第十届“飞思卡尔”杯全国大学生智能汽车竞赛

技 术 报 告

自平衡小车自动巡线运行控制的研究

学　　校：　大连海事大学

队伍名称：　逆袭二号

参赛队员：　黄翊峰 东东 汪星

带队教师：　滕国库 陈鹏

关于技术报告和研究论文使用授权的说明

本人完全了解第十届“飞思卡尔”杯全国大学生智能汽车竞赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和飞思卡尔半导体公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名： 黄翊峰 东东 汪星

带队教师签名： 滕国库 陈鹏

日　　期： 2015/8/18

引　言

第十届全国大学生“飞思卡尔”杯智能汽车竞赛是以“立足培养、重在参与、鼓励探索、追求卓越”为宗旨，鼓励创新的移向科技竞赛活动，竞赛分为光电直立组、摄像头组、电磁组三组，光电直立组要求在规定的汽车模型平台上，使用飞思卡尔半导体公司的微控制器作为核心控制模块，通过增加姿态传感器、道路识别传感器、电机驱动模块以及编写相应的控制程序，制作完成一个可以保持两轮直立姿态，能够自主识别道路的平衡车。智能汽车竞赛的赛道路面为宽度不小于 45cm 的白色面板，赛道两侧边沿有宽度为 25mm 的连续黑线作为引导线，参赛队员的目标是模型汽车需要按照规则以最短的时间完成单圈赛道。

本次比赛中，我们使用大赛组委会统一提供的 E 车模，采用飞思卡尔 MK60FX512VLQ15 单片机为核心控制器，采用 BTN7971 驱动车模电机，L3G4200D 和 MMA8452 作为姿态传感器，TSL1401 线性 CCD 作为道路识别传感器，自主构思姿态控制，速度控制和转向控制方案，引导车模按照规定路线识别行进。

在报告中，我们通过对整体方案、机械、硬件、算法等方面的介绍，详细阐述了我们在此次只能汽车竞赛中的思想和创新。包括电路设计、算法及辅助调试模块、机械结构等等。

第一章 机械结构

对于平衡车，要想实现高速稳定的运行，机械结构应当做到使重心尽可能的低，以减少电机在维持直立时的输出量，使直立控制抗干扰能力更强，同时，要想转弯灵活，机械结构应当使重心尽可能的集中，以减少车模转动惯量。

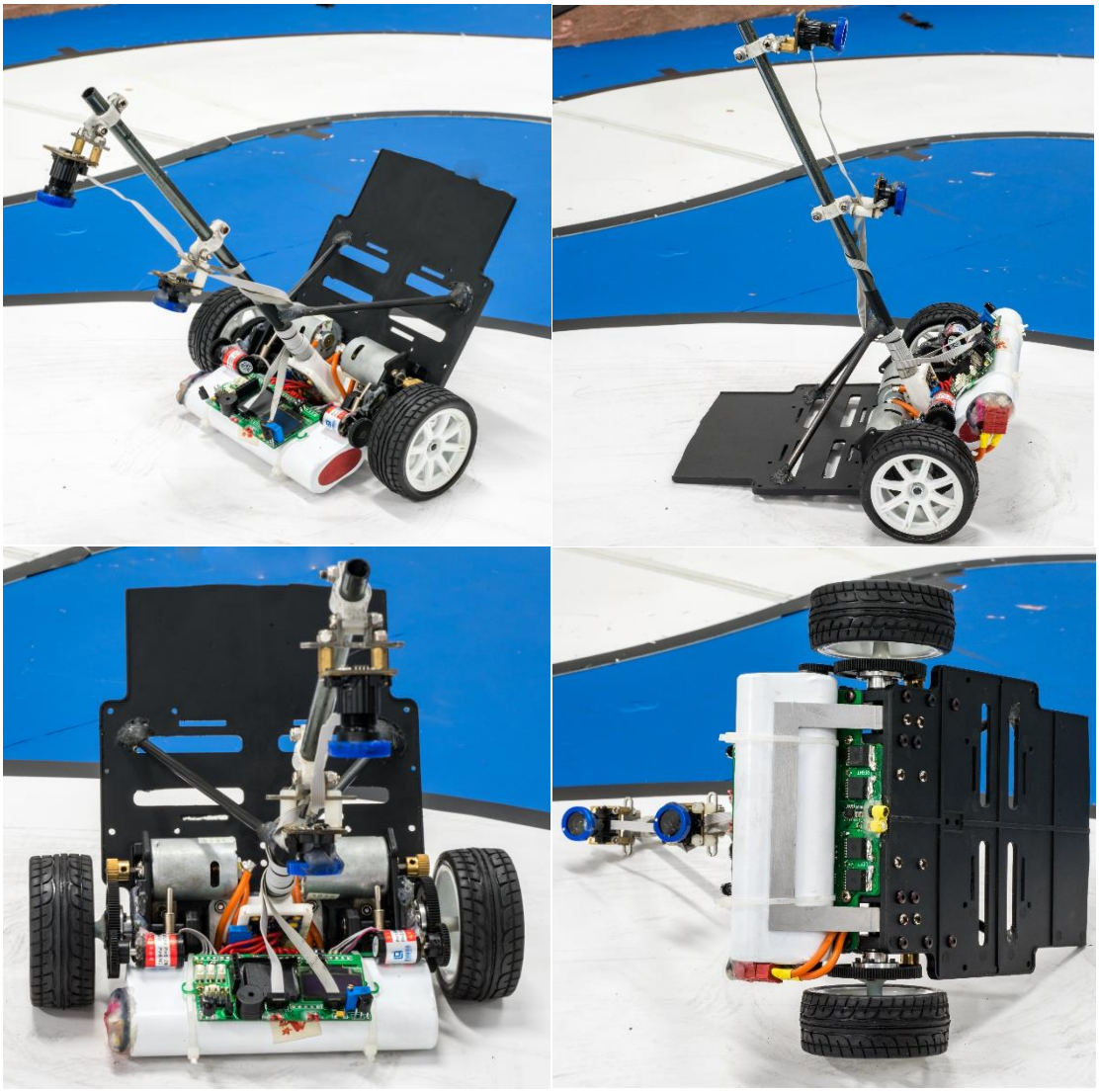
车模运行速度需要车模维持一定向前的倾角以获得加速度，车模运行也需要考虑坡道的影响，为了上下坡不挂底盘，在降低重心的同时需要保持一定离地间隙。本章介绍了平衡车机械结构，包括电池安装位置，线性CCD 安装位置，姿态传感器安装位置等。

1.1 电池位置、整体结构

为了使重心降低，我们采用了经典的”V”型机械结构，将电池放置在了紧贴车模前方的位置，和车的底盘和电机质量关于车轴对称，并将大部分的器件集中在车模前方，让车模的平衡位置尽量躺倒，这样可以使车模运行时重心更低，更稳定。

同时使用不锈钢制作了电池支架，在保证强度的同时进一步降低了重心，我们将车模原有的连接两个传动架的连接杆去掉，使用电机驱动板连接，保证了强度的同时也降低了重心，还使电池可以紧贴车模，保证了转动惯量不会太大，而为了进一步降低转动惯量和让车模平衡位置更靠后，我们还将车模头部笑脸位置锯掉。

车模结构、电池安装如下图：



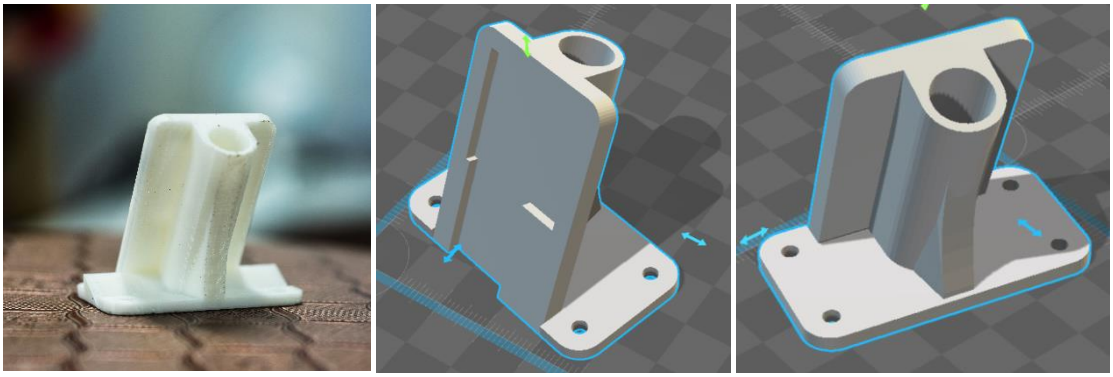
1.3 传感器位置

1.3.1 线性 CCD 安装

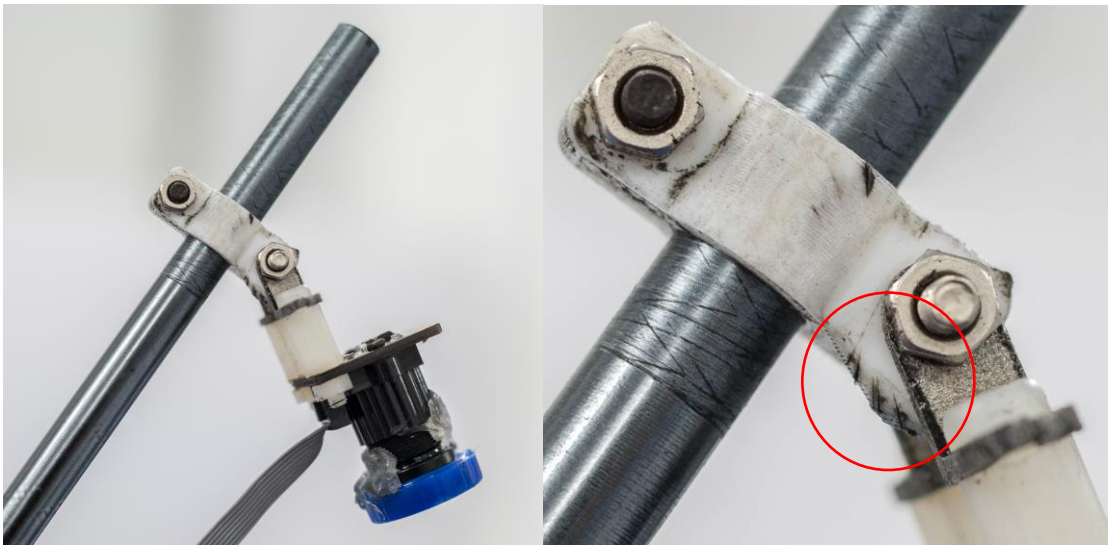
线性 CCD 是小车获取路径星系的传感器，需要一个稳定的安装方式以获得更稳定的图像质量，我们 3D 打印了线性 CCD 支架，并用空心碳纤维杆做支撑，以尽可能减小轴上方的重量。为了在车模转弯时 CCD 部分不产生额外的转动惯量和保证图像的稳定，杆子的角度应当尽量保证在车模

行驶时与地面垂直。

下图是 CCD 支架特写图和渲染图，同时它也承担了固定加速度计和陀螺仪的任务：



对于光电组直立车，在比赛时如果设定了不同的速度档位，转弯前瞻一定是不一样的，所以对于 CCD 的固定，我们使用 3D 打印和铝合金结合的方法，设计了一款带有刻度尺的 CCD 支架，安装时可用小刀在支架上划上一些刻度，可在比赛变速时精确调整需要的前瞻，如下图：



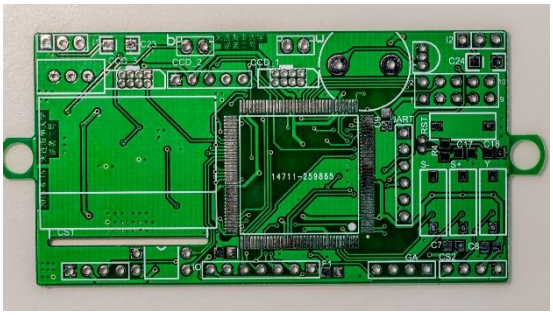
1.3.2 陀螺仪&加速度计安装

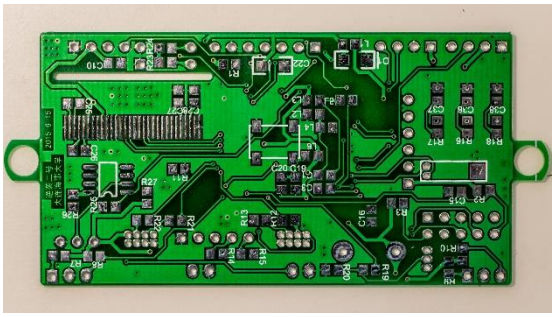
陀螺仪和加速度计用于测量车模姿态，需要和 小车紧密相连，对于

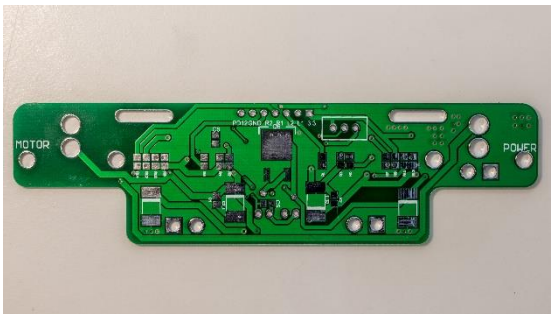
加速度计，为了减少运动产生的噪声，需要安装的尽可能靠近车轴，对于陀螺仪，则需要保证完全水平安装，否则转弯时会产生竖直方向的分量，形成加减速的现象，陀螺仪同时也测量了车模转弯时的角速度用来进行转向微分控制，我们将陀螺仪和加速度计安装到了固定 CCD 的支架上，MMA8452 的位置几乎和车轴重合，3D 打印件也保证了不会因安装误差造成陀螺仪的不水平，安装效果如下图：

第二章 硬件电路

硬件电路有两个电路板，一个是主板，包括 K60 最小系统，CCD、编码器、姿态传感器接口和电机 PWM 输出口，另一个是电机驱动板和电源系统，电池电源接入后做驱动电机和为主板供电用。两个板子如下图：







2.1 电源

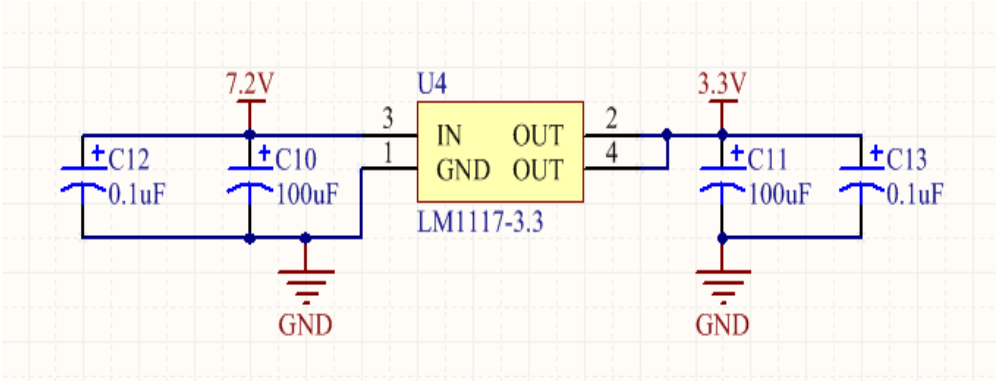
电源分为开关电源和线性电源，线性电源的电压反馈电路是工作在线性状态，开关电源是指用于电压调整的管子工作在饱和和截至区即开关状态的。线性电源一般是将输出电压取样然后与参考电压送入比较电压放大器，此电压放大器的输出作为电压调整管的输入，用以控制调整管使其结电压随输入的变化而变化，从而调整其输出电压，但开关电源是通过改变调整管的开和关的时间即占空比来改变输出电压的。

从其主要特点上看：线性电源技术很成熟，制作成本较低，可以达到很高的稳定度，波纹也很小，而且没有开关电源具有的干扰与噪音，开关电源效率高、损耗小、可以降压也可以升压，但是交流纹波稍大些。电源模块对于一个控制系统来说极其重要，关系到整个系统是否能够正常工作，因此在设计控制系统时应选好合适的电源模块。

竞赛规则规定，比赛使用智能汽车竞赛统一配发的标准车模用 7.2V 2000mAh Ni-cd 供电。系统中 3.3V 电路功耗较小，考虑到姿态传感器、CCD 对于电源低纹波的要求，我们决定使用线性稳压芯片。此外，当直流电机在高速运行时的，电池作为一个恒功率源输出电流增大，势必带来输

出电压减小。因此，为了提高系统工作的稳定性，我们选择了 TI 公司的 LM1117-3.3V。我们在实际测量小车上的各个模块及芯片时得知，车上的模块及芯片均可以用 3.3V 供电，同时，经实验，各个模块同时工作时电流总和不到 300MA，出于精简硬件电路的目的，最终我们决定使用一片 TI 公司生产的 LM1117-3.3V 为小车所有模块供电。LM1117-3.3 是低压差线性电源芯片，具有完善的保护电路,包括过流,过压,电压反接保护。使用这个芯片只需要极少的外围元件就能构成高效稳压电路。

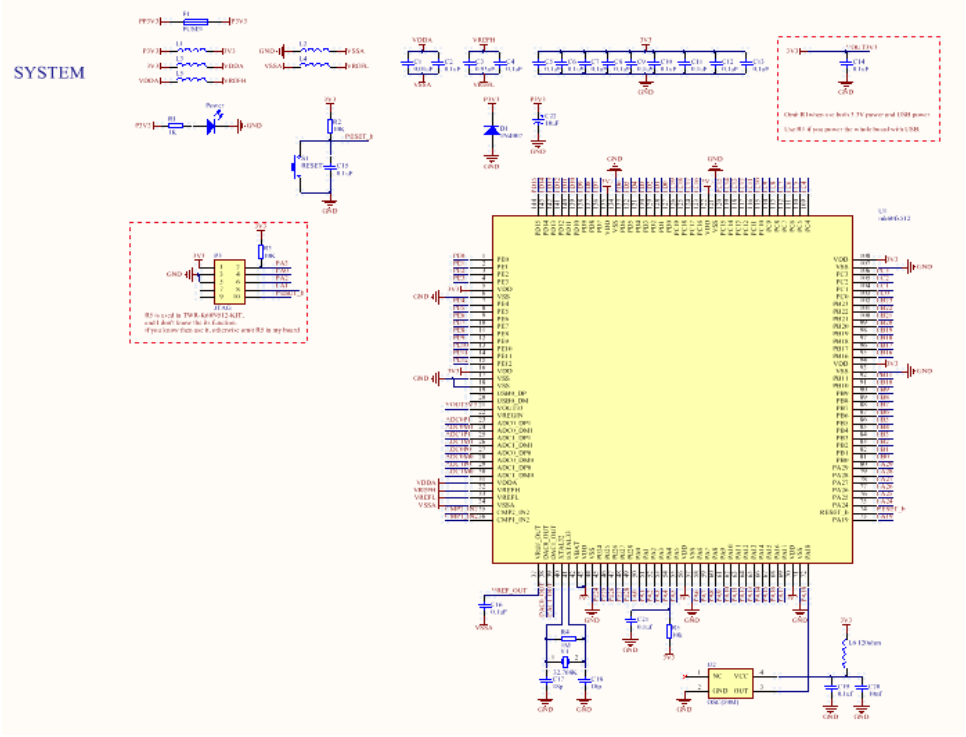
具体电源模块原理图如下：



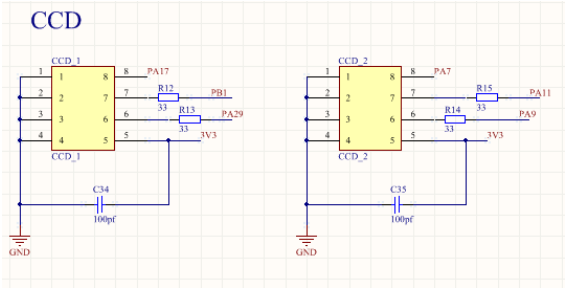
2.2 主板

我们采用的是飞思卡尔基于 Cortex-M4 内核的 MK60FX512ZVLL15 单片机，市面上有很多此芯片的最小系统销售，但这些模块外设较多，质量较重，更注重扩展性，为了使车模总重更轻，重心更低，要求主板尽可能的小而轻，所以我们根据 K60 技术手册列举的外围电路自行设计最小系统板。经过测试可以稳定运行，原理图如下：

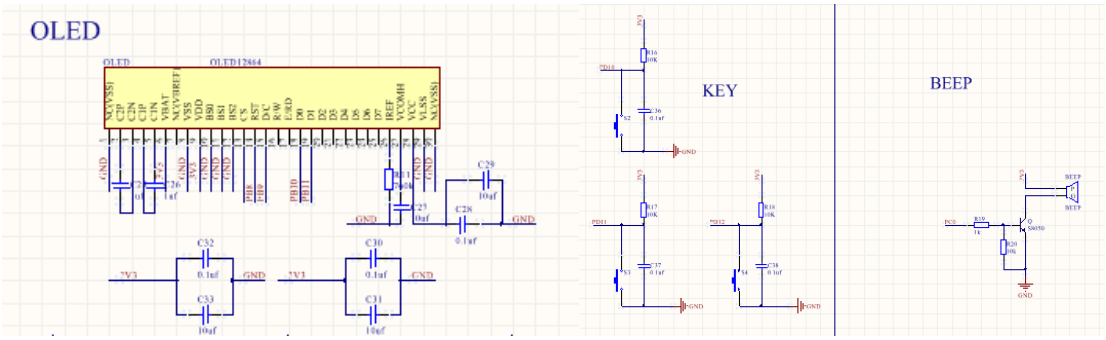
主板：



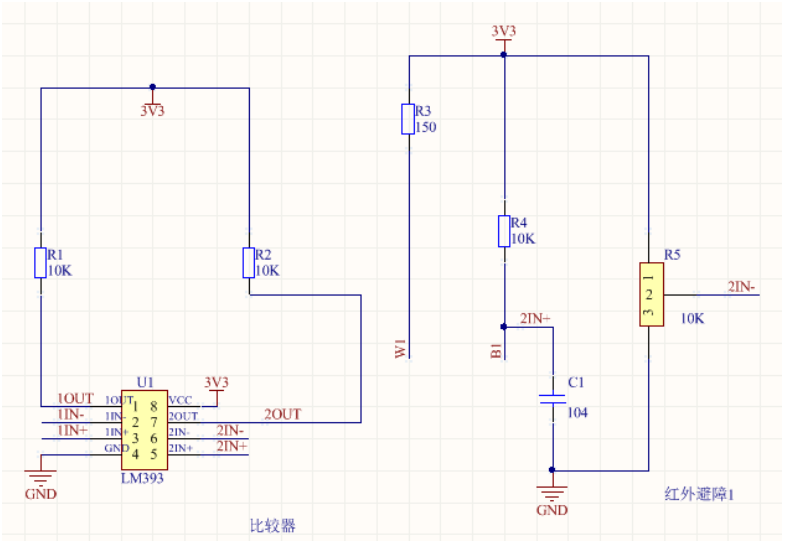
CCD：



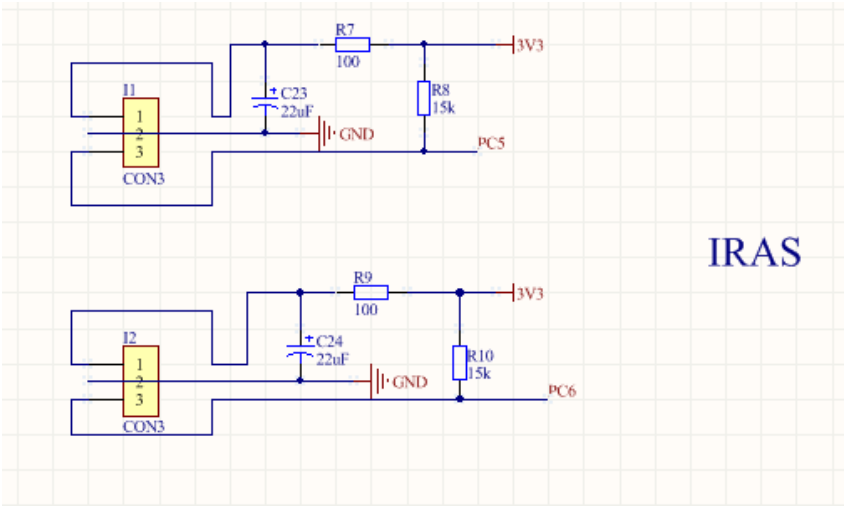
OLED&KEY：



对管直角黑色标志位检测：



起跑接受电路：

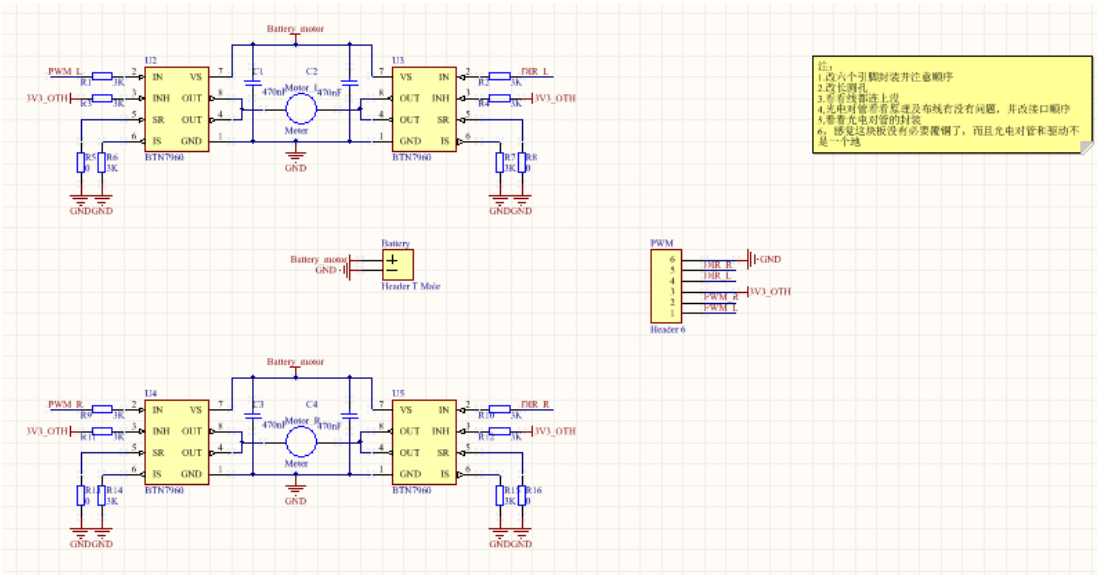


2.3 电机驱动

目前最常用的电机驱动有两种方式，一是使用 N 沟道 MOSFET 和专用栅极驱动芯片，这种驱动方案驱动能力强劲，驱动电流大，死区电压小，二是采用集成的电机驱动芯片，例如使用英飞凌公司的 BTN7971 芯片，这种方案线性度高，方案成熟，输出稳定纹波小，对于直立车模所用的

RS380 电机，又有足够的驱动电流，所以我们决定采用成熟的 BTN7971 方案。

电机驱动采用了 BTN7971 芯片，其应用非常简单，只需要向芯片第 2 引脚输入 PWM 波就能控制。当系统中只需要单向控制时，只需要让电机一端接地，另一端接 BTN7971 第 4 引脚。如果需要电机双向旋转控制，则需要另一片 BTN7971 共同组成全桥。我们使用 4 片 BTN7971 构成两个全桥分别控制两个电机。原理图如下：



第三章 姿态控制算法

车模姿态控制包括直立环、速度环和方向环三个闭环，这三个环相互影响，相互制约，为了使车模高速稳定运行，需要综合考虑很多因素，确定这三个环的控制方法，下面将介绍我们的控制方式。

3.1 车模直立控制

车模直立控制包括姿态检测、直立 PD 控制，姿态检测使用加速度计和陀螺仪互补滤波完成，通过 PD 控制使车模直立。

3.1.1 互补滤波

为了实现车模直立控制，需要获得车模的倾角和倾角速度，测量车模倾角和倾角速度通过安装在车模上的加速度计和陀螺仪实现。

1：加速度传感器

加速度计可以测量由地球引力作用或物体运动所产生的加速度。我们使用了 MMA8452 加速度计测量加速度，读取车模垂直方向的加速度值可得出与之成正比的车模角度。

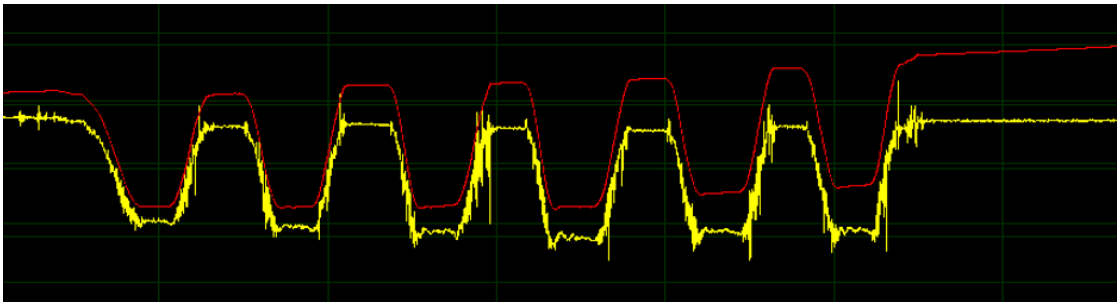
但是在实际车模运动过程中，由于车模本身摆动所产生的加速度会产生很大的干扰信号，它叠加在上述测量型号上会让输出信号无法准确反映车模倾角，加速度计信号采集如下图黄色线所示。

2：陀螺仪

陀螺仪可以测量物体旋转角速度，我们使用了 L3G4200D 陀螺仪测量

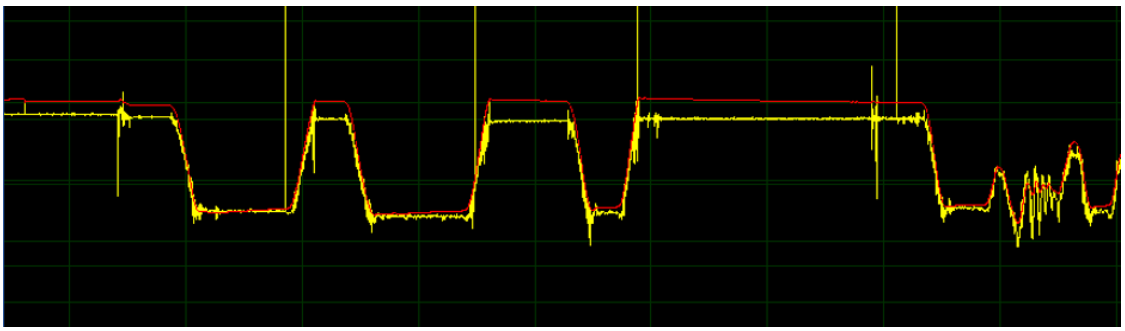
角速度，对车模垂直方向的角速度积分也可以获得车模角度，该信号中噪声很小，使角度信号更稳定。

角速度信号的微小偏差在积分后容易形成误差积累，使电路逐渐饱和，无法形成正确的角度信号，陀螺仪积分得到角度的信号如下图红色线所示：



3：互补滤波

我们通过加速度传感器获得的角度信息对陀螺仪积分得到的角度进行校正，通过对比积分所得到的角度和重力加速度所得到的角度，使用它们之间的偏差改变陀螺仪的输出，从而积分的角度逐步跟踪到加速度传感器所得到的角度，同第七届清华方案，滤波效果如下图，红色线是滤波后，黄色为加速度计采集的角度值，已经完全可用与直立控制：



3.1.2 直立 PD 控制

对于一个倒立摆系统，使它处在平衡位置需要两个力，一个是回到原

点的回复力，另一个则是使它停下来的粘斥力，类似单摆，我们使用角度值和角速度值进行位置式 PD 控制可使小车稳定的直立。

直立控制就是下面的这一句代码，原型来自位置式 PD 公式。

```
//直立PID,fixed parameter
Speed_L=Ang_IGyro*Zhili_P-GYRO_X*Zhili_D*Zhili_P*0.025;
```

3.2 车模速度控制

为了获得车模速度，我们使用 512 线编码器进行速度检测，车模速度控制使用位置式 PI 控制，控制周期为 10ms 并使用 100 次均分输出平滑处理，经过验证可以较为稳定的控制车模速度。

速度控制代码如下，程序运行周期是 1ms：

```
if(100 == Speed_Con)
{
  Speed_Con=0;
  Speed_Last=Speed_Final_Out;
}

//速度控制
if(10==Speed_Time)
{
    Speed_Time=0;
    Check_Speed=(LeftWheel_Count+RightWheel_Count)/2;
    Speed_Error=Speed_Set-Check_Speed;
    LeftWheel_Count=0;        RightWheel_Count=0;
    Speed_Int=Speed_Int+Speed_Error*Sudu_I;
    Speed_Control_Out=Sudu_P*(float)Speed_Error+Speed_Int;
    SpeedAverOut=(Speed_Control_Out-Speed_Last)/100;
}

Speed_Final_Out=Speed_Final_Out+SpeedAverOut;
```

3.3 车模方向控制

车模方向控制给定值来自路径识别算法，我们使用了动态 PD 控制做车模方向控制，对于线性 CCD 车，前瞻收到了限制，所以对于单纯的 PD，提速变得较为困难，因为较大的速度往往需要较大的前瞻。

此外，由于车模连续转弯或过坡道容易引起速度的变化，这样车模转向走线将会变得不理想，严重情况还容易掉轮子，所以我们将速度值反馈至方向 P 处，允许车模有一定程度的速度变化。

方向控制代码：

```
Dynamic_P_L=Fangxiang_P_L*(1+Speed_Rate*qianzhan*(1+Turning/64));
Dynamic_P_R=Fangxiang_P_R*(1+Speed_Rate*qianzhan*(1-Turning/64));
Turning_Out_R=Speed_Rate*Dynamic_P_R*Turning+Fangxiang_D_R*GYRO_Y;
Turning_Out_L=Speed_Rate*Dynamic_P_L*Turning+Fangxiang_D_L*GYRO_Y;
```

3.4 坡道与综合控制

对于平衡车模调试，有直立环、速度环和方向环三个 PID，这三个环相互制约，相互影响，尤其是在车模通过坡道的时候。

3.4.1 坡道、速度环和直立环

当车模过坡道的时候，会发生很大的震动，这种震动需要很硬的直立环否则车模容易失去平衡，但是对于速度控制，本身就是对直立环的一种干扰(正反馈)，为了控制好速度，直立环硬了，速度环也应当相应的硬，否则在长直道，速度容易控制不住，车模容易加速而在下一个弯道冲出赛道，但如果速度环过硬，在车模过坡道的时候又很难维持稳定，为了解决这一矛盾，我们首先是对速度控制平滑输出，再对速度控制输出量和反馈

量分别限幅，避免了车模在冲坡的时候引起的速度毛刺和过度的速度控制输出，选择较为硬的直立控制参数，提高车模在过坡道时的稳定性，在直道改变角度偏移抑制过快的加速。经验证，该方案效果不错。

第四章 路径识别算法

今年的赛道信息有直角、20º的陡坡道以及黑色砖头路障元素，对于单 CCD 的识别有很大的要求，我们组在各元素的识别上用的都是横 CCD，而在出直角时容易误判，所以在此基础上加了一个竖 CCD，用以判断出直角，具体方法后面给出。

4.1 原始图像的采集和处理

市场上所卖的线性 CCD 模块可谓琳琅满目，经过多方面的查找和比较最终我们使用了蓝宙带运放的线性 CCD 3S 模块。CCD 的采集我们先后试过蓝宙的自适应曝光，曝光和采集分离的采集，曝光、采集一体化，经过多次的尝试，我们选择了代码量更少、运行时间最短的曝光、采集一体化的采集算法。 通过曝光、采集一体化所采集回来的原始图像大致如图

如果所采集图像下降沿不够明显，可以调节 CCD 背后的运放，使得图像较为清楚。

4.2 中心线的提取和偏差的寻找

通过处理后的图像，就可以通过下降沿算法找出左右黑线的位置，然后提取实际中心线，至于具体的下降沿阀值大小需要好好调节。再和理想中线作差即可得到偏差。在偏差的提取上我们先后尝试了两种方案：

方案一：从两边往中间寻线，其优点是可以快速找到左右的黑色边界线，减少 for 循环的次数，一般不会丢线，也有效地避免了白色赛道的噪点对寻线的干扰。其缺点是，容易和其它赛道串道，不能适应白色背景，在坡道图像乱的情况下会提取到错误信息。

方案二：从中间往两边寻线，单纯的从中间往两边寻线，在大弯处丢线情况下会误判左右线。我们采用的是从中间往两边寻线，同时利用了动态中线来加以约束。其基本算法是：我们定义了一个静态的变量 static u8 centre_line1 在刚开机时，我们从 128 个点的第 64 个像素点开始往两边寻线，找到左右黑线就终止循环，这时会得到左右边界线 left_line1 和 right_line1，然后求和除以 2 得到当前的实际中线。那么此次的实际中线和理想中线 64 作差就得到了偏差，同时把该次的实际中线作为下一个循环周期寻求左右线的起始点。依次循环下去，这样就可以在直道、小弯道、大弯道处得到可靠的偏差。 通过前后期调车的比较，最终决定用方案二来提取中线，寻找偏差。实际证明，该方案可靠、稳定。

4.3 特殊元素的识别

本届比赛设定了直角、障碍、坡道、单线这些赛道元素，对车模图像处理能力提出了更高的要求，我们使用一横一竖两个线性 ccd 完成了对这些元素的识别，具体识别算法如下：

4.3.1 直角

入：直角前一米有黑色胶带区，为了减少 CCD 的识别负担，使用了红外对管识别。在对管识别到一段时间内，再看 CCD 是否单边丢线，如果满足这两个条件就证明到达直角前了，然后直接转向。

出：由于转直角时速度、等车的状态不可能一样，所以转向定时是不可取的，那么出直角的标志就很重要。第一点是 CCD 看到双线，如果就只一条条件的话，车可能会过早停止转向，而冲出赛道，所以我们加了一个竖 CCD，当竖 CCD 的前方都是白色时才说明车已经摆正，已经出直角。

4.3.2 坡道

很多队伍用了 CCD 来识别坡道，主要有两种方法：第一种是在上坡道时，CCD 的图像会比正常时变宽，但没有为全白。第二种是上坡道后会有个 CCD 图像突然变窄的过程，这时也可以作为坡道识别，但是我个人坡道用 CCD 这两种方法识别并不好，坡道直接调 PID 可以过去，如果用 CCD 识别坡道，会加重 CCD 的负担，偶尔会有误判出现。

4.3.3 障碍

障碍图像如图：

在障碍前 CCD 的图像和正常直道一样，当到障碍后图像一边突然变窄，另一边不变，可由此判断障碍，以及判断障碍在那边。

然后控制车模向障碍的相反方向转一个定值一段时间，就可以通过障碍，需要注意的是这个时间也是和速度挂钩的，否则当车模的速度发生略微改变的时候(比如下坡或者弯道失去抓地力造成的速度突变)容易踩到障碍或者掉轮。

4.3.4 单线

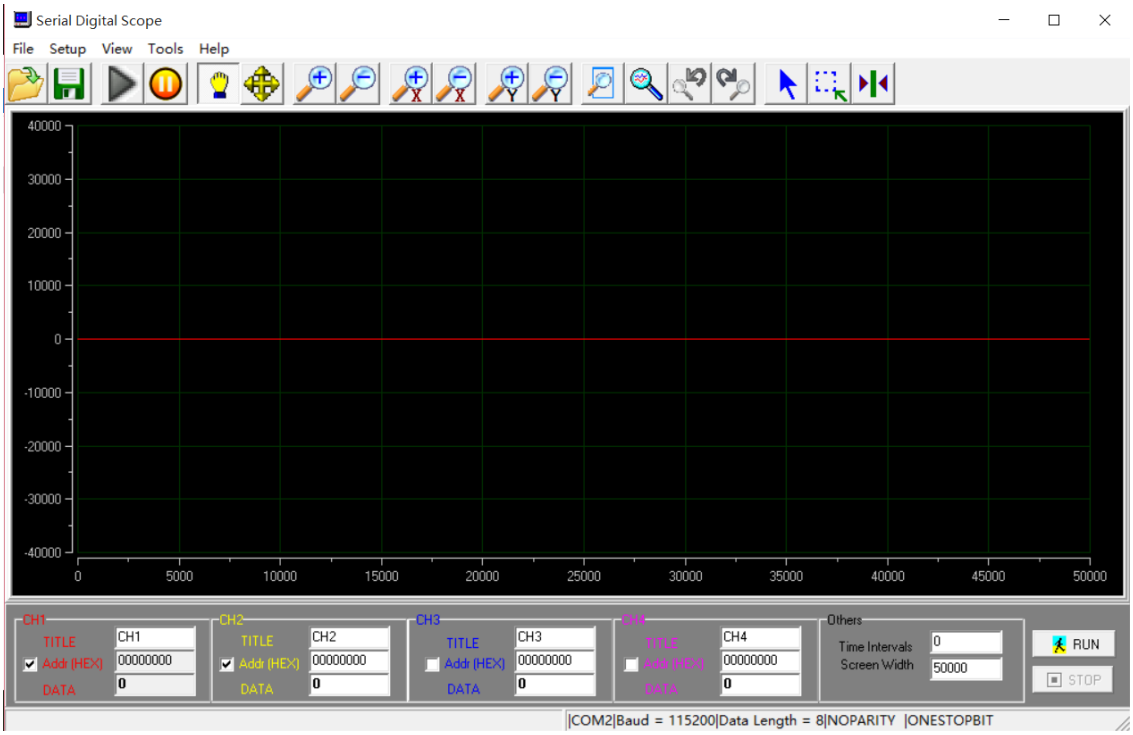
单线如图所示：

单线的查找主要是搜索中间的凹陷，即两个相邻的下降沿，但是要注意凹陷的大小、宽度等要素，调试时要特别注意，若不然容易引起误判，同时在看到单线后要关闭直角的红外对管检测，然后让车模沿着这个凹槽前进。

第五章 其它

5.1 调试软件

我们使用了一款叫 visualscope 的上位机，使用蓝牙串口通信，可以在车模运行时实时的查看变量，大大方便了调试，软件截图如下：



同时我们使用 USB 摄像头和 snagit 软件配合上位机，可录制调试视频，并查看对应时刻的一些变量的值：

5.2 车模参数信息

长：23cm

宽：21cm

高：34cm

电容总量：

传感器数量：8 个(tsl1401 线性 ccd2 个，L3G4200D 陀螺仪一个，MMA8452 加速度计一个，龙邱编码器两个，红外对管两对)

5.3 心得体会

深夜坐在实验室里，想想已经是做飞思卡尔的第三年，最后一年，终于也算是进入国赛，获得个写技术报告的资格，准备国赛时间紧张，赛后还会更深入的整理这次比赛，并在论坛分享。

第八届，那时我还大一，没什么原因，几个志同道合的小伙子，就这么组队了，看学长们调车，觉得很酷，而刚入大学的我们决定试一把，就买了东西，组了辆车，开始了我们的飞思卡尔之旅，那一年由于实力原因，虽然很努力，最终还是没能完成校赛。

第九届，开始做直立车，基本也是从零开始，每天都泡在实验室里调试，车模速度也是逐渐的加快，那年赛区赛在吉林大学，预赛成绩不错，第十一名，决赛因为坡道没过去而未完成，没跑完拿着车去车检，一直忍着眼泪。最后也不忍了，愣是哭了个底朝天，努力过，就那么的容易动感情，管它脆弱还是坚强，还有第十届，能力，都是慢慢练出来的。

第十届，成了赛区东道主，众望所归，压力和动力都很大，更深入的研究了直立车的控制，理论分析也更详细，尤其是坡道的处理，几乎这一年都在和它打交道，能力有限，也不是什么神车，还有很多地方有问题没解决，只是尽力做到最精致。

到最后，成绩什么的也释然了，何必总要以成败论英雄，想想这三年来飞思卡尔让你的能力提升了多少吧，大一学长带着调程序，大二大三从软件到硬件，从编码到测试，从系统到细节都练了个遍，而对一个控制对象的持久的研究也深深的加强了我们的项目能力，或许比赛带给我们的不是奖杯，不是荣誉，而是实实在在的锻炼。

文笔不好，征文就不投稿了，一些心得放在这里，也不知合适与否。

20150822 黄翊峰

参考文献

［1］胡寿松. 自动控制原理（第五版）. 科学出版社，2007.

［2］卓　晴. 电磁组直立行车参考设计方案，2012.

附录

赛道识别代码：

```
#include "common.h"
#include "include.h"



uint8 Pixe3[128];

int zhijiao=0;
int Into_zhijiao_right=0;
int Into_zhijiao_left=0;
int zhijiaoqian=0;
//int guanbizhijiao=0;
int d=0;
int shutdown_zhijiao=0;
int shutdown_zhijiao2=0;
int zhijiao_bai=0;

int danxian_check_sum=0;
int danxian=0;
int danxian_zhi=0;                //本次单线的值

int danxian_zhi2[2]={'0'};              //上次单线的值
```

```
int danxian_ccd2=0;              //CCD2 所看单线的值
int danxian_get=0;               //CCD2 是否检测到单线标志位
int danxian_get1=0;
/********************  直角         ***************************/

int zhijiao_allow = 1;       //允许红外，防单线
int zhijiao_hei=0;
int zhijiao_line_left=0;
int zhijiao_line_right=0;
int zhijiao_turning=0;
int zhijiao_CCD3_chu=0;
int zhijiao_chu=0;
int zhijiao_chu_time=0;
extern int zhijiao_hei_time;
int targe_error3=15;
int zhijiao_chu_shu=0;
int zhijiao_turning_time=1;
int zhijiao_heheda=0;
int zhijiao_hei_error=25;
int po=0;

/**********************二值化*******************************/
int targe_error2=15;
int16 targe_error=40;                          //下降沿阀值
int danxian_error=40;                          //单线下降沿阀值
int32 ccd_sum=0,ccd_avg=0;                     //CCD 数值的和、平均值
uint8 ccd_max=0,ccd_min=0;                     //CCD 最大值、最小值
int16 ccd_target;                              //上次左右黑线的中间值，即中
点
```

```
int16 ccd_target1;
int Ccd_getline_flag=0;
int Turning1=0;
uint8 B_flog=0,H_flog=0;                    //全白标志位，即十字
int16  line_right[3]={90};                  //右黑线位置存放数组，最近 3
次的
int16 line_left[3]={30};                    //左黑线位置存放数组，最近 3 次
的
int line_right1=90;                         //最新一次的右黑线位置
int line_left1=30;                          //最新一次的右黑线位置
int line_right2=90;                         //最新一次的右黑线位置
int line_left2=30;                          //最新一次的右黑线位置


/**********************      障碍      **********************/
int zhangai_hei=0;
int zhangai_hei_zuo=0;
int zhangai_hei_you=0;
int zhangai_time=0;
int zhangai_time1=7;
int B_flog_wangxing=0;

int Coner=1;
/**********************      ccd2      **********************/


void ccd_deal(uint8 ch[])          //求数组 ch，即 ccd 的平均值、最大值、最
小值
```

```
{
   uint8 i;
   int8 max=1,min=1;
   for(i=5;i<123;i++)
   {
      if(ch[max]<ch[i]) max=i;
      if(ch[min]>ch[i]) min=i;
      ccd_sum+=ch[i];
   }
   ccd_max=max;
   ccd_min=min;
   ccd_avg=ccd_sum/118;
   ccd_sum=0;//这里没清零平均值会飞起。。。
}

void ccd3_work()
{
   int i=0;

/*****************      弯道检测      ****************************/

   Coner=1;
   for(i=30;i<90;i++)
   {
      if(Pixe3[i]-Pixe3[i+4]>targe_error3)
      {
         Coner=0;

         break;
      }
   }

/*****************      出直角检测      ************************/
   zhijiao_CCD3_chu=0;
   if(zhijiao_line_right==1 || zhijiao_line_left==1)
   {
      zhijiao_CCD3_chu=1;
```

```
        for(i=10;i<65;i++)
        {
          if(Pixe3[i]-Pixe3[i+4]>targe_error3)
          {
            zhijiao_CCD3_chu=0;
            break;
          }
        }
    }

}
void ccd2_work()
{
    uint8 i=0;

//                                                  int16                    error;
//error=Pixe2[i]-Pixe2[i-4]，用于判断是否大于阀值

    //uint8 no_left_line=0,no_right_line=0;                    //是否寻到左右线的
标志位

    if(danxian==1)              //若入单线，CCD2 开始搜单线
    {
        danxian_get=0;          //标志位清 0
        for(i=10; i<=117; i++)
        {
          if(Pixe2[i-5]-Pixe2[i]>targe_error2                                    &&
Pixe2[i+5]-Pixe2[i]>targe_error2)
          {//CCD2 检测单线
            danxian_ccd2=i;   //单线所在的点
            danxian_get=1;     //检测到单线标志位
            break;
          }
```

```
    }
/*****************************************************************
****

    下面搜索 CCD2 的左右黑线

*****************************************************************
*****/

    // if(danxian_ccd2<1) danxian_ccd2=1;          //如果中间值小于 1，则默
认为 0

    // if(danxian_ccd2>126) danxian_ccd2=126;     //如果中间值大于 126，则
默认为 126

    //单线为搜线的中心值

      if(danxian_get==0)
      {
        danxian=0;
        danxian_ccd2=0;
        zhijiao_allow = 1;
        zhijiao_hei=0;
      }
   }
}

void ccd_getline()                                  //求 CCD 的左右黑
线
{
   uint8 i=0;
   int16                                                    error;
//error=Pixe2[i]-Pixe2[i-4]，用于判断是否大于阀值
```

```
uint8  no_left_line=0,no_right_line=0;                    //是否寻到左右线的
标志位
ccd_deal(Pixe2);                                          //求数组 Pixe1，
即 ccd1 的平均值、最大值、最小值

Ccd_getline_flag=0;
if(danxian==0)                  //判断入单线的算法
{
  for(i=20; i<=107; i++)
  {
    if(Pixe2[i-3]-Pixe2[i]>danxian_error                  &&
Pixe2[i+3]-Pixe2[i]>danxian_error)
    {    //从 i 点向左右各 6 个点检测下降沿，判断单线
      if((i-line_left[0]>5)&&(line_right[0]-i>5))
      {    //判断单线是否在左右双线的中间

        danxian=1;          //入单线标志位

        danxian_zhi=i;    //初始化本次及上次的单线值
        danxian_zhi2[1]=danxian_zhi;
        danxian_zhi2[0]=danxian_zhi;
        danxian_get1=1;
        zhijiao_allow = 0;
        zhijiao_hei=0;
        Ccd_getline_flag=1;
        break;
      }
    }
  }
}else
{
  zhijiao_allow = 0;
  danxian_get1=0;
```

```
for(i=20; i<=106; i++)
{

   if(Pixe2[i-3]-Pixe2[i]>targe_error && Pixe2[i+3]-Pixe2[i]>targe_error)
   {//从 i 点向左右各 6 个点检测下降沿，判断单线
      danxian_zhi=i;
      danxian=1;

      if(danxian_zhi>=danxian_zhi2[0])
      {
         if(danxian_zhi-danxian_zhi2[0]<10)
         {//本次单线是否在上次单线的左右 15 个点之内
            danxian_zhi2[1]=danxian_zhi2[0];
            danxian_zhi2[0]=danxian_zhi;
            danxian_get1=1;
            Ccd_getline_flag=1;
            break;
         }
      }else if(danxian_zhi<danxian_zhi2[0])
      {
         if(danxian_zhi2[0]-danxian_zhi<10)
         {
            danxian_zhi2[1]=danxian_zhi2[0];
            danxian_zhi2[0]=danxian_zhi;
            danxian_get1=1;
            Ccd_getline_flag=1;
            break;
         }
      }
   }
}
if(danxian_get1==0)
{
   danxian_zhi2[1]=danxian_zhi2[0];

danxian_zhi2[0]=danxian_zhi2[0]+(int)(1.0*(danxian_zhi2[0]-danxian_zhi2[1]
));
   Ccd_getline_flag=1;
```

```
    }
  }

  if(danxian == 0)
    ccd_target=(line_left[0]+line_right[0])/2;          //上次左右黑线的中间
值
  else ccd_target=danxian_zhi2[0];
  if(ccd_target<1)  ccd_target=1;                       //如果中间值小于 1，
则默认为 0
  if(ccd_target>126)  ccd_target=126;                   //如果中间值大于
126，则默认为 126

/*****************        左 线 从 上 次 的 中 间   向 左 寻 线
*********************/
//   if(zhijiao_hei>=1) zhijiao_hei_error=30;
//   else zhijiao_hei_error=15;
  no_left_line=1;                                       //设没寻到左线
  if(ccd_target>121) ccd_target1=126;
  else ccd_target1=ccd_target+5;
  for(i=ccd_target1;i>10;i--)                           //由于左右有畸变，
从第 5 点之后有效
  {
    error=Pixe2[i]-Pixe2[i-5];          //              //4  点之间的下降沿
与所定的阀值判断
    if(error>targe_error && Pixe2[i-5]<100 ) {
        no_left_line=0;                                 //找到左线
        line_left1=i-2;
```

```
            break;
        }
    }
/*****************          右 线 从 上 次 的 中 间   向 右 寻 线
*********************/
    no_right_line=1;                                        //设没寻到右线
    if(ccd_target<6) ccd_target1=1;
    else ccd_target1=ccd_target-5;
    for(i=ccd_target1;i<118;i++)                            //由于左右有畸变，
从第 118 点之前有效
    {
        error=Pixe2[i]-Pixe2[i+5];  //                      //4  点之间的下降沿
与所定的阀值判断
        if(error>targe_error && Pixe2[i+5]<100 )    {
            no_right_line=0;                                //找到右线
            line_right1=i+2;
            break;
        }
    }
/********************          讨    论    丢    线    情    况
*****************************/
  // gpio_set      (PTC0,      0);
    if((no_right_line==1)&&(no_left_line==0))           //no_right_line=1，说
明丢右线
    {
        //gpio_set      (PTC0,      1);//
        //if( line_right[0]<114) gpio_set      (PTC0,      1);//
        if(danxian==0 && zhijiao_hei>=1 && line_right[0]<98) {//丢右边，直角，
拧一下
```

```
        zhijiao_hei=0;
        zhijiao_line_right=1;//进入直角准备右转
    }else
    {
        Ccd_getline_flag=1;
        line_left[2]=line_left[1];
        line_left[1]=line_left[0];
        line_left[0]=line_left1;                    //更新数组，即最新的左线
        line_right[2]=line_right[1];
        line_right[1]=line_right[0];
        line_right[0]=line_right[0]+(line_left[0]-line_left[1])*10/10;// 丢一边的线，按另一边的偏差算
    }
  } else if((no_right_line==0)&&(no_left_line==1))          //no_left_line=1，说明丢左线
  {
    //gpio_set     (PTC0,     1);//
    //if(line_left[0]>14 ) gpio_set     (PTC0,     1);//
    if(danxian==0 && zhijiao_hei>=1 && line_left[0]>30) {
        zhijiao_hei=0;
        zhijiao_line_left=1;//进入直角准备左转
    }else
    {
        Ccd_getline_flag=1;
        line_right[2]=line_right[1];
        line_right[1]=line_right[0];
        line_right[0]=line_right1;                  //更新数组，即最新的左右线
        line_left[2]=line_left[1];
```

```
        line_left[1]=line_left[0];
        line_left[0]=line_left[0]+(line_right[0]-line_right[1])*10/10;     // 丢一边的线，按另一边的偏差算
      }
   }else  if((no_right_line==1)&&(no_left_line==1))            //都丢线，全白或全黑
   {
      B_flog=0;
      if(ccd_avg>100)
      {
        B_flog=1;                                  //ccd 平均值大于阀值即全白在十字，其他是看外边丢线
        Ccd_getline_flag=1;
      }
   }else   if((no_right_line==0)&&(no_left_line==0))                  // 不丢线 &&(line_right1-line_left1)>30
   {
      if(zhijiao_CCD3_chu==1)
      {
        zhijiao_line_left=0;
        zhijiao_line_right=0;
        zhijiao_CCD3_chu=0;
        zhijiao_turning=0;
        zhijiao_hei=0;
        zhijiao_chu_shu=1;
        zhijiao_heheda=1;
        //gpio_set     (PTC0,     1);
      }else          if(danxian==1          &&          danxian_get1==0          &&  line_right1-danxian_zhi2[0]>8 && danxian_zhi2[0]-line_left1>8)
      {
        danxian=0;
        zhijiao_allow = 1;
        zhijiao_hei=0;
```

```
    }else if(zhijiao_heheda>=30 && B_flog_wangxing==0 && danxian==0&& line_left1-line_left[0]>12 && line_right[0]-line_right1<3)
    {
      zhangai_hei_zuo=1;
      zhangai_time1=3+(zhangai_time1-4)/Speed_Rate;

    }else if(zhijiao_heheda>=30 && B_flog_wangxing==0 && danxian==0 && line_right[0]-line_right1>12 && line_left1-line_left[0]<3)
    {
      zhangai_hei_you=1;
      zhangai_time1=3+(zhangai_time1-4)/Speed_Rate;
    }


    if(zhijiao_line_right==0 && zhijiao_line_left==0)
    {
      Ccd_getline_flag=1;
      //if(line_right1-line_left1>30)
      //{

        line_right[2]=line_right[1];
        line_right[1]=line_right[0];
        line_right[0]=line_right1;                    //更新数组，即最新的左右线

        line_left[2]=line_left[1];
        line_left[1]=line_left[0];
        line_left[0]=line_left1;
      //}
    }
  }


/*********        计算 Turning      ***************************/
  if(Ccd_getline_flag==0)
  {
    Turning1=Turning;
    Turning=Turning+(Turning-Turning1);
  }else
```

```
    {
       Turning1=Turning;
       Turning=64-(line_left[0]+line_right[0])/2;
    }


/*********      直角和障碍的辅助时间处理          ********************/
    if(zhijiao_line_right == 1 || zhijiao_line_left == 1)
    {
       zhijiao_turning++;       //看到直角后延时转
    }else zhijiao_turning=0;

    if(zhangai_hei_zuo == 1 || zhangai_hei_you == 1)
    {
       zhangai_time++;
    }else zhangai_time=0;
    if(zhangai_time>=zhangai_time1)
    {
        zhangai_time=0;
        zhangai_hei_zuo=0;
        zhangai_hei_you=0;
        zhangai_time1=7;
    }
    if(B_flog==1 ) B_flog_wangxing=1;
    if(B_flog_wangxing>=1) B_flog_wangxing++;
    if(B_flog_wangxing>=20) B_flog_wangxing=0;
/*****************       特殊处理          **************************/
    gpio_set     (PTC0,     0);
    if(danxian==1)
    {
       Turning = (int)(1*(64-danxian_zhi2[0]));
       gpio_set     (PTC0,     1);
    }else if(zhijiao_turning>=zhijiao_turning_time && zhijiao_line_right == 1)
    {
       zhijiao_turning=zhijiao_turning_time;
       gpio_set     (PTC0,     1);
    }else if(zhijiao_turning>=zhijiao_turning_time && zhijiao_line_left == 1)
    {
```

```
        zhijiao_turning=zhijiao_turning_time;
        gpio_set     (PTC0,     1);
    }else if(zhangai_time <= 20 && zhangai_hei_zuo == 1)
    {
        //Turning = Turning-20;
        gpio_set     (PTC0,     1);
    }else if(zhangai_time <= 20 && zhangai_hei_you == 1)
    {
        //Turning = Turning+20;
        gpio_set     (PTC0,     1);
    }else if(B_flog==1)
    {

        Turning = 0;//十字

        B_flog=0;
        gpio_set     (PTC0,     1);
    }

}

void Delay(uint32 a)
{
    uint32 b=900000,c=900000;
    for(;a>0;a--)
        for(;b>0;b--)
            for(;c>0;) c--;

}
```

姿态控制代码：

```
/*!
 * @file            MK60_it.c
 * @brief           逆袭二号_中断服务
 * @author          黄翊峰  汪星
 * @version         v1.2
 * @date            2015-04-06
 */
```

```
#include       "MK60_it.h"
#include       "include.h"
//#define Canshu //如果定义，会关闭速度控制和弯道控制


uint8 CCD_succeed_flag=0;       //ccd 数据已采集标志位


float Speed_L=0;                    //用于计算的速度值

int danxian_kai=0;                 //用于延时

int duiguan_time=0;
int duiguan_guanbi=0;
int Speed_Change=0;
extern int zhijiao_allow;
extern int zhijiao_hei;
extern int zhijiao_chu;
int zhijiao_allow_time=0;
int zhijiao_hei_time=0;
extern int zhijiao_chu_time;

int Stright_Count=0;//直线延时，在弯道清零，在直道增加

extern int zhijiao_chu_time;
extern int zhijiao_turning_time;
extern int zhijiao_turning;
extern int zhijiao_heheda;

//int Stright_Count=0;//直线延时，在弯道清零，在直道增加

int zhijiao_chu_time1=0;
float Speed_Rate1=0;


float Dynamic_P_L=0;//动态转弯 P，可适当降低前瞻，提高系统响应速度，

但参数过大过弯会不稳定

float Dynamic_P_R=0;
```

```
float Turning_Out_R=0;
float Turning_Out_L=0;
float Turning_Out_R_Lo=0;
float Turning_Out_L_Lo=0;
int podao=0;//检测到坡道后，延时减速
int Shut_Hill=0;//检测到坡道后，3 秒不再坡道检测
float qianzhan=0.2;

float Speed_Control_Out=0;//速度控制输出量，直接给定到电机上


void Go(float Speed) //更新电机输出，包括转弯值和速度值
{
#ifndef Canshu

   if(gear==1)
   {
      zhijiao_Power=210;
      Fangxiang_P_R=18.0;//left
      Fangxiang_D_R=0.1;
      Fangxiang_P_L=18.0;//right
      Fangxiang_D_L=0.1;
   }
   else if(gear == 2)
   {
      zhijiao_Power=210;
      Fangxiang_P_R=18.6;//left
      Fangxiang_D_R=0.11;
      Fangxiang_P_L=18.6;//right
      Fangxiang_D_L=0.11;
   }

   if(Turning>0)//动态 P 很有效果
   {
```

```
Dynamic_P_L=Fangxiang_P_L*(1+Speed_Rate*qianzhan*(1-Turning/64));

Dynamic_P_R=Fangxiang_P_R*(1+Speed_Rate*qianzhan*(1-Turning/64));
   }
   else
   {

Dynamic_P_L=Fangxiang_P_L*(1+Speed_Rate*qianzhan*(1+Turning/64));

Dynamic_P_R=Fangxiang_P_R*(1+Speed_Rate*qianzhan*(1-Turning/64));
   }
   gpio_set     (PTC0,     0);
   if(zhijiao_turning>=zhijiao_turning_time && zhijiao_line_right == 1)
   {
      Turning_Out_R = -zhijiao_Power;
      Turning_Out_L = -zhijiao_Power;
      //gpio_set     (PTC0,     1);
   }
   else if(zhijiao_turning>=zhijiao_turning_time && zhijiao_line_left == 1)
   {
      Turning_Out_R = zhijiao_Power-20;
      Turning_Out_L = zhijiao_Power+80;//
      //gpio_set     (PTC0,     1);
   }else if(zhangai_time <= 10 && zhangai_hei_zuo == 1)
   {
      Turning_Out_R = -450;//*Speed_Rate
      Turning_Out_L = -450;
      gpio_set     (PTC0,     1);
   }
   else if(zhangai_time <= 10 && zhangai_hei_you == 1)
   {
      Turning_Out_R = 450;
      Turning_Out_L = 450;
      gpio_set     (PTC0,     1);
   }
   else
   {

Turning_Out_R=Speed_Rate*Dynamic_P_R*Turning+Fangxiang_D_R*GYR
O_Y;
```

```
Turning_Out_L=Speed_Rate*Dynamic_P_L*Turning+Fangxiang_D_L*GYRO_Y;

        if(qipaoyanshi>=510)//对一个 bug 的修复
        {
           if(Turning_Out_R>270) Turning_Out_R=270;
           if(Turning_Out_R<-270) Turning_Out_R=-270;
           if(Turning_Out_L>270) Turning_Out_L=270;
           if(Turning_Out_L<-270) Turning_Out_L=-270;

        }
        else
        {
           Turning_Out_R=0;
           Turning_Out_L=0;

        }

        if(podao!=0)
        {
           if(Turning_Out_R>70) Turning_Out_R=70;
           if(Turning_Out_R<-70) Turning_Out_R=-70;
           if(Turning_Out_L>70) Turning_Out_L=70;
           if(Turning_Out_L<-70) Turning_Out_L=-70;
        }

   }
#endif

   if(pao_end==0)
   {

      if( (Speed+Turning_Out_R)>=0 && (Speed+Turning_Out_R)<=990)
      {
           FTM_PWM_Duty(FTM0,                                    FTM_CH0,
(uint32)(Speed+Turning_Out_R) );//r
           FTM_PWM_Duty(FTM0, FTM_CH1, 0);
           //gpio_set     (PTC0,     0);
```

```
    }else if((Speed+Turning_Out_R) < 0 && (Speed+Turning_Out_R) > (0-990))
    {
        FTM_PWM_Duty(FTM0, FTM_CH0, 0);
        FTM_PWM_Duty(FTM0, FTM_CH1, (uint32)(0-Speed));
        //gpio_set     (PTC0,     0);
    }
    else
    {

      if( Speed+Turning_Out_R>=990)
      {
        FTM_PWM_Duty(FTM0, FTM_CH0,990);//r
        FTM_PWM_Duty(FTM0, FTM_CH1, 0);
      }
      if( Speed+Turning_Out_R<=-990)
      {
        FTM_PWM_Duty(FTM0, FTM_CH0,0);//r
        FTM_PWM_Duty(FTM0, FTM_CH1, 990);
      }
    }

    if( (Speed-Turning_Out_L)>=0 && (Speed-Turning_Out_L)<=990)
    {
        FTM_PWM_Duty(FTM0,                                  FTM_CH3, (uint32)(Speed-Turning_Out_L) );
        FTM_PWM_Duty(FTM0, FTM_CH2, 0);
        //gpio_set     (PTC0,     0);
    }
    else if((Speed-Turning_Out_L) < 0 && (Speed-Turning_Out_L) > (0-990))
    {
        FTM_PWM_Duty(FTM0, FTM_CH3, 0);
        FTM_PWM_Duty(FTM0, FTM_CH2, (uint32)(0-Speed));
        //gpio_set     (PTC0,     0);
    }
    else
    {
      if( Speed-Turning_Out_L>=990 )
      {
```

```
            FTM_PWM_Duty(FTM0, FTM_CH3,990);//r
            FTM_PWM_Duty(FTM0, FTM_CH2, 0);
          }
          if( Speed-Turning_Out_L<=-990 )
          {
            FTM_PWM_Duty(FTM0, FTM_CH3,0);//r
            FTM_PWM_Duty(FTM0, FTM_CH2, 990);
          }
          //gpio_set     (PTC0,     1);
        }

    }
    else
    {
      FTM_PWM_Duty(FTM0, FTM_CH0, 0);
      FTM_PWM_Duty(FTM0, FTM_CH1, 0);
      FTM_PWM_Duty(FTM0, FTM_CH2, 0);
      FTM_PWM_Duty(FTM0, FTM_CH3, 0);
    }

}

//定时器 0 中断服务
void PIT0_IRQHandler(void)
{
    CCD_Time++;//用来调曝光时

    Speed_Time++;//测一次速度的间隔时间
    DSpeed_Time++;
    Speed_Con++;
    if(zhijiao_chu_shu==1)    zhijiao_chu_time1++;

    if(qipaoyanshi>=510) dengta_yanshi++;

    //采集陀螺仪数据

    GYRO_X=Get_Gyro(1,'X');//直立控制陀螺仪
```

```
GYRO_Y=Get_Gyro(1,'Y');//方向控制陀螺仪

//采集加速度计数据
ACC_X=Get_mma8451_once('X');//这里有一个硕大的 bug!
ACC_Z=Get_mma8451_once('Z');

//以加速度计数据计算角度
Ang_Acc=Ang_Set+Ang_Cha+Parameter3-186*ACC_Z;
//计算陀螺仪积分角度+++++互补滤波
Ang_IGyro-=(GYRO_X+Ang_Error*3.5)*0.00025;//此时的 Ang_IGyro 就是最终使用的角度

//计算两个角度之间的偏差
Ang_Error=Ang_IGyro-Ang_Acc;

//直立 PID,fixed parameter
Speed_L=Ang_IGyro*Zhili_P-GYRO_X*Zhili_D*Zhili_P*0.025;// 直 立 PID

if(100 == Speed_Con)
{
   Speed_Con=0;
   Speed_Last=Speed_Final_Out;
}

//计算速度和 D 速度、I 速度
if(10==Speed_Time)                    //50ms 测一次速度
{
      Speed_Time=0;
```

```
        Check_Speed=(LeftWheel_Count+RightWheel_Count)/2;    //平均一
下,Speed_Change 没有加上


        //速度
        SpeedQueue[4]=SpeedQueue[3];
        SpeedQueue[3]=SpeedQueue[2];
        SpeedQueue[2]=SpeedQueue[1];
        SpeedQueue[1]=SpeedQueue[0];
        SpeedQueue[0]=Check_Speed;

        if(qipaoyanshi >1000)
        {
            if(              (Check_Speed-SpeedQueue[4])>15              )
Check_Speed=SpeedQueue[4];
            if(              (SpeedQueue[4]-Check_Speed)>15              )
Check_Speed=SpeedQueue[4];
        }

        if(qipaoyanshi > 1000)
        {
            if(Check_Speed>(Speed_Set-Speed_Down+7))
Check_Speed=Speed_Set-Speed_Down+7;//速度测量限幅
            if(Check_Speed<(Speed_Set-Speed_Down-7))
Check_Speed=Speed_Set-Speed_Down-7;
        }


        Speed_Rate=(float)(Check_Speed)/Speed_Set;//速度漂移


        Speed_Error=Speed_Set-Speed_Down-Check_Speed;
        LeftWheel_Count=0;          RightWheel_Count=0;

        if(qipaoyanshi>900)
        {
          Speed_Int=Speed_Int+Speed_Error*Sudu_I;
```

```
        }

        if(Speed_Int>100) Speed_Int=100;
        if(Speed_Int<-100) Speed_Int=-100;

        Speed_Control_Out=Sudu_P*(float)Speed_Error+Speed_Int;
        SpeedAverOut=(Speed_Control_Out-Speed_Last)/100;

        dengta1=1;
    }

    Speed_Final_Out=Speed_Final_Out+SpeedAverOut;

    //最终输出限幅
    if(qipaoyanshi > 1000)
    {
        if(Speed_Final_Out>300)          Speed_Final_Out=300;
        else if(Speed_Final_Out<-300)          Speed_Final_Out=-300;
    }

    //速度 PID
#ifndef Canshu
    Speed_L=Speed_L-Speed_Final_Out;
#endif

    //启动曝光，可以不曝光
    if(CCD_Time==0)
    {
      StartIntegration1() ;       //启动 CCD1,即曝光

      //StartIntegration2() ;     //启动 CCD2,即曝光
      StartIntegration3() ;
    }else if(CCD_Time==5)//调整曝光时间采集数据
    {
      ImageCapture1(Pixe1) ;
```

```
}
else if(CCD_Time==7)//调整曝光时间采集数据
{
   ImageCapture3(Pixe3) ;
}else if(CCD_Time==3)//调整曝光时间采集数据
{
  // ImageCapture2(Pixe2) ;
}
else if(CCD_Time>=10)//调整采集时间间隔
{
   CCD_Time=0;
   CCD_succeed_flag=1;//ccd 数据已采集
}

//ccd 数据采集一次后，计算左右黑线，防止重复计算

zhijiao_heheda++;
if(zhijiao_allow >= 1) zhijiao_allow++;
if(zhijiao_allow >= 200 && zhijiao_heheda>=1000) {
   zhijiao_allow=300;
   zhijiao_heheda=1010;
   if(gpio_get(PTD12)==1)
   {
      zhijiao_hei=1;
      Speed_Rate1=Speed_Rate;
   }
}


if(zhijiao_hei>=1)
{

   zhijiao_hei++;//检测到黑线标志
   Speed_Down=0;
```

```
   Ang_Cha=0;
}
if(zhijiao_hei>=1000){    //检测到黑线一段时间内检测丢线

  zhijiao_hei=0;            //超时则认为黑线无效

}
if(CCD_succeed_flag==1)
{
   if(qipaoyanshi>520) { //未起跑时不处理主横线

      ccd_getline();
      ccd3_work();
   }


   CCD_succeed_flag=0;//处理完毕再次打开采集

}
//更新电机

Go(Speed_L);//大于 0 左拐


if( ((Turning>=0 && Turning<20) ||
        (Turning<=0 && Turning>-20))
                                                  && qipaoyanshi>720)
{
   Stright_Count++;
}
else
{
   Stright_Count=0;
}

if(Stright_Count>920 &&Coner)
{
   Ang_Cha=29;
   Stright_Count=921;

}
```

```
        else
        {
           Ang_Cha=0;
         Speed_Down=0;
        }

        if(gpio_get(PTD13)==1)
        {

        }
        else if(qipaoyanshi>700 && Shut_Hill == 0)
        {

             podao=500;

        }

        if(podao!=0)
        {
           //gpio_set     (PTC0,     1);
           podao--;
           Shut_Hill=1500;
        }
        else
        {

           //gpio_set     (PTC0,     0);
           if(Shut_Hill!=0)
              Shut_Hill--;
        }




//      if(dang == 1)
//      {
//            if(zhijiao_chu_time1>=140)
//            {
//               if(gear == 1)
//               {
```

```
//                  Speed_Down=38;
//                  Ang_Cha=50;
//                  //gpio_set     (PTC0,      1);
//              }
//              else if(gear == 2)
//              {
//                  Speed_Down=25;
//                  Ang_Cha=80;
//              }
//          }
//
//          if(zhijiao_chu_time1>=820)
//          {
//              if(gear == 1)
//              {
//                  Speed_Down=38;
//                  Ang_Cha=-40;
//                  //gpio_set     (PTC0,      1);
//              }
//              else if(gear == 2)
//              {
//                  Speed_Down=38;
//                  Ang_Cha=-50;
//                  //gpio_set     (PTC0,      1);
//              }
//          }
//
//          if(zhijiao_chu_time1>=1900)
//          {
//              if(gear == 1)
//              {
//                  Speed_Down=8;
//                  Ang_Cha=-3;
//                  //zhijiao_chu_shu=0;
//
//                  //gpio_set     (PTC0,0);
//              }
//              else if(gear == 2)
//              {
//                  Speed_Down=38;
```

```
//                  Ang_Cha=-3;
//                  //zhijiao_chu_shu=0;
//
//                // gpio_set     (PTC0,     0);
//               }
//               zhijiao_turning_time=3;
//           }
//     }


    //更新上位机输出数据
    //OutData[0]=(float)Turning_Out_R;
    OutData[1]=(float)dengta;
    //OutData[2]=(float)(Check_Speed-SpeedQueue[4]);
    //清中断标志位
    PIT_Flag_Clear(PIT0);
}

void FTM1_INPUT_IRQHandler(void)       //第一路测速，右轮
{
      uint8  s  =  FTM1_STATUS;                  //读取捕捉和比较状态    All
CHnF bits can be checked using only one read of STATUS.
      uint8 CHn;

      FTM1_STATUS = 0x00;                        //清中断标志位

      CHn = 0;
      if( s & (1 << CHn) )
      {
         if(gpio_get(PTC10))
         {
             RightWheel_Count++;
         }
         else
         {
```

```
            RightWheel_Count--;
        }
    }

}

void FTM2_INPUT_IRQHandler(void)       //第一路测速，左轮
{
    uint8  s  =  FTM2_STATUS;                  //读取捕捉和比较状态    All
CHnF bits can be checked using only one read of STATUS.
    uint8 CHn;

    FTM2_STATUS = 0x00;                        //清中断标志位

    CHn = 0;
    if( s & (1 << CHn) )
    {

        if(gpio_get(PTC11))
        {
            LeftWheel_Count--;
        }
        else
        {
            LeftWheel_Count++;
        };
          /**********************/
    }
}

/*!
 *    @brief          PORTD 端口中断服务函数
 *    @since          v5.0
 */
void PORTC_IRQHandler(void)
{
```

```
    uint8    n = 0;       //引脚号
    n = 5;
    if(PORTC_ISFR & (1 << n))                  //PTD7  触发中断
    {
        dengta++;

        PORTC_ISFR    = (1 << n);              //写 1 清中断标志位


    }
}
```