第十三届“恩智浦”杯全国大学生智能汽车竞赛



学 校：南京师范大学

队伍名称：ALUS

参赛队员：陆亮亮

盛姜聪

沈舒雨

带队教师：陈 敏

褚红燕

关于技术报告和研究论文使用授权的说明

本人完全了解第十三届“恩智浦”杯全国大学生智能汽车竞赛关于保留、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。

参参赛队员签名：陆亮亮

盛姜聪

沈舒雨

带队教师签名：陈 敏

褚红燕

日期：20180.8.18

日

摘要

本文以第十三届全国大学生智能车竞赛为背景，介绍了三轮电磁车控制系统的软硬件结构和开发流程。该比赛采用大赛组委会统一指定的F型车模，以NXP半导体公司生产的32位单片机S9KEAZ128AML为核心控制器，在IAR8.20开发环境中进行软件开发，使用电磁传感器进行赛道信息采集。整个系统涉及硬件电路设计、控制策略、整车机械架构等多个方面。为提高在高速运行下的稳定性，进行了不同方案的设计，并使用Matlab进行了数据分析以及上位机的设计调试，确定了现有的整车架构和相关控制参数。根据多个不同位置的电感采集到的赛道信息进行路径规划，通过后两轮电机差速实现转向。

关键词：磁导航、三轮、硬件构造、软件调试

目录

引言…………………………………………………………………………………1

第一章 系统总体设计 …………………………………………………………2

第二章 机械结构设计与优化 …………………………………………………3

2.1 整车布局……………………………………………………………………3

2.2 传感器的设计与安装………………………………………………………3

2.2.1 电磁传感器安装………………………………………………………3

2.2.2 姿态安装………………………………………………………………4

2.3 系统电路板的安装及连接…………………………………………………4

2.3.1 电路板的固定及连接…………………………………………………5

2.3.2 电池的安装……………………………………………………………5

第三章 电路设计说明…………………………………………………………7

3.1 电源供电模块………………………………………………………………7

3.2 驱动模块设计………………………………………………………………8

3.3 电磁传感器模块设计………………………………………………………8

3.4 起跑线检测传感器设计……………………………………………………9

第四章 三轮车软件控制设计 …………………………………………………10

4.1 电磁传感器采集与处理……………………………………………………10

4.1.1 传感器信号为模拟值…………………………………………………10

4.1.2 传感器信号具有方向…………………………………………………10

4.1.3 传感器信息处理………………………………………………………11

4.2 速度控制……………………………………………………………………11

第五章 调试与优化 ……………………………………………………………13

5.1 开发软件……………………………………………………………………13

5.2 MATLAB调试…………………………………………………………………13

心得体会……………………………………………………………………………I

参考文献……………………………………………………………………………II

附录 A 程序 ……………………………………………………………………III

附录 B 主要技术参数汇总 ……………………………………………………XIII

引言

随着现代科技的飞速发展，人们对智能化的要求已越来越高，而智能化在汽车相关产业上的应用最典型的例子就是汽车电子行业，汽车的电子化程度则被看作是衡量现代汽车水平的重要标志。同时，汽车生产商推出越来越智能的汽车，来满足各种各样的市场需求。

第十三届“恩智浦”杯全国大学生智能汽车竞赛就是在这个背景下举行的。比赛要求在大赛组委会统一提供的竞赛车模，我们选择了飞思卡尔微控制器kea128为核心控制单元的基础上，自主构思控制方案及系统设计，包括传感器信号采集处理、控制算法及执行、动力电机驱动等，最终实现能够自我识别路线，并且可以实时输出车体状态的智能车控制硬件系统。

本文先从总体上介绍了智能车的设计思想和方案论证，然后分别从机械、硬件、软件等方面的设计进行论述，重点介绍了芯片的选择和路径识别的方法，接着描述了智能车的制作及调试过程，其中包含本队在制作和调试过程中遇到的问题及其解决方法。

第一章 系统总体设计

电感电容谐振

放大滤波电路

旋转编码器

蓝牙串口模块

旋转编码器

Kea128

按键

OLED屏幕显示

按键

直流电机

图1.1 系统总体概括图

第二章 机械结构设计与优化

2.1整车布局



图2.1 整车的俯视图

2.2传感器的设计与安装

车模中的传感器包括有：电磁传感器，速度传感器，车模角度传感器等。电磁传感器采用7个，速度编码器采用 mini1024J 型 10 位精度无限角度绝对式编器，车模角度传感器采用加速度传感器MMA7361以及MPU6050的陀螺仪模块。下面分别介绍部分传感器的安装。

2.2.1电磁传感器安装

电磁传感器的安装方式极大地影响小车运行的速度和姿态，由于今年规则中规定限制了小车前瞻不得超过40cm，我们为了能够让小车更加提前地进行赛道环境判断，选择将传感器到车后轴中线上的距离选定在40cm，最大化效果。同时为了不让小车整体重量变重，我们选择用质量轻，硬度好的碳素杆来固定电感，同时为了防止前面的抖震，我们选择了双杆结构，这样的搭法虽然增加了一点重量，但由于能增强传感器的稳定性也无足轻重了。



图2.2 电磁传感器的安装

2.2.2姿态传感器安装

由于赛道上存在坡道这个元素，我们在尝试用电感来检测坡道时，经常发生误判，所以选择了用一款mpu6050的姿态传感器来识别坡道，我们将传感器按照如下方式后，测得的数据也会更加精准。

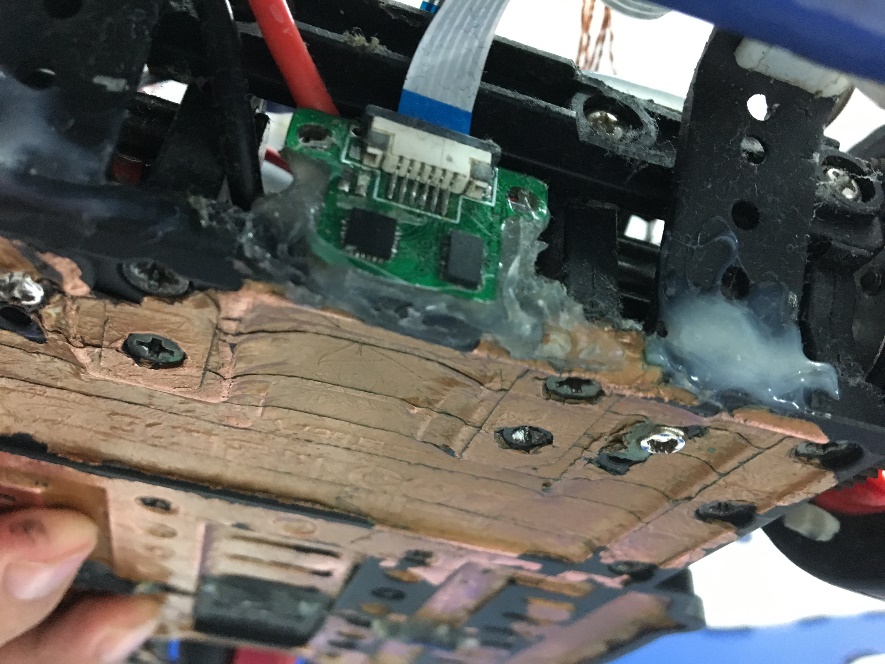


图2.3 姿态传感器的安装

2.3系统电路板的安装及连接

2.3.1电路板的安装

PCB电路板也有一点的重量，它的安装位置的不同也会影响小车整体的重心分布，在经过多次对电路板摆放位置的尝试，我们选择将电路板主要放在小车的后半部分，以免前面太重，影响转向。

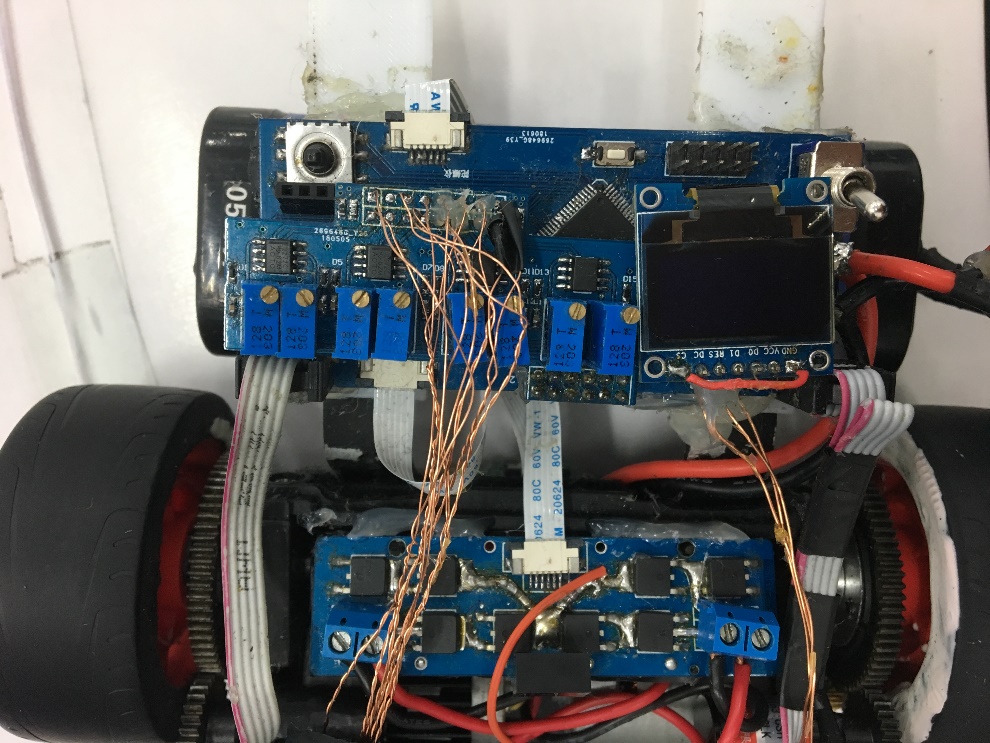


图2.4 PCB电路板摆放的俯视图

2.3.2电池的安装

由于三轮车不同于四轮车模，它主要靠双电机的差速来控制转向，为了转向的灵活性，我们借鉴了以往平衡车中电池摆放的搭法，选择了将电池放在后轮的后面，并通过铝合金结构件来固定住电池，由于小车的重心分布对转向有一定的影响，尽可能要选用结实硬度好的结构件，不然可能导致重心的偏移。



图2.5 电池摆放位置侧视图

第三章 电路设计说明

从最初进行硬件电路设计时我们就既定了系统的设计目标：可靠、高效、简洁，在整个系统设计过程中严格按照规范进行。

可靠性是系统设计的第一要求。我们对电路设计的所有环节都进行了电磁兼容性设计，做好各部分的接地、屏蔽、滤波等工作，将高速数字电路与模拟电路分开，使本系统工作的可靠性达到了设计要求。

3.1 电源供电模块

首先了解一下不同电源的特点，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。

系统中 3.3V 电路功耗较小，考虑到姿态传感器、MCU 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 TPS7333 作为稳压芯片。TPS7333 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。与 7805、LM1117 相比具有更低的工作压降和更小的静态工作电流。

下图即为tps7333的原理图，另一款稳定输出5V的tps7350同理

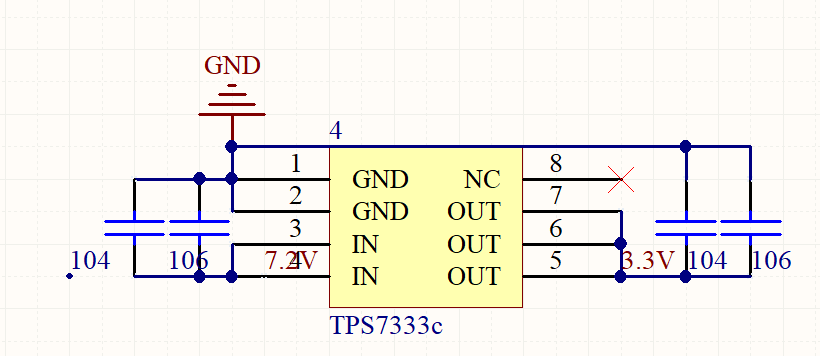


图3.1 TPS7333原理图

3.2 驱动模块设计

电机驱动的设计，采用两片IR2104S+4NMOS构成H桥进行驱动。IR2104S是H桥式N通道MOS管驱动器集成电路，特别针对于脉宽调制电动机控制。它使基于桥式电路的设计更为简单和灵活。MOS管选用IRLR7843，具有大电流内阻极小的特点。其它外围电路还有BS0512，5V升12V集成电源模块。

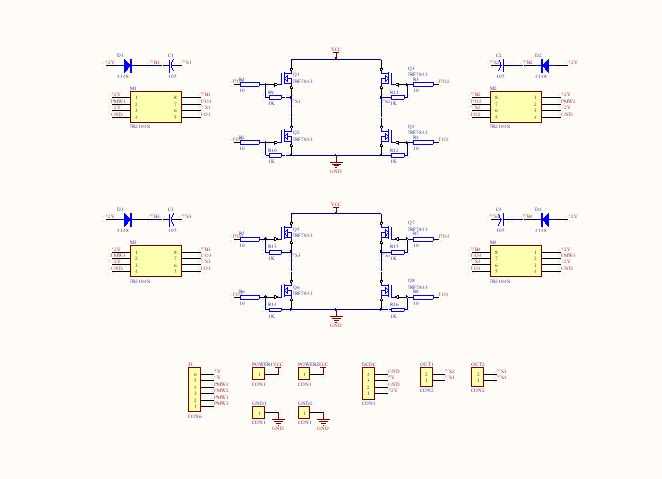


图3.2 驱动模块原理图

3.3电磁传感器模块设计

传感器是电磁车的眼睛，所以传感器模块即放大电路对于小车能否快速准确的捕捉路况信息十分重要。本系统根据LC谐振电路检测小信号的原理，选取10mH工字电感和6.8nf电容搭建LC谐振电路，但采集到的正弦信号的振幅一般只有几十毫伏，对于单片机来说太微弱。为了使单片机能够识别这种微弱信号，采用LMV358放大信号，该芯片具有稳定，放大倍数大，外围电路简单等特点。再通过检波模块得到所需要信号。如电路图所示：

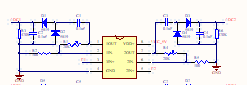


图3.3 LMV358原理图

3.4起跑线检测传感器设计

起跑线附近的永磁铁的分布是在跑道中心线两边对称分布的，如下所示：

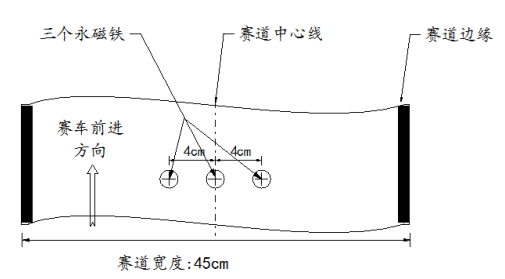


图3.5 赛道磁铁摆放位置

我们采用干簧管作为检测元器件，当干簧管靠近到永磁铁时，会产生一个通断信号。由于干簧管防抖动的性能较差，且碰撞易损坏，如果选择常闭的干簧管，在赛车运动过程中发生碰撞或抖动时，干簧管会由闭合变成断开，产生误检测起跑线信号。为了避免误触发，且让单片机的信号是低电平触发，我们选用的是常开干簧管。选型上，我们采用塑封型的干簧管，稳定耐碰撞。



图3.4 干簧管实物摆放图

第四章 三轮车软件控制设计

4.1 电磁传感器采集与处理

在本届大赛中，电磁组的规则与上一届发生了一些变化，三轮车属于一种全新的车模第一次投入比赛，传感器依旧使用电感串联电容。增加了新的赛道元素：颠簸路面、圆环。

4.1.1 传感器信号为模拟值

电磁组需要检测的信号为大小 100mA，频率为 20KHz 的方波信号，赛道由导线铺成，导线周围分布着交变的电磁场，由于赛道的各种形状，使得磁场发生叠加，不同的赛道形状形成不同的特征磁场，如下图为十字线附近的磁场。赛道信息相对于传统黑白线具有信号可以提供模拟信息的优势，我们利用电磁赛道这种优势，完善小车控制算法，达到了较好的控制效果。

磁场是矢量，在空间的分布为具有方向性，所以传感器检测到的信号也具有特定的方向性。在实际检测的时候发现，不同方向传感器的变化规律有很大的不同，这也和磁场的分量变化规律相一致。比如，磁场垂直分量变化的比较早，但是受相邻赛道的影响较大，而磁场的水平分量恰好相反。

4.1.2传感器信号具有方向性

磁场是矢量，在空间的分布为具有方向性，所以传感器检测到的信号也具有特定的方向性。在实际检测的时候发现，不同方向传感器的变化规律有很大的不同，这也和磁场的分量变化规律相一致。比如，磁场垂直分量变化的比较早，但是受相邻赛道的影响较大，而磁场的水平分量恰好相反。

采集程序如下：

/\*\*\*\*\*\*\*\*\*\*电感数据第一次采集\*\*\*\*\*\*\*\*\*/

AD\_Get[0][0][0]=LPLD\_ADC\_Get(ADC0, DAD3);

AD\_Get[0][1][0]=LPLD\_ADC\_Get(ADC1, DAD3);

AD\_Get[0][2][0]=LPLD\_ADC\_Get(ADC1, AD20);

AD\_Get[0][3][0]=LPLD\_ADC\_Get(ADC0, AD14);

AD\_Get[0][4][0]=LPLD\_ADC\_Get(ADC0, AD20);

AD\_Get[0][5][0]=LPLD\_ADC\_Get(ADC0, DAD1);

/\*\*\*\*\*\*采集数据保存，额外存放 4 组\*\*\*\*\*\*/

for(i=0;i<=5;i++) for(j=4;j>0;j--)

AD\_Get[0][i][j]=AD\_Get[0][i][j-1];

/\*\*\*\*\*\*中值滤波，冒泡法排序\*\*\*\*\*\*/

for(i=0;i<=5;i++) for(j=0;j<4;j++) for(k=0;k<4-j;k++) if(AD\_Get[i][k]>AD\_Get[i][k+1])

{ temp=AD\_Get[0][i][k];

AD\_Get[0][i][k]=AD\_Get[0][i][k+1];

AD\_Get[0][i][k+1]=temp;

} if(AD\_Get[i][k]>AD\_Get[i][k+1])

{ temp=AD\_Get[0][i][k];

AD\_Get[0][i][k]=AD\_Get[0][i][k+1];

AD\_Get[0][i][k+1]=temp;

}

4.1.3传感器信息处理

电磁信号本身会随着偏离赛道而改变，所以采集到的每个电感信号的值反映出了每个电感偏离赛道的距离。这里的距离包括水平距离和垂直距离，所以需要结合小车行进过程电感的高度和采集的值才能计算出真正偏离赛道的距离。

4.2速度控制

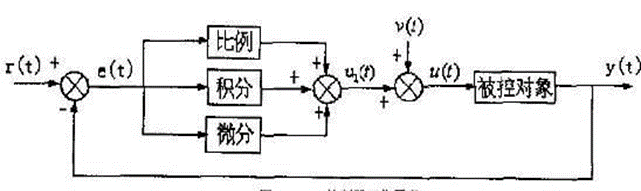


图4.1 控制流程图

PID 控制策略其结构简单，稳定性好，可靠性高，并且易于实现。其缺点在于控制器的参数整定相当繁琐，需要很强的工程经验。相对于其他的控制方式，在成熟性和可操作性上都有着很大的优势。所以最后我们选择了 PID 的控制方式。

在本方案中，使用试凑法来确定控制器的比例、积分和微分参数。

试凑法是通过闭环试验，观察系统响应曲线，根据各控制参数对系统响应的大致影响，反复试凑参数，以达到满意的响应，最后确定 PID 控制参数。试凑不是盲目的，而是在控制理论指导下进行的。在控制理论中已获得如下定性知识：

比例调节（P）作用：是按比例反应系统的偏差，系统一旦出现了偏差，比例调节立即产生调节作用用以减少偏差。比例作用大，可以加快调节，减少误差，但是过大的比例，使系统的稳定性下降，甚至造成系统的不稳定。

积分调节（I）作用：是使系统消除稳态误差，提高无差度。因为有误差，积分调节就进行，直至无差，积分调节停止，积分调节输出一常值。积分作用的强弱取决与积分时间常数 Ti，Ti 越小，积分作用就越强。反之 Ti 大则积分作用弱，加入积分调节可使系统稳定性下降，动态响应变慢。积分作用常与另两种调节规律结合，组成 PI 调节器或 PID 调节器。

微分调节（D）作用：微分作用反映系统偏差信号的变化率，具有预见性，能预见偏差变化的趋势，因此能产生超前的控制作用，在偏差还没有形成之前，已被微分调节作用消除。因此，可以改善系统的动态性能。在微分时间选择合适情况下，可以减少超调，减少调节时间。微分作用对噪声干扰有放大作用，因此过强的加微分调节，对系统抗干扰不利。此外，微分反应的是变化率，而当输入没有变化时，微分作用输出为零。微分作用不能单独使用，需要与另外两种调节规律相结合，组成 PD 或 PID 控制器

第五章 调试与优化

5.1开发软件

程序的开发是在组委会提供的 IAR 下进行的，包括源程序的编写、编译和链接，并最终生成可执行文件。IAR Systems 是全球领先的嵌入式系统开发工具和服务的供应商。公司成立于 1983 年，提供的产品和服务涉及到嵌入式系统的设计、开发和测试的每一个阶段，包括：带有 C/C++编译器和调试器的集成开发环境(IDE)、实时操作系统和中间件、开发套件、硬件仿真器以及状态机建模工具。

5.2MATLAB调试

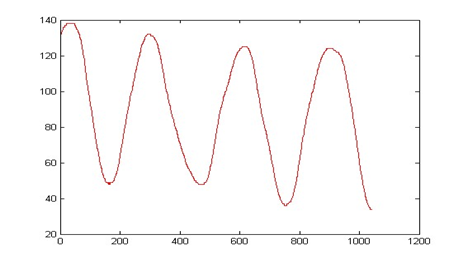


图5.1 MATLAB 仿真图

MATLAB 是矩阵实验室（Matrix Laboratory）的简称，是美国 MathWorks 公司出品的商业数学软件，用于算法开发、数据可视化、数据分析以及数值计算的高级技术计算语言和交互式环境，主要包括 MATLAB 和 Simulink 两大部分。在本项目中，我们主要使用 MATLAB 完成数据的可视化处理以及算法仿真。我们通过蓝牙串口将需要观测的数据发至电脑，使用 MATLAB 进行绘图处理。根据采集到的数据进行相关参数修改。

心得体会

智能车竞赛是一项涵盖多个学科多个领域的实践性竞赛，涉及到计算机、控制、传感技术、电子信息、模式识别、机械等。自从新规则发布以来的几个月准备过程中，我们不仅将在书上学到的理论知识用到实践上，也在做车的实践中丰富了理论实践知识。在做车的过程中也更加明白一个团队的重要性，增强了我们的实践动手能力，开拓了我们的视野。

本文主要介绍的是电磁三轮车的机械结构，硬件电路以及软件控制。在准备的过程中，为了能够让小车从零起步，从慢速到提速，我们小组都在不停地查阅资料，学习其他学校的硬件结构，控制思想。不停地在优化机械结构，为车子减负，针对电路pcb板可能存在的不稳定性因素都在一步一个脚印地优化，软件上也在尝试不同的控制算法。从能跑到提速到提速了不稳定后再优化，再到后来车速到达瓶颈，那就拆了小车，重新再搭，一次次地尝试中，也终于让我们有机会能够在分区赛中突围，有机会来到国赛的现场来目睹其他学校的风采。

在三轮车的这半年硬件与软件的积累下，我们真正地感受到了什么叫做“工匠”精神，有时候遇到问题，甚至不知道是软件还是硬件问题的时候，就是觉得车子跑的不正常，然后就得一步步地排查问题，很多时候也会遇到困惑，心态也容易崩溃，但还是得咬牙前行。现在当我快要完成这篇技术报告的时候，猛然回首过往，感觉一切都是值得的。在这说短不短的两年智能车生涯中，从第一年的分区赛三等奖，再到如今有机会进入全国总决赛，两年来为这个比赛也付出了很多，更从中学到了很多，不仅仅是将书本知识贯通在实践中，更是一种心智的历练，我们组也很感谢能有这次锻炼的机会，这种锻炼也必将会让我们终身受益。

参考文献

[1] 冯智勇, 曾瀚, 张力, 等. 基于陀螺仪及加速度计信号融合的姿态角度测量[J]. 西南师范大学学报, 2011, 36(4): 137-141.

[2] 康华光. 电子技术基础模拟部分[M]. 北京: 高等教育出版社, 2006, 第五版.

[3] 罗华飞. .MATLAB GUI 设计学习手记[M]. 北京: 北京航空航天大学出版社, 2010.

[4] 谭浩强. C 程序设计[M]. 北京: 清华大学出版社, 2003.

[5] 罗华飞. MATLAB GUI 设计学习手记[M]. 北京: 北京航空航天大学出版社, 2010.

[6] 梁慧冰, 孙炳达编著. 现代控制理论[M]. 北京: 机械工业出版社, 2007, 第二版.

[7] 屠运武, 徐俊艳, 张培仁, 等. 自平衡控制系统的建模与仿真[J]. 系统仿真学报, 2004, (4).

[8] 沈长生．常用电子元器件使用一读通[M]．北京．人民邮电出版社．2004．

[9] 张文春. 汽车理论[M]．北京．机械工业出版社．2005．

[10]徐向民．Altium Designer 快速入门（第 2 版）．北京：航空天大学出版社．2011 .

[11]张笑天，杨奋强．《MATLAB7.x 基础教程》．西安-西安电子科技大学出版社． 2008． [12] 吴建平．《传感器原理及应用》．机械工业出版社．2001．

[13] F Grasser, A D’Arrigo, S Colombi, A Rufer. Joe: A mobile, inverted. pendulum. [J]. IEEE Trans. Ind. Electron (S0278-0046), 2002, 49(1): 107-114.

附录A 程序（部分）

void precursors\_logo(void)

{

unsigned char x=20,y=0;//x,y位置变量

unsigned char ch[]={"南京师范大学"};

uint8 NextFlag=0;

while(NextFlag==0)

{

SpdL=Speed\_Get(0);//读取左旋编

SpdR=Speed\_Get(1);//读取右旋编

if(SpdL<0) SpdL=-SpdL;

if(SpdR<0) SpdR=-SpdR;

GyroX=GetData(GYRO\_XOUT\_H);//读取Roll（水平方向)陀螺仪角速度信号

GyroY=GetData(GYRO\_YOUT\_H);//读取Pitch(竖直方向)陀螺仪角速度信号

if(GyroX<32768) GyroX+=65536;//数据处理

if(GyroY<32768) GyroY+=65536;//数据处理

GyroX-=62000;//减小值，方便OLED显示

GyroY-=62000;//减小值，方便OLED显示

x=10;

y=0;

OLED\_P14x16Str(x,y,ch);

x=0;

y=7;

OLED\_P6x8Str( x, y,"SpdL");

x+=24;

OLED\_Print\_Num1( x, y,SpdL);

x+=40;

OLED\_P6x8Str( x, y,"SpdR");

x+=24;

OLED\_Print\_Num1( x, y,SpdR);

x=0;

y=5;

OLED\_P6x8Str( x, y,"GyrY");

x=24;

OLED\_Print\_Num1( x, y,GyroY);

x=50;

y=2;

OLED\_P8x16Str(x,y,"SJC");

for(int16 j=0;j<35;j++)

{

if(5==read\_key()) {NextFlag=1;break;} //按下则进入下个界面

systick\_delay\_ms(1);

}

}

}

#ifndef \_KEA128\_port\_cfg\_h

#define \_KEA128\_port\_cfg\_h

#define StartTime 220

//----------------------------------- IRQ -----------------------------------

// 模块通道 端口 可选范围 建议

#define IRQ\_PIN I5 //A5 I0 I1 I2 I3 I4 I5 I6 A5不要用（与复位按键冲突）

//----------------------------------- ADC -----------------------------------

#define AD1 ADC0\_SE5

#define AD2 ADC0\_SE15

#define AD3 ADC0\_SE4

#define AD4 ADC0\_SE13

#define AD5 ADC0\_SE6

#define AD6 ADC0\_SE12

#define AD7 ADC0\_SE7

//----------------------------------- UART -----------------------------------

// 模块通道 端口 可选范围 建议

//在切换引脚时，务必成对切换。编号相邻得两个引脚为一对

#define UART0\_RX\_PIN A2 //A2 B0

#define UART0\_TX\_PIN A3 //A3 B1

#define UART1\_RX\_PIN F2 //C6 F2

#define UART1\_TX\_PIN F3 //C7 F3

#define UART2\_RX\_PIN I0 //D6 I0

#define UART2\_TX\_PIN I1 //D7 I1

//----------------------------------- FTM -----------------------------------

// 模块通道 端口 可选范围 建议

#define FTM0\_CH0\_PIN A0 //B2 A0

#define FTM0\_CH1\_PIN A1 //B3 A1

// 模块通道 端口 可选范围 建议

#define FTM1\_CH0\_PIN H2 //H2 C4 C4不要用（与SWD\_CLK冲突）

#define FTM1\_CH1\_PIN C5 //E7 C5

// 模块通道 端口 可选范围 建议

#define FTM2\_CH0\_PIN C0 //C0 H0 F0

#define FTM2\_CH1\_PIN C1 //C1 H1 F1

#define FTM2\_CH2\_PIN C2 //C2 G4 D0

#define FTM2\_CH3\_PIN C3 //C3 G5 D1

#define FTM2\_CH4\_PIN B4 //B4 G6

#define FTM2\_CH5\_PIN B5 //B5 G7

//外部计数模块通道 端口 可选范围 建议

#define FTM0\_COUNT\_PIN E0 //E7 A5 E0 A5不要用（与复位按键冲突）

#define FTM1\_COUNT\_PIN E7 //E0 A5 E7 A5不要用（与复位按键冲突）

#define FTM2\_COUNT\_PIN A5 //E0 E7 A5 A5不要用（与复位按键冲突）

//----------------------------------- I2C ------------------------------

//在切换引脚时，务必成对切换。编号相邻得两个引脚为一对

// 模块通道 端口 可选范围 建议

#define I2C0\_SCL\_PIN B7 //A3 B7

#define I2C0\_SDA\_PIN B6 //A2 B6

#define I2C1\_SCL\_PIN H4 //E1 H4

#define I2C1\_SDA\_PIN H3 //E0 H3

//----------------------------------- SPI ------------------------------

// 模块通道 端口 可选范围 建议

//在切换引脚时，务必成对切换。编号相邻得四个引脚为一对

#define SPI0\_SCK\_PIN B2 //E0 B2

#define SPI0\_SOUT\_PIN B3 //E1 B3

#define SPI0\_SIN\_PIN B4 //E2 B4

#define SPI0\_PCS0\_PIN B5 //E3 B5

#define SPI1\_SCK\_PIN D0 //G4 D0

#define SPI1\_SOUT\_PIN D1 //G5 D1

#define SPI1\_SIN\_PIN D2 //G6 D2

#define SPI1\_PCS0\_PIN D3 //G7 D3

//----------------------------------- CAN ------------------------------

//在切换引脚时，务必成对切换。编号相邻得两个引脚为一对 H2与E7为一对

#define CAN0\_TX\_PIN C7 //E7 C7

#define CAN0\_RX\_PIN C6 //H2 C6

#endif

uint16 adc\_once(ADCn\_Ch adcn\_ch, ADC\_nbit bit)

{

if(1 == KEA128\_CLOCK)

{

//超频后，设置ADC的CLK为总线时钟/2

ADC->SC3 = (0

| ADC\_SC3\_ADIV(1) //分频系数

| ADC\_SC3\_MODE(bit) //分辨率

| ADC\_SC3\_ADICLK(0) //使用总线时钟做为ADC得时钟源

//| ADC\_SC3\_ADLSMP\_MASK //1：长采样时间 0：短采样时间 注释为0 长采样时间采集更稳定

);

}

else

{

//未超频，设置ADC的CLK为总线时钟

ADC->SC3 = (0

| ADC\_SC3\_ADIV(0) //分频系数

| ADC\_SC3\_MODE(bit) //分辨率

| ADC\_SC3\_ADICLK(0) //使用总线时钟做为ADC得时钟源

//| ADC\_SC3\_ADLSMP\_MASK //1：长采样时间 0：短采样时间 注释为0 长采样时间采集更稳定

);

}

ADC->SC1 = ADC\_SC1\_ADCH(adcn\_ch); //启动转换

while(!(ADC->SC1 & ADC\_SC1\_COCO\_MASK)); //等待转换完成

return (ADC->R & ADC\_R\_ADR\_MASK); //返回结果

}

\_\_ramfunc void ring\_control(void)

{

static int ringcount=0;

if(ringflag==0)

{

ring\_off();

ringcount=0;

}

else if(ringflag==1)

{

ring\_on();

ringcount=0;

}

else if(ringflag==2)

{

ringcount++;

if(ringcount<=25) ring\_on();

else ring\_off();

if(ringcount>=50) ringcount=0;

}

else if(ringflag==3)

{

ringcount++;

if(ringcount<=50) ring\_on();

else ring\_off();

if(ringcount>=100) ringcount=0;

}

else if(ringflag==4)

{

ringcount++;

if(ringcount<=50) ring\_on();

else ring\_off();

if(ringcount>=250) ringcount=0;

}

}

void sys\_init(void)

{

gpio\_init (E5,GPO,0);//蜂鸣器

gpio\_init (E4,GPI,1);//干簧管

//gpio\_init (I3,GPI,0);

FLASH\_Init();//FLASH初始化

///////////////按键///////////

gpio\_init (H6,GPI,0);//上

gpio\_init (D0,GPI,0);//下

gpio\_init (H7,GPI,0);//左

gpio\_init (G1,GPI,0);//右

gpio\_init (G3,GPI,0);//按下

///////////adc口/////////////

/\*adc\_init(ADC0\_SE4);//B0

adc\_init(ADC0\_SE5);//B1

adc\_init(ADC0\_SE7);//F5

adc\_init(ADC0\_SE15);//F7\*/

adc\_init(ADC0\_SE4);//B0

adc\_init(ADC0\_SE5);//B1

adc\_init(ADC0\_SE6);//B2

adc\_init(ADC0\_SE7);//B3

adc\_init(ADC0\_SE12);//F4

adc\_init(ADC0\_SE13);//F5

adc\_init(ADC0\_SE14);//F6

adc\_init(ADC0\_SE15);//F7

OLED\_Init();//显示屏

uart\_init(DEBUG\_PORT,DEBUG\_BAUD);//初始化蓝牙

ftm\_pwm\_init(ftm2,ftm\_ch0,17000,0);//PWM0 PTH0左正zuofan

ftm\_pwm\_init(ftm2,ftm\_ch1,17000,0);//PWM1 PTH1左反youfan

ftm\_pwm\_init(ftm2,ftm\_ch2,17000,0);//PWM0 PTB4右正youzhen

ftm\_pwm\_init(ftm2,ftm\_ch3,17000,0);//PWM1 PTB5右反zupozhen

gpio\_init(H5,GPI,1);//输入 旋编初始化

gpio\_init(H2,GPO,1);//输出

gpio\_init(E7,GPO,1);//输出

gpio\_init(C7,GPI,1);//输入

gpio\_init(A0,GPO,1);//输出

gpio\_init(A1,GPO,1);//输出

Speed\_Get(0);

Speed\_Get(1);

Speed\_Get(0);

Speed\_Get(1);

InitMPU6050();//陀螺仪

systick\_delay\_ms(200);

pit\_init(pit0,4);//4ms中断

set\_irq\_priority(PIT\_CH0\_IRQn,1);//设置优先级1

enable\_irq(PIT\_CH0\_IRQn);//使能中断

welcome();

void ftm\_pwm\_duty(FTMn ftmn, FTM\_CHn ch, uint32 duty)

{

uint32 cv;

//占空比 = (CnV-CNTIN)/(MOD-CNTIN+1)

switch(ftmn)

{

case ftm0:

cv = (duty \* (period[ftmn] - 0 + 1)) / FTM0\_PRECISON;

break;

case ftm1:

cv = (duty \* (period[ftmn] - 0 + 1)) / FTM1\_PRECISON;

break;

case ftm2:

cv = (duty \* (period[ftmn] - 0 + 1)) / FTM2\_PRECISON;

break;

default:

break;

}

FTMX[ftmn]->CONTROLS[ch].CnV = cv; //设置占空比

}

void ftm\_pwm\_init(FTMn ftmn, FTM\_CHn ch, uint32 freq, uint32 duty)

{

uint32 clk\_hz ;

uint16 mod;

uint8 ps;

uint16 cv;

ftm\_pwm\_mux(ftmn,ch);

// 以 CPWMS = 0 ，边沿对齐

clk\_hz = ftm\_clk\_khz\*1000;

//clk\_hz/(ps\*mod) = freq =>> clk\_hz/freq = (ps\*mod) =>>

//clk\_hz/freq < ((1<<n)\*65536) =>> (clk\_hz/freq)/65536 < (1>>n) =>> ((clk\_hz/freq)/65536)>>n < 1

mod = (clk\_hz >> 16 ) / freq ; // 临时用 mod 缓存一下

ps = 0;

while((mod >> ps) >= 1) // 等 (mod >> ps) < 1 才退出 while 循环 ，即求 PS 的最小值

ps++;

if(ps>0x07) return ; //超出设置范围，直接不设置跳出本函数，原因一般是由于PWM频率太低，或者总线频率太高导致

mod = (clk\_hz >> ps) / freq; // 求 MOD 的值

period[ftmn] = mod;

switch(ftmn) // 初值 CNTIN 设为0 ，脉冲宽度：CnV - CNTIN ，即 CnV 就是 脉冲宽度了。

{

// EPWM的周期 ： MOD - CNTIN + 0x0001 == MOD - 0 + 1

// 则 CnV = (MOD - 0 + 1) \* 占空比 = (MOD - 0 + 1) \* duty/ FTM\_PRECISON

case ftm0:

cv = (duty \* (mod - 0 + 1)) / FTM0\_PRECISON;

break;

case ftm1:

cv = (duty \* (mod - 0 + 1)) / FTM1\_PRECISON;

break;

case ftm2:

cv = (duty \* (mod - 0 + 1)) / FTM2\_PRECISON;

break;

default:

break;

}

FTMX[ftmn]->MOD = mod; //设置PWM周期

FTMX[ftmn]->CONTROLS[ch].CnSC &= ~FTM\_CnSC\_ELSA\_MASK;

FTMX[ftmn]->CONTROLS[ch].CnSC = FTM\_CnSC\_MSB\_MASK | FTM\_CnSC\_ELSB\_MASK;

// set FTM clock to system clock

FTMX[ftmn]->SC = ( 0

//| FTM\_SC\_CPWMS\_MASK //0：边沿对齐 ，1： 中心对齐 （注释了表示 0）

| FTM\_SC\_PS(ps) //分频因子，分频系数 = 2^PS

| FTM\_SC\_CLKS(1) //时钟选择， 0：没选择时钟，禁用； 1：系统时钟； 2：固定时钟； 3：外部时钟

//| FTM\_SC\_TOIE\_MASK //溢出中断使能（注释了表示 禁止溢出中断）

);

if(ftm2 == ftmn)FTMX[ftmn]->CNTIN = 0; //设置计数器初值，一般没特殊用途就清零

FTMX[ftmn]->CONTROLS[ch].CnV = cv; //设置占空比

FTMX[ftmn]->CNT = 0; //计数器，写任意值都会加载CNTIN的值

}

void flashconf\_write(void)

{

int16 i=0;

FLASH\_EraseSector(FlashSector); //擦除最后一个扇区

/\*for(i=0;i<ConfigNum;i++)

{

FLASH\_WriteSector(FlashSector,(const uint8 \*)(conf[0][i]),2,2\*i);

}\*/

FLASH\_WriteSector(FlashSector,(const uint8 \*)conf[0],ConfigNum\*2,0);

}

//从D\_flash读取参数

void flashconf\_read(void)

{

int16 i=0;

uint32 temp=0;

for(i=0;i<ConfigNum;i++)

{

temp=flash\_read(FlashSector,2\*i,uint16);

conf[0][i]=temp;

}

}

void adc\_init(ADCn\_Ch adcn\_ch)

{

SIM->SCGC |= SIM\_SCGC\_ADC\_MASK; //开启ADC时钟

ADC->APCTL1 |= 1<<adcn\_ch; //使能ADC引脚

ADC->SC2 = ADC\_SC2\_REFSEL(1); //基准电压选择

}

附录B 主要技术参数汇总

表 1 车模尺寸参数

|  |  |
| --- | --- |
| 车模总重（含电池） | 1560 g |
| 长度 | 40 cm |
| 宽度 | 25 cm |
| 高度 | 13 cm |

表 2 车模技术参数

|  |  |
| --- | --- |
| 电路功耗 | 25.2 W |
| 电容总量 | 1500 uF |
| 传感器种类 | 电磁传感器 |
| 传感器个数 | 7 个 |
| 伺服电机个数 | 0 |
| 赛道检测频率 | 1 ms |