Smart Car Communication

整个系统分成三个部分：主控（MasterControl）, 主车（Master\_Car）, 从车（Slave\_Car）,现阶段没有考虑更多车的情况。其主要数据传输块图如下：

MainControl

msg

Master Car

monitorThread

Slave Car reflect

Slave Car

hd

rcvMasterMsgThread

MainThread

MainThread

rcvControlMsgThread

Master car reflect

mySendSlaveMsgThread

Key input ut

msg

msg

MainThread

MainThread

msg

msg

msg

msg

Turn msg

Speed msg

monitorThread / colorToCoord

Turn msg

Slave Car reflect

一、MainControl 模块介绍

该模块包括两个线程Mainthread, msgConsumeThread,下面分别介绍两个线程的功能。

1. Mainthread

该线程主要处理键盘的输入，将键盘的输入包装成结构体struct Msg\_Item { }表示的消息，并存入队列。

1. msgConsumeThread

该线程从队列中取出每个键盘消息，并通过网络发送给Master Car模块。

二、Master Car 模块介绍

该模块分室内和室外两种处理逻辑

* 室外逻辑

该逻辑包括Mainthread，rcvControlMsgThread，gpsThread和sendSlaveMsgThread四条线程。

1. Mainthread

该线程将做必要的网络初始化操作，它将建立网络服务端的监听逻辑，等待Maincontrol 和 Slave Car两个网络客户端的连接，接着将启动rcvControlMsgThread，不停得读取Maincontrol发送过来的控制消息，并将这些消息存入队列。

2、rcvControlMsgThread

有键盘消息存入队列时，该线程就取出消息进行处理，在处理消息时，分两种方式，

1. 如果是速度类消息，就直接发送给小车，并将消息存入到sendSlaveMsgThread线程将要处理的队列。
2. 如果是转弯类消息，就要通过GPS获取到当前小车的位置信息，并与gpsThread记录的起点信息一起计算当前的小车行驶的参考距离，并将该距离值加入到消息包，等待小车转弯完成，再将消息存入sendSlaveMsgThread线程将要处理的队列。

3、gpsThread

该线程主要获取小车转弯时用于计算参考距离的GPS起点经纬信息。在每次转弯后，经过一定的时间，就会重新记录参考距离的起点信息。

1. sendSlaveMsgThread

该线程直接将存入队列的消息包，转发到Slave Car。

- 室内逻辑

该逻辑包括Mainthread，rcvControlMsgThread和 sendSlaveMsgThread三条线程.

1. Mainthread

该线程将做必要的网络初始化操作，它将建立网络服务端的监听逻辑，等待 Maincontrol 和 Slave Car两个网络客户端的连接，接着将启动 rcvControlMsgThread，不停得读取Maincontrol发送过来的控制消息，并将这些消

息存入队列。

2. rcvControlMsgThread

有键盘消息存入队列时，该线程就取出消息进行处理，在处理消息时，分两种方 式，如果是速度类消息，就直接发送给小车，并将消息存入到 sendSlaveMsgThread线程将要处理的队列。如果是转弯类消息，就要通过 颜色传

感器获取到当前小车的位置地图的坐标信息，并将该距离值加入到消息包，等待小车转弯完成，再将消息存入sendSlaveMsgThread线程将要处理的队列。

3. sendSlaveMsgThread

该线程直接将存入队列的消息包，转发到Slave Car。

三、Slave Car 模块

该模块分室内和室外两种处理逻辑

- 室外逻辑

该逻辑包括Mainthread, rcvMasterMsgThread 和 monitorThread三条线程

1. Mainthread

该线程中将通过网络去连接Master Car， 连接上之后，就会不停查询并读取Master Car发送过来的控制消息，对消息分两种逻辑处理：

1. 如果是非转弯消息，就交由rcvMasterMsgThread线程处理。
2. 如果是转弯信息，就交由monitorThread线程处理。
3. rcvMasterMsgThread

如果是设置速度类消息，就由该线程取出消息直接发送给小车。

1. monitorThread

在每次转弯过后的一定时间（这个时间要与Master Car中用的时间一致）后就开始记录小车的当前位置信息，通过这个位置信息与小车的当前GPS信息计算出小车行驶的参考距离是否与Master Car传过来的参考距离匹配，若匹配Slave Car就进行转弯；否则就继续参考距离匹配的检测。

- 室内逻辑

该逻辑包括Mainthread, rcvMasterMsgThread 和 colorToCoord三条线程.

1. Mainthread

该线程中将通过网络去连接Master Car， 连接上之后，就会不停查询并读取 Master Car发送过来的控制消息，对消息分两种逻辑处理：

A. 如果是非转弯消息，就交由rcvMasterMsgThread线程处理。

B. 如果是转弯信息，就交由colorToCoord线程处理。

2. rcvMasterMsgThread

如果是设置速度类消息，就由该线程取出消息直接发送给小车。

3. colorToCoord

该线程从转弯信息队列中取出主车转弯时的颜色地图坐标信息, 并与自身采集到 的颜色坐标进行匹配,如果完全匹配就做转弯操作. 否则继续进行匹配,直到转弯信 息队列为空为止。

四、数据包定义

整个系统中数据的传递是以如下的结构体封装起来，细节见下：

struct Msg\_Item {

int index;

char type;

short key;

int val;

union {

struct color {

short r;

short g;

short b;

} color;

struct array {

int row;

int col;

} array;

struct gps {

double distance;

int angle;

}gps;

struct ecompass {

int angle;

}ecompass;

char buf[24];

}u;

};

-index: 数据包序号

-type： 数据包类型

-key： 键编码

-val： 按键对应的物理属性值

-struct color：颜色值的封装（目前没用）

-struct array：颜色地图行列坐标表示

-struct gps： gps算出的距离值（angle：目前没用）

-struct ecompass： 电子罗盘信息（目前没用）

type的取值范围：

enum { TYPE\_GENERIC = 1, TYPE\_SPECIAL = 2, TYPE\_DIRECTION = 4,TYPE\_ECOMPASS = 8};

TYPE\_GENERIC：up、down加减速按键类型

TYPE\_SPECIAL：p 停车按键类型

TYPE\_DIRECTION： left、right左右转弯按键类型

次数据报的流转经过以下几个阶段：

1 、maincontrol 将按键封装成数据包发给mastercar

2 、mastercat收到数据包后，如果是加减速包则不需改动直接转发到后面slavecar；

如果是转向包则要将地图信息或gps信息填充到该数据包中，再转发到slavecar

3 、slavecar收到数据包后，如果是加减速包，就直接下发给小车；若果是转向包，就要

通过传感器采集数据与数据包中的信息进行比较，若转弯条件满足，就控制小车转弯。

四、运行命令

1. MainControl

start D: \bin\main\_control.exe -rh 172.20.85.109 -sport 9999 –speed\_step 200

-rh: MasterCar的IP地址

-sport：MasterCar设置的网络监听端口

–speed\_step: 设置小车的初始速度

2. MasterCar

室外：（ip：172.20.85.109）

start D:\bin\master\_car.exe -rp COM5 -rb 9600 -sport 9999 -gps COM6 -sleepTime 5000 -masterGpsTest 0

-rh: 小车串口号

-sport： 初始采用网络端口号

-rn：等待几个客户端连接，现在最大为2

-gps： 接GPS的串口号

-hasClient：表示该小车将作为服务端，将数据包转发给后面的从车

-slaveServerPort：本车作为服务端时，设置的监听端口

-sleepTime：转弯后等待多长时间开始采集参考位置

-masterGpsTest 0: 在实际中该参数设为0

室内：

start D:\bin\master\_car.exe -rp COM5 -rb 9600 -sport 9999 -rn 2 -sensor 5 5 0 2 0 0 1 1 COM6

-sensor 5 5 0 2 0 0 1 1 COM6： 表示传感器串口号COM6，颜色地图是5\*5的矩阵，小车初始坐标位置（x, y）是（0, 2）, （0 0 1 1）表示小车行驶方向是沿着x轴正向。

3. SlaveCar

室外：

1、从车将作为服务端将数据转发到后一辆从车（ip：172.20.85.110）

start D:\bin\slave\_car.exe -rp COM4 -rb 9600 -server 172.20.85.109 -sport 9999 -hasClient -slaveServerPort 9998 -gps COM6 -sleepTime 5000 -slaveGpsTest 0 -delayGo 7000

-rh: 小车串口号

-server：mastercar 的IP地址

-sport： mastercar服务端监听端口号

-gps： 接GPS的串口号

-hasClient：表示该小车将作为服务端，将数据包转发给后面的从车

-slaveServerPort：本车作为服务端时，设置的监听端口

-sleepTime：转弯后等待多长时间开始采集参考位置

-masterGpsTest 0: 在实际中该参数设为0

-delayGo：该车在mastercar启动多久时，再开始运行

2、从车是车队最后一辆车 （ip：172.20.85.111）

start D:\bin\slave\_car.exe -rp COM4 -rb 9600 -server 172.20.85.110 -sport 9998 -gps COM6 -sleepTime 5000 -slaveGpsTest 0 -delayGo 14000

-rh: 小车串口号

-server：前一辆从车的IP地址

-sport： 前一辆从车作为服务端时的监听端口号

-gps： 接GPS的串口号

-sleepTime：转弯后等待多长时间开始采集参考位置（该值在车队中应该相同）

-masterGpsTest 0: 在实际中该参数设为0

-delayGo：该车在mastercar启动多久时，再开始运行（该值要根据第一辆从车设置的值进行计算所得）

室内：

1 、从车将作为服务端将数据转发到后一辆从车（ip：172.20.85.110）

start D:\bin\slave\_car.exe -rh 127.0.0.1 -rrtp 8101 -server 172.20.85.109 -sport 9999 -hasClient -slaveServerPort 9998 -sensor 5 5 0 0 0 0 1 1 COM6

- sensor 5 5 0 0 0 0 1 1 COM6： 表示传感器串口号COM6，颜色地图是5\*5的矩阵，小车初始坐标位置（x, y）是（0, 0）, （0 0 1 1）表示小车行驶方向是沿着x轴正向。

-hasClient：表示该小车将作为服务端，将数据包转发给后面的从车

-slaveServerPort：本车作为服务端时，设置的监听端口

2 、从车是车队最后一辆车 （ip：172.20.85.111）

start D:\bin\slave\_car.exe -rh 127.0.0.1 -rrtp 8101 -server 172.20.85.110 -sport 9998 -sensor 5 5 0 0 0 0 1 1 COM6

五、注意

整个程序分成室内与室外两个大的部分，在编译阶段需要分开编译，这个编译开关有宏 OUTDOOR 指定。

1. 室内逻辑

#define OUTDOOR 0

1. 室外逻辑

#define OUTDOOR 1

运行脚本

1 、Main\_Control.bat: 主控启动

2 、Master\_Car.bat: 主车启动

3 、Slave\_Car\_As\_Master\_Slave.bat: 作为服务端的从车启动

4 、Slave\_Car\_As\_Slave.bat: 最后一辆从车启动