## 无人机APP控制端开发工作日记

8月6日（Day 01）

今天我们开发的第一天，具体任务就是定出下面9天我们的具体计划，然后明确一个目标。每个人都有好多很好的想法，我觉得其中最好的几个就是3G／4G信号控制无人机、拍摄全景地图、利用无人机进行传感器节点的数据收集。最后我们投票决定做两个方面：3G／4G网络控制无人机、拍摄全景地图。

明确了目标以后，我们进行了分组——3G组：杨知水、王晓达、我（App端），任乐健、朱哥、刘云飞（Onboard组）；全景地图组：王帅，林高杰。

在听了大家的介绍ppt后，我们从刘云飞那里知道了树莓派——一种微型主机，可以承载系统从而进行编程化操作。我们也制定了初始的传输计划：3G点对点——这个想法在后面进行的过程中发现是不合适的。

在我们App组讨论的过程中，一开始本来想杨知水和王晓达做Android端，我做IOS端，但后来分析任务之后，觉的还是3个人一起做Android端比较好，这个决定也让我们在之后解决问题的时候效率非常高，这绝对是一个非常正确的决定。

在决定我们做Android端App开发之后，我们开始分头查找资料，配置Android环境：

1.在查找资料的过程中，我们发现移动网络的IP地址，需要经过很多层NAT转换，所以我们并没有办法得到移动端的真正公网IP，所以想实现移动端1对1是非常难实现的，所以后来我们决定通过服务器转发的机制，进行对飞机的控制。

2.因为对飞机的控制要实时，并且尽量提高网络传输的稳定性与安全性，我们选择了用TCP协议的Socket编程。（在之后开发的过程中，我们也发现了从来没有想到过的问题－粘包）

3.配置Android环境：询问过开发Android程序的同学之后，我们决定要用AndroidStudio。在配置环境的时候，我们首先改了自己本地的Hosts文件，这样才能访问外网，从而进行Android SDK的更新。这个工作在Windows上很快就解决了，但在Mac上，一直没有找到一个好的Hosts，由于之前完全没有经验，就一直一边查找资料一边尝试，最后弄到晚上2点多才把环境配置出来。Mac配置Android环境具体步骤：Android官网下载AndroidStudio🡪打开AndroidStudio，在自动更新Sdk部分可能会卡住，说找不到代理🡪改变AndroidStudio中的idea文件，使AndroidStudio可以不检查SDK直接进入界面🡪去官网下载简洁版SDK🡪设置AS路径到简介SDK🡪更改Hosts文件（一定要找到一个有效的）🡪打开SDK Manager进行SDK更新 🡪 如果找到的Hosts文件是对的（里面一定会包含一个203.208.49.162 dl.google.com ， 203.208.49.162 dl-ssl.google.com 才能正确下载），那么SDK就会进行更新。🡪一定要更新所有的Tools类文件和至少1款API 🡪至少有10多个G🡪更新成功之后，就可以进行使用了。

8月7日（Day 02）

今天我们走了许多得弯路。因为我们是第一次接触Android编程，好多东西都一知半解。由于之前对java控件的事件监听机制有一些了解，便去网上查了相关的博客，了解到如何获取控件id等方法。一开始看到Android工程中的Activity和.xml文件，不是很理解它们的关系，感觉Android工程目录非常乱，不是很能缕清楚，所以就请教了任乐健帮我们讲解了一下，因为他以前做过相关的开发。

具体的结构是：

1. Manifest.xml 是一个总的工程概述类文件，所有的工程配置都在其中，比如说app的应用图标，Activity入口，所需要的权限，还有最基本的配置信息都在其中。
2. Activity.java 是一个界面的后台java代码，就相当于一个手机app某个界面的大脑，指挥着界面的跳转和响应。
3. ＊＊＊.xml 是对应某个Activity的界面布局，里面保存了界面的id等相关属性，即UI。

在了解了相关结构之后，我们便开始了初步Android编程。

今天遇到的第二个问题，就是我想把按钮的背景用图片替代，从而作出一个比较好的效果。一开始我尝试的是直接将图片拖拽进drawable文件夹中，未果。

后来在网上找了好久（因为好多篇博客说的方法都不一样），最后的解决方法是将图片复制进drawable文件夹，然后只需要直接调用它的名字即可。这里要注意的是：

1. 图片尽量是png格式
2. 图片不能过大
3. 图片的名字里不能有大写字母，并且不能以数字开头
4. 调用图片时，用findViewById（R.drawable.图片名）来获取图片从而进行使用。

最后，我们进行了分工，最初的分工是我写Socket和界面部分，杨知水也百度地图部分，王晓达做整体app的规划和UI设计。

8月8日（Day 03）

今天我们开始按着昨天的分工干活了。

我第一件事做的就是查找并实现界面之间的跳转，任乐健告诉我要用Intent，然后我就去网上找了它的用法，并且实现了界面的跳转。

第二件事，我开始编写Socket部分。我之前对Socket有所了解，写了一个代码，不过一直报错，历经好多波折，才知道Android编程中，Socket等获取数据的任务，都必须放在子线程中执行，不能放在主线程。最后我是这么理解的：主线程负责给用户呈现界面，而获取数据等需要等待的活动需要放在子线程中，当拿到数据后就可以呈现在主线成上。以上过程，还需要借助的一个东西就是Handler，下面是具体的步骤：

1. 定义子线程
2. 建立Message对象
3. 用Bundle将数据打包
4. 用Message 将数据传出
5. 编写Handler接收Message传出的数据给主线程
6. 更新主线程UI

8月9日（Day 04）

今天主要进行了Android端Socket编程，服务器Socket编程以及测试其运行效果。由于前一天懂了Android中Socket编程的原则和套路，今天进展的就非常快了。不过在修改服务器端的程序时，一直有一个异常，我们Debug了一下午，依然没有什么进展。

8月10日（Day05）

今天的任务很艰巨，首先我要将界面的每一个按键和地图写入响应，其次我们要和飞机进行通信调试，最后我们要商量数据包的结构和编写解包模块。

1.写入响应相对比较容易，很快就完成了。

2.和飞机之间的通信，这个部分花费了我们大量的时间。由于我的Socket是用java代码编写，而飞机端用的是C＋＋编写，一开始我用的是字符流进行读取，但并没有成功，后来使用字节流便可成功得到数据。

3.最后就是编写解包代码。在做这件事之前，我们首先要商量包的结构，这也是我们后期最意想不到的问题。由于我们的包结构设计的不是很合理，而且考虑的范围非常狭窄，导致当我们想对数据的种类进行扩展时，造成了非常打的麻烦。

首先先是包头，由几个大写字母开始，这个代表了它是指令，或者是传回的参数等分类。接下来，如果是数据包的话，就按着一个格式进行发送，那时我们并没有考虑把包长度作为参数加入包中，便利用正则表达式写出了一个解包程序。但在后来的数据传输中，发现有时这样的包格式会出现问题，所以我们又稍加改动，最后定格了一种包格式。

下面最大的问题就来了，这个问题也是我们意想不到且十分不好解决的问题——粘包。

用Socket通信时，当我们的数据发送量过快的时候，或者网络不稳定的时候，就会出现粘包现象，尤其是一些飞行数据，需要大批量的传回app端，但有时3G信号不好的时候，大量的粘包现象就会产生，从而程序就会报错，最后直到崩掉。

我们在发现这个问题的时候，已经是晚上了，所以我们决定用明天全力解决这个问题。

8月10日（Day06）

今天一大早来，我们便开始解决粘包问题。王晓达之前说过我们包的结构设计的不合理，不过当时我们都没有很在意，但这个时候，这个严重的问题暴露了出来，我们面临的是两个选择：一是重新更改包结构，这个需要花费大把的时间，而且就意味着我们之前写的好多代码都废掉了；二是在原来包的基础上多加一个包长度，这样就可以在所有粘包的包中提取出来一个完整的包了。

我们选择了后者。

由于飞机端的数据传递非常快，基本10毫秒刷新一次，而手机端却跟不上这样的速度，所以使用后者的方法，既可以得到完整的包，又可以将保证飞机上数据的实时性。

这个解包程序是由王晓达来写的，最后只需要放到之前预留在主程序上的位置，就可以进行使用了。

至此，我们基本上完成了整个App的开发，下面的几天就是对App进行优化了。

1. 代码优化
2. 界面优化
3. 数据压力测试
4. 进一步Debug