1. 课题研究概述（不少于300字）

随着车联网的迅速发展，车载智能终端逐渐成为车辆配置的标配，而车辆合理调度监控以及交通问题的优化解决迫在眉睫。目前GPRS网络覆盖已经非常完善，基于Android智能手机平台的电子地图应用也迅速发展，使车辆的移动定位以及实施导航更加便捷。

本课题先从开发难度、开发成本、用户体验等角度比较了目前各种车载智能终端操作系统的优缺点，最终说明采用Android智能手机作为车载终端的数据采集设备。从Android智能手机的WLAN定位以及GPRS定位原理出发，重点研究车载智能终端工作原理，学习了Android智能手机端调用百度SDK实现定位的原理和流程；以及围绕车载智能终端而实现的包括车内语言图像实时记录与保存，车辆速度等数据的记录与保存功能的实现方式；基于Android的UI设计与实现机制。最终实现一款操作方便的车载智能终端，能够实现车辆定位功能，以及包括录音、录像在内的车内驾驶员的操作数据监控与记录上传功能。目前仍有一些操作逻辑不清晰导致软件异常崩溃、功能实现不成熟等问题，需要后续改进优化。

1. 目前已完成学位论文工作的内容及进展（不少于800字）

1. 操作系统的选择

操作系统是管理软件与硬件层面的处理程序，需要处理管理、内存配置、控制输入输出等一系列事件。车载智能终端作为一款移动终端，目前包括Linux、Symbian、iOS、Android、鸿蒙OS等。各种操作系统都有适应各自的生态环境，而Android作为移动智能手机的操作系统，与车载智能终端有很多相似的地方，并且Android开发简单，适配容易，开源性好，并且为触摸屏专门做了优化。这些优点使Android操作系统成为车载智能终端操作系统的主流。

|  |  |  |  |
| --- | --- | --- | --- |
| Windows | Linux | iOS | Android |
| 闭源 | 开源 | 闭源 | 开源 |
| 开发成本高 | 开发成本低 | 开发成本高 | 开发成本低 |
| 应用数量少 | 应用数量少 | 应用数量多 | 应用数量多 |

手机是每个人每天都在使用的智能工具，而且基本都配备了GPS等定位功能，依靠手机自身的硬件传感器、百度地图的开源SDK以及Android端一些围绕车载需求的小功能的实现，一部Android手机就可以变成一套完整的车载移动智能终端。

2. 车载定位技术

2.1 GPS定位技术

GPS定位技术就是依靠围绕地球的几颗卫星来确定当前GPS接收器的所在位置。具有全球性、全天候、高精度、自动化、高效益的显著特点。原理比较简单：通过卫星已知的位置坐标以及与地面接收器收发信号的时间差，计算出地面坐标。为了提高精度以及避免时钟不同步带来的定位误差，一般至少同时需要4颗卫星的信号坐标来参与计算。

随着全球定位系统技术的不断发展，美国以外的其他国家也建立了自己的一套卫星定位系统，例如中国的北斗（Compass）、俄罗斯的GLONASS、欧盟的伽利略（GALILEO）等。

用户体验上要面临多定位导航系统信号的组合以及选取问题，要根据各个卫星导航系统的不同特点及精度优势，选择最佳的导航卫星数据。目前我国北斗卫星定位系统已经成熟，正被广泛使用。提供免费的服务区开放服务以及面向授权用户提供的更安全的授权服务。

2.2 WLAN定位

WLAN定位就是无线定位的一种。最早做Wi-Fi定位的是Skyhook公司，目前数据库最大的是Google。Wi-Fi定位首先需要建立离线采集的数据库（例如谷歌的街景车），当一台设备准备从采用Wi-Fi定位时，它可以找到很多Wi-Fi热点，根据每个热点的信号强弱可以知道每个热点的SSID和Mac地址。由于Mac地址全球唯一，而且不容易移动，参照庞大的数据库就可以计算得到当前的位置。

2.3 GPRS定位

GPRS定位是指GSM系统上发展出来的一种定位业务，GPRS适用于间断频繁、少量的数据传输，并根据数据流量计费。目前国内的GPRS运营商有中国移动、联通、电信。GPRS在有信号的条件下，可以完全通过基站的经纬度来进行定位。精度较低，一般在百米左右，通常用来弥补GPS定位受天气、高楼等影响。

3. Android 操作系统编写

3.1 Android开发环境搭建

本终端采用 Java + Android Studio + Android 10.0为主要开发环境，终端最终在一加6T收手机上进行模拟调试。

在Android Studio 官网上下载所需的安装包，进行安装即可，过程比较简单。Android Studio 开发软件比较友好，自带Java开发环境。

Android操作系统框架是分层结构 ，包括应用程序层、应用程序框架层、系统运行库层以及Linux核心层。

其中开发者所需的API接口以及功能模块的实现都在应用程序框架层编写设计，包括活动管理、视图、资源管理等。Android系统包括四大组件：Activity、Service、Content provider、Broadcast Receiver。Android代码的调用一般在Activity的可见生命周期onStart()内进行。

3.2 Android 定位模块设计与实现

本终端依靠手机自带的定位模块，软件上主要调用了百度地图开源SDK，核心功能类包括BDLocation类、BDLocationListener类、LocationClientOption类、BaiduMap类等。具体设计与实现步骤如下：

1. 登录百度地图开发者网站获取Application开发密钥key

2. 下载库文件，并导入到开发环境的libs文件夹下，并将jar添加到工程环境中

3. 配置权限，在AndroidManifest.xml中添加所用定位权限，并在application标签下添加获取的密钥key。

4. 在AndroidManifest.xml文件中声明service组件。

5. 在MainActivity.java中对LocationClient类进行初始化，

6. 配置定位模式，显示参数等，通过LocationClientOption类传递给LocationClient类，并设置监听函数。

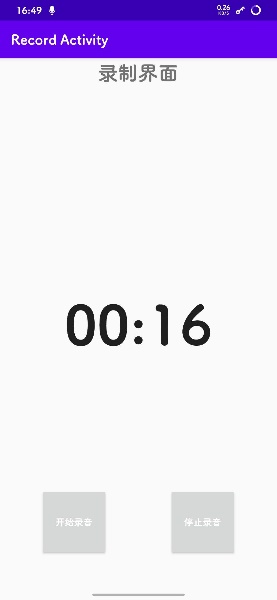
7. 初始化BDLocationListener接口，通过监听器获得定位信息，将数据记录下来并通过Toast类显示在屏幕上（后期考虑采用TextView控件进行显示并对信息进行存储）。

8. 初始化BaiduMap类，并通过MyLocationData类传输过来进行图层显示。

9. 最终与设计的Android UI中的Button控件进行绑定，以备后续调用。如下图所示：

3.3 Android 录音模块设计与实现

本终端设计硬件方面调用手机自带话筒，Android提供了两个API用于实现录音功能：MediaRecorder和AudioRecord。考虑到车内语音录制的时长以及压缩、编码格式等需求，MediaRecorder使用起来比较简单，因此软件方面采用Android系统的MediaRecorder类实现录音功能。具体设计与实现步骤如下：

1. new一个MediaRecorder类，设置音频格式、编码器以及保存录音文件的配置。
2. 编写startrecord函数和stoprecord函数，并与布局中的Button控件进行绑定。
3. 由于MediaRecorder类没有界面计时功能，考虑到用户的界面友好性，布局文件中加入了chronometer控件进行计时的显示，并编写timestart函数以及timestop函数，与Button分别绑定。
4. 最后注意内存释放。
5. 最终与设计的Android UI中的Button控件进行绑定，以备后续调用。如下图所示：

3.4录像功能模块的设计与实现

本终端设计采用调用硬件CMOS摄像头，通过软件代码控制，进行视频录像及存储。Android同样提供了两种方式实现视频信息的采集，本课题采用调用系统的Camera类进行摄像头的操控实现录制，以方便后续的视频参数及格式设定。软件方面使用SurfaceView类嵌套在视图页面中，通过SurfaceHolder控制Surface的接口，SurfaceView可以在自主创建的线程内部进行画面更新，避免了UI主线的堵塞问题。

代码实现如下：

//RESET MEDIA对象

mediaRecorder = new MediaRecorder();

camera = Camera.open(Camera.CameraInfo.CAMERA\_FACING\_BACK);

Camera.Parameters parameters = camera.getParameters();

parameters.setRotation(90);

camera.setParameters(parameters);

camera.setDisplayOrientation(90);

camera.unlock();

mediaRecorder.setCamera(camera);

mediaRecorder.reset();

//设置声源

mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);

//设置图像源

mediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);

//视频文件的输出格式

mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG\_4);

//声音编码

mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR\_NB);

//图像编码

mediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.H264);

当监听到Button的动作信息时，创建视频文件并进行录制

videoFile = new File(getExternalCacheDir() + "/video.mp4");

mediaRecorder.setOutputFile(videoFile.getAbsoluteFile());

mediaRecorder.prepare();

//开始录制

mediaRecorder.start();

timeStart();

Toast.makeText(mMediaRecorder.this,"开始录制",Toast.LENGTH\_SHORT).show();

Log.d("VideoRecord","开始录制");

startFLAG = true;

button\_startstop.setText("stop");

}

catch (Exception e){

e.printStackTrace();

}

}else {

//停止录制

mediaRecorder.stop();

timeStop();

mediaRecorder.release();

mediaRecorder = null;

Toast.makeText(mMediaRecorder.this,"停止录制",Toast.LENGTH\_SHORT).show();

Log.d("VideoRecord","停止录制");

button\_startstop.setText("start");

电脑屏幕的照片上有文字

描述已自动生成电脑屏幕截图

描述已自动生成最终效果如下图所示：

**停止录制画面**

**开始录制**

3.5数据上传至云端存储

对象存储（Cloud Object Storage，COS）是腾讯云提供的一种存储海量文件的分布式存储服务，具有高扩展性、低成本、可靠安全等优点。通过控制台、API、SDK 和工具等多样化方式，用户可简单、快速地接入 COS，进行多格式文件的上传、下载和管理，实现海量数据存储和管理。

本终端采用腾讯云存储iezlvsdr 产生的数据，COS可通过控制台、API、SDK等工具简单方便地接入，进行数据上传、下载和管理。并且成本很低，易于采用。

首先要注册腾讯云开发者账号，新建对象存储数据桶，，安卓端采用自动集成SDK的方式，在根目录的build.gradle中添加依赖：compile 'com.tencent.qcloud:cosxml:5.4.31'，为了方便调试，现阶段采用获取永久密钥的方式进行授权，代码如下：

public void send\_message(String Message,String cosPath){

context = getApplicationContext();

// 创建 CosXmlServiceConfig 对象，根据需要修改默认的配置参数

CosXmlServiceConfig serviceConfig = new CosXmlServiceConfig.Builder()

.setRegion(region)

.isHttps(true) // 使用 HTTPS 请求, 默认为 HTTP 请求

.builder();

String secretId = "My\_ID"; //永久密钥 secretId

String secretKey = "My\_Key"; //永久密钥 secretKey

QCloudCredentialProvider credentialProvider = new ShortTimeCredentialProvider(secretId, secretKey, 300);

CosXmlService cosXmlService = new CosXmlService(context, serviceConfig, credentialProvider);

// 初始化 TransferConfig

TransferConfig transferConfig = new TransferConfig.Builder().build();

// 初始化 TransferManager

TransferManager transferManager = new TransferManager(cosXmlService, transferConfig);

String bucket = "car-message-1301782340"; //存储桶，格式：BucketName-APPID

// 上传对象

//COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath, srcPath, uploadId);

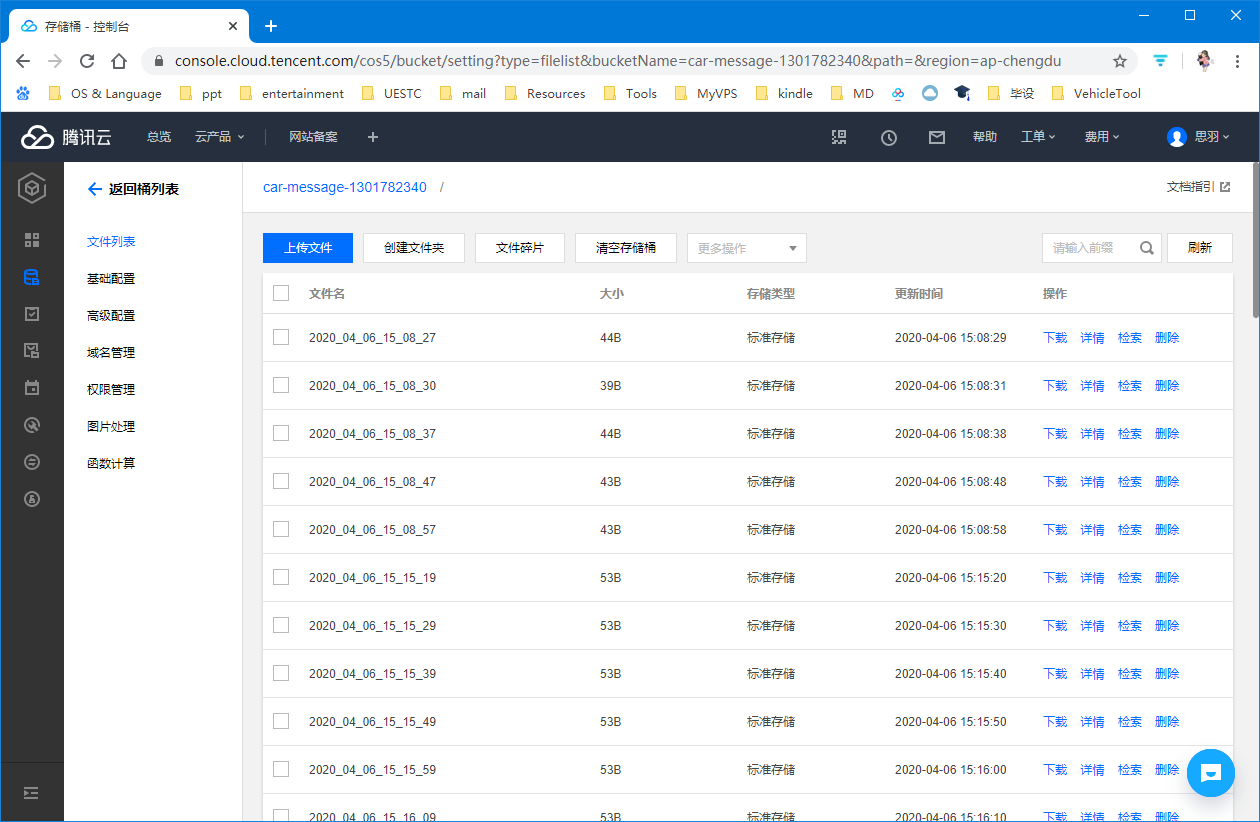
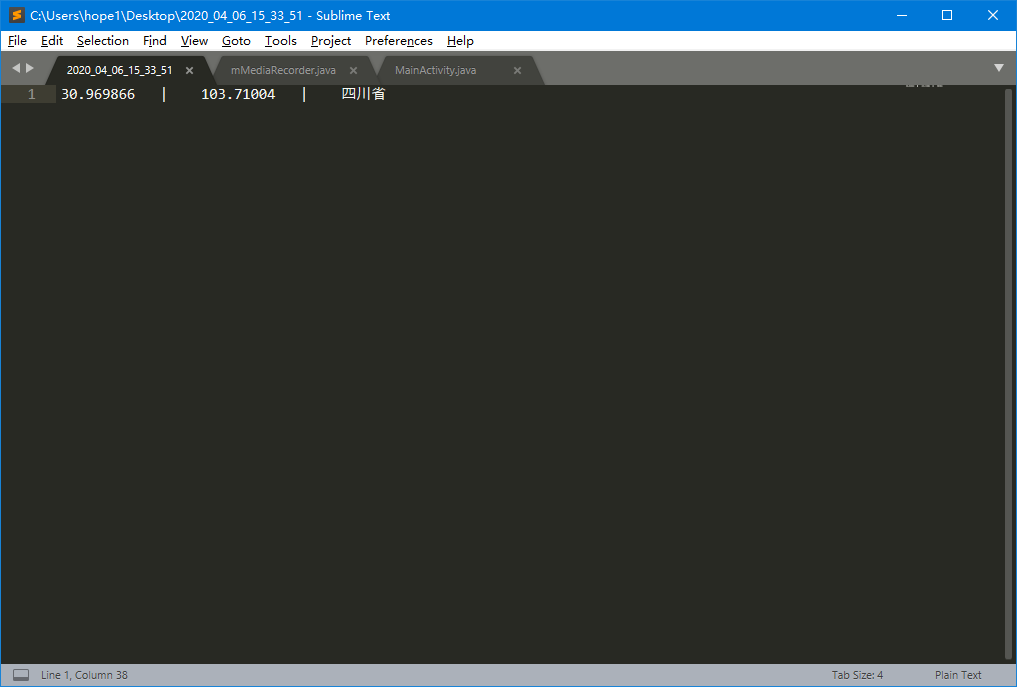
byte[] bytes = Message.getBytes(Charset.forName("UTF-8"));

COSXMLUploadTask cosxmlUploadTask= transferManager.upload(bucket, cosPath, bytes);

Log.d("SEND\_\_\_\_\_\_:","Success!");

}

在获取定位信息时将经纬度以字符串的形式保存至存储桶，保存数据间隔与获取位置信息间隔采取一致，最终数据上传效果如下：

可以看出设置的每十秒上传一次经纬度信息，云端的文件可以及时下载至本地，信息如下：