毕业设计说明书中文摘要

|  |
| --- |
| 传统地面站软件或二次开发难度大、或需要运行与PC设备上，在户外场合使用不便，本文针对在移动设备对无人机器人进行辅助遥控和可视化的需求，基于Android平台开发一款地面站APP，实现数据状态、现场视频可视化及存储记录、远程临场感可视化界面等。该地面站APP能够实现空中场景下对无人机的辅助遥控，能够实现陆空场景下的可视化功能。  在研究了地面站软件相关理论后，分析了地面站APP的需求，进行了系统的总体技术方案设计，将地面站APP分为地图导航模块、飞行控制模块、图像传输模块三部分。并使用百度地图SDK实现了地图导航模块，使用MAVSDK实现了飞行控制模块，使用UVC摄像头实现了图像传输模块。开发完成后在仿真和实验两种环境下对系统各功能进行了测试，测试结果表明，地面站APP一切功能运行良好。  关键词 地面站 可视化 无人机 Android |

毕业设计说明书外文摘要

|  |
| --- |
| **Title**  Android-based Assisted Remote Control for Land and  Sky Unmanned Platforms Visualization Software Development  **Abstract**  The traditional ground station software is either difficult to develop twice or needs to run on PC devices, which is inconvenient to use in outdoor occasions. This paper develops a ground station APP based on Android platform for the needs of auxiliary remote control and visualization of unmanned robots in mobile devices, realizing data status, field video visualization and storage records, remote proximity visualization interface, etc. This ground station APP can realize the auxiliary remote control of the unmanned robot in the air scene and can realize the visualization function in the land and air scene.  After studying the theories related to UAV ground station software, this paper analyzes the needs of the ground station APP, designs the basic architecture of the system, and divides this ground station APP into three parts: map navigation module, flight control module, and image transmission module. The map navigation module is realized by using Baidu map SDK, the flight control module by using MAVSDK, and the image transmission module by using UVC camera. After the development, the system functions were tested in both simulation and real environment, and the test results showed that all functions of this ground station APP worked well.    Keywords Ground Control Station Visualization  UVA Android |

目 录

[1 绪论 1](#_Toc103637044)

[1.1 研究背景 1](#_Toc103637045)

[1.2 国内外研究现状 1](#_Toc103637046)

[1.3 论文主要工作及章节安排 3](#_Toc103637050)

[2 基础知识 5](#_Toc103637051)

[2.1 Android平台介绍 5](#_Toc103637052)

[2.2 MAVLink通信协议介绍 6](#_Toc103637053)

[2.3 MAVSDK介绍 8](#_Toc103637054)

[2.4 本章小节 9](#_Toc103637055)

[3 地面站APP的分析与设计 10](#_Toc103637056)

[3.1 需求分析 10](#_Toc103637057)

[3.2 总体设计 10](#_Toc103637058)

[3.3 模块详细设计 11](#_Toc103637061)

[3.4 界面设计 14](#_Toc103637065)

[3.5 数据存储设计 16](#_Toc103637066)

[3.6 本章小结 17](#_Toc103637067)

[4 地面站APP的实现 18](#_Toc103637068)

[4.1 开发平台与运行环境 18](#_Toc103637069)

[4.2 地图导航相关功能的实现 19](#_Toc103637073)

[4.3 飞行控制功能的实现 22](#_Toc103637078)

[4.4 图像传输功能的实现 24](#_Toc103637082)

[4.5 本章小结 26](#_Toc103637086)

[5 地面站APP的测试 28](#_Toc103637087)

[5.1 仿真环境测试 28](#_Toc103637088)

[5.2 实验环境测试 29](#_Toc103637091)

[5.3 本章小结 33](#_Toc103637094)

[结 论 34](#_Toc103637095)

[致 谢 35](#_Toc103637096)

[参 考 文 献 36](#_Toc103637097)

**1 绪论**

**1.1 研究背景**

无人驾驶机器人（英文：“Unmanned Driving Robot”），是一种利用无线遥控器或预设程序代码控制的机器人设备，包括无人机（UAV）、无人车（UGV）等陆空多种场景设备。由于其小体积、轻重量、高机动性的特点，最初主要在军事领域，用于替代载人设备在高危领域、狭小空间完成电子对抗、无人侦查、中继通信等任务[1]。随着科学技术的进步和机器人设备生产成本的降低，无人驾驶机器人已逐步运用到社会生活的各个领域，如：地面巡检、农用植保、森林防火、灾后救援、影视拍摄等，且其应用仍在不断扩宽。

典型的无人系统由动力系统、遥控系统、控制系统、视频系统等组成。其中控制系统是无人系统的关键，而相较于二维空间无人车，对三维空间无人机的操作由于增加了高度控制，其实现更为复杂，可运用的场景更为丰富，是当前无人领域研究的一大重点。

无人机的控制系统包括无人机飞行控制和地面站系统两部分。无人机飞行控制由控制无人机自主飞行的飞行控制软硬件模、无人机与地面站系统之间的数据链路硬件模块以及机载的电子设备和任务设备等组成[2]，负责接收来自地面站的控制信号，并根据机身传感器的参数控制无人机，调整飞行姿态，执行相应任务；地面站在整个无人机系统中有着至关重要的地位，是整个无人机系统的控制核心[3]，负责接收、发送、解析飞行数据包，显示、存储飞行数据，以及完成电子地图、实时定位显示、航迹显示等功能[4]；同时，无人机飞行任务计划制订、发射控制、飞行状态控制和调整及回收、传感器运行控制、与外部硬件设备通讯等重要功能的实现全都依赖于地面站系统[5]。

视频系统是实现现场画面可视化的关键，用于辅助无人机器人完成避障等操作。基于陆地和空中环境的不同，陆地场景需要获取前、后、左、右方位的现场画面，空中场景需要获取前、后、左、右、上、下方位的现场画面。现有的视频系统实现方案一般分为两种：一是使用多个摄像头，每个摄像头覆盖一个方位，多个摄像头共同实现全方位视频显示；二是使用单个摄像头加云台，由云台控制摄像头调整拍摄角度，实现多方位画面显示。其中，方案一的实现效果更好，但成本较高；方案二的实现更简单，且操作更灵活。现如今越来越多的地面站系统同时集成有视频显示功能。

然而，当前市面上大多数地面站系统是为专用设备设计、仅具有某些特定功能，不具备通用性，且大多只能运行在PC系统上。随着智能手机的普及、Android系统的发展，传统地面站系统逐渐不能满足用户的需要。因此研究并开发一款基于Android系统的地面站APP实现陆空场景下的辅助控制和可视化很有意义。

**1.2 国内外研究现状**

**1.2.1 地面站软件**

地面站软件作为地面站系统的重要组成部分之一，操作人员可以通过地面站系统提供的外部设备与地面站软件进行交互，在飞行开始前预先规划好本次飞行任务、飞行过程中对无人机的飞行状况进行实时监控和修改任务以操作无人机飞行，任务完成后还可以对任务的执行记录进行回放分析 。地面站软件主要的功能有：

（1）地图导航功能。主要用于显示无人机发送给地面站的实时位置、飞行轨迹信息，以便操作人员观察，随时调整无人机飞行姿态，确保飞行任务顺序完成。

（2）飞行控制功能。主要用于控制无人机的飞行姿态。由于无人机在飞行过程中，会受到强风、气流、无线电信号、障碍物等影响，为了保证良好的飞行品质，飞行控制功能尤为重要。其实现依赖于GPS模块、陀螺仪、加速计、气压传感器、超声波传感器等多种模块。

（3）航线规划与航迹操作功能。主要用于帮助地面站操作人员，根据需要执行的飞行任务，事先规划出飞行路线、标记任务航点，并将规划好的路线和航点信息，通过有线或无线方式传输到无人机的飞行控制系统[6]。

**1.2.2 国外研究状况**

国外对地面站研究起步早，开源项目多、功能完善，以下为两款当下流行的国外地面站软件。

（1）QGroundControl

QGroundControl是一款适用于所有启用MAVLink协议无人机的地面站软件。该地面站软件能够为运行ArduPilot和PX4系统的无人机提供完备的飞行控制和任务规划功能，能够对无人机进行各种设置和参数配置；能为所有使用MAVLink协议通讯的无人机实时显示和分析飞行数据，自主设定飞行任务规划和飞行航线[7]；可连接到多台无人机，同时管理多台设备。QGroundControl支持Windows系统和Android系统，软件所有代码均为开源，开发者可以根据自己的需要，使用C++语言、Qt框架自行扩展功能，市面上多数地面站软件也都是据此修改而来。

（2）Mission Planner

Mission Planner地面站也是一款基于MAVLink通信协议的开源地面站软件[8]，相较于QGroundControl，其功能更加全面，涵盖了调试、飞行、模拟等一整套功能，不仅具备基础的数据配置、任务规划、定位、航迹显示、现场画面显示功能，还能够从飞控上下载和分析飞行日志信息，能为APM、Pixhawk飞控升级最新版本固件，能够操控如固定翼无人机、多旋翼无人机、无人车等多种设备，并支持一个地面站系统控制多架无人机[9]。

**1.2.3 国内研究状况**

（1）DJI GS

DJI GS是由全球领先的大疆创新公司开发的专供大疆无人机使用的地面站软件[10]。与其他形式单一、界面老旧、重在数据显示的地面站系统相比，DJI GS的3D操作界面十分独特、直观、便捷，更易于用户上手，大大降低了使用的复杂度。且凭借大疆创新公司在飞行控制芯片领域的技术优势，搭配先进的自研飞行控制算法，地面站在保持无人机飞行的稳定性、机动性和精准度方面有着优异表现。DJI GS能完美实现一键返航、自动避障、自主起飞降落等功能；支持任务飞行航线规划，可预先在航点设置拍照、空投动作；支持在飞行任务执行期间修改飞行航线或航点，编辑飞行任务，实时调整飞行任务目标；支持通过键盘或自定义摇杆进行飞行控制操作。

（2）CUAV GS

CUAV GS是雷迅创新公司专属的Android手机地面站，基于雷迅云平台开发。该地面站软件支持使用PX4和APM固件的开源飞控，能满足不同用户的需求；支持自定义飞行控制设备的各种参数，能够设置返航高度、最大飞行高度、失控保护等，确保飞行安全；支持通过机载摄像机拍照与录像，并可将视频与照片记录在LTE Link的内存卡中；支持团队协作功能，用户可以操作已加入团队的其他设备，查看其它设备的实时现场画面与飞行数据。

**1.3 论文主要工作及章节安排**

本论文通过研究地面站有关内容、陆空可视化背景，从地面站软件发展现状以及开发的实际需要出发，开发了一款基于Android系统的地面站APP。利用Android平台、MAVSDK、百度地图SDK、UVC摄像头等多种技术，建立APP与硬件设备的连接，实现数据状态和现场视频可视化及存储记录、远程临场感可视化界面等。并在地面站APP开发完成后，对系统进行了仿真环境与实验环境下的测试。

该地面站APP针对更为复杂的空中无人机实现了辅助遥控，针对陆地无人车和空中无人机，采用单摄像头加云台方案，实现了现场画面可视化。通过WiFi与使用MAVLink协议的无人机建立连接后，用户可通过各功能模块实现对无人机的定位、飞行控制、实时现场画面显示等操作，同时兼容陆地无人车的可视化功能。APP的开发使用Java语言，依靠Java强大的生态，有利于进行后续功能的扩展；基于Android Studio平台，符合当下Android应用开发的潮流。

本文将分为五大章节，各章节的内容安排如下。

第一章：绪论。本章主要介绍了本文研究的地面站背景以及国内外有关地面站软件的研究现状。

第二章：基础知识。本章主要介绍了本地面站APP开发过程所需的一些基础知识。包括：MAVLink通信协议、Android平台、MAVSDK。

第三章：地面站APP的分析与设计。本章首先分析了地面站APP的需求，根据需求设计了总体技术方案，将软件系统划分为三大模块，并对每个模块进行了详细设计。

第四章：地面站APP的实现。本章根据第三章中的设计给出了地面站APP各功能模块的详细实现方法，为各个模块绘制了流程图。

第五章：地面站APP的测试。本章介绍了开发完成后对软件进行的仿真和实验环境下的功能测试，验证软件的可用性。

**2 基础知识**

**2.1 Android平台介绍**

Android系统采用了典型的分层体系架构，其体系结构分为5层：Linux内核层（Linux kernel）、硬件抽象层（HAL）、系统运行库层（Libraries）、JAVA API框架层（Application Framework）、应用层（Application）。系统整体架构图如图所示：



图2.2 Android平台体系架构

自底向上的，第一层是Linux内核层。该层提供了内存管理、进程管理、安全管理、网络协议等核心系统服务，也为各种硬件提供了驱动程序，如显示驱动、相机驱动、蓝牙驱动、电池管理等。

第二层是硬件抽象层（HAL）。该层对Linux内核驱动程序进行了封装，屏蔽Android低层的实现细节，并为上层提供标准界面，向更高一层的Java API框架层提供显示硬件设备的功能。

第三层是Android系统运行层。该层包括原生C/C++库和安卓运行时（ART）两部分。原生C/C++库使得开发者在开发基于C或C++ 代码Android应用时，可以使用Android NDK直接从native代码中访问native平台库，大大降低了开发难度；安卓运行时部分则提供了预先和即时编译、优化的垃圾回收等功能。

第四层是Java API框架层。该层提供了构建应用程序时可能用到的各种API，开发者使用这些API可以简化核心模块化系统组件和服务的重复使用，以方便应用的构建。

第五层是系统应用层。系统自带的软件和用户自己下载的软件均属于这一层。

**2.2 MAVLink通信协议介绍**

MAVLink是一种十分轻量级的用于微型飞行器MAV（Micro Air Vehicle）的消息传输协议，用于实现地面控制终端（地面站）与无人机之间 （以及机载无人机组件之间)）的通信，实现无人机起飞、降落、移动等操作。该协议的设计采用现代混合发布-订阅和点对点设计模式，这种模式的特点是将数据流作为主题发布，供用户订阅，而其配置子协议是点对点的，具有重新传输功能。MAVLink协议在无人机领域被广泛应用，飞行控制设备如APM、PIXHAWK，地面站软件如Mission Planner、QGroundControl等均使用了MAVLink协议。

MAVLink协议在传输时，以消息包作为基本单位的，其数据长度最短8字节，最长263字节。消息包的结构如下图所示：

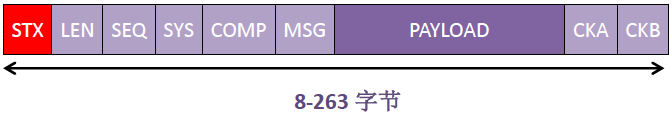


图2.1 MAVLink消息报结构图

除PAYLOAD字段外，每一个格子均代表了一个字节的数据。MAVLink官方对消息帧的定义如下：

表2.1 MAVLINK V1官方定义

|  |  |  |
| --- | --- | --- |
| Filed name | Index | Purpose |
| Start-of-frame | 0 | Denotes the start of frame transmission (v1.0:0xFE) |
| Payload-length | 1 | length of payload (n) |
| Packet sequence | 2 | Each component counts up their send sequence. Allows for detection of packet loss. |
| System ID | 3 | Identification of the SENDING system. Allows to differentiate different systems on the same network. |
| Component ID | 4 | Identification of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot. |
| Message ID | 5 | Identification of the message - the id defines what the payload “means” and how it should be correctly decoded. |
| Payload | 6 to (n+6) | The data in the message, depends on the message id. |
| CRC | (n+7) to (n+8) | Check-sum of the entire packet, excluding the packet start sign (LSB to MSB) |

其中，第一个字节标识起始标志位（STX），在v1版本中以“FE”作为起始标志。

第二个字节标识有效载荷长度（LEN），即有效载荷（PAYLOAD）数据的字节长度。

第三个字节标识消息帧序号（SEQ），用于接收端根据序号与接收到的消息帧数量，计算发送过程中消息丢失的比例，即信号强度，在每一个消息帧发送完毕后，该序号会加1。

第四个字节标识系统ID编号（SYS），用于标识MAVLink消息帧的不同发送设备。

第五个字节标识部件ID编号（COMP），用于接收端区分MAVLink消息帧是由发送设备的哪个单元发送的。

第六个字节标识消息包编号（MSG），用于标识有效负载的含义，以便接收端选择负载解码的正确方式。

最后两个字节标识校验号（CRC），用于校验消息帧在传输过程中是否出现差错。

**2.3 MAVSDK介绍**

MAVSDK是一个封装了MAVLink协议的开源项目，该项目支持C、C++、Python、Go、Swift、Java等多种语言，极大方便了使用MAVLink协议设备的项目开发，并已得到了业界领先机构的认可。该SDK提供的主要API包括：

表2.2 MAVSDK主要API

|  |  |
| --- | --- |
| 名称 | 简介 |
| Action | 提供了基本动作API |
| Geofence | 提供了操作禁飞区的API |
| Camera | 提供了操作摄像机的API |
| Gimbal | 提供了操作云台的API |
| Mission | 提供了操作飞行任务的API |
| Telemetry | 提供了获取GPS等信息的API |
| Action | 提供了基本动作API |
| Geofence | 提供了操作禁飞区的API |

其中MAVSDK-Java使用了RxJava编程框架，该框架是响应式编程的Java语言版本，其基于“观察者模式”实现了异步编程接口，可以链式的调用，简化了代码的逻辑。RxJava的核心思想是将事件分为起点、终点两部分，起点（又称上游）通过“订阅”将事件依次传递给终点（又称下游），事件在流动过程中，可以使用“操作符”进行操作，每个操作获取上一个操作传递下来的结果，然后对事件进行各自需要的处理，并将处理结果传递给下一个操作，这样事件就从起点经一次次处理流向终点。

**2.4 本章小节**

本章首先介绍了基于Android系统开发地面APP所需的Android平台架构，接着介绍了用于地面站系统与无人机器人设备通信所需的MAVLink协议，以及开发过程中所需用到的封装MAVLink协议的项目MAVSDK，为之后的设计与实现工作奠定了良好的理论基础。

**3 地面站APP的分析与设计**

**3.1 需求分析**

由于对陆地场景下无人车的辅助控制可以简单理解为去除高度控制后的、对空中场景下无人机的辅助控制，因此只需要实现对空中场景下无人机的辅助控制即可。而在陆地和空中场景下的可视化需求有所不同，陆地场景下需要能控制云台，调整摄像头角度，获取前、后、左、右方位的现场画面，而空中场景还需要获取上、下方位的现场画面，因此需要能满足两种场景下的需求。

综上，在地面站APP中，需要实现的功能有：

（1）无人机定位与飞行数据显示。地面站APP需要能够显示地图信息，要求能在与无人机建立连接后，根据从无人机实时获取的飞行数据，更新无人机位置，显示在地图上，并显示飞行轨迹。

（2）无人机的飞行控制。地面站APP通过WiFi连接数据传输设备地面端后，需要能够向设备发送MAVLink消息包，进而操作飞行控制器实现无人机的飞行控制功能。

（3）实时现场画面显示。地面站APP需要能够满足陆空两种场景下的可视化需求。需要能从空中无人机或地面无人车装配的摄像头上，以相对较低的延迟获取实时画面信息，显示在APP界面，并能左、右、上、下调整可视角度，同时能拍照、录像保存现场画面。

（4）数据存储。地面站APP需要能够记录每次飞行的数据，并以文本的形式保存；能够记录实时现场画面，并以JPG格式保存；能够记录实时现场视频，并以MP4格式保存。

**3.2 总体设计**

**3.2.1 总体设计方案**

根据上文的背景介绍和需求分析，本APP将针对使用Mavlink协议飞控和WiFi数传的四旋翼无人机开发辅助遥控等功能，同时针对陆空两种场景的无人机器人开发可视化功能。

本APP按功能将系统分为地图导航模块、飞行控制模块、图像传输模块三大模块，基于Android平台开发，使用Java语言设计逻辑代码、XML语言设计显示页面，主要使用百度地图SDK、MAVSDK和UVC摄像头SDK实现有关功能，并在开发过程中结合Gazebo仿真环境测试，开发完成后在实验环境中测试。地面站APP的总体设计如下图所示：



图3.1 地面站APP总体设计图

**3.2.2 模块功能设计**

（1）地图导航模块用于实现地图显示、无人机定位、飞行轨迹显示等功能。该模块通过地图视图，可以在界面上设置中心点，显示初始位置，并能根据从无人机获取的位置信息更新无人机位置、绘制飞行轨迹。

（2）飞行控制模块用于控制无人机进行起飞、降落、上升、下降、前进、后退、向左、向右、悬停等动作，可以将操作结果以一定形式展现给用户，并更新位置信息，以方便地图导航模块绘制信息。同时在飞行结束后可以将飞行数据以文本形式保存至手机本地。

（3）图像传输模块用于实现预览实时画面、调整角度、拍照、录像功能。地面站APP连接到无人车或无人机图像传输设备接收端后，可以获取实时现场画面，调整可视角度，并能根据获取到的画面数据拍照保存或录像记录，存储至Android手机。

**3.3 模块详细设计**

**3.3.1 地图导航模块设计**

地图导航模块是将飞行操作结果可视化的核心，本系统地图导航模块逻辑代码将在MapActivity.java中完成，主要基于“百度地图SDK”实现功能。百度地图SDK能够提供地图显示、定位、绘制等多种功能，完美满足本地面站APP地图导航模块的需要。用户进入地图页面后，系统将首先加载地图视图，将\*\*\*大学作为中心点，并以一定比例尺缩放。当用户与无人机建立连接后，地图导航模块从无人机获取实时数据，显示飞行参数、重设中心点位置，以图标形式定位无人机，并在每次移动指令发出后绘制飞行轨迹。

地图导航模块的功能如下图所示：



图3.2 地图导航模块的功能图

其中，地图显示功能将使用到百度地图SDK提供的View子类com.baidu.mapapi.map.MapView，该视图放置在activity\_map.xml中；无人机定位功能将使用百度地图SDK提供的“绘制点标记”功能，通过在地图绘制无人机形状的OverlayOptions类标记，并根据无人机实时位置刷新标记位置来实现；飞行轨迹功能将使用百度地图SDK提供的“绘制线”功能，通过将无人机飞行过的路径抽象为一条折线，并使用PolylineOptions类在地图上绘制来实现。

**3.3.2 飞行控制模块设计**

飞行控制模块用于操作无人机，也是本地面站APP的核心模块，本系统飞行控制模块的逻辑代码也将在MapActivity.java中完成。该模块功能的实现依赖于MAVSDK，基于其提供的接口，无人机能实现的动作有：起飞、降落、上升、下降、前进、后退、向左、向右、悬停等。用户必须首先与无人机建立连接，将无人机起飞后才能进行移动操作。在飞行过程中，APP需要实时记录下无人机当前飞行数据，暂存在List集合中，并在飞行结束后记录到文本文件中，实现数据持久化。在无人机飞行过程中，用户可以悬停无人机以中断移动，并可在合适位置降落或返回起飞点，紧急情况下还可强制结束迫使无人机降落。飞行控制模块的功能如下图所示：



图3.3 飞行控制模块的功能图

其中，连接功能的实现将借助MavsdkServer类的run方法，通过UDP协议，向开放14550端口的WiFi数据传输设备请求建立连接；连接建立后，通过Telemetry类从无人机订阅数据，通过Action类从无人机订阅有关飞行操作；飞行数据记录功能将通过Java输出流写入到Android手机文件中实现。

**3.3.3 图像传输模块设计**

图像传输模块用于实现多方位的现场视频可视化，并能存储记录。图像传输模块的逻辑代码将在USBCamearaActivity.java中完成，有关功能的实现依赖于UVC摄像头（USB Video Class Camera）。UVC是一种专门为USB视频捕获设备定义的协议标准，得到了Windows、Linux、Android等多操作系统的支持。在Android平台上使用UVC摄像头可以借助开源项目UVCCamera提供的SDK，该项目对UVC摄像头使用过程中所需的类进行了很好的封装，本地面站APP可借助其提供的API实现实时预览、拍照、录像等多种功能，同时借助云台实现角度调整。图像传输模块的功能如下图所示：



图3.4 图像传输模块的功能图

其中，实时画面预览功能将通过UVCCamera提供的监听器实现，当系统检测到UVC摄像头插入时，自动获取画面数据，显示在TextureView中；角度调整功能将通过控制云台俯仰角与偏航角实现；拍照功能将通过SDK提供的capturePicture方法实现，只需传入图片保存位置；录像功能将通过startPusher方法开始录像，并通过stopPusher方法停止录像，保存视频至预设位置。

**3.4 界面设计**

地面站APP主要有地图导航界面和图像传输界面两个界面，均使用Android活动组件Activity实现，界面之间通过点击按钮跳转。

地图导航界面将在activity\_map.xml中完成，用于显示无人机位置、展示飞行数据、进行飞行控制操作。该界面采用相对布局，主体使用MapView视图显示地图内容，无人机实时位置、飞行轨迹信息也在地图视图上显示；顶部为ToolBar，左侧为“返回”按钮，右侧以菜单的形式存放无人机操作有关按钮，其中“强制结束”、“返航”按钮默认显示在界面上，以方便用户在危险时快速做出反应，其余按钮则需要点击“菜单”按钮才显示；左上方为一个半透明的TextView，用于显示经度、纬度、高度等飞行数据；右上方为“移动距离调整”按钮；下方无人机移动按钮使用ImageButton实现，为保证清晰度，按钮图片资源均使用矢量图。用户点击按钮即可移动无人机，按钮自左向右、自上到下依次为“前进”、“悬停”、“后退”、“左移”、“右移”、“上升”、“下降”。地图导航模块的界面设计如下图所示：

图3.5 地图导航界面

图像传输界面将在activity\_usbcamera.xml中完成，用于显示实时现场图像，提供角度调整、拍照、录像等操作按钮。该界面采用相对布局，主体使用TextureView视图显示预览画面；上方为ToolBar，以菜单的形式存放“拍照”、“录像”、“对焦”、“分辨率”按钮；左上方为Switch按钮，用于开启或关闭预览视频的声音；下方采用四个ImageButton，用户点击即可操作云台进行左、右、上、下方位调整，实现多方位的现场画面可视化。图像传输模块的界面设计如下图所示：

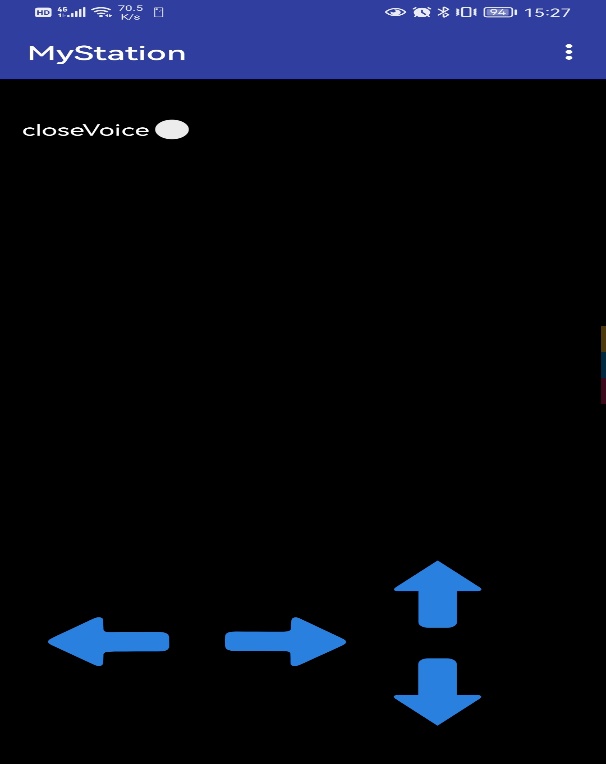
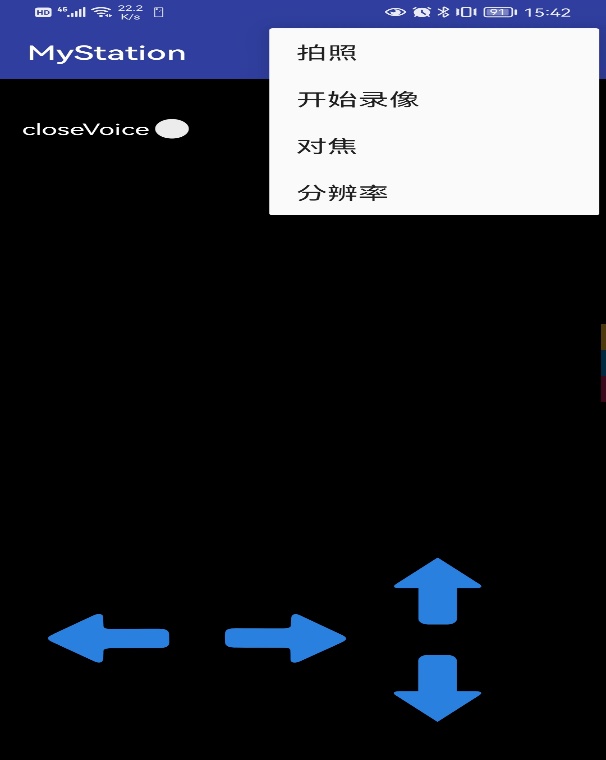
 

图3.6 图像传输界面

**3.5 数据存储设计**

由于本地面站APP需要存储的数据较为简单，因此不再额外设计数据库存储数据，飞行过程中需要记录的数据均以文件形式保存。根据上文需求分析，需要保存的数据包括无人机飞行数据、实时现场画面的图像、实时现场画面的录像。

其中，飞行数据包括无人机飞行过程中的实时经度、纬度、高度、俯仰角、偏航角、翻滚角、时间戳信息，系统每500ms新建一个MyPosition对象存放当前所有飞行数据，并将该对象添加至List集合中，在飞行结束后使用RandomAccessFile类将集合中存放的飞行数据以String的形式统一写入到文本中。

存储实时现场画面的图像首先需要使用Bitmap类获取图像数据，再通过其compress方法写入到BufferedOutputStream字节缓冲输出流中，生成JPEG格式的图片实现持久化。

对实时现场画面进行录像需要使用ByteBuffer字节缓冲类读取H.264编码的视频流，并通过Mp4MediaMuxer类重新封装视频流，生成MP4文件存储在本地。

**3.6 本章小结**

本章节按照软件开发中的设计流程，简单分析了地面站APP系统的需求，据此进行了地面站APP系统总体技术方案设计，按功能将系统划分为三大模块，又对每个模块的功能进行了详细设计。对系统各个界面进行了设计，确定了页面布局类型和需要用到的Android控件类型。最后对数据存储进行了设计。

**4 地面站APP的实现**

**4.1 开发平台与运行环境**

**4.1.1 软件开发环境介绍**

Android Studio是Google官方专为Android应用开发设计的、基于IntelliJ idea改造的集成开发环境，也是官方唯一推荐的开发功能。与传统的Eclipse、QT for Android软件相比，Android Studio优势明显，其启动、响应速度更快，UI界面美观易于操作，整合了Gradle构建工具，编辑器更加智能，插件生态完善，版本控制系统完美。因此，本地面站APP将基于Android Studio Arctic Fox (2020.3.1)版本开发。

**4.1.2 百度地图SDK环境配置**

地图导航模块有关功能的实现需要借助百度地图SDK，该SDK使用环境搭建过程如下。

（1）在Gradle中导入依赖：

dependencies {

implementation 'com.baidu.lbsyun:BaiduMapSDK\_Map:7.4.0'

implementation 'com.baidu.lbsyun:BaiduMapSDK\_Util:7.4.0'

}

（2）在中AndroidManifest.xml下的application中配置开发密钥，该密钥需要在百度地图开放平台控制台中获取：

<meta-data

android:name="com.baidu.lbsapi.API\_KEY"

android:value="key of develepment" />

（3）在AndroidManifest.xml中添加百度地图SDK需要的网络、储存读写等权限：

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

（3）在地图界面activity\_map.xml中新建一个MapView视图，用于显示地图：

<com.baidu.mapapi.map.MapView

android:id="@+id/mapView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:clickable="true" />

（4）最后还需要在Application.java中初始化地图：

SDKInitializer.initialize(this);  
SDKInitializer.setCoordType(CoordType.BD09LL);

配置完成后，即可在MapActivity.java中通过Java代码显示地图并使用百度地图提供的相关API。

**4.1.3 UVC摄像头SDK环境配置**

图像传输模块有关功能的实现需要用到UVCCamera提供的USB视频类SDK，该SDK的配置过程如下。

（1）首先需要到Github上下载开源项目UVCCamera的源代码，在Android Studio中打开并编译该项目的libusbcamera模块，可在outputs目录得到所需的libusbcamera-release.aar包。将该包与libs目录下的libusbcommon.aar包复制到本地面站APP项目的libs目录下，在Gradle中导入后即可使用：

dependencies {

implementation(fileTree("libs"))

}

（2）在USBCameraActivity.java中初始化mUVCCameraView对象用于显示实现预览视图，初始化mCameraHelper摄像头工具类对象，设置预览界面的分辨率、默认视频格式、绑定USB设备监听器，之后即可使用mCameraHelper提供的API操作UVC摄像头：

mUVCCameraView = (CameraViewInterface) mTextureView;  
mUVCCameraView.setCallback(this);  
mCameraHelper = UVCCameraHelper.getInstance(640, 480);  
mCameraHelper.setDefaultFrameFormat(UVCCameraHelper.FRAME\_FORMAT\_MJPEG);  
mCameraHelper.initUSBMonitor(this, mUVCCameraView, listener);

**4.2 地图导航相关功能的实现**

地图导航功能是地面站APP的基本功能之一，无人机的定位、轨迹显示都将在此实现。本功能的实现逻辑为：用户进入页面后初始化地图，默认将\*\*\*大学作为中心点；点击“连接”按钮后，系统从无人机订阅数据，通过TextView显示；点击“起飞”按钮后，系统将当前位置设置为Home点，用于用户“返航”操作；之后每次移动，地图视图将更新无人机位置、绘制无人机飞行路径。该部分的流程图如下图所示：



图4.1 地图导航模块流程图

**4.2.1 初始地图的加载**

当用户点击“开始”按钮进入后，地面站APP会进入MapActivity。在该页面中加载地图，显示无人机位置与飞行控制按钮。APP会默认将\*\*\*作为初始点，并以5m的比例尺显示。

通过4.1.2中的百度地图SDK环境配置后，在MapActivity中使用如下代码，设置地图缩放比例等属性，即可在页面中加载出地图视图：

MapStatus.Builder builder = new MapStatus.Builder();  
builder.zoom(20.0f);  
BaiduMap myMap = findViewById(R.id.mapView).getMap();

myMap.setMapStatus(MapStatusUpdateFactory.newMapStatus(builder.build()));

**4.2.2 地图中心点的实现**

地图创建完成后，会将\*\*\*所在位置设置为中心点，并在地图上显示一个中心点的标志。中心点的设置需要使用到百度地图SDK提供的OverlayOptions类，该类可用于设置地图覆盖物的属性，并通过BaiduMap.addOverlay()方法添加到地图上：

BitmapDescriptor bitmap = BitmapDescriptorFactory.*fromResource*(R.drawable.home);

OverlayOptions option = new MarkerOptions()  
 .position(GEO\_NJUST)  
 .icon(bitmap)  
 .anchor(0.5f, 0.5f);

myMap.addOverlay(option);

**4.2.3 定位功能的实现**

点击“连接”按钮后，地面站APP系统与无人机通过WiFi建立连接，订阅飞行数据，并根据数据以4.2.2中相同的方法刷新无人机实时位置，实现定位跟踪功能。由于订阅功能为一个内置子线程，而在Android应用子线程中操作UI控件会导致线程不安全，必须在主线程中操作UI控件。因此需要使用Handler机制，在子线程中向Handler发送Message，并在主线程的Handler中进行无人机位置更新操作。主线程中收到消息后的操作如下：

if (msg.what == MapActivity.UPDATE\_POSITION) {

clearOverlay(droneOverlay);

droneOverlay = drawOverlay(GEO\_DRONE, R.drawable.plane);

}

**4.2.4 飞行轨迹显示功能的实现**

无人机的飞行轨迹显示通过PolylineOptions类实现，当用户发送移动指令后，系统将无人机移动过的经纬度位置信息加入List<LatLng> points 集合中，通过百度地图SDK提供的OverlayOptions折线类设置宽度、颜色、路径等属性并绘制出来：

OverlayOptions myOverlayOptions = new PolylineOptions()

.width(10)

.color(0xAAFF0000)

.points(points);

myMap.addOverlay(myOverlayOptions);

**4.3 飞行控制功能的实现**

飞行控制功能是地面站APP的核心功能，用于操作无人机进行动作。本模块功能实现的逻辑为：用户点击“连接”按钮与无人机建立连接，接收从无人机传来的数据并根据数据初始Home点等信息；点击“起飞”按钮后无人机起飞至预设高度，起飞后即可进行“移动”操作，执行飞行任务；到达所需位置后可“悬停”以进行拍照、录像等任务；任务完成后可以在当前点“降落”或“返航”至Home点结束飞行。该部分的流程图如下图所示：



图4.2 飞行控制模块流程图

**4.3.1 连接功能的实现**

当用户点击“连接”后，MavServer开始运行，通过UDP协议向数传设备开放的14550端口发送数据，建立连接，同时实例化一个drone对象作为代理，接收订阅后的信息：

int port = mavsdkServer.run("UDP://14550");

System drone = new System("127.0.0.1", port);

Action action = drone.getAction();

Telemetry telemetry = drone.getTelemetry();

Gimbal gimbal = drone.getGimbal();

**4.3.2 移动功能的实现**

当无人机“起飞”后，用户可以操作无人机完成“前进”、“后退”、“上升”、“悬停”等动作，动作的逻辑基本相同，以“前进”动作为例进行介绍，MAVSDK没有直接提供完成前进动作的接口，因此需首先获取无人机当前位置，根据操作的类型增加或减少经度、纬度、高度，将新位置的坐标作为移动的目的地，发送移动指令，并在指令发送成功后绘制路径信息：

action.gotoLocation(next.getLatitude(), next.getLongitude(), next.getAam(), next.getYaw())

.doOnComplete( () -> drawTrajectory();).subscribe();

**4.3.3 飞行数据存储功能的实现**

在每次无人机位置更新时，将当前位置信息存入List<MyPosition> myPosition集合中，并在结束飞行时通过文件输出流保存为txt文件：

FileOutputStream outputStream = new FileOutputStream(FILEPATH, true);

outputStream.write(myPosition.toString().getBytes());

outputStream.close();

**4.4 图像传输功能的实现**

图像传输功能的实现逻辑为：用户进入图像传输界面时，系统将首先检查APP是否有所需权限，没有则弹出Toast提示用户；接着检查是否插入了UVC摄像头，若有则直接显示预览画面；之后用户可以调整角度，进行拍照或录像操作。该部分的流程图如下图所示：



图4.3 图像传输模块流程图

**4.4.1 预览功能的实现**

使用图像传输功能时，用户需首先检查图传设备发送端是否开启、接收端与手机是否正确连接。点击“图传”按钮，检查完权限后即可进入USBCameraActivity图传页面。预览功能是通过监听器实现的，经4.1.3的配置后，地面站APP检测到UVC摄像头设备已连接便可自动显示画面：

public void onSurfaceCreated(CameraViewInterface view, Surface surface) {  
 if (!isPreview && mCameraHelper.isCameraOpened()) {  
 mCameraHelper.startPreview(mUVCCameraView);  
 isPreview = true;  
 }  
}

**4.4.2 角度调整功能的实现**

用户点击“向左”、“向右”、“向上”、“向下”按钮时，会操作云台向相应角度偏转，调整摄像头朝向。向各方位调整的逻辑类似，以“向上”为例，云台控制的代码如下：

drone.getGimbal()  
 .setPitchAndYaw(90F,0F)  
 .subscribe();

**4.4.3 拍照功能的实现**

用户点击“拍照”按钮，调用mCameraHelper.capturePicture()方法截取实时视频图像，并保存到指定路径，保存成功后通过回调函数显示保存位置，回调函数的代码如下：

AbstractUVCCameraHandler.OnCaptureListener() {  
 @Override  
 public void onCaptureResult(String path) {  
 new Handler(getMainLooper()).post(new Runnable() {  
 @Override  
 public void run() {  
 Toast.*makeText*(this, "Save path:"+path, 0).show();  
 }  
 });  
 }

**4.4.4 录像功能的实现**

用户点击“录像”按钮后，系统开始录制现场视频，将视频保存路径、视频长度等参数设置到params中，通过mCameraHelper.startPusher()方法获取视频数据并存入本地，再次点击按钮停止录像，通过mCameraHelper.stopPusher()方法结束视频数据的写入并向用户发出录像完成的提示。开始录像功能的代码如下：

mCameraHelper.startPusher(params, new AbstractUVCCameraHandler.OnEncodeResultListener() {  
 @Override  
 public void onEncodeResult(byte[] data, int offset, int length, long timestamp, int type) {  
 FileUtils.*putFileStream*(data, offset, length);}  
 });

**4.5 本章小结**

本章在上一章节APP设计的基础上，对系统的具体开发过程进行详细的介绍，给出了系统各功能模块的流程图、阐述了实现方法。首先介绍了本地面站APP的开发平台与运行所需环境，详细介绍了实现功能所需的百度地图SDK和UVC摄像头SDK的配置方法。接着详细描述了地图导航模块、飞行控制模块、图像传输模块主要功能的实现过程与关键代码。

**5 地面站APP的测试**

地面站APP开发完成后，还需进行测试，由于疫情影响和场地限制，直接在实验环境中使用真机测试会有很多不便，本文将首先在仿真环境中进行测试，从理论上验证地面站APP是否可行，待系统功能基本完善后再进行实验环境测试。同时，由于陆地与空中场景的可视化功能实现逻辑类似，且在空中场景下的实现难度更高，本文将简单测试陆地场景下的可视化功能，而着重于测试空中场景下用于无人机的有关功能。测试过程中地面站APP将安装在系统版本为Android 10的手机上。

**5.1 仿真环境测试**

**5.1.1 仿真环境介绍**

仿真软件使用的是gazebo，这是一款免费的机器人仿真软件。该软件能够提供高保真度的物理模拟环境，具备一整套传感器模型，人机交互方式也十分友好，能够在复杂的室内和室外环境中准确高效地模拟机器人工作的功能，因而为无人设备开发者提供了优异的仿真环境。传统的gazebo仿真软件体积庞大、下载编译复杂，为此gazebo公司提供了px4-gazebo-headless docker镜像，模拟具备px4飞控的四旋翼无人机，且通过docker环境免去了繁琐的环境配置过程，大大提高了开发效率。使用命令： docker run --rm -it jonasvautherin/px4-gazebo-headless:1.12.3 <your-phone-ip> 即可下载镜像并使用，该镜像对外提供了14550端口监听QGroundControl软件、14540端口监听Mavserver。

**5.1.2 地面站APP功能测试**

将装有本地面站APP的Android手机与PC连接在同一网络环境下，PC端运行docker镜像、手机端运行地面站APP后即可使用。

在测试过程中，使用本地面站APP进行连接、起飞、前进、悬停、返航等操作，选用主流地面站软件QGroundControl作为参照，通过对比飞行参数、飞行轨迹可知，本地面站APP一切功能正常。

表5.1 仿真环境下地面站APP功能测试

|  |  |  |
| --- | --- | --- |
| 序号 | 待测试的APP功能 | 测试结果 |
|  | 地图显示 | 地面站APP能正确显示地图并能流畅移动、缩放地图 |
|  | 连接 | 地面站APP能够与无人机建立连接，接收无人机传递的飞行数据并显示在左上角 |
|  | 起飞 | 地面站APP能够操作无人机起飞至预设高度 |
|  | 移动 | 地面站APP能够操作无人机进行前进、后退、左移、右移、上升、下降等操作 |
|  | 定位 | 地面站APP能够正确显示无人机实时位置 |
|  | 轨迹显示 | 地面站APP能够正确显示无人机飞行轨迹 |
|  | 悬停 | 地面站APP能够操作无人机悬停在当前位置 |
|  | 降落 | 地面站APP能够操作无人机在当前位置平稳下降 |
|  | 返航 | 地面站APP能够操作无人机返回起飞点 |
|  | 飞行数据存储 | 地面站APP能够将飞行数据以文本形式储存 |
|  | 预览实时画面 | 地面站APP能够预览由摄像头传来的实时现场画面 |
|  | 拍照 | 地面站APP能够对实时现场画面拍照保存 |
|  | 录像 | 地面站APP能够对实时现场画面录像保存 |

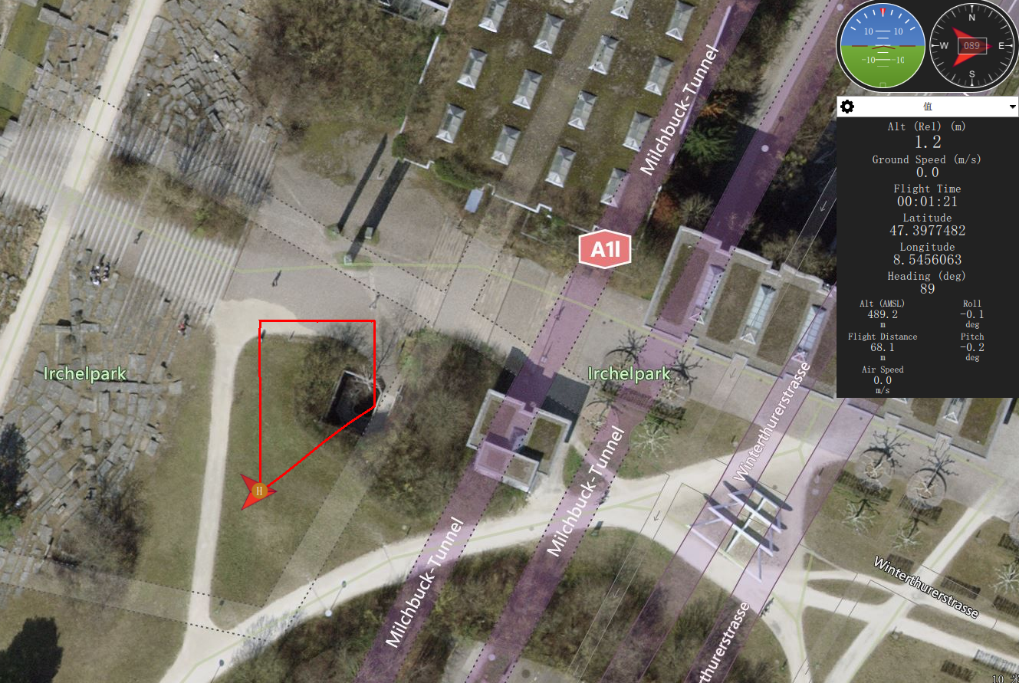
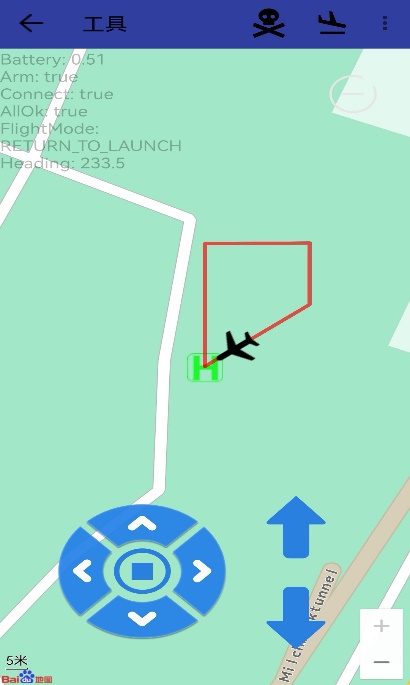
 

图5.1 仿真环境下的飞行对比

**5.2 实验环境测试**

**5.2.1 陆地场景下地面站APP可视化功能测试**

在陆地场景下，将UVC摄像头与云台安装在无人车上，建立连接后即可开始测试。测试过程中操控云台调整摄像头，预览前、后、左、右四个方位的现场画面，测试结果表明，地面站APP可视化功能一切正常。

** **

图5.2 陆地场景下的前方现场画面 图5.3 陆地场景下的后方现场画面

** **

图5.4 陆地场景下的左方现场画面 图5.5 陆地场景下的右方现场画面

**5.2.2 空中场景下实验环境介绍**

本地面站APP需结合硬件使用，在飞行场景下的实验环境中，使用的硬件设备介绍如下。

飞行控制器设备为雷迅创新（CUAV）公司生产的PIXHACK V3X模块，该飞控配有陀螺仪、电子罗盘、加速器、气压计等多种传感器，可运行PX4和APM环境，支持固定翼、多旋翼、无人车、无人船等多种机型。

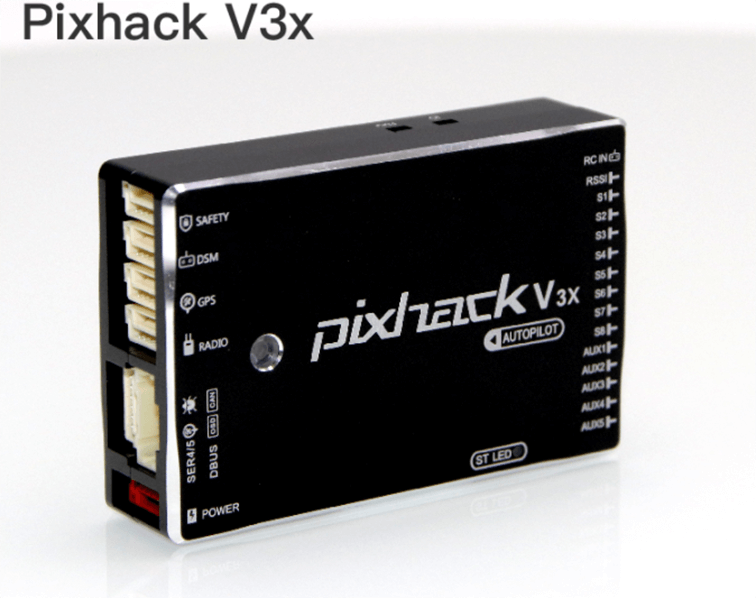


图5.6 V3X飞行控制模块

数据传输设备为雷迅创新公司的PW-Link模块，这是由该公司研发的一款2.4G无线WIFI数传模块，可以用于无人机近距离通信或调试，对外提供了UDP:\\14550端口用于连接。

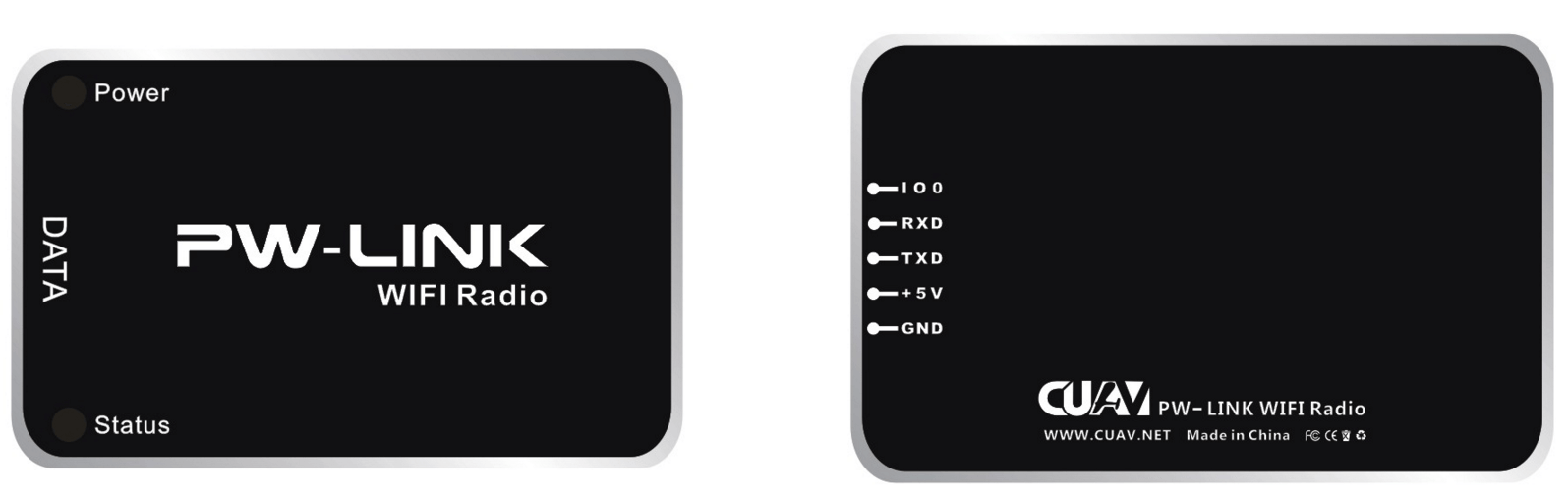


图5.7 PW-Link数据传输模块

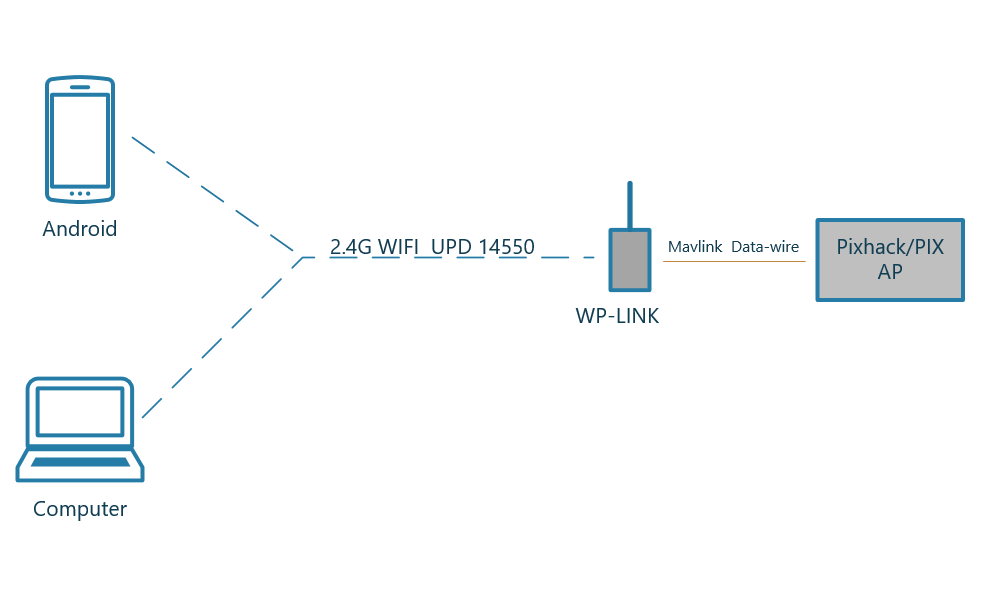


图5.8 PW-Link数传工作原理图

图像传输器设备为雷迅创新公司的VMR32模块，是用于移动设备的5.8G视频接收机。采用UVC免驱标准，支持大多数Android、Windows、Linux设备，具备OLED显示屏，可以显示频道、信号强度、电压信息，通过移动设备连接即可实时预览无线视频，使用方便。



图5.9 VMR32图像传输模块

**5.2.3 空中场景下地面站APP功能测试**

与仿真环境相比，实验环境更为复杂，不稳定因素更多。本次空中场景的实验测试选在学校荒地草坪，行人稀少、场地空旷、飞行安全。测试方法与5.1.2大致相同。经测试，本地面站APP一切功能正常，飞行实况、飞行数据文件等内容如下图所示：

表5.2 实验环境下地面站APP功能测试

|  |  |  |
| --- | --- | --- |
| 序号 | 待测试的APP功能 | 测试结果 |
|  | 地图显示 | 地面站APP能正确显示地图并能流畅移动、缩放地图 |
|  | 连接 | 地面站APP能够与无人机建立连接，接收无人机传递的飞行数据并显示在左上角 |
|  | 起飞 | 地面站APP能够操作无人机起飞至预设高度 |
|  | 移动 | 地面站APP能够操作无人机进行前进、后退、左移、右移、上升、下降等动作，但受信号干扰影响有时指令会丢失 |
|  | 定位 | 地面站APP能够正确显示无人机实时位置，但受GPS精度影响定位准确度有所下降 |
|  | 轨迹显示 | 地面站APP能够正确显示无人机飞行轨迹 |
|  | 悬停 | 地面站APP能够操作无人机悬停在当前位置 |
|  | 降落 | 地面站APP能够操作无人机在当前位置平稳下降 |
|  | 返航 | 地面站APP能够操作无人机返回起飞点附近 |
|  | 飞行数据存储 | 地面站APP能够将飞行数据以文本形式储存 |
|  | 预览实时画面 | 地面站APP能够预览由摄像头传来的实时现场画面 |
|  | 调整角度 | 地面站APP能够控制云台调整画面角度 |
|  | 拍照 | 地面站APP能够对实时现场画面拍照保存 |
|  | 录像 | 地面站APP能够对实时现场画面录像保存 |

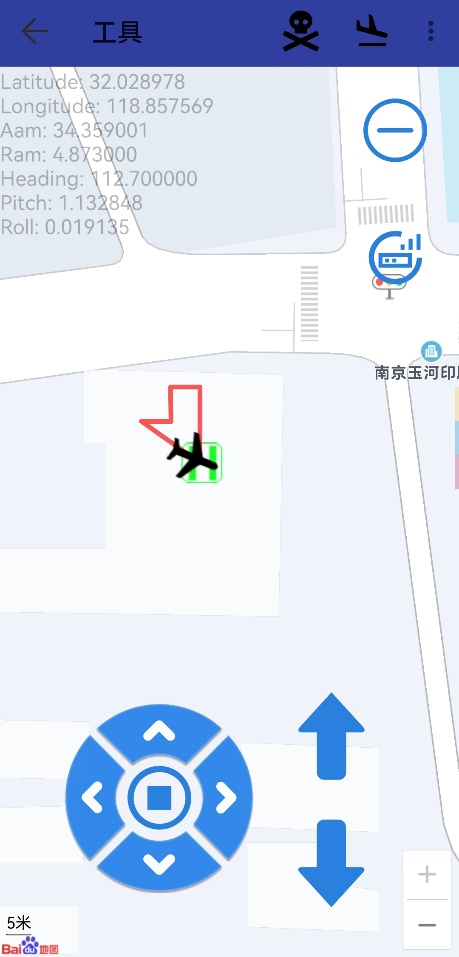
 

图5.10 空中场景下的实时现场画面 图5.11 实验环境下的地图界面

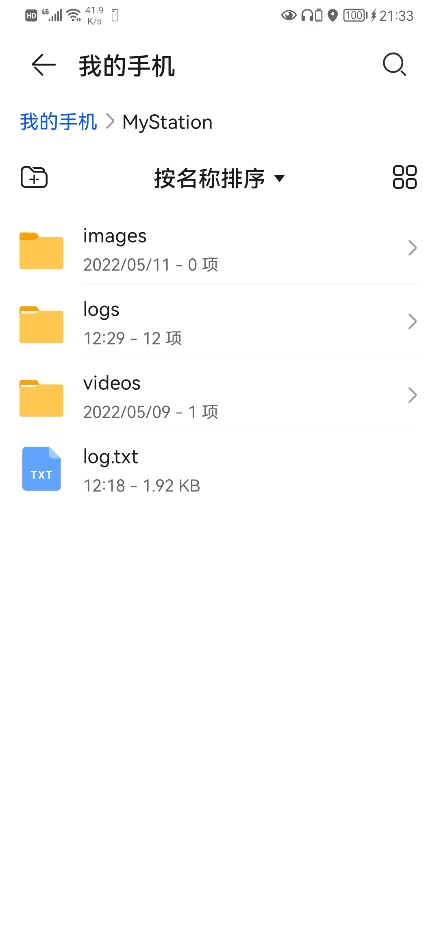
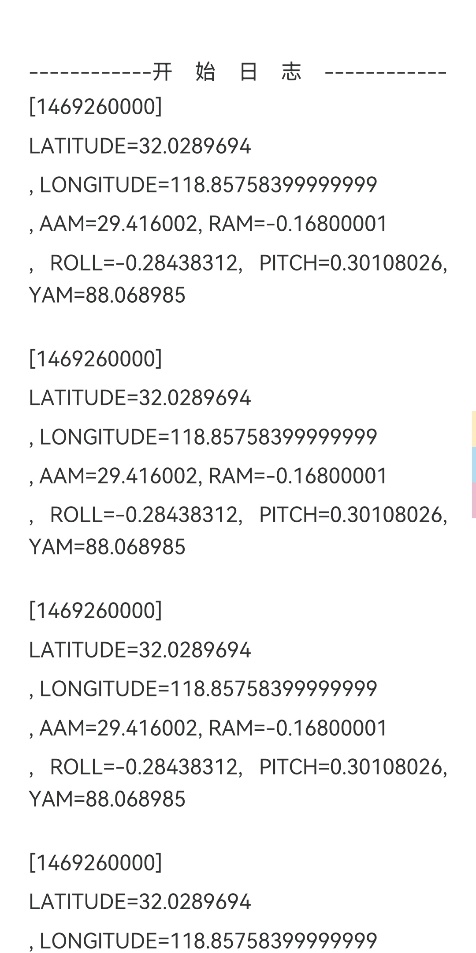
 

图5.12 实验环境下的数据存储位置 图5.13 实验环境下的飞行数据文件

**5.3 本章小结**

本章主要对地面站APP进行了测试，首先介绍了测试过程中使用到的软硬件环境，接着在仿真环境下对地面站APP的各项功能进行了测试，最后在实验环境下，对陆地场景下的可视化功能进行了简单测试，对空中场景下的辅助遥控、可视化等功能进行了较详细的测试。测试结果表明：地面站APP的所有功能运行良好。

结 论

本文主要开发了基于Android系统的地面站APP，该软件能够实现基本的无人机位置显示、飞行控制、图像传输等功能，代码逻辑较为完整，人机交互界面简单易懂，达到了毕业设计所需要满足的要求和实现的功能。本文所做的工作如下：

（1）介绍了地面站软件的研究背景，并分析了国内外地面站软件的研究现状。

（2）分别介绍了开发基于Android的地面站APP所需的Android平台架构、MAVLink协议以及MAVSDK的有关知识。

（3）基于以上知识准备，遵循软件开发的流程规范，经过需求分析、总体技术方案设计、软件功能模块设计、软件界面设计、数据存储设计，编码开发了地面站APP，实现了地图导航、飞行控制、图像传输、数据存储等多种功能。

（4）对地面站APP进行了仿真与实验两种环境下的测试，验证了地面站APP的可用性。

由个人能力有限、时间仓促，加之疫情影响，进行实验环境测试困难，本文虽取得了一定研究成果，但仍在某些方面存在这不足，还需进一步改善，具体内容如下：

（1）APP实现的功能较为简单，与主流地面站软件相比，缺少自定义飞行任务、设置飞行参数等功能。

（2）数据传输模块通过WiFi与Android手机连接，受外界干扰影响较大，控制功能随距离的增加会出现不稳定的情况。

（3）图像传输设备性能落后，在传输时延与传输图像质量上与当前主流设备存在一定差距。

随着硬件设备的更新以及软件后续的完善，相信该地面站APP能够凭借其简单易用性、可扩展性在无人机器人领域得到一定应用。