第十二届“恩智浦”杯全国大学生

智能汽车竞赛

技 术 报 告

学　　校：　　中南大学

队伍名称：　　比亚迪金牛座 2017

参赛队员：　　黄竞辉

任宏宇

魏佳雯

带队教师：　　王　击

周成训

关于技术报告和研究论文使用授权的说明

本人完全了解第十二届“恩智浦”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，比赛组委会和恩智浦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：

带队教师签名：

日　　期：

2017 年 8 月 14 日

第一章 引言

1.1 背景介绍

全国大学生智能汽车竞赛是在统一汽车模型平台上，使用恩智浦公司的16位、32位微控制器作为核心控制模块，通过增加道路传感器、设计电机驱动电路、编写相应软件以及装配模型车，制作一个能够自主识别道路的模型汽车，按照规定路线行进，以完成时间最短者为优胜的竞赛。该竞赛是涵盖了控制、模式识别、传感技术、电子、电气、计算机、机械及车辆工程等多个学科的科技创意性比赛。

全国大学生智能汽车竞赛已经成功举办了十一届，比赛规模不断扩大、比赛成绩不断提高。通过比赛促进了高等学校素质教育，培养大学生的综合知识运用能力、基本工程实践能力和创新意识，激发大学生从事科学研究与探索的兴趣和潜能，倡导理论联系实际、求真务实的学风和团队协作的人文精神，为优秀人才的脱颖而出创造条件。

从2017年1月开始，我们就开始着手准备这项赛事。历时7个月时间，经历了机械构造、硬件方案、算法思路的不断创新，这些创新体现在设计理念上，也贯穿赛车制作过程的始终。由于这些创新，赛车各方面综合性能得到提升，并且获得了良好的赛场表现。本技术报告将详细介绍我们为第十二届“恩智浦”杯全国大学生智能汽车竞赛而准备的智能车系统方案。

1.2 整车设计思路

1.2.1 控制系统

智能车系统是一个相对复杂的反馈系统(如图1.1)。CMOS摄像头采集的赛道图像信息、编码器采集到的车体运行速度、姿态传感器采集到的车体姿态信息，是反馈控制系统的输入量。执行器（直流电机）以及模型赛车构成反馈系统的装置。恩智浦 Kinetis 系列的32位单片机 MK66FN2MVLQ18 是系统的控制器。在智能车系统的搭建到赛车快速、稳定地按照赛道行驶的整个过程中，反馈原理是我们分析问题和解决问题的基本原理。

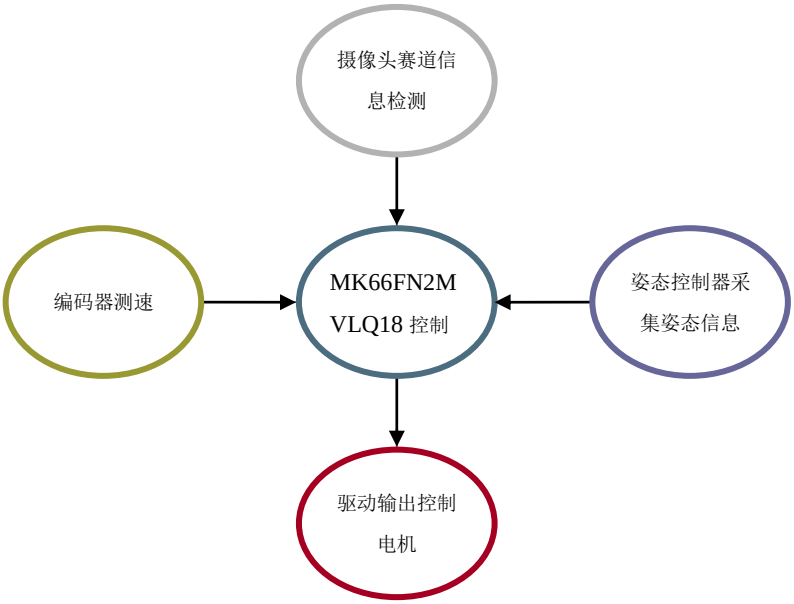


图 1.1，智能车控制系统

1.2.2 赛车整体结构设计

关于赛车的整体结构布局（如图 1.2 所示），我们主要思路是：减轻车体重量，电路板一体化处理，尽量降低并合理调整车体重心。为此，我们将电路板体积做到了最小，在四层板情况下集成度做到了最高；选用轻质单板 CMOS 摄像头（鹰眼 OV7725 摄像头）；选用轻质 3K 碳卷杆安装摄像头；尽可能的降低了摄像头高度(碳杆长度约 25CM)；

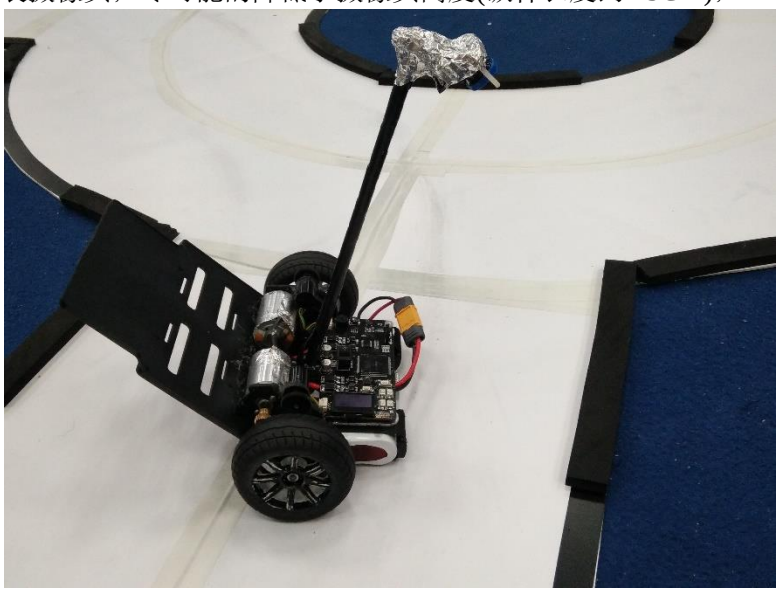


图 1.2，车模整体结构

第二章 机械结构设计与调整

2.1 总体设计思路

关于赛车的整体机械机构，我们从以下几个方面考虑：

① 重心

平衡车模的重心相较于四轮车来说尤为重要，这将最终决定该车模的加减速性能与过弯最大速度，我们采用了第十届重庆大学“不慢队”的电池支架方案，并适当的掰折，使得最终安装上电池后，处于运行状态下的车模重心极低。同时降低摄像头高度，使车模在俯仰时受摄像头的影响减小。

② 重量

对于平衡车来说，重量越轻越好，这样其转动惯量小过弯将更加迅速。我们仅使用了一块集成的四层板作为主控板与驱动板，尽可能的不使用复杂的外部接线，尽可能使用尼龙螺丝螺母而不是金属材料，尽可能减小 PCB 板体积。刚搭车模时，总质量达到了 850g，进行一系列的修改后，最终质量稳定在 810g，最轻时可以做到 780g。

③ 重量分配

平衡车的加减速来源于其机械结构的重量分配，有无零位决定了该车模能否静止站立。一般来说无零位的车模加速会快，但是这样的车模会在长直道上翻车。因此适当的给一点零位，运行时可以适当减速保证安全。

2.1 车模底盘调整

切除笑脸部分，并切除底部横杆的中间部分，只保留两端，最终状态如图 2.1。

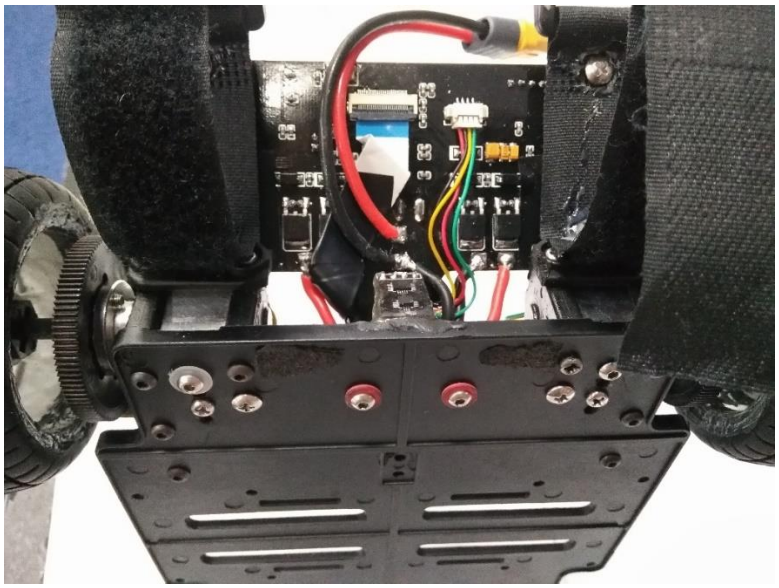


图 2.1 车模底部

2.2 测速机构的安装

E 车模原厂配置的测速装置是光电码盘，其测速精度不够，而车底盘上预留的欧姆龙编码器安装孔可以安装精度更高的欧姆龙编码器，但是其体积太大，重量太大，故最终采用 MINI512 线磁编码器。因为没有安装位置预留，所以我们用两片迷你编码器安装支架用胶水粘合在车模上，将编码器安装在支架上，最后用胶枪加强固定。如图 2.2 所示。

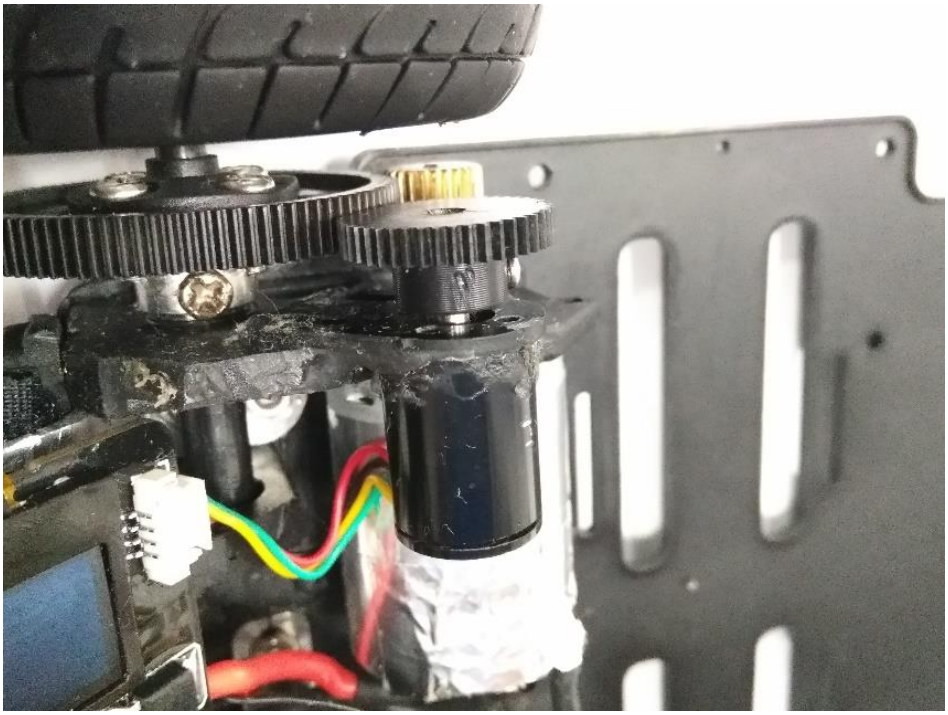


图 2.2 mini 编码器安装

2.3 姿态传感器的安装

我们自制了一块极小的九轴姿态传感器单元（FXAS2100 + FXOS8700），并将其安装在底盘最低的位置，这样它受车模抖动的影响就会很小。同时姿态传感器的其中一轴与摄像头的碳杆水平，在进行逆透视变换时可以更加方便。如图 2.1 所示

2.4 摄像头的安装

通过 3D 打印，我们自制了一个摄像头支架，如 2.3 所示。

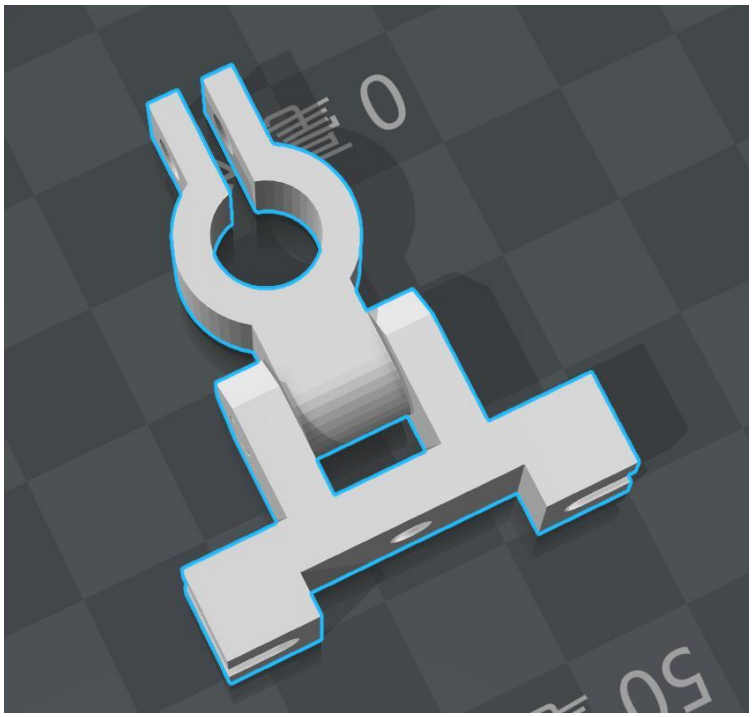


图 2.3 摄像头支架

安装时全部采用尼龙螺丝螺母，最终调整好后，用 502 胶水固定。

2.5 轮胎处理

这是一个避之不及的问题，绝大多数队伍会说自己不处理轮胎（那是不可能的），或多或少地，轮胎决定了一个车模运行的稳定程度与速度。虽说将智能车比赛戏称为“智能轮胎大赛”有点嘲讽的意味，但是一副优秀的轮胎确实可以解决不少控制上的问题。厂家的质量已经摆在那里，其余的发挥全靠选手。我对轮胎的处理没有多少研究，仅有的只是去年在 B 车轮胎上涂了一年的软化剂，它的轮胎变大了不少，摩擦力也变大了。因此今年的比赛也是一样，偶尔涂抹一点软化剂，不跑时用保鲜膜包裹保持其湿润，将车模置空放置保持轮胎的自然形态，经常跑。对于 E 车的轮胎来说，一般跑一到两周轮胎就会变鼓，差不多就可以上场了。

第四章 硬件设计

硬件稳定是一部车稳定的前提，只有稳定可靠的硬件基础才能为软件算法调试提供极大的帮助。

4.1 总体方案

系统硬件电路主要由两块 PCB 板构成，主控板和姿态传感器。如图 4.1 所示。集成了整个系统的逻辑电路和驱动电路，主控制芯片采用官方推荐的 32 位微控制器 MK66FN2M0VLQ18。

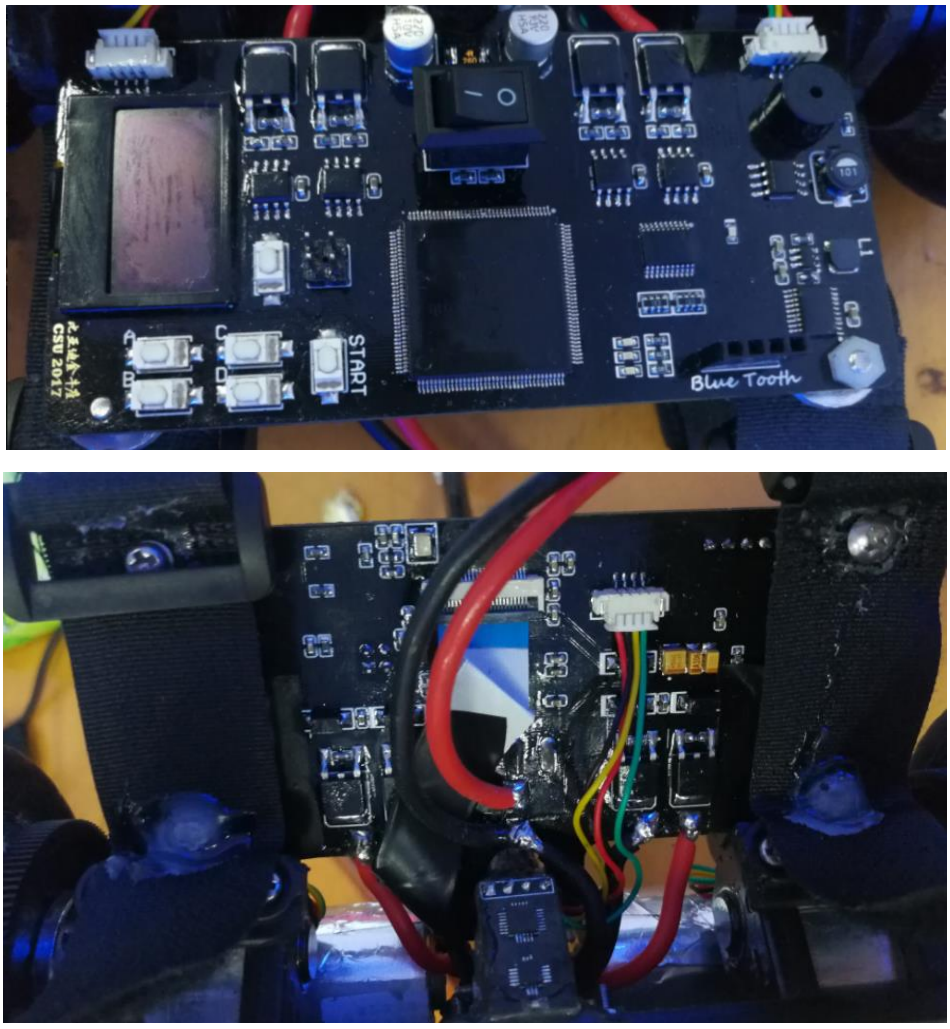


图 4.1 主控板、姿态传感器

4.2 传感器模块

4.2.1 摄像头的选择

对于CMOS摄像头分为数字和模拟两种。其中数字摄像头OV7725信噪比更高、速度更快、稳定性更好和微光灵敏度更高，速率可达150帧每秒，去噪点能力极强，效果非常理想，因此，最终我们选择了鹰眼硬件二值化摄像头。

4.2.2 编码器测速模块

磁编码器是一种通过磁转换将输出轴上的机械几何位移量转换成脉冲或数字量的传感器，这也是目前应用最多的测速传感器之一。其获取信息准确、精度高、应用简单。

采用增量式 512 线磁编码器，其供电电压为 3.3V，输出为小幅值的正弦信号，送入单片机进行运算。

4.2.3 倾角及角速度传感器

姿态传感器芯片采用恩智浦公司生产的 fxas21002 陀螺仪芯片和 fxos8700 加速度计芯片，其为汽车级传感器芯片，具有精度高，体积小，稳定性好等优点。原理图如图 4.2，实物图如 4.3 和 4.4 所示。

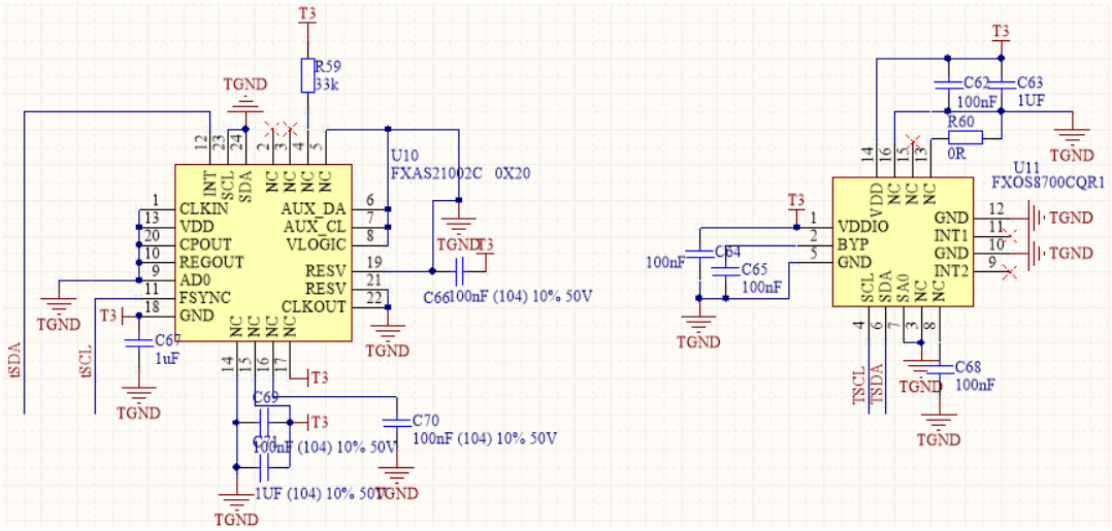


图 4.2 姿态传感器原理图

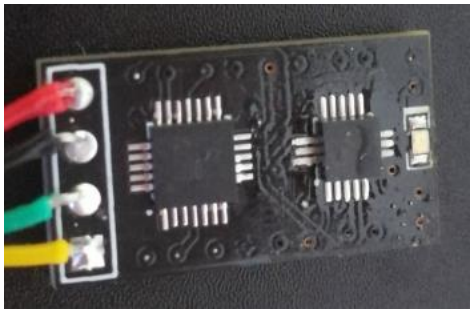


图 4.3 姿态传感器模块正面

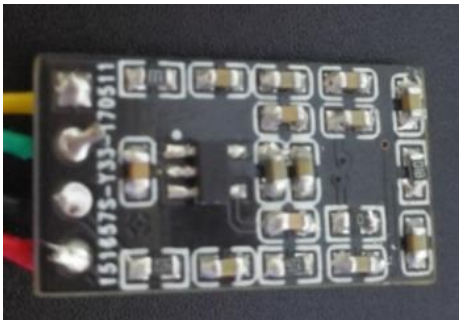


图 4.4 姿态传感器模块反面

4.3 MCU 主控部分

MCU 控制板包括滤波电路，晶振模块，主控芯片 MK66FN2M0VLQ18。

Kinetis-32 位系列针对一系列成本敏感型汽车车身电子应用进行了优化。Kinetis 产品满足了用户对设计灵活性和平台兼容性的需求，并在一系列汽车电子平台上实现了可升级性、硬件和软件可重用性、以及兼容性。

紧凑的封装使得这些器件适于空间受限应用，如小型执行器、传感器模块和转向柱集成模块。

4.4 电源模块

由于电源对高频干扰具有较强的抑制作用，同时由于其低功耗特点，在进行电路板设计时，可以减少散热片的体积和 PCB 板的面积，有时甚至不需要加装散热片，方便了电路设计与使用，提高了稳定性能。经过设计，对所有模块设计为 3.3V 或 12V 电源输入，保证了电源部分的稳定性。对于 3.3V 稳压输出，采用 TI 公司生产的 TPS 系列降压稳压芯片，确保电源的稳定性及可靠性。对于规定的电池输入电压在 8.4-7.0V，为保证电压的稳定，先对电池使用 TPS564201 进行 5V 稳压，再对 5V 使用 TPS75233 进行 3.3V 稳压。理论上 TPS564201 最大电流 4A，TPS75233 最大电流可达 2A，完全满足线路板的供电需求。电路图如图 4.5 所示。

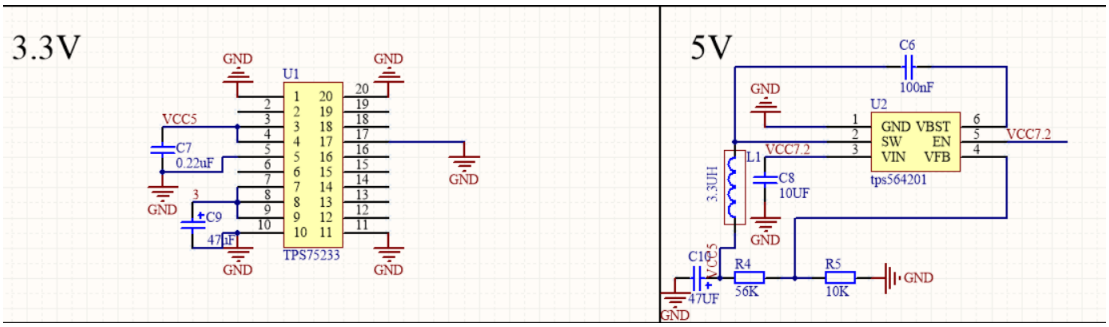


图 4.5　3.3V 稳压原理图

对于 12V 需求的驱动模块，采用 MC34063 的经典升压方案。电压调节电阻选择 1.2K 和 10K，电压泵升至 11.4V 左右，供驱动模块使用。

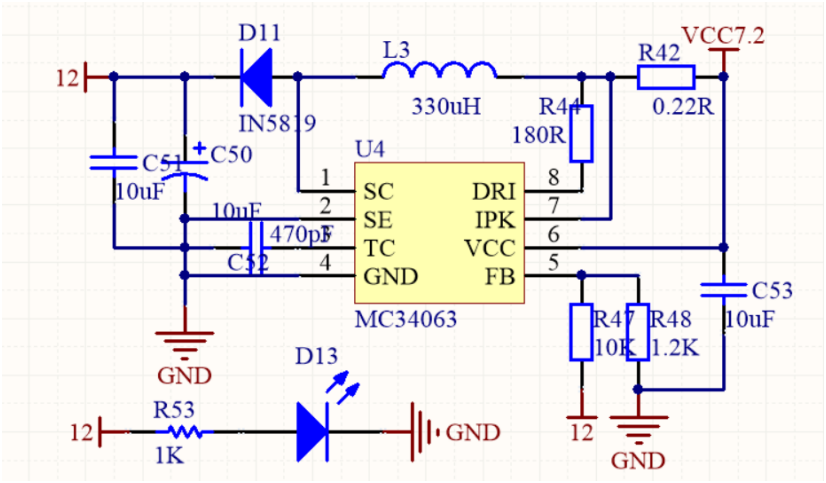


图 4.6　12V 稳压原理图

4.5 电机驱动模块

驱动单元是控制系统的重要组成部分，驱动电路经过改进，最终选取以 IR2184 作为驱动芯片的 MOS 驱动电路，其电路结构简单，负载能力强，为赛车的加速和制动性能以及上限速度得到了很大程度的提高。由于将驱动与主控集成在一块线路板上，所以在绘制时务必做好隔离，电路如图 4.7 所示。

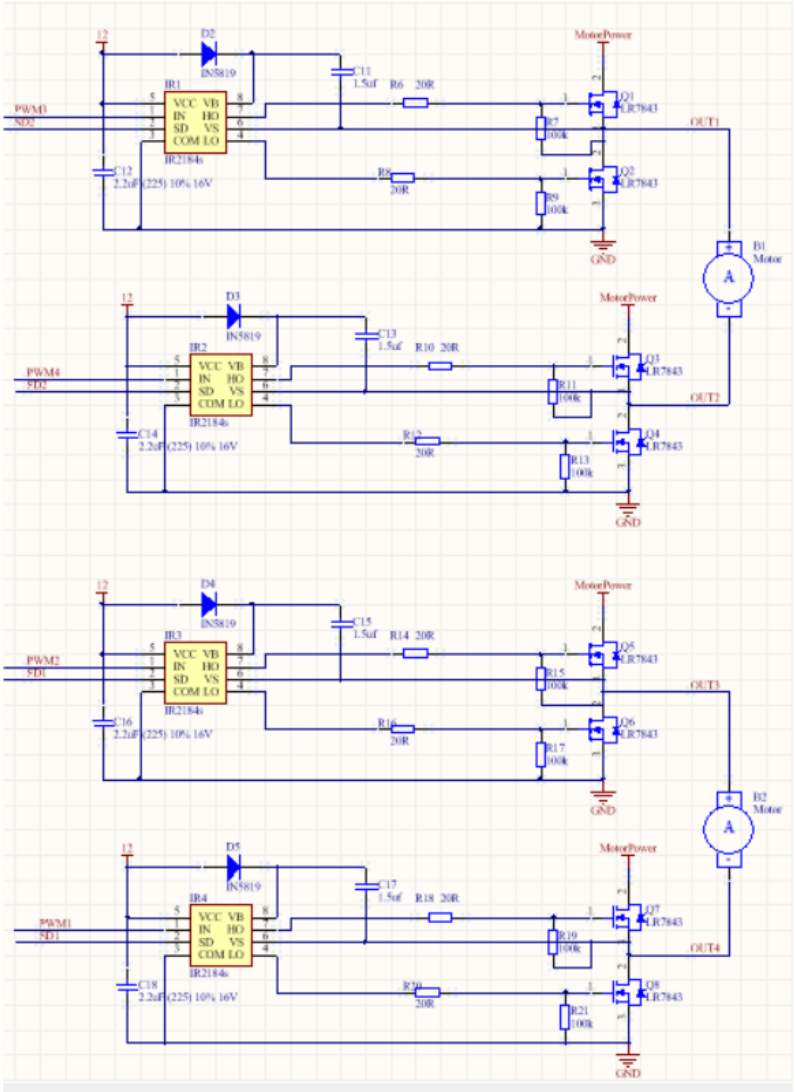

MotorPower
IR1
IR2
IR3
IR4
IR2184s
IN5819
PWM3
SD2
PWM4
SD2
PWM2
SD1
PWM1
SD1
OUT1
OUT2
OUT3
OUT4
GND
B1 Motor
B2 Motor

图 4.7　驱动电路

第五章 软件设计

5.1 姿态计算与姿态控制

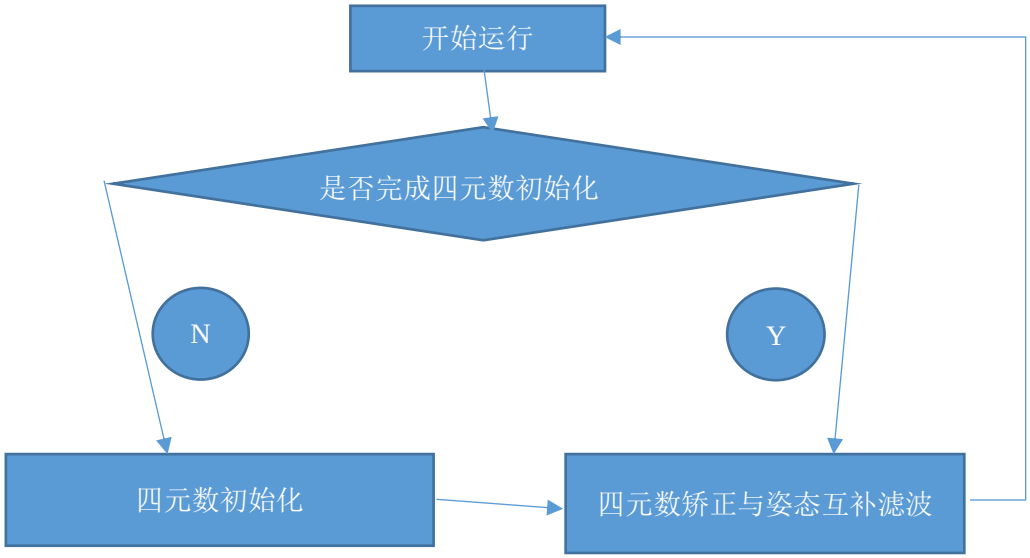
5.1.1 基于四元数方法的姿态解算

载体的姿态解算算法是实现捷联式惯性导航系统精确导航的核心技术之一。捷联惯导是一种自主式的导航方法。该方法将陀螺仪和加速度计直接安装在载体上，省掉机电式导航平台，利用计算机软件建立一个“数学平台”来代替机电平台实体。由于其结构简单且抗干扰能力强，目前已成为航空航天、航海、机器人、智能交通等领域的研究热点之一。

姿态解算是捷联式惯性导航系统的关键技术，通过姿态矩阵可以得到载体的姿态和导航参数计算需要的数据，是捷联式惯导算法中的重要工作。载体的姿态和航向体现了载体坐标系与导航坐标系之间的方位关系，确定两个坐标系之间的方位关系需要借助矩阵法和力学中的刚体定点运动的位移定理。通过矩阵法推导方向余弦表，而刚体定点运动的位移定理表明，定点运动刚体的任何有限位移都可以绕过定点的某一轴经过一次转动来实现。目前描述动坐标相对参考坐标系方位关系的方法有多种，可简单地将其分为 3 类，即三参数法、四参数法和九参数法。三参数法也叫欧拉角法，四参数法通常指四元数法，九参数法称作方向余弦法。欧拉角法由于不能用于全姿态飞行运载体上而难以广泛用于工程实践，且实时计算困难。方向余弦法避免了欧拉法的“奇点”现象，但方程的计算量大，工作效率低。随着飞行运载体导航控制系统的迅速发展和数字计算机在运动控制中的应用，控制系统要求导航计算环节能更加合理地描述载体的刚体空间运动，四元数法的研究得到了广泛重视。

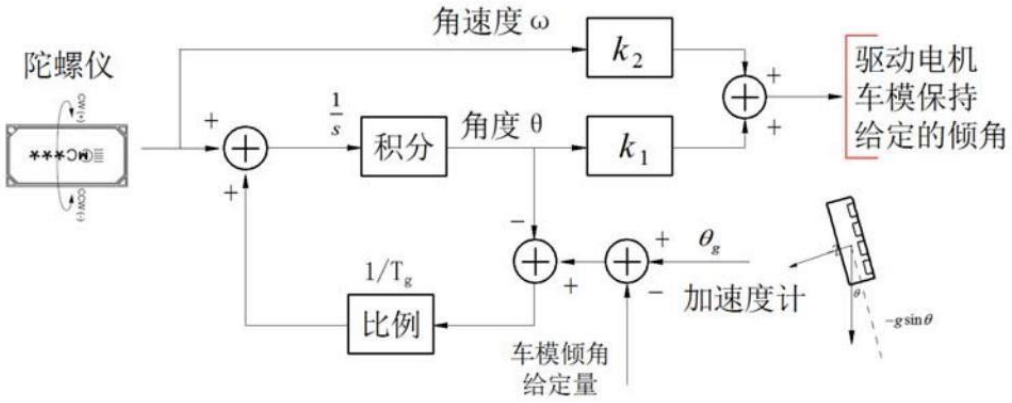
四元数的数学概念是 1843 年由哈密顿首先提出的，它是代数学中的内容之一。随着捷联式惯性导航技术的发展，为了更简便地描述刚体的角运动，采用了四元数这个数学工具，用它来弥补通常描述刚体角运动的 3 个欧拉角参数在设计控制系统时的不足。四元数可以描述一个坐标系或一个矢量相对某一个坐标系的旋转，四元数的标量部分表示了转角的一半余弦值，而其矢量部分则表示瞬时转轴的方向、瞬时转动轴与参考坐标系轴间的方向余弦值。因此，一个四元数既表示了转轴的方向，又表示了转角的大小，往往称其为转动四元数。

在二轮平衡智能车控制系统中，使用较多的是欧拉角中的俯仰角（Pitch）、偏航角（Yaw）、俯仰角速度以及偏航角速度，用俯仰角与俯仰角速度控制车体平衡，用偏航角与偏航角速度控制车体转向，从而实现精确导航。其步骤如下：



5.1.2 俯仰角姿态控制

在俯仰角控制上，采用了传统 PD 控制，且调整参数使其控制强度加大（即直立环很硬），由于速度控制与俯仰角姿态控制是外环与内环的关系，因此作为内环的俯仰角姿态控制需要对输入期望进行最终限幅与变化限幅，使其运行稳定。



5.1.3 姿态稳定性评估

车模运行过程中因为抖动、遇到坡道或各种不可遇见的情况发生时，姿态稳定性会下降，此时应该修正自身的控制参数以匹配环境的突变。实现方法如下：

采集运行过程中的加速度、横滚角速度、偏航角速度信息，对其进行方差计算以及数据

突变判定从而获取稳定性判据，当车模出现不稳定状况时，其方差会不稳定且某些实时信息会有较大突变，根据不同的突变情况做出相应的参数修正。

5.2 速度控制

二轮平衡车的速度控制受制于其欠驱的特性，速度跟随性很差，且很容易受到机械的影响，因此对其进行精准速度控制的可能性不大。

平衡车的速度控制本质上是俯仰角度的控制，因此速度闭环的内环应为俯仰角度闭环，从视觉上看，就是低头加速，抬头减速。由于速度的跟随性问题，为了尽可能的加快响应，在速度控制上我们采取了单纯比例控制，其输入为速度偏差，输出为角度偏差（与给定正常跑动角度的偏差），同时为了保证弯道速度的极限，只对直道进行速度控制，实践结果表明，这样的控制能够在一定限度内将速度控制在一定范围内。

5.3 基于 ESP 车身电子稳定系统车身安全控制算法

如后轮驱动汽车常出现的转向过多情况，此时后轮失控而甩尾，ESP 便会刹慢外侧的前轮来稳定车子；在转向过少时，为了校正循迹方向，ESP 则会刹慢内后轮，从而校正行驶方向。

ESP 系统是汽车上一个重要的系统，通常是支持 ABS 及 ASR 的功能。它通过对从各传感器传来的车辆行驶状态信息进行分析，然后向 ABS、ASR 发出纠偏指令，来帮助车辆维持动态平衡。ESP 可以使车辆在各种状况下保持最佳的稳定性，在转向过度或转向不足的情形下效果更加明显。ESP 一般需要安装转向传感器、车轮传感器、侧滑传感器、横向加速度传感器等。

ESP 系统应用与二轮平衡车上的三个控制方案即为点刹（满反偏占空比）、满正偏占空比、零占空比，根据不同的情况做出相应的动作。当内轮转速过大时，根据其与姿态速度的差值合理选择点刹或零占空比控制，过小时反之，而当外轮轮速过大或过小时亦然。同时，由于系统的欠驱特性，为保证车模的平衡与稳定，某些动作不可以同时完成，如一轮满正偏而一轮满反偏，这样的动作要取消。应用 ESP 系统进行辅助控制后，可以应对侧滑、车轮抬起、车轮抬起后对地摩擦、碰撞等危险情况。

5.4 基于局部路径记忆的路径优化与参数动态修正

大多数的控制方案均为高频率实时控制，这样的控制方案相对于路径来说，是开环的，并不可以随着历史路径信息的改变而做出相应的修正，具体表现如下：当某个或某几个控制周期未达到期望时，或者控制补偿不及时时，路径出现偏差，那么就会对以后的所有路径产生很大的影响。这样的情况很容易在提速时出现，从而限制最高速度。

局部路径记忆即为记录最近一米或两米内的所有轨迹，用一套自定义的算法对其进行评判得到几个评判参数，由这些参数来确定是否对此时的控制参数进行修正，使得今后的轨迹能够在一定误差范围内吻合期望路径。大致实现方法如下：

将路径内某点期望值与实际值进行比较，得出相对误差，在一定距离内计算范围内误差偏移总量 D，当其为负数时，表现为控制不足，为正数时，表现为控制超调，根据不足量与超调量合理修正转向控制参数。

将路径内的速度曲线描绘出来，由时间最短的终极目标，得出这条速度曲线必须能够满足减速小，加速大的要求。直立车因为速度控制响应慢的原因，其弯道降速会很明显，原因之一是其自身欠驱，另一原因是轮胎，当到轮胎满足要求时，即可针对第一个原因进行调整。当降速大时，减小转向曲率，使其能够充分加速，但是减小转向曲率的弊端就是路径会稍差，因此必须对减小曲率的值进行限幅，在路径最优与速度最优之间做出相对合适的选择。

当车模速度足够快时（>= 3.3M/s）,会因为法向离心加速度过大而抬轮，此时应该减小转向曲率，缓解一部分的压力。这是远远不够的，还应该将车模俯仰角降低，即降低重心。两者结合后，加上原有的 ESP 车身稳定保护，可以使抬轮极限值提高至 3.8-4.0M/s,当然这样的情况很少发生，以目前的赛道来说，很少有这样的直道可以加速到很高的速度。

5.5 转向控制

在转向控制上，我们将角速度作为内环，路径曲率作为外环进行控制，使用了传统的 PD 控制，加上一系列的控制补偿，使得车模能够在高速时与低速时达成一致的控制效果。由于不采用电流环作为输出内环，需要将电池电压进行采集，使用一阶滤波对电池电压滤波与标准电压的比值作为电机输出的补偿。

5.6 图像处理

5.6.1 基本边缘提取

我们采用了 OV7725 山外鹰眼二值化摄像头，其输出为压缩二值化图像，将其解压缩后，得到 60*80 的黑白图像。对于边缘提取，有两种解决方案，一是直接对压缩二值数据进行按位提取，这样的做法速度快，二是对解压后的图像进行边缘追踪提取，这样的做法可以省却后续对真实边界的判定，综合考虑后，采取第二种方法。即对靠近车模的底部边界进行连续五行的边界初始化，然后由上一边的值追踪下一边的值，同时进行边缘判定（正常黑白跳变边、无边、全黑）。

5.6.2 基本元素判断

本届智能车竞赛中，添加了圆环这一元素，使得难度有所提高。我们将圆环判定的优先级提到最高，即在基本边缘提取后就直接进行圆环判定，判定方法如下：

① 首先找到双边的折角部分，确定这一对折角相对行距离与列距离排除干扰。能找到双边折角的，只能是圆环入口或十字出入口。

② 沿着折角部分前部的边缘趋势做延长边，直至延长双边内部全黑。

③ 排除十字即为圆环。

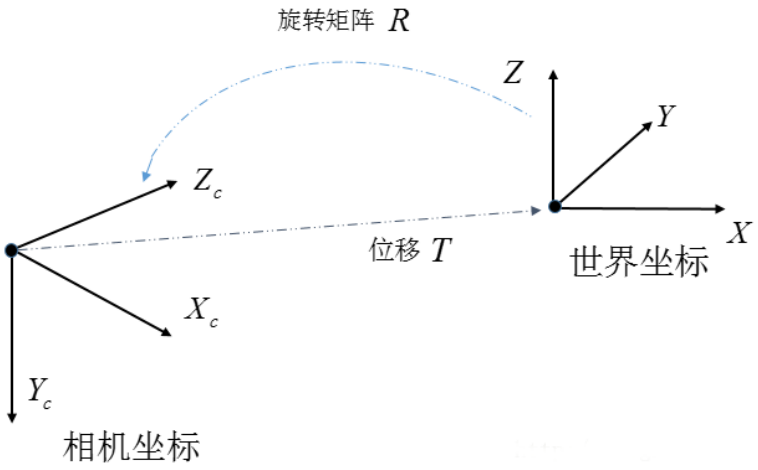
判断为圆环后，还需进行障碍判断，对于障碍，因为是边缘追踪寻找基本边缘，是不能显示的，因此需要再对边缘内部进行可行区域分析。找到其中较大的一块不可通行区域标定为障碍，然后延迟一段距离靠着另一边行驶，延迟距离根据测得的第一次发现该障碍时相对距离来决定。

坡道判断：首先要求全部的边均为黑白跳变边，其次要求中线满足一定范围，然后对其进行边缘逆透视，找到边界的折点从而判断坡道。

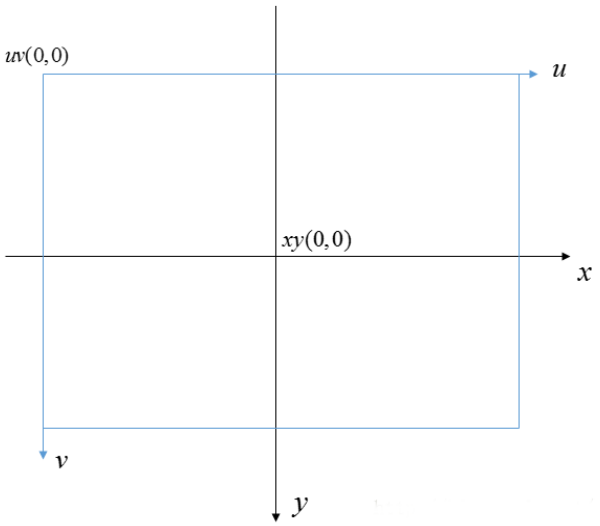
5.6.3 中线处理

平衡车的加减速来自于平衡车的俯仰角，因此其俯仰角会经常做出改变，这会极大地影响摄像头的图像，所以必须要对中线轨迹做逆透视变换，得到真实的轨迹。

将图像坐标转换到世界坐标将进行如图的变换，其旋转矩阵为一个 4×4 的矩阵



数字图像在计算机内为 MXN 数组，M 行 N 列的图像中每一个元素（pixel）数值就是图像点的亮度（灰度）。如图，在图像上定义直角坐标系 U，V，每一个像素为单位的图像坐标系坐标，由于（u,v）只能表示像素位于数组中的列数与行数，并没有使用物理单位表示该像素在图像中位置，所以需要再建立以物理单位（mm）表示的图像坐标系，该图像坐标系以图像内某一点 uv(0,0)为原点，x 轴和 y 轴分别平行于 u、v。



如图中，（u、v）表示以像素为单位的图像坐标系的坐标，（X、Y）表示以 mm 为单位的图像坐标系的坐标。

假设每一个像素在 X 轴与 Y 轴方向上的物理尺寸为 dx、dy，则图像任意一个像素在两个坐标系下的坐标由式 1 所示

$$u = \frac{x}{dx} + u_0$$

$$v = \frac{y}{dy} + v_0$$

其中(u0 ，v0) = xy(0,0)　　（式 1）

进而得到矩阵表达式

$$\begin{Bmatrix} u \\ v \\ 1 \end{Bmatrix} = \begin{Bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{Bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$$

（式 2）

以上为像素与像平面的变换关系，再经过旋转变换，投影变换后，最终逆透视变换矩阵完成，各矩阵对应关系如图 11 所示。

$$Z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

相机与世界

投影关系

像素与像平面

搭好车模后，就对相机的信息进行测量（高度、焦距、与支撑杆的夹角），然后由姿态传感器得出支撑杆与地面的夹角，从而将矩阵内的全部参数获得。

对中线轨迹进行逆透视变换后，取其中的几个坐标进行曲率计算，得到最终的实时转向曲率，然后对更远方的几个坐标进行曲率预测，描绘出预测路径，以方便后续的路径控制。由于图像分辨率的原因，并不能做到大幅度俯仰角内的曲率一致性，但是保证俯仰角+-20°内转向一致还是可以的。

5.7 路径曲率与图像曲率的跟随及最优路径的选取

受制于图像处理的低级算法，图像曲率的得出是期望车模一直运行于赛道中心，而通过转向上对于速度与角速度的处理及 ESP 车身自稳定系统，在高速时会强制使车模运行于安全转向离心加速度下，因此其实际路径会和期望路径产生较大的偏移。其根本原因是因为算法本身是在控制高频率的情况下通过控制参数的调整使得实际与期望一致，在第十届以前直立车大多采取的方案就是这样，最终通过模糊控制使其在各种速度下跟随一致。而后由于车模速度不断提高，又出现了车身安全稳定系统，使得车模能够在高速下对转向进行自适应控制。本篇技术报告中提及的局部路径记忆方法，虽然可在一定限度内对高速路径进行约束，但如果想要把速率（非速度，路径与实时速度共同决定了最终成绩，速率至上）提升到极致，还需要一些更加革新的控制方案。

智能车竞赛早年时，清华大学提出了全赛道记忆的方案，但是在目前的比赛情况下不太可能实现（目前赛道情况有点复杂）。如果要路径最优，则必须要开全图视野，这样才能达到 K1999 那样的最优路径，难度很大。在当前情况下，不如把对最优的要求降低些，给出一些约束条件：最小降速路径、最短路径、最大转向离心加速度、全局速率最高与局部速率最高。这些条件是相互影响的，要做到最小降速必须加大转向半径那么路径肯定加长，同样地高速时无法以小半径转向必须限定在最大转向离心加速度范围内那么路径也会加长，而全局速率最高与局部速率最高本身也相互制约，虽然局部最优后全局也会最优，但考虑到之前那几个硬性条件，没有全图视野的情况下，局部必定不能最优。这是一个很麻烦地事，至少对于一个 MCU 来说。因此我的解决方案是：

1、 由中线逆透视后得出基本赛道骨骼，并向两边延伸一段距离，车模中心位于这个区间内的任意点车轮不会出界。由此得到一个可以安全通行的区域，对于实时控制来说，由该区域可以得到一个曲率范围。最后在这个曲率范围内计算一定距离内的图像预测轨迹。

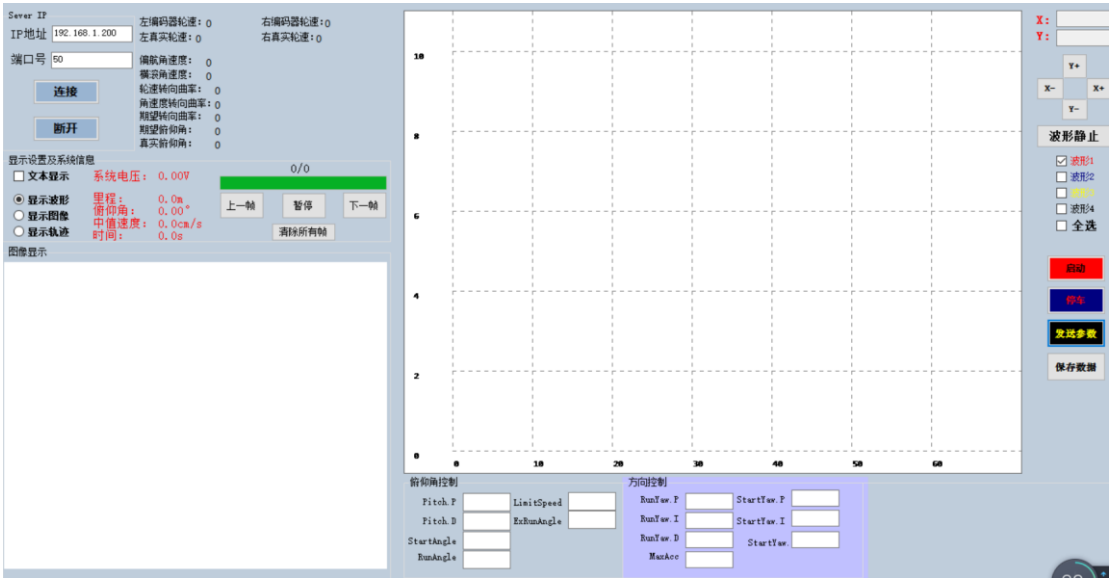
2、 由当前速度与偏航角速度计算出一个转向向心加速度，再由差速以同心圆定理得到另一个转向向心加速度，给出两个权重得到最终转向向心加速度，评估此时的加速度是否在安全行驶范围内，如果是危险行驶，那么就给出一个可供选择的安全曲率范围。最后在这个安全曲率范围内计算在当前速度下并随曲率增大而降速的实际预测轨迹。

3、 以上操作均由单片机完成，两个轨迹会有交集部分（如果没有则优先图像轨迹并取曲率最小的那一条），之后根据不同的路况与局部记忆的内容对交集内的轨迹进行排除，最终选取交集内曲率最大的那一条（就是路径最短的那一条）。

5.8 上位机制作与调试

我们使用了 ESP8266 串口 WIFI 模块作为通讯设备，其最大传输速率可达 4M，而 K66 的 UART 速率最大可达 6.8M，因其高速时易发生丢包等问题，所以最终选用 3M 的通信速率。

在上位机制作上，使用 VS2015 平台（基于 C#），最终界面如下：



相对于蓝牙来说，WIFI 理论上可以做到超长距离通信，在日常使用时，只需要一个信号良好的路由器即可。更深入的话可以连入 Internet。

第六章 赛车主要技术参数

表 6.1 赛车技术参数统计

项目	参数
路径检测方法（赛题组）	光电类直立组
车模几何尺寸（长、宽、高）（毫米）	200*200*300
车模轴距/轮距（毫米）	210（轴距）
车模平均电流（匀速行驶）(毫安)	2000
电路电容总量（微法）	1000
传感器种类及个数	OV7725 摄像头 1 个，迷你 512 线编码器 2 个，FXAS2100 陀螺仪 1 个，FXOS8700 加速度计地磁仪 1 个
新增加伺服电机个数	0 个
赛道信息检测空间精度（毫米）	10
赛道信息检测频率（次/秒）	75
主要集成电路种类/数量	数字 74 系列集成电路
车模重量（带有电池）（千克）	0.820

第七章 总结与展望

7.1 总结

要实现对高速行驶汽车的自主智能控制并不是一个简单的自动控制问题，它涵盖了控制、模式识别、力学、光学、电磁学、传感技术、电子、电气、计算机、机械及车辆工程等多个学科。本文采用的控制核心是一款恩智浦公司生产的 32 位微控制器——MK66FN2MVLQ18，利用了微控制器的强大功能实现了智能小车对路径的自主寻迹，以及在未知环境下，结合一定的算法，实现了对智能小车的高速导航控制，从最终测试结果来看，本系统具有较好的控制性能与对未知环境的适应能力。

对智能小车系统的自主控制，可以分为三大部分：首先是对路径信息的提取与识别，即智能小车的寻迹；然后是计算并规划路径，得出可以安全行驶的路径；最后是对双轮车速的控制，即智能小车的驱动控制。

7.2 展望

智能车系统的研究十分复杂，需要解决的问题很多，任务非常艰巨，不是一蹴而就的，必须经过长期的理论研究和实践探索才能够取得突破和进展。展望未来，对于我们所研究的这类智能小车，我们认为今后还可在以下几个方面做进一步的研究和提升。

1）、采用优质的数字摄像头，数字摄像头可以非常方便的调整其参数，对环境适应能力更强。

2）、设计二轮车悬挂系统，可以极大提升其转向能力，有希望超过四轮车的速度

3）、在图像处理方面，可以研究更先进的算法，不过，这需要配合微控制器性能的提升，可以考虑使用 FPGA、DSP 等更先进的处理器。

4）、硬件传感器方面。可以尝试采用 CMOS 摄像头装于摄像头云台方案。

谢　辞

今年能够参加第十二届恩智浦杯全国大学生智能汽车赛我们首先要感谢中南大学信息科学与工程学院王击老师的悉心指导，指导老师多次询问研究进程并为我们指点迷津，甚至和我们一起奋战至凌晨三四点，其严谨的治学态度以及对我们的关怀使们获益甚多，感动颇深。

在准备比赛期间我们得到了以前参加过比赛的学长的许多非常实用的帮助和建议，在这里要特别感谢他们。另外也感谢一起准备比赛的队员，没有他们的帮助我们不可能完成工作。

另外我们还要感谢德州仪器公司以及恩智浦公司为我们提供的芯片支持，没有一个大范围的比较是不可能选出最优最合适元器件出来的，我们的小车能够顺利的参赛要感谢这两大公司的帮助。

参考文献：

[1] 教育部高等学校自动化专业教学指导分委员会. 关于举办第十一届全国大学生“恩智浦”杯智能汽车竞赛的通知[EB/OL]. 2015-11-01.

[2] 邵贝贝. 单片机嵌入式应用的在线开发方法. 北京：清华大学出版社. 2004 年 10 月第 1 版.

[3] 卓晴，黄开胜，邵贝贝.学做智能车[M].北京：北京航空航天大学出版社，2007.24～27.

[4] Freescale Semiconductor Inc. S12PWM8B8CV1/D Block User Guide V01.17[Z]. 2004. 1~51.

[5] National Semiconductor. LM1881 Video Sync Separator General Description[Z]. June 2003

[6] 杨明，宋雪峰，王宏，张钹.面向智能交通系统的图像处理[J].计算机工程与应用.2001 年 09 期.

[7] 余灿键，程东成，李伟强.PID 算法在智能汽车设计上的应用.——《学做智能车——挑战“飞思卡尔”杯》[C].北京：北京航空航天大学出版社，2007.133～135.

[9] Todd D.Motton. 嵌入式微控制器 Embedded Microcontroller[M].北京:科学出版社.2003 年 8 月第 1 版.

附录：

```
void main(void)
{
    EnableInterrupts;                  /////关闭全部中断
    OLED_init();                       ///OLED 初始化完成
    EventControlTimer_init();          ///时钟事件管理器初始化完成，不打开
    systick_timing(220000);            ///1MS 系统时钟
    IIC_init();                        ////I2C 初始化
    FLASH_init();                      ////flash 初始化
    CommonDelay(100);                      ////FLASH 初始化后需要延时
    My_FlashRead(0);                   ////读取末尾 flash 数据
    key_init();                        ///按键初始化
    ParameterInit();                   ///参数初始化

    SystemSettings.Wifi_Init = 'F';
    if (SystemSettings.Wifi_ON == 'T')
        ESP8266_Init();                    ////wifi 初始化

    BUZZUP;
    CommonDelay(100);
    BUZZDOWN;
    CommonDelay(50);
    BUZZUP;
    CommonDelay(100);
    BUZZDOWN;

    if (SystemSettings.ResetTest == 'T')
    {
        if (RCM->SRS0 & RCM_SRS0_POR_MASK) OLED_PrintStr(0, 0, "Power_On Reset", 16, 1);
```

```
        if (RCM->SRS0 & RCM_SRS0_PIN_MASK) OLED_PrintStr(0, 1, "External Reset", 16, 1);

        if (RCM->SRS0 & RCM_SRS0_LVD_MASK) OLED_PrintStr(0, 2, "LowPower Reset", 16, 1);

        if (RCM->SRS1 & RCM_SRS1_SW_MASK) OLED_PrintStr(0, 3, "Soft Reset", 16, 1);

        if (RCM->SRS1 & RCM_SRS1_LOCKUP_MASK) OLED_PrintStr(0, 4, "CoreLock Reset", 16,
1);

        if (RCM->SRS0 & RCM_SRS0_LOC_MASK) OLED_PrintStr(0, 5, "LossClock Reset", 16, 1);


        while (GPIO_get(14) && GPIO_get(15) && GPIO_get(16) && GPIO_get(17));
    }




    while ('F' == MotorPwm_init());   ////电机初始化
    QuadratureDecode_init();              ///正交解码初始化

    ADC_init();                           ///电源电压检测


    ImageSensor_init((uint8 *)ImageSensorData, 600);////摄像头驱动初始化
    NVIC_EnableIRQ(PIT0_IRQn);
    NVIC_EnableIRQ(PIT1_IRQn);
    OS_menu();
    while (1);
}
```

```
void Attitude_UpdateAcc(void)//深度融合更新
{
	QuaternionTypedef	EstQuaternion;
	EulerAngleTypedef	EstEulerAngle;
	QuaternionTypedef	DivQuaternion;
	QuaternionTypedef	ComAxisangle;
	QuaternionTypedef	Compensate;
	QuaternionTypedef	Last;


	QuaternionFromAcc(&MeaQuaternion, 0, YA, ZA, -1, 0, 0);
	Quaternion_ToEulerAngle(&MeaQuaternion, &MeaEulerAngle);
	Quaternion_ToAxisAngle(&MeaQuaternion, &MeaAxisAngle); //计算当前加速度计姿态

	EstEulerAngle.Roll = EulerAngle.Roll;
	EstEulerAngle.Pitch = EulerAngle.Pitch;
	EstEulerAngle.Yaw = 0;

	Quaternion_FromEulerAngle(&EstQuaternion, &EstEulerAngle);	//估计欧拉角转四元数

	  //计算估计与测得四元数偏差
	Quaternion_Invert(&DivQuaternion, &EstQuaternion);
	Quaternion_Multi(&ErrQuaternion, &DivQuaternion, &MeaQuaternion);
	Quaternion_Normalize(&ErrQuaternion);
	Quaternion_ToEulerAngle(&ErrQuaternion, &ErrEulerAngle);
	Quaternion_ToAxisAngle(&ErrQuaternion, &ErrAxisAngle);


	//轴角校正限幅

	memcpy(&ComAxisangle, &ErrAxisAngle, sizeof(QuaternionTypedef));
	if (ComAxisangle.W > ATTITUDE_COMPENSATE_LIMIT)
	{
		ComAxisangle.W = ATTITUDE_COMPENSATE_LIMIT;
	}
	Quaternion_FromAxisAngle(&Compensate, &ComAxisangle);


	//执行校正

	memcpy(&Last, &EstQuaternion, sizeof(QuaternionTypedef));
	Quaternion_Multi(&EstQuaternion, &Last, &Compensate);
```

```
    Quaternion_ToEulerAngle(&EstQuaternion, &EstEulerAngle);
    EstEulerAngle.Yaw = EulerAngle.Yaw;//不使用加速度计测偏航角
    Quaternion_FromEulerAngle(&Quaternion, &EstEulerAngle);
    Quaternion_ToEulerAngle(&Quaternion, &EulerAngle);
    Quaternion_ToAxisAngle(&Quaternion, &AxisAngle);
}

void SystemStart(void)
{
    static uint8 IsCallStop = 'F';
    if (CarInfo.Motor_ON == 'T')
        CarInfo.time_s = GetSysTime();

    Get_Car_Angle();            ///获取角度信息
    Get_Car_Speed();            ///获取编码器转速
    Get_Car_curvature();
    Get_Car_BAT();              ///获取电池电压
    Get_Car_Length();           ////计算行驶路程



    if (SystemSettings.Run_DistanceLimit == 'T'
        && CarInfo.length > CarInfo.Run_Distance
        && IsCallStop == 'F')
    {
        IsCallStop = 'T';
        CarInfo.IsStop = 'T';
    }                   /////全程停车一次
    if (SystemSettings.Wifi_Init == 'T')
    {
        SendFloat[0] = FloattoUint32(CarInfo.NowSiteX);
        SendFloat[1] = FloattoUint32(CarInfo.NowSiteY);
        SendFloat[2] = FloattoUint32(CarInfo.Battery);
        SendFloat[3] = FloattoUint32(CarInfo.time_s);
        SendFloat[4] = FloattoUint32(CarInfo.length);
        SendFloat[5] = FloattoUint32(CarInfo.pitch);
        SendFloat[6] = FloattoUint32(CarInfo.rollrate);
        SendFloat[7] = FloattoUint32(CarInfo.yawrate);
        SendFloat[8] = FloattoUint32(CarInfo.speed_Mtrue);
        SendFloat[9] = FloattoUint32(CarInfo.speed_Ltrue);
        SendFloat[10] = FloattoUint32(CarInfo.speed_L);
            SendFloat[11] = FloattoUint32(CarInfo.speed_Rtrue);
        SendFloat[12] = FloattoUint32(CarInfo.speed_R);
```

```
                SendFloat[13] = FloattoUint32(1 / CarInfo.curvature);
                SendFloat[14] = FloattoUint32(CarInfo.speed_Mtrue / CarInfo.yawrate * 100);
                SendFloat[16] = FloattoUint32(ImageStatus.ExpectCur);

        if (RoadType == Normal)
            SendFloat[17] = 0;
        else if (RoadType == Obstacle)
            SendFloat[17] = 1;
        else if (RoadType == Cirque)
            SendFloat[17] = 2;
        else if (RoadType == Straight)
            SendFloat[17] = 3;
        else if (RouteStatus[RealCrossType] == 'I')
            SendFloat[17] = 4;

        SendFloat[18] = 10;
        SendFloat[19] = 20;
        SendFloat[20] = 30;
        SendFloat[21] = 40;
        SendFloat[22] = 50;
        SendFloat[23] = 60;
        SendFloat[24] = 70;
    }

    if (RoadType == Ramp)
        BUZZUP;
    else
        BUZZDOWN;
}
```