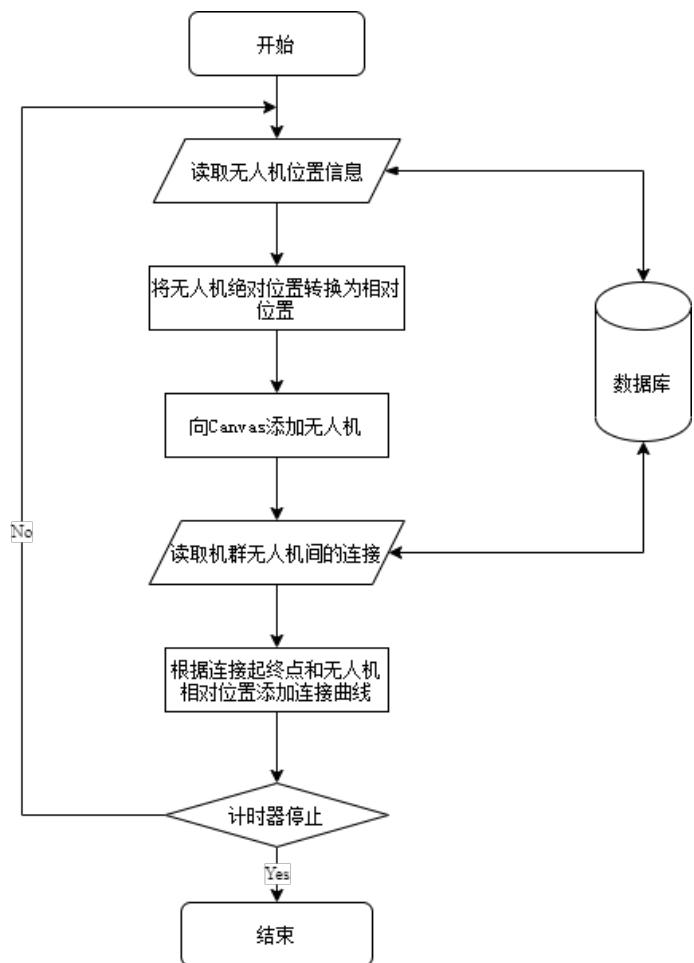


图 5.3 网络拓扑模块流程图

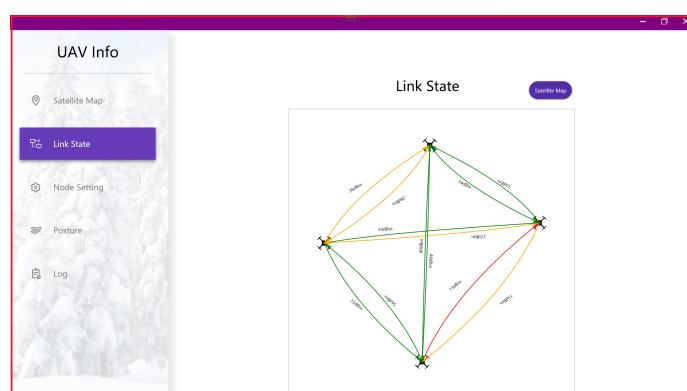


图 5.4 网络拓扑模块运行效果

5.4 网络节点模块实现和测试

5.4.1 网络节点模块实现

网络节点模块主要包含无人机网络节点信息展示和数据保存两方面，当用户更改无人机网络节点设置后，保存数据将会同时用于更新数据库和发送给服务器更新无人机设置。

本模块的流程图如图 5.5 所示。

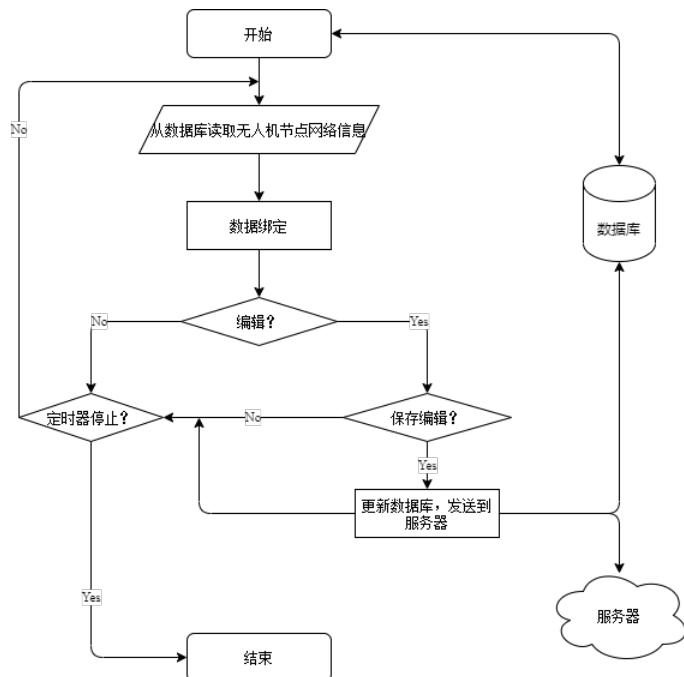


图 5.5 网络节点模块流程图

5.4.2 网络节点模块测试

测试目的：验证节点数据是否正确显示，验证数据保存是否正常运行。

测试数据：无人机群各无人机网络节点的相关设置数据来自附录 A.3。

测试结果：数据记载正常，数据保存正常。运行效果如图 5.6 所示。

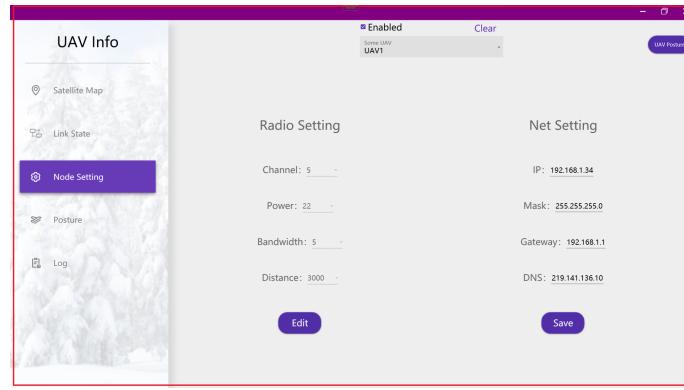


图 5.6 网络节点模块运行效果

5.5 飞行姿态模块实现和测试

5.5.1 飞行姿态模块实现

飞行姿态模块主要是读取数据库内的无人机姿态信息，绘制虚拟仪表盘。此模块的流程图如图 5.7 所示。

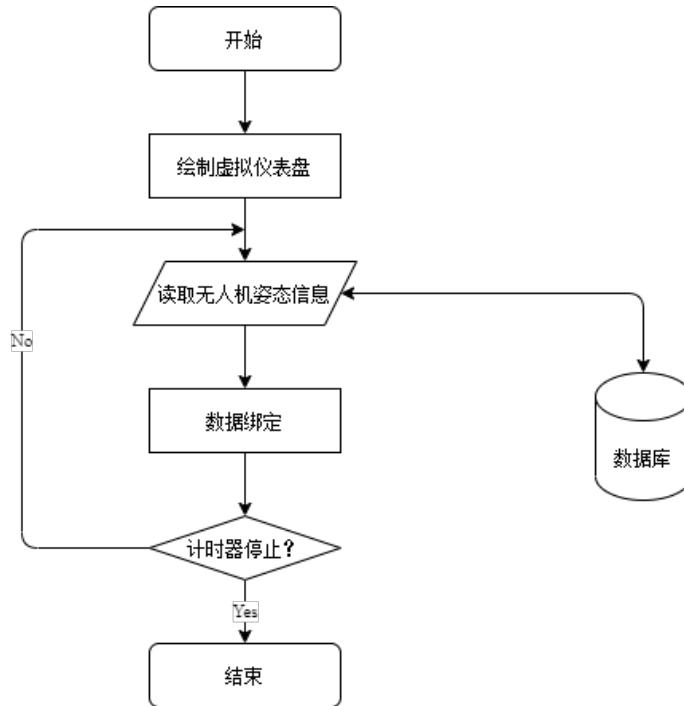


图 5.7 飞行姿态模块流程图

5.5.2 飞行姿态模块测试

测试目的：验证虚拟仪表盘加载正常，实时姿态可以正常展示。

测试数据：测试用的姿态数据来自附录 A.1。

测试结果：飞行姿态模块加载正常，如图 5.8。



图 5.8 飞行姿态模块运行效果

5.6 飞行记录模块实现和测试

5.6.1 飞行记录模块实现

飞行记录模块包含以下步骤：

1. 加载飞行任务索引列表，方便用户查找相关飞行任务。
2. 加载指定的飞行任务数据。
3. 绘制历史飞行轨迹和飞行过程中的实时姿态。

此模块的流程图如图 5.9 所示。

5.6.2 飞行记录模块测试

测试目的：验证历史飞行任务加载正常，用户可查看飞行数据。

测试数据：历史飞行任务索引表来自附录 A.4。飞行任务示例数据来自附录 A.5。

测试结果：历史飞行任务加载正常，可以描绘相应轨迹和姿态信息。如图 5.10。

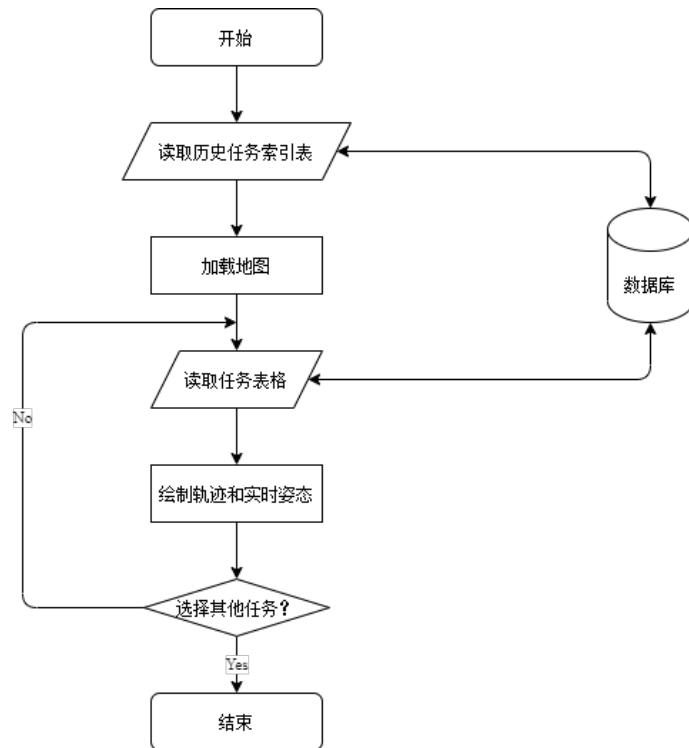


图 5.9 飞行记录模块流程图

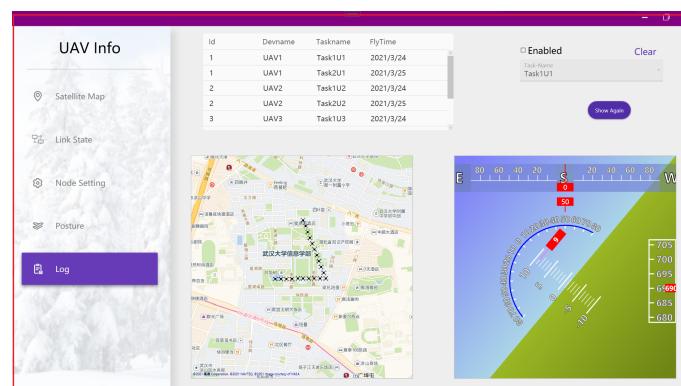


图 5.10 飞行记录模块运行效果

5.7 本章小结

本章主要介绍了相关功能模块的实现流程，以及在测试数据下的运行结果。运行结果表明，程序设计合理，系统运行一切正常，验证了整个系统的可行性。下一张，我们将简单介绍本设计的总结和展望，关于开发过程的不完善之处将详细说明。

6 总结与展望

6.1 本文工作总结

随着科技的发展和无人机技术的进步，在可预见的将来，无人机将会更加微型化，其成本也将得到大规模的降低，进一步可以促进无人机在生活中的应用。当单无人机无法满足工业和生活需求时，针对无人机群的管理系统将越来越重要。

本文力图从网络管理的角度初步实现无人机群的集群管理，论文首先研究了国外无人机管理系统和无人机群网络的相关进展和发展趋势；然后研究了相关的技术背景，包括 WPF 技术、C# 和 XAML 语言以及 SQLite 数据库；进一步设计了整个系统的相关功能和软件的界面布局；本设计所设计的关键技术主要包含 GMap.NET 电子地图、WPF 绘图、Socket 通信、数据绑定等；最后关于各页面进行了具体的实现和测试，验证了系统可行性。

论文主要完成了以下几个方面的具体设计和实现：

1. 研究了电子地图技术加载的一般性原理，分析了国内高德地图的瓦片地址，利用 GMap.NET 实现了电子地图的加载和离线缓存瓦片地图，进一步实现了无人机群内各无人机实时位置的展示。
2. 设计绘制了无人机群网络连接的拓扑结构，通过将绝对位置转换为相对位置，保持了无人机之间的相对空间位置，通过绘制带文字的二次贝塞尔曲线展示无人机之间的网络连接质量。
3. 展示了集群内指定无人机的网络节点信息，展示信息的过程使用了 WPF 的数据绑定以实现数据更新，当用户修改无人机网络设置后，可以使用 Socket 通信将相关信息回传至服务器，Socket 通信使用 UDP 协议，数据使用 UTF-8 编码，传输格式使用 JSON 格式。
4. 设计了一个虚拟仪表盘以实现无人机飞行姿态的实时展示，虚拟仪表盘是一个自定义的用户控件，使用 XAML 实现页面布局，C# 控制相关图形的添加，加载时定时使用 SQLite 读取数据库以实现实时姿态。
5. 设计了一个历史飞行任务保存的数据库，以某个时间频率定时保存无人机的位置和姿态，当用户调取历史任务时，实现飞行轨迹重现和实时姿态复现。
6. 自行设计了相关测试数据，实现了对整个系统各模块的本地测试，测试结果

表明，系统运行一切正常，实现了设计目的，验证了系统可行性。

6.2 本文不足与工作展望

本设计初步实现了无人机群网络的多方位展示，包括机群整体状态和单个无人机状态，系统的可行性也得到了较为充足的验证。但是，项目本身在设计上和功能实现上还有很多可以完善补充的方面，受限于项目周期有限，很多可研究的方向可以在未来进一步展开。主要包含以下几点：

1. 受限于开发经验，在 GMap.NET 电子地图控件的应用方面应该还有很多可以拓展的方面，诸如：可以考虑将网络拓扑加载到电子地图中，这样可以提高用户的使用便捷性；可以开发电子围栏系统，设定无人机的运行范围等。
2. 在描绘网络拓扑的过程中，采用了带文本的二次贝塞尔曲线，其文本描绘方向，中间控制点的计算等方面还可以更加优化。
3. 无人机网络节点信息展示时，采用了数据库进行本系统和服务器之间的数据过渡，是不是可以考虑越过数据库，直接进行系统和服务器之间的数据通信。
4. 关于无人机姿态的展示，在进行文献调研时，曾经看过基于 OpenGL 构建无人机的三维立体模型，改变姿态信息后，模型就会发生相应的旋转，这样看起来可以更直观，如果时间充足的条件下，可以进一步研究。
5. 设计历史飞行任务的保存时，还应该考虑到更多实地应用场景下的数据归类，诸如无人机短时间断连后再次连接是否应该归纳为新一次飞行任务的问题等。
6. 本设计基于 Windows 平台开发，如果时间充足，可以考虑使用其他开发语言开发跨平台的应用程序。
7. 本设计囿于 TCP 的报文粘连现象，采纳了 UDP 的连接协议，那么关于报文粘连是否可以探索某种解决方案，如果可以解决将扩大本设计的拓展应用。
8. 关于无人机群网络连接的部分研究不足，诸如无人机群无线自组织方式，飞行过程中的动态路由，关于在无人机群中使用新型通讯方式(例如 5G)的应用等等，都可以成为很好的研究方向。

参考文献

- [1] HOSSEIN MOTLAGH N, OTHERS. An Efficient System Orchestrator and a Novel Internet of Things Platform for Unmanned Aerial Systems[J], 2018.
- [2] KUMAR A, SHARMA K, SINGH H, et al. A drone-based networked system and methods for combating coronavirus disease (COVID-19) pandemic[J]. Future Generation Computer Systems, 2021, 115 : 1 – 19.
- [3] 鲁杰. 无人机技术在智能电网中的典型应用及发展趋势 [J]. 中国科技投资, 2020, 000(003) : 57 – 59.
- [4] 艾必心. 多旋翼无人机地面监控系统的设计 [D]. [S.l.]: 东南大学, 2017.
- [5] 赵婷婷. 无人机地面站与航迹规划的研究 [D]. [S.l.]: 天津大学, .
- [6] KENDOUL F, YU Z, NONAMI K. Guidance and nonlinear control system for autonomous flight of minirotorcraft unmanned aerial vehicles[M]. [S.l.] : Guidance and nonlinear control system for autonomous flight of minirotorcraft unmanned aerial vehicles, 2010.
- [7] NAMUDURI K, WAN Y, GOMATHISANKARAN M, et al. Airborne network: a cyber-physical system perspective[A]. Proceedings of the first ACM MobiHoc workshop on Airborne Networks and Communications[C], 2012 : 55 – 60.
- [8] MOTLAGH N H, TALEB T, AROUK O. Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives[J]. IEEE Internet of Things Journal, 2016, 3(6) : 899 – 922.
- [9] BEKMEZCI I, SAHINGOZ O K, TEMEL S. Flying ad-hoc networks (FANETs): A survey[J]. Ad Hoc Networks, 2013, 11(3) : 1254 – 1270.
- [10] HAN Z, SWINDLEHURST A L, LIU K. Optimization of MANET connectivity via smart deployment/movement of unmanned air vehicles[J]. IEEE Transactions on Vehicular Technology, 2009, 58(7) : 3533 – 3546.

- [11] MOZAFFARI M, SAAD W, BENNIS M, et al. Mobile Unmanned Aerial Vehicles (UAVs) for Energy-Efficient Internet of Things Communications[J]. IEEE Transactions on Wireless Communications, 2017, PP(11): 7574–7589.
- [12] LEE J Y, LEE W, KIM H, et al. Adaptive TCP Transmission Adjustment for UAV Network Infrastructure[J]. Applied Sciences, 2020, 10(3): 1161.
- [13] 刘宏波, 孟进, 赵奎. 蜂群无人机数据链自组网协议设计 [J]. 火力与指挥控制, 2018, 43(009): 163–168.
- [14] 游文静、董超、吴启晖. 大规模无人机自组网分层体系架构研究综述 [J]. 计算机科学, 2020, v.47(09): 232–237.
- [15] 张倩怡. 无人机群网络智能拓扑控制机制 [D]. [S.l.]: 电子科技大学, .
- [16] 周奕捷. 浅谈无人机集群组网通信方式及其发展趋势 [J]. 企业科技与发展, 2019, 000(007): 130–132.
- [17] 王婧宇, 贾秋玲, 付玮. 基于 VC++ 的无人机便携式地面站设备的设计 [J]. 电子设计工程, 2014, 22(004): 39–42.
- [18] 张浩然. 基于 LabVIEW 的无人机地面站系统研究与设计 [D]. [S.l.]: 长春理工大学, .
- [19] 金强, 方春华, 王亮. 基于 Qt 的四旋翼无人机地面站航空虚拟仪表的设计 [J]. 现代机械, 2020, No.217(03): 85–89.
- [20] 杨青, 赵锋, 李阳. 基于 C# 的无人机地面站软件设计 [J]. 电子质量, 2017, 000(005): 48–51,56.
- [21] 刘静超. 基于 Android 的小型无人机地面站研究与设计 [D]. [S.l.]: 西京学院, .
- [22] 孙博. 基于微软新一代图形系统 WPF 和 Silverlight 的数据可视化研究与实现 [D]. [S.l.]: 东北师范大学, .
- [23] 王鑫. 基于 WPF 技术无人机地面站控制软件的设计 [D]. [S.l.]: 北京理工大学, 2015.

- [24] 董杰, 汪漪, 蒋同海. 基于.NET Framework 的多语种软件 UI 构件 [J]. 计算机应用研究, 2009, 26(011): 4108–4110.
- [25] 艾迪明. .NET 框架体系结构 [J]. 计算机工程与应用, 2003, 39(2): 174–176.
- [26] LIANG R. THE STUDY ON IMPLEMENTING THE MILITARY DRAWING SYSTEM WITH THE INTEGRATION OF MAPX INTO VISUAL C++ 6.0[J]. Computer Applications and Software, 2005.
- [27] 王斌. 多旋翼无人机地面监控系统设计 [D]. [S.l.]: 南京信息工程大学, 2017.
- [28] 周炤, 李少梅, 杨佳. Web 墨卡托投影及其性质分析 [J]. 测绘科学技术学报, 2021.
- [29] 赵永辉, 段云龙, 郭新望, et al. 谷歌地图关键技术剖析与应用 [J]. 海洋测绘, 2019(3).
- [30] 寇曼曼, 王勤忠, 谭同德. Google Map 数字栅格地图算法及应用 [J]. 计算机技术与发展, 2012(04): 204–206.
- [31] 黄梦龙. 瓦片地图技术在桌面端 GIS 中的应用 [J]. 地理空间信息, 2011, 000(004): 149–151.
- [32] 李辉杰. 无人机地面站监控软件的设计与实现 [D]. [S.l.]: 哈尔滨工程大学, .
- [33] 黄富东. 无人飞行器状态参数地面监控系统设计与应用 [D]. [S.l.]: 南京航空航天大学, .
- [34] 郑华美. 小型无人机地面站软件系统的设计与实现 [D]. [S.l.]: 电子科技大学, .
- [35] 刘艳. 基于 GMAP.NET 飞行在线地图系统的设计与实现 [J]. 科技创新导报, 2013(14): 228+230.
- [36] 谢希仁. 计算机网络. 第 7 版 [M]. [S.l.]: 计算机网络. 第 7 版, 2017.
- [37] 刘烨. 用 Socket 实现基于 TCP 和 UDP 的原理探索 [J]. 智能计算机与应用, 2009(3): 6–8.
- [38] 李巧玲. 基于 C#-Socket 的网络通信程序设计 [J]. 福建电脑, 2009, 25(004): 135–136.

- [39] 何诚, 邵乾飞, 袁浩, et al. 基于 Socket 实现 Android(java) 与 C# 的同步通信 [J].
无线互联科技, 2015, 000(002): 15–16.
- [40] 任楷川. 基于 WPF+SQL 的军队涉密文件资料管理系统的设计与实现 [D].
[S.l.]: 吉林大学, 2018.
- [41] 瑾彬. 基于 WPF 平台的自定义控件开发 [D]. [S.l.]: 西安电子科技大学, 2008.

附录 A 测试数据

A.1 无人机节点信息

表 A.1 无人机节点表格结构

名称	数据类型
Id	INTEGER
Latitude	DOUBLE
Longitude	DOUBLE
Altitude	DOUBLE
Yaw	DOUBLE
Pitch	DOUBLE
Roll	DOUBLE

表 A.2 无人机节点表格数据

Id	Latitude	Longitude	Altitude	Yaw	Pitch	Roll
1	30.52904698265222	114.36134562411276	200	20	15	10
2	30.52813480200784	114.36343806864225	150	-40	-10	-30
3	30.52650274657577	114.361198897819	500	50	10	-20
4	30.52789851283774	114.35932973242562	700	-20	40	50

A.2 无人机连接信息

表 A.3 无人机连接表格结构

名称	数据类型
source	INTEGER
target	DOUBLE
quality	DOUBLE

表 A.4 无人机连接表格数据

source	target	quality
1	2	34
1	3	44
1	4	28
2	1	33
2	3	15
2	4	27
3	1	40
3	2	13
3	4	56
4	1	26
4	2	30
4	3	33

A.3 无人机网络节点信息

表 A.5 无人机网络节点表格结构

名称	数据类型
Id	INTEGER
Channel	INTEGER
Power	INTEGER
Bandwidth	INTEGER
Distance	INTEGER
IP	TEXT
Mask	TEXT
Gateway	TEXT
DNS	TEXT

表 A.6 无人机网络节点表格数据

Id	Channel	Power	Bandwidth	Distance	IP
1	5	22	5	3000	192.168.1.34
2	5	15	10	5000	192.168.1.45
3	7	25	10	11000	192.168.2.78
4	9	25	40	9000	192.168.2.93

表 A.7 无人机网络节点表格数据续表

Id	Mask	Gateway	DNS
1	255.255.255.0	192.168.1.1	219.141.136.10
2	255.255.255.0	192.168.1.1	219.141.136.10
3	255.255.255.0	192.168.2.1	219.141.136.10
4	255.255.255.0	192.168.2.1	219.141.136.10

A.4 无人机历史飞行任务索引表

表 A.8 无人机历史飞行任务索引表结构

名称	数据类型
Id	INTEGER
Devname	TEXT
TaskId	TEXT
FlyTime	TEXT

表 A.9 无人机历史飞行任务索引表数据

Id	Devname	TaskId	FlyTime
1	UAV1	Task1U1	2021/3/24
1	UAV1	Task2U1	2021/3/25
2	UAV2	Task1U2	2021/3/24
2	UAV2	Task2U2	2021/3/25
3	UAV3	Task1U3	2021/3/24
3	UAV3	Task2U3	2021/3/25
4	UAV4	Task1U4	2021/3/24
4	UAV4	Task2U4	2021/3/25

A.5 无人机历史飞行任务示例

表 A.10 无人机历史飞行任务示例结构

名称	数据类型
Latitude	DOUBLE
Longitude	DOUBLE
Altitude	DOUBLE
Yaw	DOUBLE
Pitch	DOUBLE
Roll	DOUBLE

表 A.11 无人机历史飞行任务示例数据

Latitude	Longitude	Altitude	Yaw	Pitch	Roll
30.5271563997661	114.359712416647	500	-10	-10	-45
30.527156933637	114.360048337113	510	0	-9	-40
30.5271574675079	114.36038425758	520	10	-8	-35
30.5271580013788	114.360720178046	530	20	-7	-30
30.5271585352497	114.361056098512	540	30	-6	-25
30.5271590691206	114.361392018978	550	40	-5	-20
30.5271596029915	114.361727939445	560	50	-4	-15
30.5271601368624	114.362063859911	570	60	-3	-10
30.5271606707333	114.362399780377	580	70	-2	-5
30.5271612046042	114.362735700843	590	80	-1	0
30.5274608907667	114.362601860343	600	90	0	5
30.5277605769293	114.362468019842	610	100	1	10
30.5280602630919	114.362334179341	620	110	2	15
30.5283599492544	114.36220033884	630	120	3	20
30.528659635417	114.36206649834	640	130	4	25
30.5289593215795	114.361932657839	650	140	5	30
30.5292590077421	114.361798817338	660	150	6	35
30.5295586939047	114.361664976838	670	160	7	40
30.5298583800672	114.361531136337	680	170	8	45
30.5301580662298	114.361397295836	690	180	9	50