数据采集系统概要设计文档

## 1.系统的功能结构设计

基于 windows系统下采集GPS数据特点，采用多线程的方式实现串口数据的采集以及用不同的频率和数据格式发送到不同的平台。 实现与各个需求平台的对接，满足需求平台的不同的需求，进而让平台间更方便地完成自己的工作。

### 1.1 模块介绍

数据采集分为三大模块，即选择和连接串口、GPS数据中心和数据采集与解析，数据的发送线程。

串口选择和打开即选择适合的串口设置串口的参数，数据解析模块即将从串口收到消息按照对应的数据格式解析成需要的格式，再从中筛选有用的数据，数据发送模块即设置相应的发送对象和发送频率。

其中数据采集是主线程，数组采集线程和数据发送线程存活在项目的整个生命周期。

其中SinanReadingThread模块是用来完成串口的选择以及调用，收集GPS数据并进行数据的解析和筛选。Com模块是对串口进行参算设置，以及打开串口的设置。Udp模块来完成udp的设置以及发送。

### 1.2模块间的参数传递

模块间是通过相同的内存空间对参数的值进行初始化和修改的。这样多线程就可以保证自己所获取的数据是自己使用的最新的实时数据。

各个模块在不影响其他模块的的情况下对数据进行修改，使用和传递。

## 2. 系统处理流程

系统各个模块的具体处理流程及设计说明如下。

2.1 SinanReadingThread处理流程设计

#### 主要类的功能和调用关系

CopeSerialData

串口以及定位设备的选择，以及读取串口数据。其中包括三个函数

while(true)

{

int head = 0;

if(navType==3) //如果是华测

{

while((head+3<tail) && (strncmp(&buffer[head],"$GP",3) != 0)){

//printf("abcd\n");

head++;}

if(head>0){

tail-=head;

strncpy(buffer1,&buffer[head],tail);

strncpy(buffer,buffer1,tail);

}

if(tail<6)

return;

if(((strncmp(buffer,"$GPHCD",6) == 0) && (tail < packHCLen))) //packHCLen=103

return;

if((strncmp(buffer,"$GPHCD",6) == 0) && (tail >= packHCLen)){

tail -= packHCLen;

strncpy(hcMessage,buffer,packHCLen);

QString message=QString("%1").arg(hcMessage);

// qDebug()<<message;

QStringList list1=message.split(",");

QString latitude=list1.at(9); //纬度

QString longtitude=list1.at(10); //经度

QString speed=list1.at(8);

QString heading=list1.at(5);

//Car::p\_Instance1=latitude+longtitude+speed+heading;

double lat=latitude.toDouble();

double lng=longtitude.toDouble();

double truelat=pack2angle(lat);

double truelng=pack2angle(lng);

QString truelatitude=QString::number(truelat,'f',7);

QString truelongtitude=QString::number(truelng,'f',7);

double trueSpeed=speed.toDouble();

double trueHeading=heading.toDouble();

// qDebug()<<latitude<<longtitude<<speed<<heading;

Car::getInstance()->setp\_Instance1(latitude,longtitude,speed,heading);

Car::getInstance()->setjj(message);

Car::getInstance()->setPositionSpeed(message,truelongtitude,truelatitude,trueSpeed,trueHeading);

如上所示。我们用的是华测的设备，所以就会选着这一模块的代码进行数据的读取，以及华测设备的数据格式解析，最终放到我们需要的结构体中让udp数据进行调用。

Com

选择串口以及串口的初始化，对串口的各个数据进行设置：

Udp模块

发送的消息包括两种形式，结构体类型和string类型

第一个结构体类型：

|  |  |  |
| --- | --- | --- |
| **数据来源** | | |
| **name** | netPackage | **说明** |
| longtitude | unsigned int | **经度** |
| latitude | unsigned int | **纬度** |
| heading | short | **航向角** |
| speed | short | **速度** |
| data[CAN\_MSG\_DATA\_LEN] | unsigned char | **Can口数据** |

第二个string代表类型：

|  |  |  |  |
| --- | --- | --- | --- |
| **数据类型分类** | | | |
| **name** | **编码** | **说明** |  |
| **Haomiaoshu** | **1** | **当前的毫秒** | **自身** |
| **Longtitude** | **2** | **车辆** | **交通参与者** |
| **Latitude** | **3** | **车辆** |
| **Heading** | **4** | **车辆** |
| **Speed** | **5** | **车辆** |
| **设备号** | **6** | **设备** |