第十一届“恩智浦”杯全国大学生

智能汽车竞赛

技 术 报 告

学　　校：武汉大学

队伍名称：Super B

参赛队员：张博深

邵浩东

赵家玉

带队教师：专祥涛

张　荣

关于技术报告和研究论文使用授权的说明

本人完全了解第十一届“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：邵浩东

带队教师签名：

日　　期：2016.8.13

摘 要

本文设计的智能车系统以飞思卡尔公司生产的MK60N512VMD100微控制器为核心控制单元，采用 Sony CCD摄像头采集赛道信息，使用模拟比较器对图像进行硬件二值化，提取得到赛道两边的黑线信息，用于赛道识别和控制；利用编码器反馈模型车的实际速度，利用采样电阻和AD8210进行电机电流的采样放大，使用双闭环PID控制算法调节驱动电机的转速；根据前方黑线的信息，利用偏差计算，插值，曲线拟合，坡道，障碍识别等方法对图像进行处理，根据图像处理得到的黑线偏差，用PID 算法控制转向，实现了对模型车运动速度和运动方向的闭环控制。为了提高模型车的速度和稳定性，使用上位机、无线模块、液晶模块等调试工具，进行了大量硬件与软件测试。实验结果表明，该系统设计方案确实可行。

关键字：MK60N512VMD100，SONY CCD，双闭环控制，PID

目录

第一章 系统方案设计

本章内容主要介绍了制作智能汽车之前对智能车系统总体方案的确定以及在后来的制作过程中对系统方案的修正结果。

1．1 系统方案的选择

这本届智能汽车竞赛组别共有六个，我们选择传统的摄像头组。在摄像头选取方面，一般说来，单片机采集图像传感器的数据有两种方法，模拟式和数字式，数字式的图像采集则是利用CMOS图像传感器的数字输出特性，直接读取传感器的数字输出。该方式性能稳定，不需要A/D，图像采集工作在单片机就变成了按照一定顺序将外部数据并行读入，因此程序简单，采集速度快。但是数字式摄像头只有CMOS型，没有找到CCD型的，而且使用起来外围电路和软件部分的寄存器设置较为繁琐，故我们选用模拟式的具有较高成像质量的CCD作为图像采集模块。

1．2 系统总体方案设计

按照本届竞赛规则规定，采用飞思卡尔的32位微控制器MK60DN256ZVLL10单片机作为核心控制单元用于智能汽车系统的控制。SONY 面阵CCD采集赛道明暗信息，返回到单片机作为转向控制的依据。主控输出PWM波控制电机的转速和舵机的打角。为了控制的准确性和快速性，我们的电机控制采用双闭环控制，外环采用编码器做速度传感器，作为外环输入，内环采用采样电阻及AD8210进行电流采集，作为内环输入 ，使用PID控制电机的转速，实际测试证明双闭环控制比单环控制的跟随性好，速度控制性能大大提升 。

根据以上系统方案设计，赛车共包括六大模块：MK60DN256ZVLL10主控模块、传感器模块、电源模块、电机驱动模块、速度及电流检测模块和辅助调试模块。各模块的作用如下：

MK60DN256ZVLL10主控模块，作为整个智能汽车的“大脑”，将采集CCD传感器、电流采集模块和光电编码器等传感器的信号，根据控制算法做出控制决策，驱动直流电机和舵机完成对智能汽车的控制；

传感器模块，是智能汽车的“眼睛”，可以通过一定的前瞻性，提前感知前方的赛道信息，为智能汽车的“大脑”做出决策提供必要的依据和充足的反应时间；

电源模块，为整个系统提供合适而又稳定可靠的电源；

电机驱动模块，驱动直流电机和伺服电机完成智能汽车的加减速控制和转向控制；

速度及电流检测模块，检测反馈智能汽车轮的转速，流经电机的电流，用于

速度的双闭环控制；

辅助调试模块，主要用于智能汽车系统的功能调试，参数设置，赛车状态监控。

第二章 智能汽车机械结构的调整与优化

智能汽车所采用的车模是仿造现代汽车的模型，因此现代汽车的许多理论同样使用于智能汽车。所有的硬件、程序算法最终都是建立在机械平台的基础上实现竞速的。因此，机械结构的调节是智能汽车制作中一个不容忽视的重要环节。

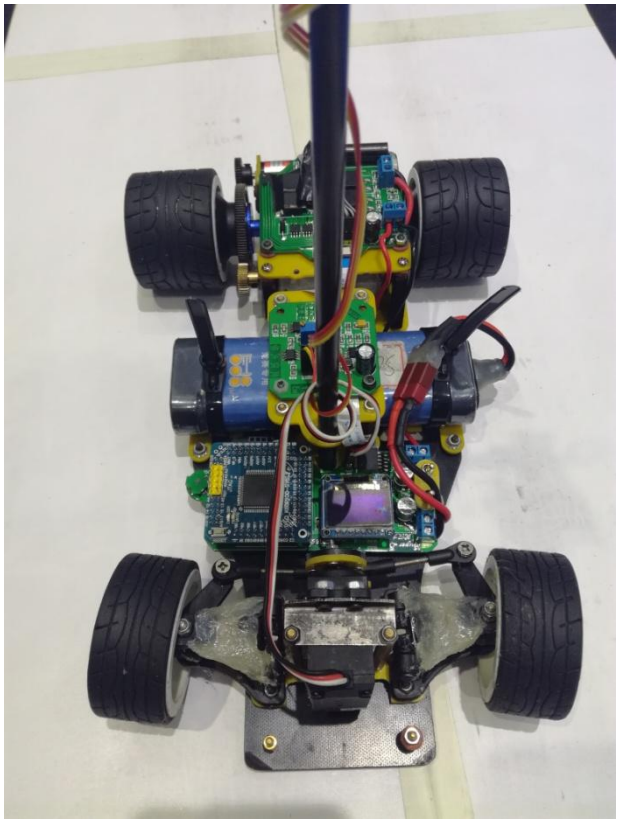


图 2-1 B 车模型图

2．1 智能汽车前轮定位的调整

前面说过，智能汽车是现代汽车的缩影，因而现代汽车的许多理论同样适用于智能汽车。比如前轮定位的调整，差速器的调整，转向系的调整等。

对于前轮转向的整体式车架的车辆，通常将转向节主销后倾角、主销内倾角、转向轮外倾角和转向轮前束统称为转向轮定位。转向轮定位的作用是减小偏转车轮的操纵力矩，保证车辆直线行驶，即有自动回正作用。

2．1．1 主销后倾角 γ

主销的上部在车辆纵向平面内向后倾斜，主销轴线和垂线间的夹角γ 叫主销后倾角。如下图所示：

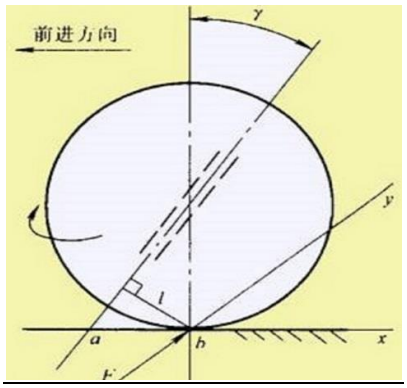


图 2-2 主销后倾角

主销后倾将使主销轴线与路面的交点 a 位于车轮与路面接触点 b 的前方。车辆直线行驶时，若转向轮偶受外力作用而偏转（如图 2-2 中箭头所示向右偏转）时，由于车辆本身离心力的作用，在车轮与地面接触点 b 处，路面对车轮作用着一个侧向反力 F。F 力对主销轴线便产生一个迫使车轮回正的力矩 F1，从而保证车辆能稳定地沿直线行驶。力矩 F1 称为稳定力矩。稳定力矩不宜过大，否则将使转向操纵费力。F1 的大小取决于 1 值，故γ 角不宜过大，一般不超过 2°～3°。轮胎的弹性变形对稳定力矩亦有影响。采用低压轮胎时，因其弹性较大，在转向时，由于轮胎的侧向弹性变形而使 F 的作用点 b 后移，从而增大稳定力矩，此时，γ 角可以减小到接近于零，甚至为负值，从而减小转向时所需操纵力。

2．1．2 主销内倾角 β

主销的上部在横向平面内向内倾斜一个角度 β 叫主销内倾角。主销内倾可使转向轮有自动回正的作用。如下图 2-3 可知，当车轮没有偏转时，作用在 A 点的地面垂直反力 Z 和主销轴线位于同一平面内，故 Z 对主销轴线的力矩为零。当车辆直线行驶，因受外力作用而稍有偏转时，则车轮与路面的接触点由 A 运动到 A′点，这时地面反力 Z 显然就不和主销轴线在同一个平面内。在车轮平面内，Z 力可分为 1 和 Z2 二个分力，Z2 和主销轴线垂直但不相交， 1 则和主销轴线处在同一平面内，故 1 对主销轴线仅产生弯矩而不产生稳定力矩。Z2 则对主销轴线产生稳定力矩，使车轮自动返回直线行驶位置。

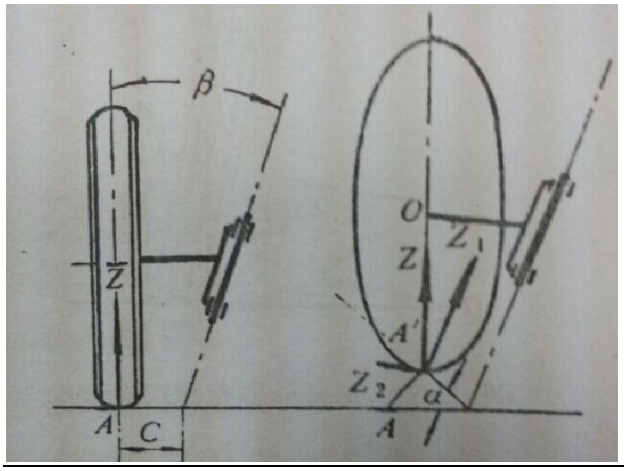


图 2-3 主销内倾角

此外，主销内倾还使主销轴线与路面交点到车轮中心平面的距离 C 减小，即减小了转向时车轮的旋转半径，从而减小转向时所需的操纵力，使转向轻便。同时也减小了从转向轮带来的冲击力。但 C 也不宜过小，即 β 角不宜过大。β 角一般不大于 8°，C 一般为 40~60mm。

2．1．3 转向轮前束 δ

转向轮外倾后，在滚动时就类似滚锥而使车轮向外拉开，这又增加了外端小轴承的压力。同时，由于转向横拉杆和车桥的约束，使车轮不能向外滚开，因此转向轮就在地面上边滚边滑，增加了轮胎的磨损。为了消除车轮外倾的不良后果，在安装时，可使两个转向轮前面轮缘之间的距离 B 小于后面轮缘间的距离 A。A-B=δ 称为车轮前束（图 2-4）。这样可使转向轮滚动时，在每一个瞬间其滚动方向均接近于正前方。车轮前束可通过改变转向横拉杆长度进行调整。δ 值一般小于 8~12mm。

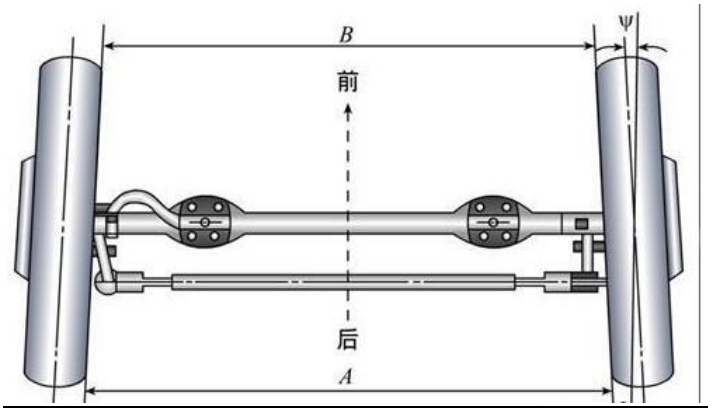


图 2-4 转向轮前束

以上说法均为理论基础，在实际操作中发现，由于 C 车模零件的制造误差导致主销内倾调节螺丝、前轮前束调节连杆等诸多地方存在左右不对称的现象。因此，在调节前轮定位的过程中，要有耐心，仔细调节，一旦发现车跑得不对称的时候要冷静分析原因，重新调整，使车模转向轮在转向时既轻便又灵敏，且虚位现象小。

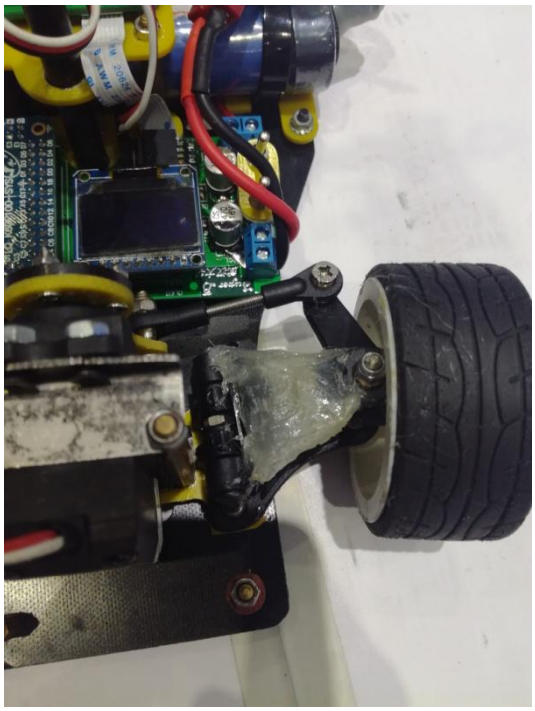


图 2-5 B 车模的前轮定位

2．2 舵机的安装方式

舵机的安装方式不外乎两种：一是卧式安装，二是立式安装。

对比这两种安装方式发现，卧式安装法在安装过程中存在几个比较大的弊端：

一是，为了让连杆水平必须把舵机架高腾空安装，这使得整体的重心变高了很多；二是，由于舵机本身自带的安装孔不适宜于卧式安装，使得必须再加工较多的零件来固定舵机，同时卧式安装架高的过程用的铜柱的数量大大多于立式安装，导致整体的重量也变重了。

基于以上各种考虑，我们放弃卧式安装的方法，最终采用了立式安装法。

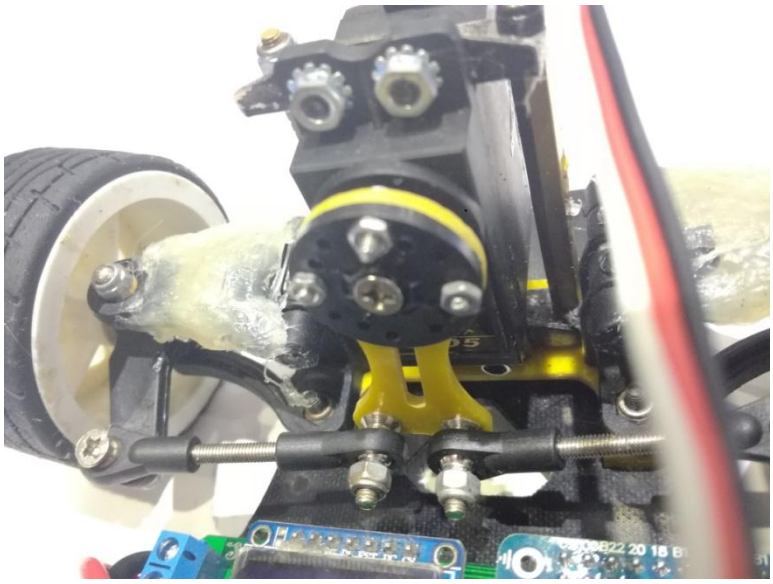


图 2-6 B 车模的舵机立式安装

2．3 前轮转向节的调整

智能汽车转向系对车辆的行驶影响很大，直接影响到过弯性能。转向系的基本要求很多，以下只列一条进行分析：

理论上要求：转向时各车轮必须作纯滚动而无侧向滑移。

为了满足上述要求，对前轮转向节的要求分析过程如下：

在过弯时，智能汽车的各个车轮如果既滚又滑，则会增加转向阻力及加重轮胎磨损的现象。由图 2-7 可知：只有当所有车轮的轴线都交于一点 0 时，各车轮才能作纯滚动，此瞬时中心 0 就称为转向中心。

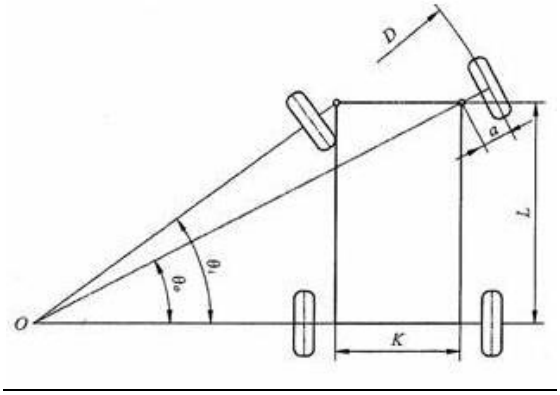


图 2-7 车轮转向简图

图2-7中，L为前后轴轴距，K为两侧主销中心距，θ_0、θ_1分别为内外转向轮的偏转角。显然θ_0、θ_1是不等的，根据理论力学相关知识分析，经过计算得到

$$\tan^{-1}\theta_0-\tan^{-1}\theta_1=\kappa L \quad \text{公式一}$$

2. 4 重心整体调整

重心对后期提速有很大影响。如若重心太高，容易导致过弯倾覆，大大限制了速度提升。为此，在底盘不刮到坡道的前提下，应该尽可能把重心调低。大概从以下几方面调整：

2．4．1 降低底盘

一方面，通过机械建模安排布局后，得到主板的大小。画 PCB 的时候保证 PCB 板大小能够贴紧底盘。另外一方面，电池、液晶、驱动板、蜂鸣器、摄像头等尽可能贴近底盘。

2．4.2低底盘硬链接

在不刮到坡道的前提下，尽可能降低底盘。降低底盘的方法不外乎调整前桥的垫片数目和升高后轮轴轴线两种。于是我们增加前桥垫片，抬升后轮车轴，尽量降低车模底盘，以达到降低重心的目的。转弯由于惯性力的存在，车子会产生外倾趋势，采用硬链接有助于保持重心稳定，有助于车子过弯，因此我们的车模采用硬链接。

2．5 其他机械方面调整

除了上述的调整之外，在许多细节也应该做好。例如，能用短螺丝就不用长螺丝，螺丝要拧紧，左右要对称，摄像头前瞻要固死等等。总之，不能忽视任何细节。 关于底盘固定等需要精确固定的地方，用雕刻机加工必要的连接件，保证结构尺寸精度。

2．6 小结

机械结构性能的好坏是关乎整个智能汽车速度最高极限的决定性因素之一。在实际操作中发现，往往程序没有改动，而对机械结构进行改良之后速度又上了一个台阶，这就证明了机械结构不可忽视的重要性。

第三章 智能汽车硬件电路设计

硬件电路是程序的载体，硬件电路的好坏也是智能汽车性能好坏的关键因素。下面从主板电路，SONY CCD 供电及二值化电路，接口电路，驱动板电路四方面对整车电路进行分析。

3．1 主板电路设计

3．1．1 最小系统版接口

经过对 K60 主控芯片技术手册的查阅学习，下图是我们队 K60 主控芯片的 I/O 口资源分布图：

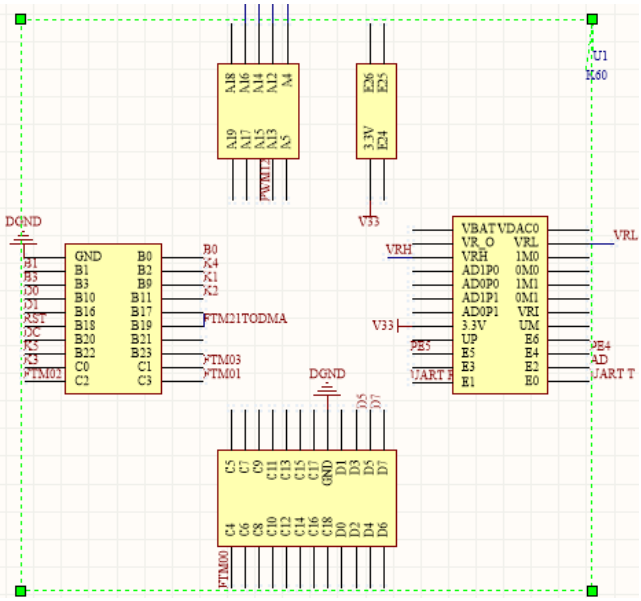


图 3-1 I/O 口资源分配

3．1．2 3.3V电源模块

由于最小系统版供电需要 3.3V 电源，因此必须采用 3.3V 稳压芯片来提供电源。经过查资料，能到达该功能的芯片有 TPS7333、HT1003、HT1004、1117 系列等许多芯片。 考虑到 TPS7333 的压降小，大概 0.1V，最终选择 TPS7333 作为最终采用的电源芯片。如图 3-2 为其主要参数：

AVAILABLE OPTIONS

T_J	OUTPUT VOLTAGE (V)			NEGATIVE-GOING RESET THRESHOLD VOLTAGE (V)			PACKAGED DEVICES			CHIP FORM (Y)
	MIN	TYP	MAX	MIN	TYP	MAX	SMALL OUTLINE (D)	PLASTIC DIP (P)	TSSOP (PW)	
−40°C to 125°C	4.9	5	5.1	4.55	4.65	4.75	TPS7350QD	TPS7350QP	TPS7350QPW	TPS7350Y
	4.75	4.85	4.95	4.5	4.6	4.7	TPS7348QD	TPS7348QP	TPS7348QPW	TPS7348Y
	3.23	3.3	3.37	2.868	2.934	3	TPS7333QD	TPS7333QP	TPS7333QPW	TPS7333Y
	2.94	3	3.06	2.58	2.64	2.7	TPS7330QD	TPS7330QP	TPS7330QPW	TPS7330Y
	2.425	2.5	2.575	2.23	2.32	2.39	TPS7325QD	TPS7325QP	TPS7325QPW	TPS7325Y
	Adjustable 1.2 V to 9.75 V			1.101	1.123	1.145	TPS7301QD	TPS7301QP	TPS7301QPW	TPS7301Y

The D and PW packages are available taped and reeled. Add an R suffix to device type (e.g., TPS7350QDR). The TPS7301Q is programmable using an external resistor divider (see application information). The chip form is tested at 25°C.

根据芯片资料的介绍和经典电路，最终采用的原理图如下：

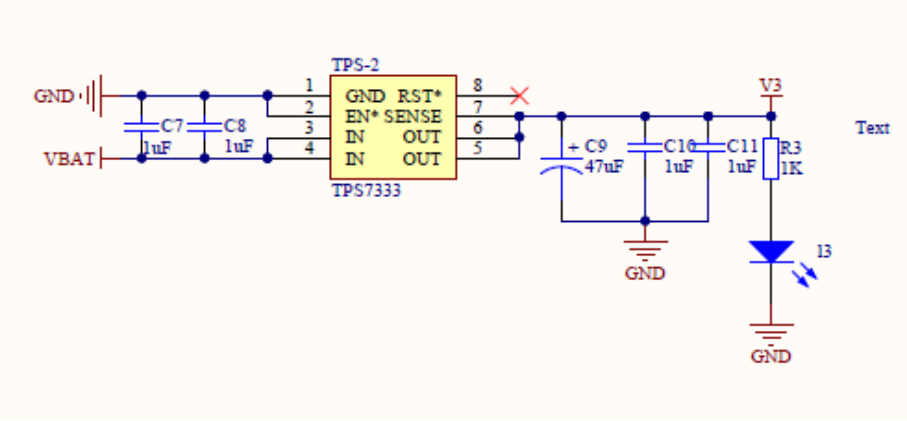


图 3-2 3.3v 电源图

5V 电源采用 TPS7350，原理图如下：

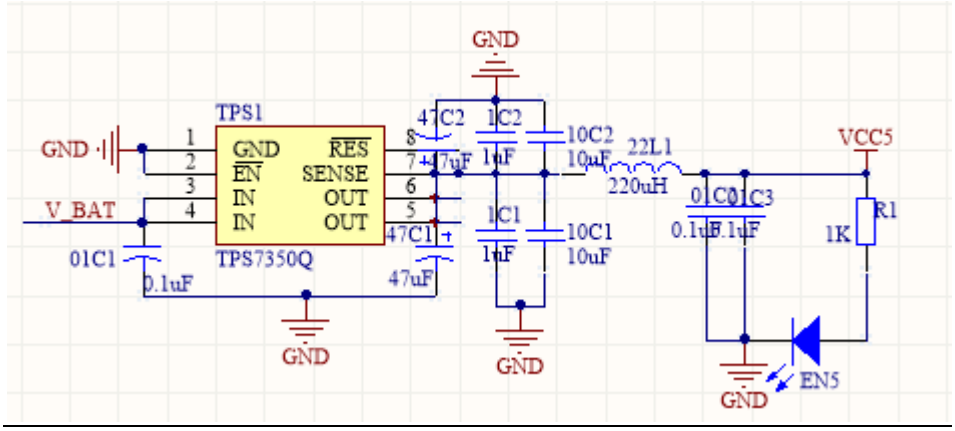


图 3-3 5V 电源图

3．1．3 舵机电源

B 车模规定采用 SD5 数字舵机，经查 SD5 舵机供电 5~6V，为了提供稳定充足的电源，我们采用 LM2941S 搭成的 6V 电源为舵机供电，其电路图如下：

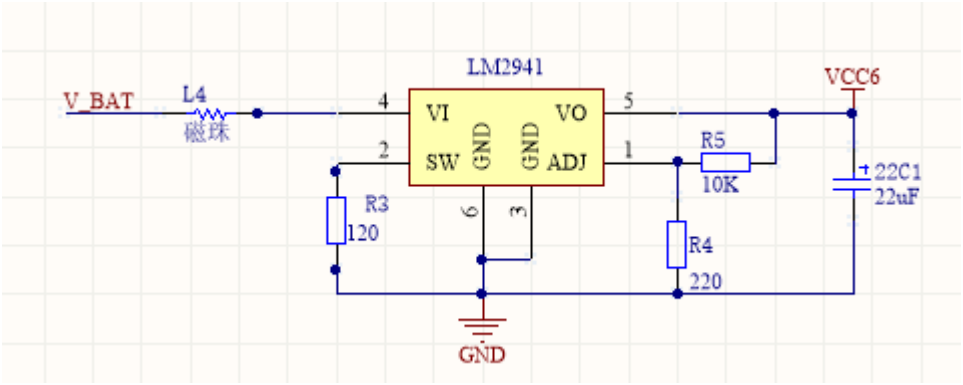


图 3-4 舵机供电

3．1．4 开关

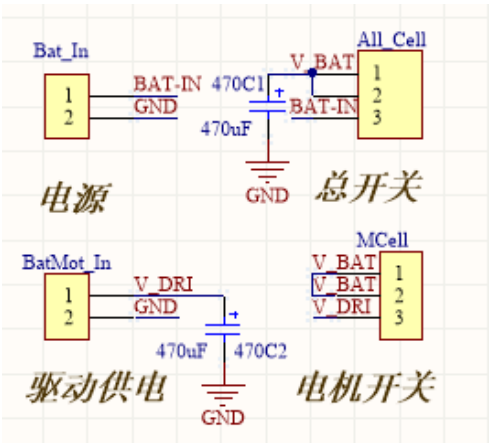


图 3-5 开关电路

3．2 SONY CCD 供电及二值化电路

由于采用 SONY CCD 模拟摄像头，需要用 12V 供电，我们采用 MC34063 及其外围电路实现。

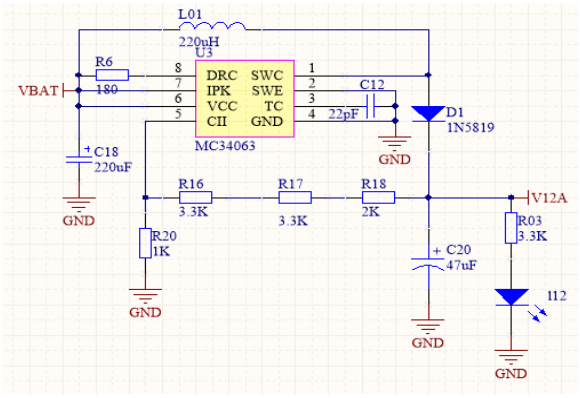


图 3-6 12V 电源图

硬件二值化电路直接处理摄像头信号，减轻 CPU 负担。电路图如下：

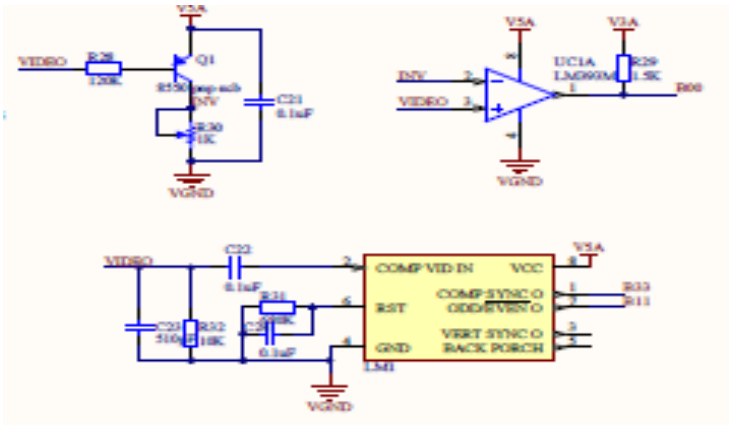


图 3-7 硬件二值化电路

3．3 各种重要接口设计

3.3.1 驱动板接口

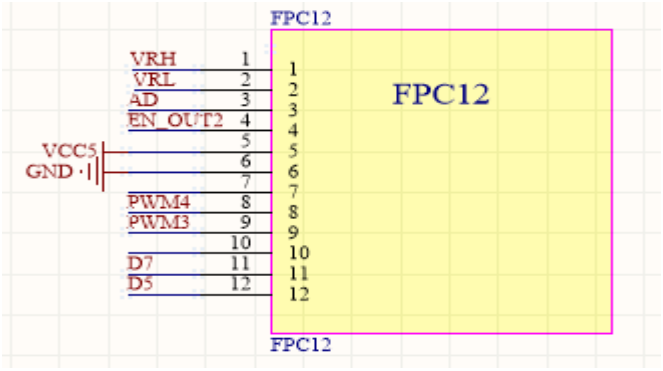


图 3-8 驱动板 FPC 接口

3.3.2 舵机接口

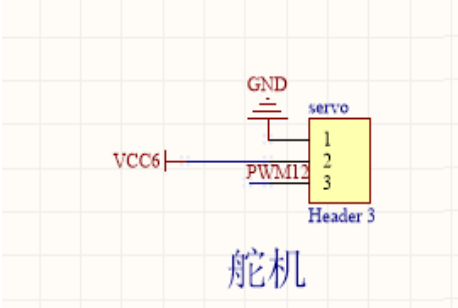


图 3-9 舵机接口

3. 3. 3 液晶接口

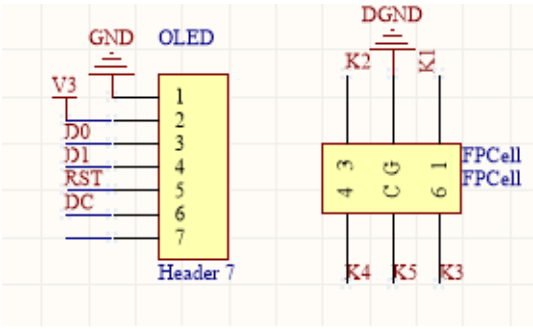


图 3-10 液晶接口

3. 3. 4 蓝牙接口

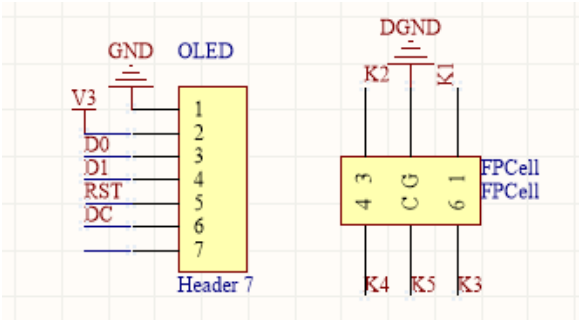
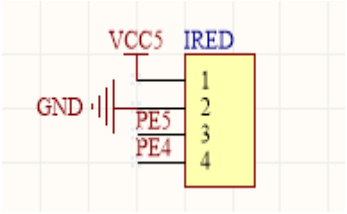
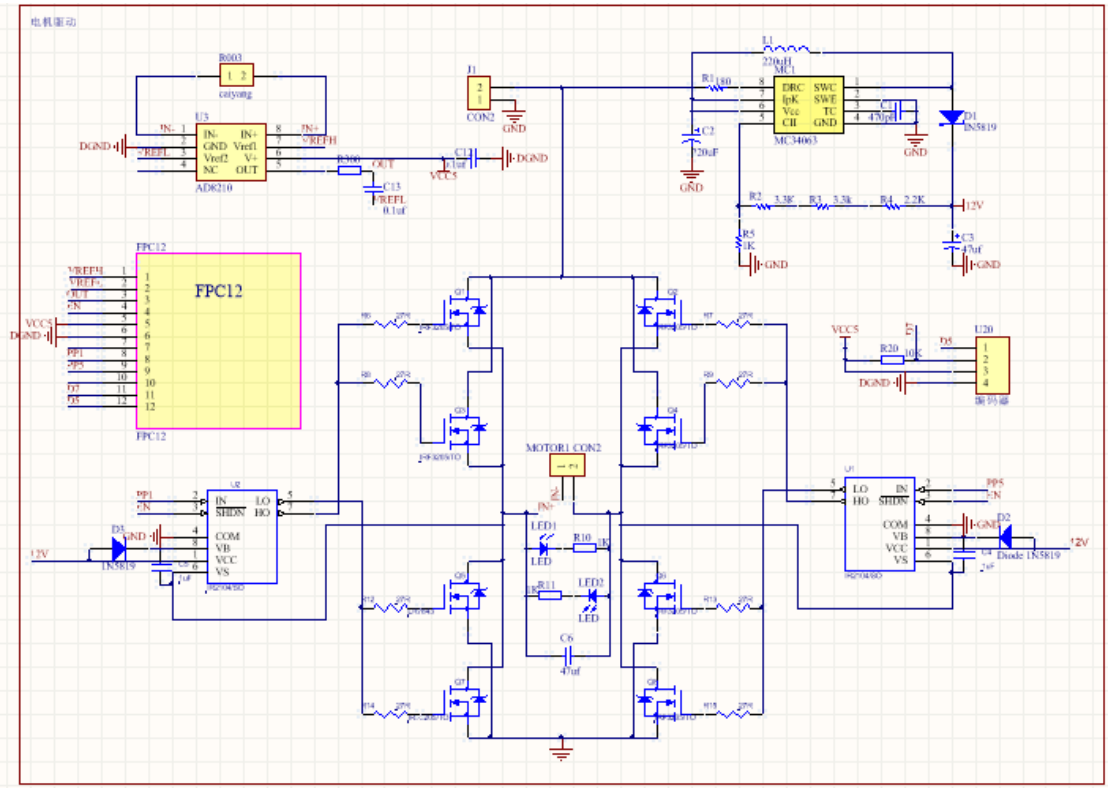


图 3-11 液晶接口

3． 3. 5 红外接收模块接口



3．4 驱动板



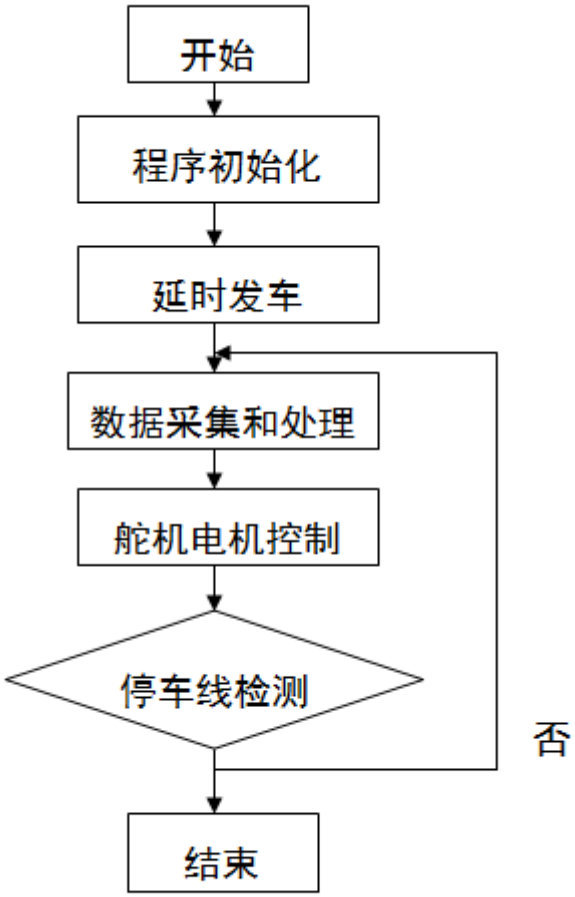
3.5 小结

硬件电路部分需要做到认真仔细无纰漏，从原理图的绘制到 PCB 的布线，都要尽善尽美，做好细节问题。另外电路板的形状大小要和车模机械结构相适配，保证电路稳定的同时也要保证良好的机械结构。

第四章 智能汽车软件设计

智能汽车的软件设计和算法优劣是直接一个决定智能汽车速度性能的关素。因此好车更要有好的算法。本章从以下几方面阐述我们智能车的软件设计方案。

4．1 程序流程图



4．2 程序初始化

4．2．1 舵机初始化

B 型车模的舵机使用频率为 100 赫兹。其初始化函数如下：

```
void servo_init(void)
{
    //舵机
    FTM_InitTypeDef Sever_ftm_init_struct = { 0 };
    Sever_ftm_init_struct.FTM_Ftmx = FTM1;    //使能FTM1通道
    Sever_ftm_init_struct.FTM_Mode = FTM_MODE_PWM;    //使能PWM模式
    Sever_ftm_init_struct.FTM_PwmFreq = 100;//PWM频率100Hz
```

```
    LPLD_FTM_Init(Sever_ftm_init_struct);

    LPLD_FTM_PWM_Enable(FTM1, //使用FTM0
        FTM_Ch1, //使能Ch1通道
        //angle_to_period(0), //初始化角度0度
        midservo,
        PTA13, //使用Ch0通道的PTA13引脚
        ALIGN_LEFT          //脉宽左对齐
        );
}
```

4．2．2 电机初始化

电机频率为 10K 赫兹，使用电平判断轮子转向的编码器，采用 DMA 计数。其初始化函数如下：

```
void motor_pwm_init(void)
{
    Motor1_ftm_init_struct.FTM_Ftmx = FTM0;
    Motor1_ftm_init_struct.FTM_Mode = FTM_MODE_PWM;
    Motor1_ftm_init_struct.FTM_PwmFreq = 10000;
    LPLD_FTM_Init(Motor1_ftm_init_struct);

    LPLD_FTM_PWM_Enable(FTM0,
        FTM_Ch0,
        0,
        PTC1,
        ALIGN_LEFT);

    LPLD_FTM_PWM_Enable(FTM0,
        FTM_Ch1,
        0,
        PTC2,
        ALIGN_LEFT);

}
```

4．2．3 PIT初始化

40us 中断一次。PIT 初始化函数如下：

```
PIT_InitTypeDef pit0_init_struct;
void pit_init(void)
{

  pit0_init_struct.PIT_Pitx = PIT0;
  pit0_init_struct.PIT_PeriodUs = 40;
  pit0_init_struct.PIT_Isr = pit0_isr;
```

```
  LPLD_PIT_Init(pit0_init_struct);
  LPLD_PIT_EnableIrq(pit0_init_struct);

}
```

4．3 SONY CCD数据采集

摄像头数据量大，采用 DMA 方式采集，程序初始化如下：

```
void Video_init()
{
    Video_GPIO_Init();
    dma_video_init();
    fclk.FTM_Ftmx = FTM2;
    fclk.FTM_Mode = FTM_MODE_PWM;
    fclk.FTM_PwmFreq = 3033333;
    LPLD_FTM_Init(fclk);
    LPLD_FTM_PWM_Enable(FTM2,
        FTM_Ch1,
        5000,
        PTB19,
        ALIGN_LEFT);
    GPIO_InitTypeDef gpio_dma_init_struct = { 0 };
    gpio_dma_init_struct.GPIO_PTx = PTB;
    gpio_dma_init_struct.GPIO_Pins = GPIO_Pin17;
    gpio_dma_init_struct.GPIO_Dir = DIR_INPUT;
    gpio_dma_init_struct.GPIO_PinControl = INPUT_PULL_DIS | IRQC_DMARI;
    //DMA触发源
    LPLD_GPIO_Init(gpio_dma_init_struct);
}
```

4．4赛道类型识别

4．4．1 十字路识别

十字识别是通过判断边沿跳变和两边全白的条件是否满足。好的十字识别可以避免章鱼弯闯道。

4．4．2障碍识别

通过边沿跳变和黑块大小条件是否满足，进行障碍识别。识别之后根据障碍的位置设置适当的延时和打角。

4．4．3坡道的识别

通过赛道形状的改变，和黑块的大小进行判断识别。识别之后设置适当的过坡速度，避免飞坡带来的不确定因素。

4. 4. 4起跑线识别

通过起跑线长度和宽度的特征进行识别，识别之后进行起跑停车操作。

4．5 舵机控制

4．5．1 舵机控制理论

舵机是一个位置随动系统，它由舵盘、减速齿轮组、位置反馈电位计、直流电机和控制电路组成。通过内部位置反馈，使它的舵盘输出转角正比于给定的控制信号，因此对于它的控制可以使用开环控制方式。在负载小于其最大输出力矩的情况下，它的输出转角正比于给定的脉冲宽度。

舵机控制是控制算法中最重要的方面之一。一旦舵机控制出错，智能汽车就极有可能出界犯规了。因此，平时调车的主要任务之一就是观察舵机，使之控制合理。本队舵机的控制采取传统的 PD 控制。关于 PID 控制的知识，由于 PID 控制是经典的控制，其理论比较成熟，且市面上相关的书籍较多，故在此不对 PID 控制展开深入阐释。

其关键部分代码如下：

```
servo_pwm_change = (wucha_Now* Kp_servo + Kd_servo*(wucha_Now - wucha_Last)) / 100;
```

其中，KP_servo、KD_servo 两个重要的参数需要经过不断调整。在实际调试中发现，针对赛道类型给定参数更符合实际。例如，在直道和小 S 路上相应的 KP_servo 可以给小，以防舵机打角过大。 分段各段之间参数的给定要连续，否则会出现舵机在参数切换之后出现较大突变，导致智能汽车的路径变差。我们设置一个数学函数控制参数的输出，减少调节参数的工作量。

当然为了防止把舵机的齿轮打坏，应该对舵机的打角进行限幅处理。

4．5．2 舵机响应速度改善

影响舵机控制性能的一个重要参数是舵机的响应速度。舵机转动一定长度有时间延迟，时间延迟正比于旋转过的角度，反比于舵机的响应速度。而舵机的响应速度正是直接影响智能汽车过弯性能和过弯最高速度的因素，因此提高舵机的响应速度是提高智能汽车平均速度的一个关键。工作电压与响应速度有一定关系，适当地提高工作电压可以提高舵机的响应速度。

根据杠杆原理，在舵机的输出舵盘上安装一个较长的输出臂，将转向传动杆连接到输出臂末端。这样可以在舵机输出较小的转角下，取得较大的前轮转角，从而提高了车模的转向控制速度。但是，安装臂并不是越长越好。加长了安装臂会使加到连杆的力减小，如果力太小的话，有可能在转弯时带不动前轮，使车模的转向性能下降，得到相反的效果。所以，应该权衡利弊。详阅第二章。

4．6 电机控制

4．6．1 控制理论

由于各处赛道摩擦力不一样，电池电压一直在变化等客观因素，因此，如果电机控制采取开环控制的话，同样的参数可能每一次的效果都是不一样的，导致

调试效果不佳。故必须采用闭环控制。第二章提及编码器，编码器的作用正是为了实现速度闭环。

本队算法中，电机速度控制采用了 PID 加 BangBang 控制的方法。

其控制的思路大概如下：

当实际速度与设定速度相差过大时，采用 BangBang 控制。如果二者相差在一定范围内，则采用 PID 进行控制。其公式如下：

$$Motor\_out=\begin{cases}100\% & speed\_cnt-speed\_set<-S\\ P\times(speed\_cnt-speed\_set) & |speed\_cnt-speed\_set|\le S\\ 0 & speed\_cnt-speed\_set>S\end{cases}$$

实际调试中发现，如果简单采用上述公式，可能会出现电机控制不平滑。故对上述公式应当适当改善，选择好合适的参数。

下面讲述电机增量式 PID 的实现。

因为控制器是通过软件实现其控制算法的，所以必须对模拟调节器进行离散化处理，这样它只能根据采样时刻的偏差值计算控制量。因此，不能对积分和微分项直接准确计算，只能用数值计算的方法逼近。用离散的差分方程来代替连续的微分方程。根据输出量 u(k)的形式可分为位置式 PID 控制算法和增量式 PID 控制算法。考虑到电机的控制量不是绝对数值，而是其增量，因此要采用 PID 增量式控制算法。

采用增量式 PID 控制算法的表达式如下：

$$\Delta(k)=u(k)-u(k-1)$$

则：

$$\Delta u(k)=K_Pe(k)+K_PTT_I\Sigma e(i)_{ki=0}+K_PT_DT[e(k)-e(k-1)]-K_Pe(k-1)-K_PTT_I\Sigma e(i)_{k-1i=0}-K_PT_DT[e(k-1)-e(k-2)]$$

令： $K_I=K_PTT_I$ ； $K_D=K_PT_DT$

则： $\Delta u(k)$

$$=u(k)-u(k-1)=K_P[e(k)-e(k-1)]+K_Ie(k)+K_D[e(k)-2e(k-1)+e(k-2)]$$

相对于位置式算法，增量式 PID 算法的优点为:位置式算法每次输出与整个过去状态有关，计算式中要用到过去偏差的累加值，容易产生较大的积累误差；而增量式只需计算增量，当存在计算误差或精度不足时，对控制量计算的影响较小。

具体实现代码如下：

```
speed_right += wucha_right_now*(Kp_Motor + Ki_Motor + Kd_Motor) -
wucha_right_last*(Kp_Motor + 2 * Kd_Motor) + wucha_right_pre*Kd_Motor;
```

4. 6. 2 PID 参数的作用与整定

为了更好地应用 PID 算法，首先必须对算法的参数有深刻的理解。下面阐述了 PID 参数的作用。

（1）、比例 P:比例项部分是对预设值和反馈值差值的放大倍数。显然比例 P 越大，电机转速回归到输入值的速度将更快，调节灵敏度就越高。从而加大其值，就可以减少从非稳态到稳态的时间，但同时也可能造成电机转速在预设值附近振荡的情形，故应引入积分 I 解决此问题。

值在时间上进行累加。当差值不是很大时，为了不引起振荡，可以先让电机按原转速继续运行；此时将这个差值用积分项累加；当这个和累加到一定值时，再一次性进行处理，从而避免了振荡现象的发生。可见，积分项的调节存在明显的滞后，而且 I 值越大，滞后效果越明显。

（3）、微分 D:微分项部分其实就是求电机转速的变化率，也就是前后两次差值的差而已。也可以说，微分项是根据差值变化的速率，提前给出一个相应的调节动作。可见微分项的调节是超前的；并且 D 值越大，超前作用越明显；可以在一定程度上缓冲振荡。比例项的作用是放大误差的幅值，而目前需要增加的是“微分项”，它能预测误差变化的趋势，这样，具有“比例+微分”的控制器，就能够提前使抑制误差的控制作用等于 0，甚至为负值，从而避免了被控量的严重超调。

通过上面的简单阐述，对 PID 算法的各个参数有了一定的了解，但是具体到智能汽车速度控制系统中，必须根据实际情况进行合理调整，不可一味抄袭前人定式。

下面对 PID 参数整定进行简单阐述。

通过上面的介绍，我们得知，比例控制使反应变快，微分控制使反应提前，积分使反应滞后。在一定范围内，比例系数、微分系数越大，调节的效果越好。调节 PID 参数的原则大致如下：

(1)、在输出不振荡时，增大比例增益 P;

(2)、在输出不振荡时，减小积分系数 I;

(3)、在输出不振荡时，减小微分系数 D.

我们先得到电机在定值输入下的响应曲线，使用 matlab 仿真得到大致的 PI 参数，再通过试凑法对 I 和 D 进行细调。

4. 7 起跑检测与停车

检测一段时间内接受到的红外中断数,如果符合要求再进行一系列的逻辑判断。由于小车要停在 1m 区之内，在识别到停车区前的红外信号时可以适当减速。

4. 8 小结

本章主要阐述了智能汽车控制算法与赛道识别等“上层建筑”的东西。实践证明，硬件与软件的结合需要不断调整才能达到完美。

第五章 开发工具、调试工具、调试过程

5. 1 开发工具

本队队员开发环境为window平台。开发工具为IAR Embedded Workbench。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. 2 调试工具

5.2.1 串口上位机

通过蓝牙模块与PC机实时交互，方面调车人员掌握智能汽车的状态和各项参数的效果。通过不同的上位机visual scope等软件，对图像处理、电机速度控制有很好的帮助。

5. 3 小结

通过上位机，方便地实时了解车的状态，有利于调车。实践证明，自主开发上位机是很有必要的。

第六章 智能汽车技术参数

智能汽车主要技术参数如下：

车长(CM)	27
车宽(CM)	18
车高(CM)	28
车重(g)	1279
摄像头数目(个)	1
舵机数目(个)	1
电路功耗（W）	30
电容总容量（uf）	1640
赛道信息检测频率（次/秒）	50

表 6-1 智能汽车技术参数表

总结

“飞思卡尔”智能汽车竞赛已经成功举办了十一 届。从第一届到第十一届，参赛队伍的规模逐年增长，参赛队伍无论是传感器应用、竞赛规则、控制算法、模式识别等均茁壮成熟起来，促使“飞思卡尔”竞赛的生命力、观赏性、趣味性、创新性得到提升。

在第十一届的比赛中，我们学校取得了很好的成绩，所以学校和学长对我们这一届都给予了很大的期望。我们不敢辜负大家的期望，自去年暑假开始学习智能汽车竞赛相关的知识。从最简单的 51 单片机开始，到飞思卡尔公司生产的 MC9S12 芯片，再到 K60 芯片，我们一步步的在打好基础。硬件方面，我队队员自学软件，在实践中摸索，不断参考各种方案，仔细检查细节，终于使得做出来的板子稳定

虽然如此，我们自身还是存在一些不足。例如，我们在系统建模与仿真方面的能力还是略显不足。但我们在不断学习中，在实践中熟练操作 MATLAB，逐渐对建模、仿真等有了进一步的了解。我们也大胆摒弃了往届前辈留下来的程序，程序框架和识别都从零开始，极大地锻炼了我们的能力。

本届竞赛相比去年，赛道元素趋于简单化，这就要求我们需要对车子的控制精益求精，提升稳定性，保证赛场上稳定发挥，我队队员在实践中不断摸索赛道识别、控制的方法，终于使智能汽车方案经受住实践考验。

有心人，天不负，百二秦关终属楚。我们相信，经过我们辛勤的努力，在全国总决赛的角逐中，胜利女神会抛给我们橄榄枝！

致谢

制作智能汽车是一个长期的工程，我们队并不是在孤军奋战。感谢一年中陪我们在实验室一起奋斗的同学。本文所涉及的，不仅仅是我们队辛勤劳作的结晶，同时还有许多人智慧的光芒。在这里也要感谢他们跟我们提的好建议、好想法、好思路。

同时也要感谢武汉大学动力与机械学院自动化系给我们提供一个自由学习的好环境，学校教务部也给我们提供了调试场地等诸多方便。还有专祥涛和杨飞老师在一年之中给我们悉心的指导和督促。在这里表示衷心的感谢。 也要感谢教育部高等学校自动化专业教学指导分委员会引进了这样富有活力的竞赛，还有清华大学自动化系老师们多年来为“飞思卡尔”智能汽车竞赛做出的贡献和协办方——飞思卡尔公司一直的支持。还有各个赛区承办学校为本届竞赛提供的场地，营造竞赛环境，组织培训裁判和志愿者等多方面的努力。

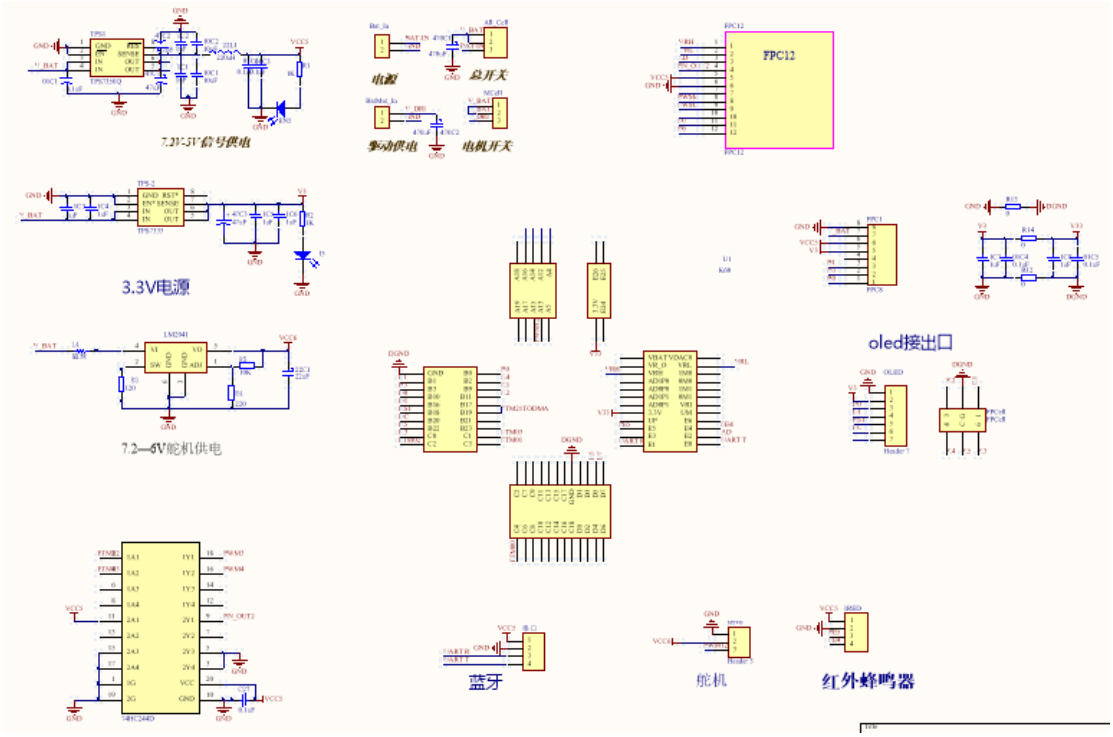
衷心祝愿这个融创新、智慧、汗水于一炉的高水平竞赛越办越好，越办越有活力。

参考文献

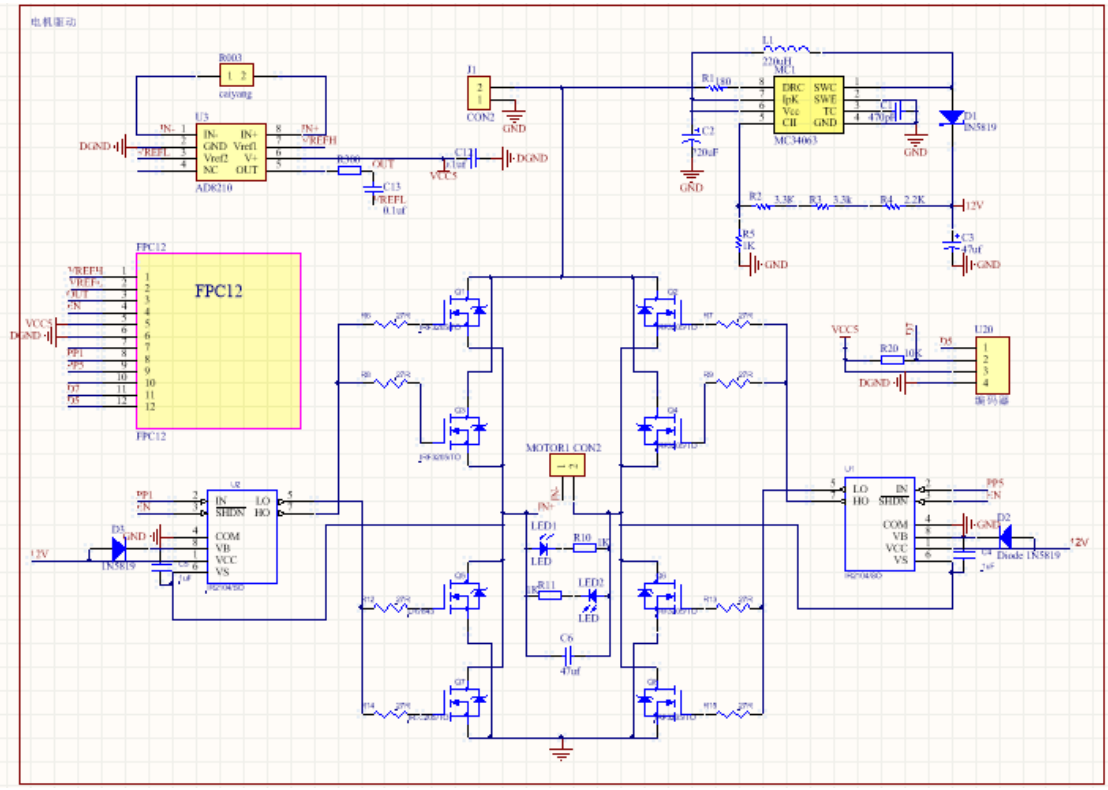
[1] 卓晴、黄开胜、邵贝贝等.学做智能车——挑战“飞思卡尔”杯.北京：北京航空航天出版社，2007 年 3 月第 1 版
[2] 蔡述庭主编.智能汽车竞赛设计与实践——基于S12XS和Kinetis K10.北京：北京航空航天大学出版社，2012年6月第1版
[3] 第九届全国大学生“飞思卡尔”杯智能汽车竞赛秘书处，竞速比赛规则与赛场纪律，2013年10月1日
[4] 王宜怀、曹金华. 嵌入式系统设计实战. 北京：北京航空航天大学出版社，2011年
[5] 余恒睦主编.工程机械.武汉：武汉大学出版社，2013年11月01日
[6] 邵贝贝.单片机嵌入式应用的在线开发方法.北京：清华大学出版社，2004年10月第1版
[7] 郑学坚、周斌.微型计算机原理与应用(第三版).北京：清华大学出版社，2001年6月第3版
[8] 侯俊杰.深入浅出MFC.武汉：华中科技大学出版社，2001年1月第2版
[9] 孙桓、陈作模、葛文杰.机械原理.高等教育出版社，2006年5月1日第7版

附录 A：电路原理图

1. 主板原理图



2. 驱动板原理图



附录B：主要源代码

```
#include "Mycommon.h"

void Delay200ns()    //延时 200ns
{
    asm("nop"); asm("nop"); asm("nop"); asm("nop");
    asm("nop"); asm("nop"); asm("nop"); asm("nop");
    asm("nop"); asm("nop"); asm("nop"); asm("nop");
}
void DelayUs(void)
{
    unsigned char i;
    for (i = 0; i < 5; i++)
        Delay200ns();
}
void DelayMs(int a)
{
    unsigned int i;
    for (i = 0; i < a*500; i++)
        DelayUs();
}

void nvic_init(void)
```

```
{
    NVIC_InitTypeDef NVIC_InitStructure0;
    NVIC_InitStructure0.NVIC_IRQChannel = PIT0_IRQn;                    //指明要
配置的通道号
    NVIC_InitStructure0.NVIC_IRQChannelGroupPriority = NVIC_PriorityGroup_2;
    NVIC_InitStructure0.NVIC_IRQChannelPreemptionPriority = 0x01;       //抢占优先
级 1
    NVIC_InitStructure0.NVIC_IRQChannelSubPriority  =  0x00;            //响应优
先级 1
    NVIC_InitStructure0.NVIC_IRQChannelEnable  =  TRUE;                 //使能
PIT0_IRQn 中断通道
    LPLD_NVIC_Init(NVIC_InitStructure0);

    NVIC_InitTypeDef NVIC_InitStructure1;
    NVIC_InitStructure1.NVIC_IRQChannel = PORTB_IRQn;
    NVIC_InitStructure1.NVIC_IRQChannelGroupPriority = NVIC_PriorityGroup_2;
    NVIC_InitStructure1.NVIC_IRQChannelPreemptionPriority = 0x00;   //抢占优先级 0
    NVIC_InitStructure1.NVIC_IRQChannelSubPriority = 0x00;        //响应优先级 0
    NVIC_InitStructure1.NVIC_IRQChannelEnable = TRUE;          //使能外部中断通道
    LPLD_NVIC_Init(NVIC_InitStructure1);        //根据 NVIC_InitStruct 中指定的参数初始化外
设 NVIC 寄存器


}


void system_Init()
{
        uart_init();
    Video_init();
        servo_init();
        motor_pwm_init();
        motor_DMA_Init();
        nvic_init();
        KEY_IO_Init();
        OLED_Init();
        //OLED_Set_OLED();
     //     pit_init();

}
//////////液晶程序////////////////////////////////
/*********************************************************************/
void OLED_Delay( void )
{
```

```
    unsigned int i, j;
    for ( i=0; i<200; i++ )
    {
        for ( j=0; j<2000; j++ )
        {
            asm( "nop" );
        }
    }
}
void OLED_Write_Cmd(unsigned char cmd )
{
    unsigned char i;
    OLED_DC = 0;;;
    OLED_SCLK = 0;;;
    for(i=0;i<8;i++)
    {
        if(cmd&(0x80>>i))
        {
            OLED_SDIN = 1;
        }
        else
        {
            OLED_SDIN = 0;
        }
        OLED_SCLK = 1;;;
        OLED_SCLK = 0;;;
    }

    OLED_DC = 1;
    OLED_SDIN = 1;
}

void OLED_Write_Data(unsigned char data )
{
    unsigned char i;
    OLED_DC = 1;;;
    OLED_SCLK = 0;;;
    for(i=0;i<8;i++)
    {
        if(data&(0x80>>i))
        {
            OLED_SDIN = 1;;;
        }
        else
```

```
            {
                OLED_SDIN = 0;;;
            }
            OLED_SCLK = 1;;;
            OLED_SCLK = 0;;;
        }

        OLED_DC = 1;
        OLED_SDIN = 1;
}

void OLED_Set_Address(unsigned char row, unsigned char column )      //?°?·
{
        OLED_Write_Cmd(0xb0+row);
        OLED_Write_Cmd(((column&0xf0)>>4)|0x10);
        OLED_Write_Cmd((column&0x0f)|0x00);
}


void OLED_Full(unsigned char data)
{
        unsigned char row,column;
        for( row=0; row<8; row++ )
        {
            OLED_Set_Address(row,0);
            for( column=0; column<128; column++ )
            {
                OLED_Write_Data( data );
                asm("nop");
                asm("nop");
            }
        }
}

void OLED_Init(void)
{
        OLED_IO_Init( );
        OLED_SCLK=1;

        OLED_RES=0;
        OLED_Delay( );
        OLED_RES=1;
        OLED_Write_Cmd(0xae);//--turn off oled panel
        OLED_Write_Cmd(0x00);//---set low column address
```

```
    OLED_Write_Cmd(0x10);//---set high column address
    OLED_Write_Cmd(0x40);//--set start line address  Set Mapping RAM Display Start Line (0x00~0x3F)
    OLED_Write_Cmd(0x81);//--set contrast control register
    OLED_Write_Cmd(0xcf); // Set SEG Output Current Brightness
    OLED_Write_Cmd(0xa1);//--Set SEG/Column Mapping     0xa0×óòò•′ ?? 0xa1?y3£
    OLED_Write_Cmd(0xc8);//Set COM/Row Scan Direction   0xc0é??? •′ ?? 0xc8?y3£
    OLED_Write_Cmd(0xa6);//--set normal display
    OLED_Write_Cmd(0xa8);//--set multiplex ratio(1 to 64)
    OLED_Write_Cmd(0x3f);//--1/64 duty
    OLED_Write_Cmd(0xd3);//-set display offset    Shift Mapping RAM Counter (0x00~0x3F)
    OLED_Write_Cmd(0x00);//-not offset
    OLED_Write_Cmd(0xd5);//--set display clock divide ratio/oscillator frequency
    OLED_Write_Cmd(0x80);//--set divide ratio, Set Clock as 100 Frames/Sec
    OLED_Write_Cmd(0xd9);//--set pre-charge period
    OLED_Write_Cmd(0xf1);//Set Pre-Charge as 15 Clocks & Discharge as 1 Clock
    OLED_Write_Cmd(0xda);//--set com pins hardware configuration
    OLED_Write_Cmd(0x12);
    OLED_Write_Cmd(0xdb);//--set vcomh
    OLED_Write_Cmd(0x40);//Set VCOM Deselect Level
    OLED_Write_Cmd(0x20);//-Set Page Addressing Mode (0x00/0x01/0x02)
    OLED_Write_Cmd(0x02);//
    OLED_Write_Cmd(0x8d);//--set Charge Pump enable/disable
    OLED_Write_Cmd(0x14);//--set(0x10) disable
    OLED_Write_Cmd(0xa4);// Disable Entire Display On (0xa4/0xa5)
    OLED_Write_Cmd(0xa6);// Disable Inverse Display On (0xa6/a7)
    OLED_Write_Cmd(0xaf);//--turn on oled panel
    OLED_Full( 0 );
    OLED_Set_Address(0,0);
}

void OLED_IO_Init(void)
{
    GPIO_InitTypeDef GPIO_OLED = { 0 };
    GPIO_OLED.GPIO_Dir = DIR_OUTPUT;
    GPIO_OLED.GPIO_Isr = NULL;
    GPIO_OLED.GPIO_Output = OUTPUT_L;
    GPIO_OLED.GPIO_PinControl = IRQC_DIS;
    GPIO_OLED.GPIO_Pins = GPIO_Pin18 | GPIO_Pin20 | GPIO_Pin10| GPIO_Pin16;
    GPIO_OLED.GPIO_PTx = PTB;
    LPLD_GPIO_Init(GPIO_OLED);
}
void KEY_IO_Init(void)
{
```

```
        GPIO_InitTypeDef GPIO_LED = { 0 };
    GPIO_LED.GPIO_Dir = DIR_OUTPUT;
    GPIO_LED.GPIO_Isr = NULL;
    GPIO_LED.GPIO_Output = OUTPUT_H;
    GPIO_LED.GPIO_PinControl = IRQC_DIS;
    GPIO_LED.GPIO_Pins = GPIO_Pin15;
    GPIO_LED.GPIO_PTx = PTA;
    LPLD_GPIO_Init(GPIO_LED);


   GPIO_InitTypeDef GPIO_KEY1 = { 0 };
    GPIO_KEY1.GPIO_Dir = DIR_INPUT;
    GPIO_KEY1.GPIO_Isr = NULL;
    GPIO_KEY1.GPIO_PinControl = IRQC_DIS|INPUT_PULL_UP ;
    GPIO_KEY1.GPIO_Pins = GPIO_Pin11 | GPIO_Pin2 | GPIO_Pin9 | GPIO_Pin22;
    GPIO_KEY1.GPIO_PTx = PTB;
    LPLD_GPIO_Init(GPIO_KEY1);

    GPIO_InitTypeDef GPIO_KEY2 = { 0 };
    GPIO_KEY2.GPIO_Dir = DIR_INPUT;
    GPIO_KEY2.GPIO_Isr = NULL;
    GPIO_KEY2.GPIO_PinControl = IRQC_DIS|INPUT_PULL_UP;
    GPIO_KEY2.GPIO_Pins = GPIO_Pin8 | GPIO_Pin9 |GPIO_Pin10 |GPIO_Pin11 |GPIO_Pin12;
    GPIO_KEY2.GPIO_PTx = PTC;
    LPLD_GPIO_Init(GPIO_KEY2);

}

void OLED_Write_Byte16X8(unsigned char row, unsigned char column, unsigned char data)
{
    unsigned char i;
    OLED_Set_Address( row*2, column*8 );
    for ( i=0; i<8; i++ )
    {
        OLED_Write_Data( OLED_ASCII16X8[data][i] );
    }
    OLED_Set_Address( row*2+1, column*8 );
    for ( i=8; i<16; i++ )
    {
        OLED_Write_Data( OLED_ASCII16X8[data][i] );
    }
}
void OLED_Write_Str16X8(unsigned char row,unsigned char column, unsigned char *data)
```

```
{
    while ( *data != '\0')
    {
        OLED_Write_Byte16X8( row, column, *data );
        column++;
        data++;
    }
}

void OLED_Write_Byte16X8_F(unsigned char row,unsigned char column, unsigned char data)

{
    unsigned char i;
    OLED_Set_Address(row * 2, column * 8);
    for (i = 0; i<8; i++)
    {
        OLED_Write_Data(~OLED_ASCII16X8[data][i]);
    }
    OLED_Set_Address(row * 2 + 1, column * 8);
    for (i = 8; i<16; i++)
    {
        OLED_Write_Data(~OLED_ASCII16X8[data][i]);
    }
}

void OLED_Write_data16X8(unsigned char row,unsigned char column, unsigned char data )
{
      unsigned char i;
      OLED_Set_Address( row*2, column*8 );
      for ( i=0; i<8; i++ )
      {
            OLED_Write_Data( OLED_ASCII16X8[data+48][i] );
      }
      OLED_Set_Address( row*2+1, column*8 );
      for ( i=8; i<16; i++ )
      {
            OLED_Write_Data( OLED_ASCII16X8[data+48][i] );
    }
}


unsigned char getbai(int a)
{
  unsigned char bai;
```

```
    bai=a/100;
    return bai;
}

unsigned char getshi(int a)
{
  unsigned char shi;
    shi=(a/10)%10;
    return shi;
}

unsigned char getge(int a)
{
    unsigned char ge;
    ge=a%10;
    return ge;
}
void Show_LCD_Image(unsigned char videodata[ROW][COLUMN])
{
        unsigned char i = 0, j = 0, temp = 0;
        OLED_Set_Address(0, 0);



        for (i = 0; i < 128; i++)
        {
                for (j = 0; j < 8; j++)
                {
                        OLED_Set_Address(j, i);
                        //按比例显示在显示屏上
                        temp = (videodata[j * 8 * (ROW - 1) / 63][i*(COLUMN - 1) / 127]) | (videodata[(j *
8 + 1)*(ROW - 1) / 63][i*(COLUMN - 1) / 127] << 1)
                                | (videodata[(j * 8 + 2)*(ROW - 1) / 63][i*(COLUMN - 1) / 127] << 2) |
(videodata[(j * 8 + 3)*(ROW - 1) / 63][i*(COLUMN - 1) / 127] << 3)
                                | (videodata[(j * 8 + 4)*(ROW - 1) / 63][i*(COLUMN - 1) / 127] << 4) |
(videodata[(j * 8 + 5)*(ROW - 1) / 63][i*(COLUMN - 1) / 127] << 5)
                                | (videodata[(j * 8 + 6)*(ROW - 1) / 63][i*(COLUMN - 1) / 127] << 6) |
(videodata[(j * 8 + 7)*(ROW - 1) / 63][i*(COLUMN - 1) / 127] << 7);
                        OLED_Write_Data(temp);
                }
        }
}
//////////////////////flash&液晶调参//////////////////////
void OLED_Show_Page( void )
```

```
{
    switch (OLEDPage)
    {
    case 1: OLED_Set_Pagestart();
        break;
    case 2: OLED_Set_Pagewarn();
        break;
    case 3: OLED_Set_Page1();
        break;
    case 4: OLED_Set_Page2();
        break;
    case 5: OLED_Set_Page3();
        break;
    case 6: OLED_Set_Page4();
        break;
    case 7: OLED_Set_Page5();
        break;
    case 8: OLED_Set_Page6();
        break;
    case 9: OLED_Set_Page7();
        break;
    case 10: OLED_Set_Page8();
        break;
    case 11: OLED_Set_Pagelogo();
        break;
    }
}

//OLED 初始化函数
void OLED_Set_OLED( void )
{
  uint8 Key_Num=0;
    //OLED_Init();
        OLED_Set_Pagestart();  //选择模式，然后刷新参数数据
        Flash_Erase(68);
        Flash_Write(68, 0, Moshi);//将模式写入 flash

        OLED_Set_Page1();
        OLED_Set_Page2();
        OLED_Set_Page3();
        OLED_Set_Page4();
        OLED_Set_Page5();
        OLED_Set_Page6();
        OLED_Set_Page7();
```

```
        OLED_Set_Page8();
    OLED_Show_Page();
    while ( SetOK == 0 )
    {
        Key_Num = OLED_Key_Scan();
        switch (Key_Num)
        {
        case 1:

            {
                Dataset = 1;
                Datasetnumsec = 1;
            OLED_Show_Page();
            }
            break;
        case 2:

            {
                if (OLEDPage < SHOW_MAX)
            {
                OLEDPage++;
            }
            else
            {
                OLEDPage = SHOW_Debug;
            }
            OLED_Show_Page();
            }
            break;
        case 3:

            {
            if ( OLEDPage > SHOW )
            {
                OLEDPage--;
            }
            else
            {
                OLEDPage = SHOW_MAX;
            }
            OLED_Show_Page();
            }
            break;
        case 4:
```

```
                {

                    OLEDPage = SHOW_Debug;
                                        OLED_Show_Page();
                }
                break;

            default : break;
            }

        }

        OLED_Full( 0 );
}
void dataset(void)
{
   Dataset = 0 ;
   Moshi=Flash_Read( 68, 0 );
   OLED_Set_Page1();
   OLED_Set_Page2();
   OLED_Set_Page3();
   OLED_Set_Page4();
   OLED_Set_Page5();
   OLED_Set_Page6();
   OLED_Set_Page7();
   OLED_Set_Page8();
   OLED_Full( 0 );
}
```