第一届“飞思卡尔”杯全国大学生智能汽车邀请赛

技 术 报 告

研究论文：智能寻迹小车控制算法研究与程序设计

学　　校：华中科技大学

队伍名称：九　头　鸟

参赛队员：程钊 刘广林 唐旋来

带队老师：彭　刚

目 录

第一章 引 言

智能小车以飞思卡尔 16 位微控制 MC9S12DG128B 为控制器，采用多传感器进行信息采集，运用反射式红外传感器设计路径检测模块和速度监测模块。同时，采用 PWM 技术，控制舵机的转向和电机转速。系统还扩展了 LCD(Liquid Crystal Display：液晶显示屏)和键盘模块作为人机操作界面，以便于智能小车的相关参数调整。此外，PID 寻迹算法结合 ABS（Anti-skid Brake System.：防抱死系统）技术，使我们的小车能在曲折的赛道上畅通无阻。

技术报告以智能小车的设计为主线，包括小车的构架设计、软硬件设计，以及控制算法研究等，分为六章。其中，第一章为引言部分，第二章主要介绍了小车的总体设计，第三章对小车的硬件设计进行了详细的介绍，其中包括机械改造，电路设计两大部分，第四章描述了小车的软件设计和相关算法，第五章对使用到的单片机资源作了说明，第六章叙述了我们在设计过程中遇到的问题和解决方法。附录 A 的研究论文介绍了小车智能行驶的控制算法，附录 B 为程序源代码清单，红外传感器参数说明见附录 C，附录 D 为小车配件清单。

第二章　　智能小车设计分析

2.1 设计要求

在本次竞赛中，要求所设计的小车具有自动寻迹的功能，能在指定跑道上高速，稳定地运行。跑道为黑白两色。其背景色为白色，跑道中央有一条黑线作为小车行进的依据。很明显，我们要设计的小车是要能沿黑线的正常行驶，并在此基础上，尽量提高小车行驶速度。

2.2 总体设计

系统框架如图 1-1 所示：

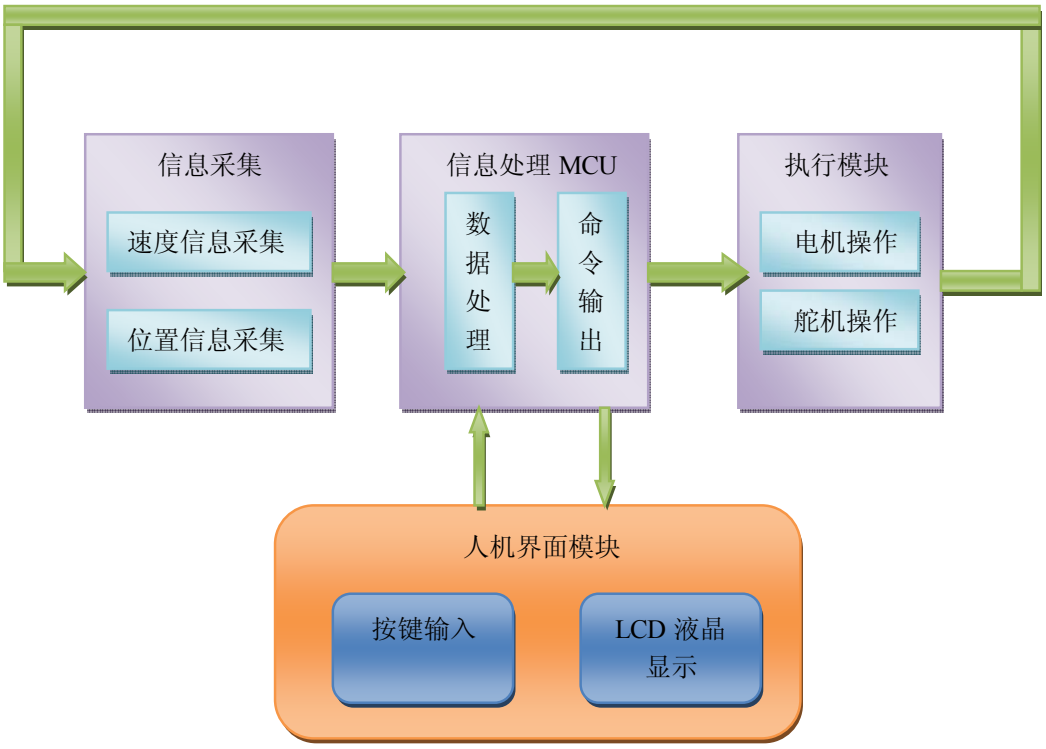

图 1.1 系统构架

如图 1-1 所示，该智能小车系统主要分为以下四大块：

（1） 信息采集模块：在该模块中包括有速度信息采集和位置信息采集两个子模块，分别采集小车当前的位置信息和速度信息，并将采集到的信息传给 MCU，其核心是传感器。
（2） 信息处理模块：信息处理模块包括信息处理和控制模块，其核心是 MCU，MCU 接收到采集来的信号，对信号进行处理后作出判断，并发出控制命令。
（3） 执行模块：该模块包括了驱动电机和舵机，当接收到 MCU 的命令后便执行相应的操作，同时信息采集模块又采集到电机和舵机的状态信息，反馈给 MCU 。从而整个系统构成一个闭环系统，在运行过程中，系统自动调节而达到正确行驶的目的。
（4） 人机交互模块：在该模块中包含了按键输入与 LCD 液晶显示，其中按键用于调节小车的工作模式，同时也用于调节小车行驶时所需的一些参数，结合 LCD 液晶显示，从而使整个小车系统更具人性化

2.3 方案论证

2.3.1 传感器设计方案

在传感器方案的选择中，有以下两种方案供参考：

方案一：使用 CCD 传感器来采集路面信息。使用 CCD 传感器，可以获取大量的图像信息，可以全面完整的掌握路径信息，可以进行较远距离的预测和识别图像复杂的路面，而且抗干扰能力强。但是对于本项目来说，使用 CCD 传感器也有其不足之处。首先使用 CCD 传感器需要有大量图像处理的工作，需要进行大量数据的存储和计算。因为是以实现小车视觉为目的，实现起来工作量较大，相当繁琐。

方案二：使用光电传感器来采集路面信息。使用红外传感器最大的优点就是结构简明，实现方便，成本低廉，免去了繁复的图像处理工作，反应灵敏，响应时间低，便于近距离路面情况的检测。但红外传感器的缺点是，它所获取的信息是不完全的，只能对路面情况作简单的黑白判别，检测距离有限，而且容易受到诸多扰动的影响，抗干扰能力较差，背景光源，器件之间的差异，传感器高度位置的差异等都将对其造成干扰。

在本次比赛中，赛道只有黑白两种颜色，小车只要能区分黑白两色就可以

采集到准确的路面信息。经过综合考虑，在本项目中采用红外光电传感器作为信息采集元件。

2.3.2 控制算法设计方案

在小车的运行中，主要有方向和速度的控制，即舵机和电机的控制，这两个控制是系统软件的核心操作，对小车的性能有着决定性的作用。

对舵机的控制，要达到的目的就是：在任何情况下，总能给舵机一个合适的偏移量，保证小车能始终连贯地沿黑线以最少距离行驶。在舵机的控制方案中，有以下两种方案可供选择：

方案一：比例控制

这种控制方法就是在检测到车体偏离的信息时给小车一个预置的反向偏移量，让其回到赛道。比例算法简单有效，参数容易调整，算法实现简单，不需复杂的数字计算。在实际应用中，由于传感器的个数与布局方式的限制，其控制量的输出是一个离散值，不能对舵机进行精确的控制，容易引起舵机左右摇摆，造成小车行驶过程中的振荡，而且其收敛速度也有限。

方案二：PID 控制

PID 控制在比例控制的基础上加入了积分和微分控制，可以抑制振荡，加快收敛速度，调节适当的参数可以 有效地解决方案一的不足。不过，P，I，D 三个参数的设定较难，需要不断进行调试，凭经验来设定，因此其适应性较差。

在我们的选择中，根据比赛规则，赛道模型与相关参数已给定，即小车运行的环境基本上已经确定，可通过不断调试来获得最优的参数。因此我们选用的是 PID 算法来对舵机进行控制。

对驱动电机的控制（即速度控制），要达到的目的就是在行驶过程中，小车要有最有效的加速和减速机制。高效的加速算法使小车能在直道上高速行驶，而快速减速则保证了小车运行的稳定，流畅。为了精确控制速度，时时对速度进行监控，我们还引入了闭环控制的思想，在硬件设计，增加了速度传感器实时采集速度信息。

第三章　　智能小车硬件设计

3.1 机械设计

3.1.1 车模结构特点

本项目采用后轮驱动，前轮转向。使用前置单排非均匀排布红外传感器探测路面信号。电源模块和 MCU 的扩展电路板置于小车顶部。整个小车重心在中部偏后，有较好的稳定性。经过改装后的车模尺寸如下

3.1.2 寻迹传感器布局

(1)分析：寻迹传感器模块的设计是整个智能小车设计中的最重要的一部分，其作用相当于人的眼睛和耳朵，采集外部路面的信息并将其送入 MCU 微控制器进行数据处理，其能否正常工作直接影响着小车对路面的判断以及小车下一步的行动，因而其布局的合理性与有效性对小车稳定而又快速的行驶起着至关重要的作用。我们认为在传感器的布局中，要解决两个问题：信息检测的精确度和信息检测的前瞻性。

寻迹传感器的布局常见的有以下几种方案

方案一：一字形布局

反射式光电传感器在小车前方一字形简单排布。在一字形中传感器的间隔有均匀布局和非均匀布局两种方式，均匀布局不利于弯道信息的准确采集，通常采取的是非均匀布局。考虑到弧度信息采集的连贯性，非均匀布局的理论依据是等角度分布原则，即先确定一合适的定点，从顶点依次等角度画射线，射线与传感器水平线相交的位置即为传感器的位置。这种方案信息检测相对连贯，准确，使控制程序算法简单，小车运行连贯，稳定。

方案二：M 形布局

传感器呈 M 形排布。这种方案的优点在于拓宽了边沿传感器的检测范围，更适合于小车快速行进中的弯道检测，但相对一字形布局来说，M 形布局不利

于信息检测的稳定，易于产生振荡，不利于小车行驶的稳定。

方案三：活动式传感器布局

前面两种方案都是固定的布局方式，使传感器对赛道有一定的依赖。在这个方案中，传感器的位置是可以在一定范围内灵活排布的。这种方案的布局思路是传感器在安装板上的位置是可调的，先将传感器排布成为矩形点阵，根据不同的赛道情况而灵活地作出调整，就可以设计出不同的布局方式而适应不同的赛道。这样对不同赛道有更强的适应性。但这种方案可调性大，临时调节较难，其次机械设计中体积较大，增加了小车的重量，不利于加减速。

在我们的方案选择中，我们采用的是上述第一种方案与第二种方案的结合，通过比较，我们对第一第二种方案进行综合，扬长避短，优势互补。由于本次比赛的赛道相关参数已知，而且赛道只有直道和弯倒两种，可以在测试中对赛道进行模拟，赛道变化不大，因此没有采取第三种方案。

传感器布局图如图 3-1 所示

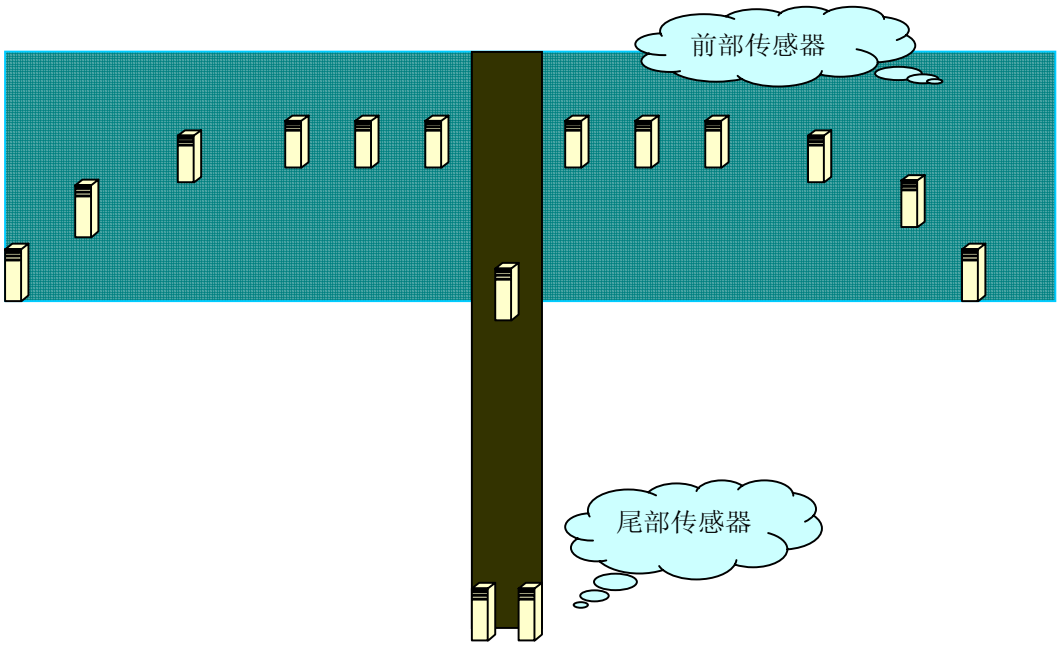

图 3-1 传感器布局示意图

如图所示，整个传感器布局呈喷泉状。中间黑线即为跑道中央黑线，在小车车头和车尾的正中心（即路面黑线处布局了三个传感器（前面一个，后面两个），用来判断小车是否处于弯道状态（当在一定时间内三个传感器都检测到黑线则小车处于直道上），且尾部传感器可方便地判断偏离较大（前部传感器完全冲出黑线）的情况，进一步防止小车冲出跑道。车头前部在黑线外围一共有十二个传感器，呈对称分布.传感器间隔遵循由中间至外围由密到疏的排布方式，中间四个在同一条水平线上，最外三个传感器呈圆弧状向内分布。在这种布局中，我们采用的是等角度非均匀排布原则（见方案一），靠近中间的 8 个传感器用于进一步判断小车是否处于弯道状态以及小车偏离的程度，适合于小车跑大圆弧和消除小车行进中的左右振荡现象。而边缘四个传感器则主要用感知小车大距离偏离黑线适合于小车在小圆弧上的高速行进。边缘四个传感器呈圆弧状向内排布，扩大了传感器的检测范围。

3.1.3 系统电路板的固定及连接

一共用到三块外接电路板，所有电路板都制作成印制板。分别为传感器主板，传感器尾板，车身主板(包括 MCU，调试电路，电源电路，测速传感器)

其详细情况见下表 3-1

表 3-1 电路板信息表

板名	规格(长 X 宽)	安装位置
车身主板	165 x 150 mm	车头最前言
传感器主板	220 x 96 mm	车身中央靠后
传感器尾板	51 x 24 mm	两后轮之间

3.2 电路设计

3.2.1 传感器电路设计

本项目中，选用的是红外对管 ST188 作为传感元件。ST188 是一个四端口元件，包括了一只红外发射管和红外接收管，用塑料外壳将对管封装起来，如

图 3-2 所示（传感器详细资料见附录 C）。

传感器电路如图 3-2 所示

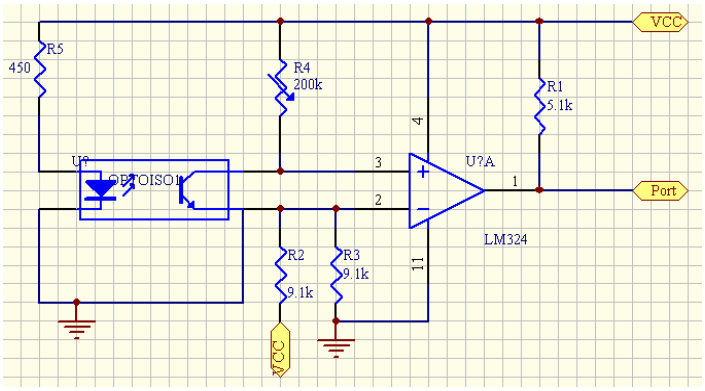

图 3-2 单对红外传感器电路图

R_1作为发射管的限流电阻，若R_1阻值过大，则发射管功率会大幅降低，所起其阻值在 50-200 欧之间可以根据需要选择。R_1,R_3,R_4的阻值要综合考虑确定。其基本工作原理是通过R_3,R_4来确定输出信号的门限值，当发射管压降V_D高于R_3上的压降时，由于运放的饱和特性，输出电压为 5V；当V_D低于R_3上的电压时，输出电压为 0V。C_6是一个滤波电容，可以过滤尖峰干扰。

当小车在白色地面行驶时，装在车下的红外发射管发射红外线信号，经白色反射后，被接收管接收，一旦接收管接收到信号，那么图中光敏三极管将导通，比较器输出为低电平；当小车行驶到黑色引导线时，红外线信号被黑色吸收后，光敏三极管截止，比较器输出高电平，从而实现了通过红外线检测信号的功能。将检测到的信号送到单片机 I/O 口，当 I/O 口检测到的信号为高电平时，表明红外光被地上的黑色引导线吸收了，表明小车处在黑色的引导线上；同理，当 I/O 口检测到的信号为低电平时，表明小车行驶在白色地面上。

3.2.2 测速传感器的设计

测速传感器安装于小车右后轮附近，我们在小车的靠近车轮的轴上贴一圈白色的胶带，再在白色胶带上贴上几条黑色的标记，这样就可用反射式红外传感器检测黑线，通过计数的方式来测量小车的速度。

测速传感器的电路设计同路面检测传感器的设计相同，都采用 ST188 来进行信息采集(见图一)，在车轮轴上作好黑色标记，通过对黑色标志的记数可得小

车车轮转过一周所用的时间 T，通过公式进行计算即可得到小车当前的运行速度 V。

3.2.3 电源管理模块

智能车系统根据各部件正常工作的需要，对配发的标准车模用 7.2V 2000mAh Ni-cd 蓄电池进行电压调节。其中，单片机系统、路径识别的光电传感器和接收器电路、车速传感器电路，LCD 显示电路，需要 5V 电压，伺服电机工作电压范围 4.8V 到 6V。智能车的电源调节图如下图所示：

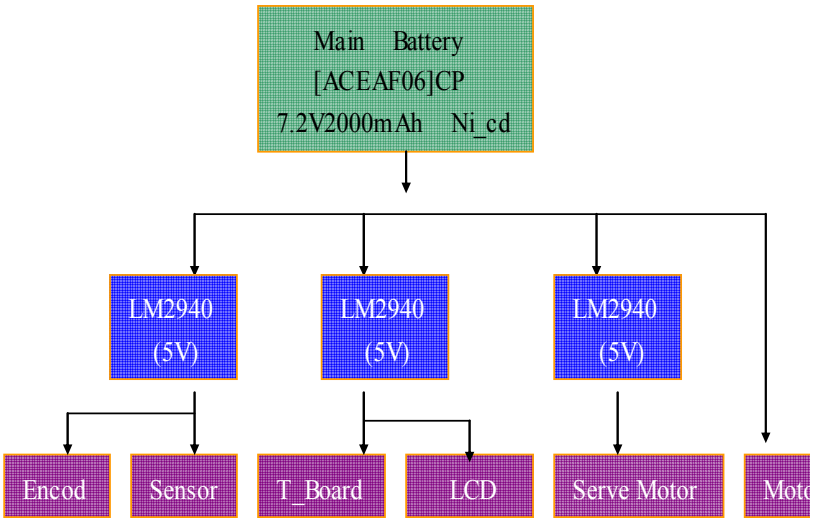

图 3-3 电压调节图

在电源管理芯片的选择中，常用的是 LM7805，LM7806。但由于电池的工作电压为 7.2V.而 LM7805，LM7806 正常工作时，其输入输出引脚之间的电压差通常为 2~3V，驱动电机工作时引起瞬间电压下降，造成芯片输出电压下降，影响其他模块的工作。经过实验我们确实也发现了此种情况，因此我们采用的是低压降的芯片 LM2940（LM2940 的电路图如下图 3-4 所示），为了避免各个模块的供电相互干扰，设计中，采用了三片 LM2940，每一芯片单独为一个模块供

电。在电路设计中，考虑到由于电机驱动所引起的电源不稳定（主要为瞬态脉冲），在电源输入端，各芯片电源引脚都加入了滤波电路。为了避免由于驱动电机转动时所引起的电磁干扰，在电路板设计中，在印制板上做了敷铜处理，将电路中的“地”与敷铜面相连接。

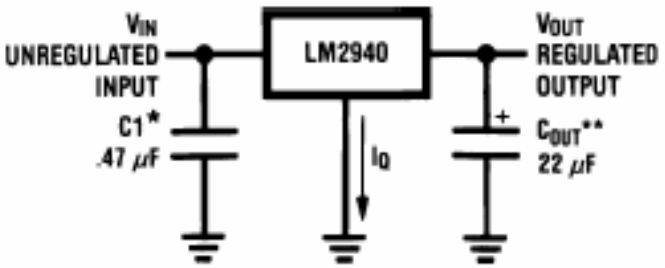

图 3-4 LM2940 应用电路图

3.2.4 驱动模块

直流电机的控制一般由单片机的 PWM 信号来完成，驱动芯片采用飞思卡尔半导体公司的半桥式驱动器 MC33886。电路图示见图 3-5

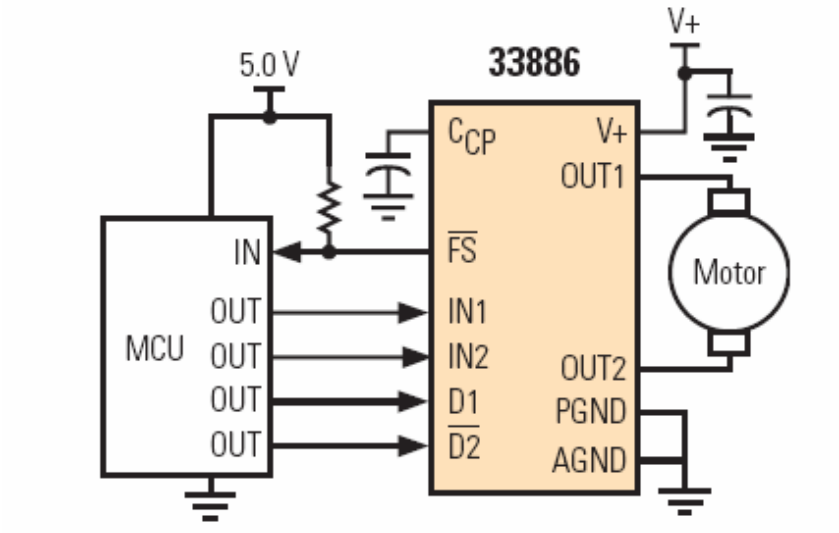

图 3-5 电机驱动电路图

MC33886 为桥式驱动电路，通过控制输入的信号，可以控制两个半桥的通断来实现电机的顺转与倒转。由于在赛车中不需要倒车，为了扩大芯片的驱动能力，

把两个半桥并联使用。

3.2.5 调试模块

为方便调试，我们设计了 LCD 显示电路和键盘电路,在调试过程中将小车的当前状态参数实时显示，并可以方便地通过键盘输入来调节参数，和切换小车的状态。

(1) LCD 显示电路设计

显示电路采用是 TS1620-1 液晶显示模块，该模块具有体积小，功耗低，操作方便的优点.该模块总共有 16 个 I/O 口与 MCU 相接,通过对模式选择位写入命令，可调节 LCD 的工作模式，分时进行命令和数据写入。

(2) 键盘电路设计

6 个键盘呈 3 x 2 阵列排布，6 个键盘分别具有模式选择，步进增，步进减，确定，退出功能。键盘统一采用带下拉电阻的连接方式，输出口与单片机 IO 口相连。键盘的工作模式为乒乓模式：当没有键按下时输出为低电平；有键按下时输出为高电平。

第四章　　智能小车软件设计

4.1 总体流程图

在软件的总体规划中，为方便现场调试与参数测试，我们将其分成了三个模式：测试模式，人机交互模式，比赛模式。软件流程图如图 4-1 所示：

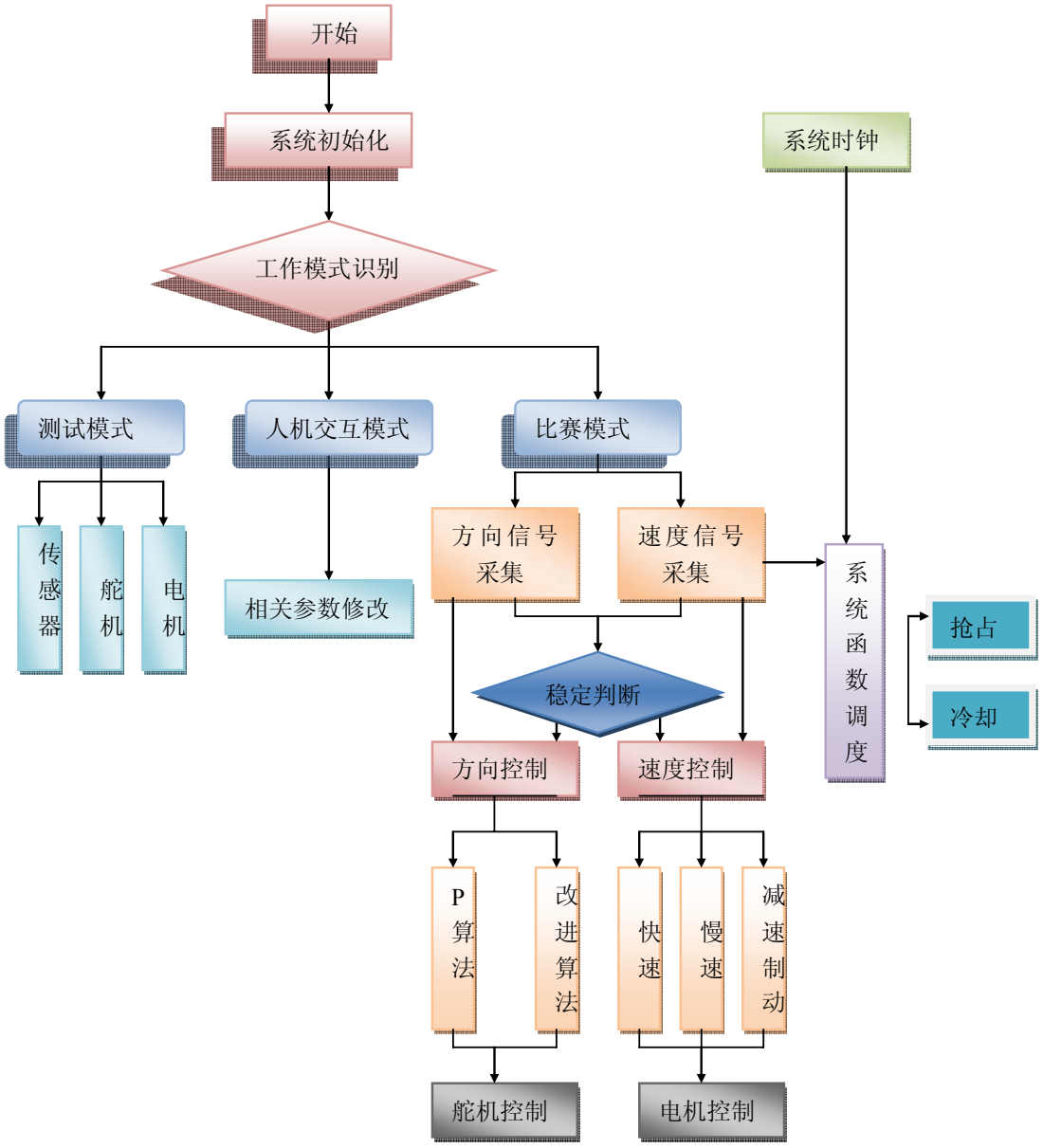

图 4-1 软件流程图

4.2 PID 控制算法

比例，积分，微分（PID）是建立在经典控制理论基础上的一种控制策略。PID 控制器作为最早实用化的控制器，已经有五十多年的历史，现在仍然是最广泛的工业控制器。PID 控制器最大的特点是简单易懂，使用中不需要精确的系统模型等先决条件，因而成为应用最广泛的控制器。

PID 控制器系统原理框图如图 4-1 所示：

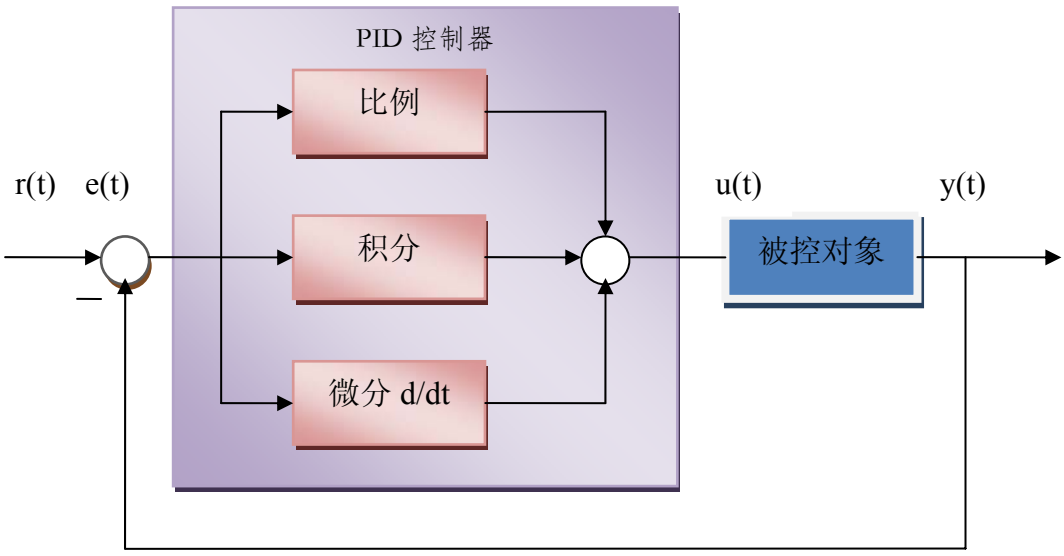

图 4-1 典型 PID 控制结构

在图 4-1 中，系统偏差信号为 e(t)=r(t)-y(t)。在 PID 调节作用下，控制器对误差信号 e(t)分别进行比例(P)，积分(I)，微分运算(D)，其结果的加权和构成系统的控制信号 u(t)，送给被控对象加以控制。

PID 控制器的数学描述为式 4-1

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{t}] \qquad \text{式 4-1}$$

式中，Kp 为比例系数，Ti 为积分时间常数，Td 为微分时间常数。

比例环节的主要作用是：Kp 的值增大时，系统的响应速度加快，闭环系统

响应的幅值增加。当达到某个 Kp 值，系统将趋于不稳定。

当增加积分时间常数 Ti 的值时，系统超调量减小，而系统的响应速度将变慢。因此，积分环节的主要作用是消除系统的稳态误差，其作用的强弱取决于积分时间常数 Ti 的大小。当增加微分时间常数 Td 时，系统的响应速度增加，同时响应的幅度也增加。因此，微分环节的主要作用是提高系统的响应速度。由于该环节产生的控制量与信号变化速率有关，因此对于信号无变化，或者变化缓慢的系统不起作用。

在本项目中智能小车位置信号为 y(t)，由传感器采集得到；小车期望的运行位置 r(t)为事先设置好的产量；输出信号 u(t)即为舵机控制信号。

4.3 舵机方向控制算法

如前面所述，我们对舵机的控制采用的是 PID 控制。

4.4 速度控制算法

重点研究一下小车的减速，这里有三种方方案：

方案一 通过机械接触小车轮或轴实现减速。这种方案使用的范围很广，但也很有效，是机械加工复杂，需要对车体作改造，另外可靠性与可控性不，不适合单片机的控制。。

方案二 使电机短时间停机。这种案可控性强，适合于轮轴自锁的电机，如减速电机与步进电机，，但由于小车的惯性作用，从刹车到停止需要一个缓冲时间，这样容易造成小车由直道进入弯道时容易冲出跑道，限制了小车的直道速度，而且使小车在弯道运行时连贯性很差第

方案三 电机瞬时反转来快速降低小车速度。这种适合于直流电机，这相当于给小车一个反向力矩来迅速降低小车的速度，很明显其减速效率优于方案二，经实验验证，其速度上连贯性也优于方案二。

为进一步精确控制小车速度，还需要引入闭环速度控制。把采用速度传感器检测到小车的实时速度，通过实际速度与期望速度之间的比较确定小车速度

状态以及决定加速或减速的强度大小。如果没有速度闭环，虽然也可以较好的实现速度控制，但是不能灵活地根据小车的实时速度来进行调整，这会降低小车对赛道的应变能力，会降低小车的整体速度。在减速的过程中，为了保证小车的连贯行驶，我们引入了 ABS 技术的思想，预设一个期望值，通过实际值与期望值的比较，分段进行减速。

第五章　开发流程

5.1 单片机资源划分

我们将单片机的 4000H 到 7FFFH 的地址设为默认程序 ROM 地址，共 16K 的存储空间，程序代码都将存储在该区域。而 RAM 地址定义为 1FFFH 到 2FFFH 的地址中。设置两个中断向量，分别是实时中断 RTI 和定时器中断 timer0。

表 5-1 单片机资源分配表

单片机接口	使用情况
PORTA	接 LCD 数据位 / 传感器 0～7（该接口复用）
PORTB	接传感器 8～14
PORTD	用于 PWM 输出，接舵机与电机
PORTE	前 5 脚接按键，后 3 位接 LCD 控制位
PORTE	接驱动芯片控制端
单片机内部资源	使用情况
RAM	1KB（0x1FFF~0x2FFF）
ROM	16KB（0xC000~0xFEFF）
堆栈	256B
中断	实时中断
	输入捕捉中断
	增强型时钟中断

5.2 编译环境

在制作过程中，运行的编译环境为 CodeWarrior 4.1

5.3 下载调试

调试下载工具应用的是由清华飞思卡尔研发中心开发的 BDM 调试器

第六章　开发总结与心得

6.1 开发与调试过程

在开发与调试的过程中，我们遵循由易到难，由简到繁，循序渐进的过程。我们将具体过程分为以下三个阶段进行。

第一阶段：基本设计阶段

这一阶段的目标就是初步分析小车的整体框架，对系统进行模块划分。通过基本的硬软件设计，初步实现小车的行走与寻迹功能。在这一阶段中，我们重点解决了以下几个问题：

(1)理论准备，重点是对单片机编译环境的熟悉。

(2)将小车系统划分为传感器，MCU 及附属电路，电源供应及电机驱动三大模块

(3)根据大赛要求，初步设计制作实现上述各模块的功能。

(4)组装小车，测试小车的整机性能。

第二阶段：方案论证与实验阶段

在这一阶段中，主要是先提出对各种具体的设计方案，然后进行理论的论证与实验的验证，根据实验的结论对各种方案进行比较与分析，并结合设计要求确定设计方案。这一阶段包括以下几个方面：

(1)传感器设计

在传感器型号的选用中，我们重点选择了 ST168，ST188，GY043W 进行测试，经反复实验，综合设计要求，最终选用了 ST188。

在传感器的布局设计中，我们先后进行了一字形布局，M 字形布局，活动式布局，在活动式布局的测试中，我们最终选用了喷泉形作为最终设计。

(2)电源管理的改进

在电源模块的设计中，我们选择了 LM7805，LM7806，LM2575，LM2940 进行了性能测试，最终选用了低压差稳压片 LM2940 作为稳压芯片。

(3)控制算法的实验

(1) 舵机控制

首先我们选用的是比例控制算法，先将 P 控制算法达到极限后再加入了积分，即应用 PI 控制，对积分参数进行调节，将 PI 控制达到极限后，最后加入微分，即实现 PID 控制，不断测试调节，选择最优参数。

(2) 速度控制

我们通过对加速和减速算法进行了多种方案的测试，并加入了测速传感器对速度进行实时检测。

第三阶段：方案改造与优化阶段

这一阶段是在第二阶段的基础上对设计方案进行局部调整与优化，通过不断的实践，不断地进行参数调整，硬件改造，软件优化。为了方便调试，我们加入了调试电路，可现场对参数进行调整

6.2 开发中遇到的几个典型问题

6.2.1 电源管理问题

在小车的运行过程中，电源管理是一个很重要的问题，在调试过程中，由于刚开始设计的不完善，小车在运行中出现了舵机运行失控，单片机工作异常。经过分布调试，我们最终发现这些都由电源管理设计失当所引发。电源设计主要有两方面的问题：

(1)稳压芯片选择失当

由于供电电池稳定时只有 7.2V，而稳压输出需 5V，6V。因此选择低压差的芯片，保证输出电压的稳定。

(2)各供电模块相互干扰

传感器模块，控制器模块，如果驱动模块的供电分配不当，会导致模块相互干扰，造成模块工作不稳定。我们采取的措施是对这三个模块，分别用相互独立的稳压电路供电。

6.2.2 PID 微分误差的问题

有许多求离散时间微分的方法。最简单的方法是：

$$D(n)=K_D\frac{E(n)-E(n-1)}{\Delta t} \qquad \text{公式 6-1}$$

实际上，这种一阶微分方程，对于噪声十分敏感。图 6-1 给出了一个带有噪声的 E（n）序列。通过该图我们可以看到，若在这种情况下使用公式 6-1 进行微分运算，会产生很大的误差。

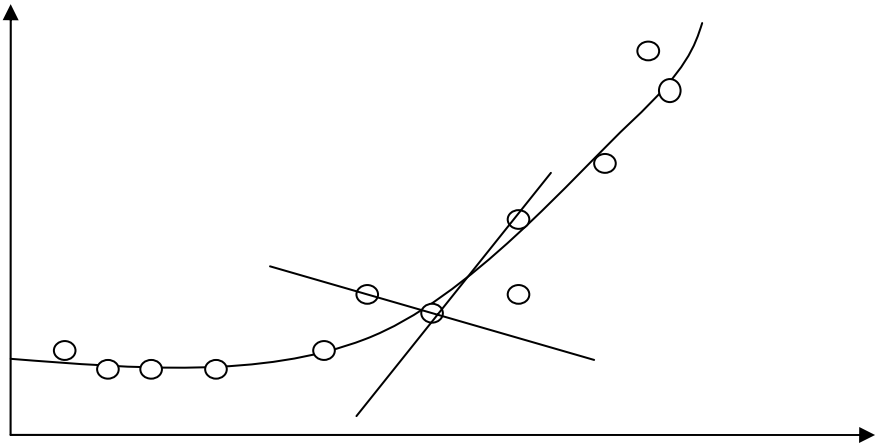

图 6-1 噪声对求离散微分的影响

为了减小这样的误差，可以通过计算两个不同时间跨度（甚至是多个不同时间跨度）的微分平均值来实现。

$$D(n)=K_D[\frac{1}{2}\frac{E(n)-E(n-3)}{3\Delta t}+\frac{1}{2}\frac{E(n-1)-E(n-2)}{\Delta t}] \qquad \text{公式 6-2}$$

公式 6-2 可以化简为如下形式:

$$D(n)=K_D\frac{E(n)+3E(n-1)-3E(n-2)-E(n-3)}{6\Delta t} \qquad \text{公式 6-3}$$

另外，在积分控制中，在开始或停止工作的瞬间，或者大幅度地给定量时，由于偏差较大，积分项的作用将会产生一个很大的超调，影响车辆调整的稳定，甚至使车辆偏出跑道。

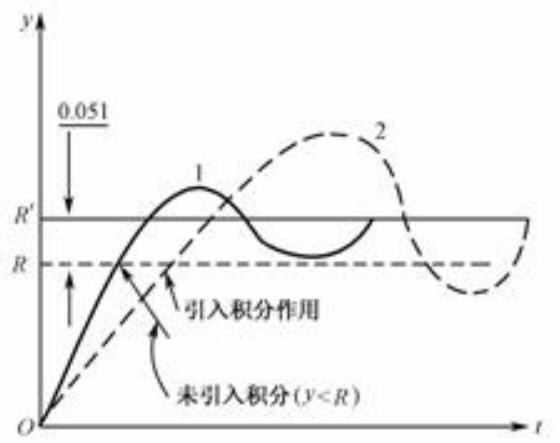

图 6-2 引入积分导致的超调效应

为此，可以采用积分分离手段，即在被控制量开始跟踪时，取消积分作用，直到被控制量接近新的给定值时，才可以在 PID 算式中，引入如下的算法逻辑功能。

$$u(k) = K_p e(k) + K_i T_i \sum_{i=0}^{k} e(i) + T_d [e(k) - e(k-1)] \qquad \text{公式 6-4}$$

式中，Ki——引入的逻辑系数。

$$K_i = \begin{cases} 1 & e(i) \leq d\% \\ 0 & e(i) > d\% \end{cases}$$

6.2.3 电机电磁干扰的问题

由于电源电路，驱动电路和 MCU 电路板均位于小车后部，距离电机距离较近，因此会受到电机的电磁干扰，造成数据的失真甚至是硬件器件的损坏。在实际项目中，为了避免干扰，对于电路板进行了重新的设计和制版，用宽大的底线包围电路板，并且在全板加设了铜敷。这对于以后类似情况的处理都有很大帮助。

6.3 总结与展望

在这次设计中，我们初步完成了小车的制作，达到了比赛的基本要求，但

由于时间有限，小车各方面的性能还有待提高，设计方案还可以进一步扩展如传感器采用 CCD 与红外相结合，控制采用模糊控制等，这需要我们以后进一步完善。

参考文献

[1] 董克,刘明锐.机器人与人工智能发展.上海，上海交通大学出版社，2004

[2] 陈懂，刘瑢，金世俊.智能小车的多传感器数据融合.现代电子技术，2005，第六期

[3] 张宏希.红外接收组件原理及其应用.石河子大学学报，2005.12

[4] MC9S12DB128B Device User Guide V1.09.Motorola,Inc

[5] Jonathan W.Vavlano 著 李曦等译.嵌入式微计算机系统 实时接口技术[M].北京.机械工业出版社,2003

[6] PWM_8B8C Block User Guilde V01.16.Motorola Inc.Document number:S12PWM8B8CV1/D

[7] 王宜怀 刘晓升.嵌入式应用技术基础教程[M] 北京清华大学出版社 2005

[8] 邵贝贝.单片机嵌入式应用的在线开发方法[M].北京.清华大学出版社,2004

[9] 吕广明，孙立宁.自动引导车轨迹偏差的智能控制.哈尔滨工业大学学报.2002 年 35（12）:1466-1467

[10] Jean J.Labrosse 著 邵贝贝等译 .嵌入式实时操作系统 uC/ OS-II（第二版）[M].北京.北京航空航天大学出版社,2003.5

附录 A：研究论文

智能寻迹小车控制算法研究与程序设计

程钊，王琰，唐旋来，刘广林

（华中科技大学 控制科学与工程系，武汉 437300）

摘要

本文研究了基于嵌入式单片机技术的一种智能寻迹小车的控制算法。算法的主要功能是使小车能对路况进行判断进而对行驶策略和控制规则进行决策，实现快速稳定的巡线行驶。论文对智能小车算法进行了深入的研究，从分析小车和跑道的特点入手，提出了基本的算法设计方案，并对方案的实现进行了详细阐述和说明，包括调度模块，速度控制模块，PID 控制模块，记忆模块和信号处理模块。

本文所阐述的算法均经过实验与仿真的验证，并根据仿真和实验结果对算法进行了优化和改进。最后对算法将来可以发展与更深入研究的部分进行了展望和探讨。

关键词：智能小车，智能算法，调度

Abstract

The paper is focused on the control arithmetic of an auto guild vehicle(AGV) based on the embedded single chip. The function of the arithmetic includes signal processing, road surface identify ,decision-making ect. The paper begins with the analysis of the characteristic of the vehicle and the road. Based on the analysis, then it discusses the further design of the program.

All the programs which the paper discusses have been test by the real AGV system. Finally some of the advanced design is discussed.

Key words: auto guild vehicle, intelligent arithmetic

1 绪论

1.1 引言

近年来，随着经济的发展和社会的进步，道路的通行能力、交通的安全性、能源的损耗、环境污染等问题越来越突出。这些问题的解决引发了新的研究和应用的热点，比如自动车辆驾驶，通过计算机控制、人工智能和通信技术实现更好的通行能力和更安全的行驶。

繁重的驾驶工作和驾驶人员的疲劳是交通事故频发的重要原因。车辆在交通拥挤的市区行使驾驶人员必须完成大量的换档和踩离合器的工作，大约在每分钟完成 20~30 个手脚协调动作。随着经济的发展，车辆拥有量的增加，非职业驾驶人员的人数增多，导致交通事故频繁发生，交通事故已经成为现代社会的第一公害。交通问题已经成为全球范围令人困扰的严重问题，因此，如何提高交通安全性已经成为急需解决的社会性问题。道路偏离系统、疲劳检测系统、自动巡航控制等都可以大大减轻驾驶人员的驾驶工作，提高交通系统的安全性。

1.2 项目研究背景及意义

1.2.1 智能车辆

智能车辆（ intelligent vehicle）又叫轮式移动机器人,是一个集环境感知、规划决策、自动行驶等功能于一体的综合系统，他集中地运用了计算机、传感、信息、通讯、导航、人工智能及自动控制等技术，是典型的高新技术综合体[1]。智能车辆，首先来自军事上的需要。利用智能车辆代替人类进行侦查从而减少伤亡取得胜利，一直都是各国军事部门的研究重点。智能车辆在科学研究领域也有广泛的应用。2005 年 1 月，美国发射的“勇气”号和“机遇”号火星探测器，实质上都是装备先进的智能车辆。随着技术的不断发展，智能车辆技术在

民用方面也获得了广泛的应用，给人们的生活带来巨大的影响。例如智能车辆技术能提高道路网络的利用率、降低车辆的燃油消耗量，尤其在改进道路交通安全等方面提供了新的解决途径[2]。

1.2.2 智能算法

车辆之所以能实现智能行驶，自动驾驶，居于核心地位是控制算法。随着微控制器技术的发展，控制器的资源愈加丰富，功能日趋强大，为实现更高智能的控制算法提供了良好的平台。智能算法的目的是使小车能够有类人的一些思维特点，如学习，记忆，判断，决策等能力，对于不同的情况进行合理高效的处理。本项目所要研究的就是基于智能巡线小车的算法设计。使小车对道路的不同情况，智能的进行决策，以达到高效行驶的目的。

2.智能小车运动与赛道特点分析

2.1 智能小车简介

本论文所研究的智能小车，是基于嵌入式单片机技术和传感器技术的自动寻迹小车。其基本功能是跟踪地面的标志线，巡线行驶，能自动根据道路的变化拐弯，加速，减速，保证快速安全的运行。在本项目中我们采用中心带有 2.5mm 宽黑线的白板作为小车行驶的道路。

小车通过设在前端的 15 个红外传感器获取位置信息，通过安装在驱动轮（后轮）上的速度传感器获取速度信息[3]。位置信息和速度信息送入MCU进行处理和运算，然后由MCU输出信号控制舵机的偏转和电机的转速，以实现小车的转向和速度控制[4]。图 1 是小车的系统框架图

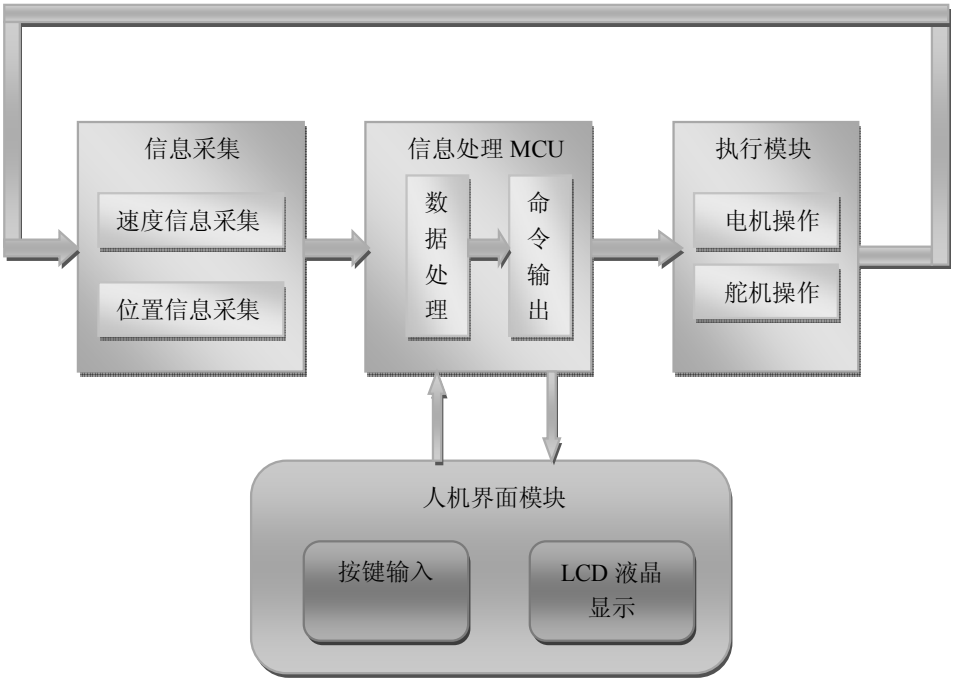

图 1 是小车的系统框架图

2.2 赛道特点分析

在智能车比赛的赛道中，包括如下几种路况：直道，圆弧弯道，连续 S 形道，十字交叉道，坡道。经过抽象，赛道可以视为由直道和弯道两种基本类型所组成。

小车在运行的过程中，路况的变化是扰动的主要来源。路况变化的越剧烈，扰动的幅值越大，对小车运行时的超调量也越大。路况变化的剧烈，一方面由赛道本身决定，另外一方面由小车运行的速度决定。当小车在高速运行的过程中进入弯道，相对于低速的情况，道路的变化扰动更加迅速，小车偏离黑线越严重；同时，小车的稳定时间也越长。前者影响稳定性，后者影响小车运行速度。为了兼顾稳定和快速，在小车控制的基本原则是位置控制结合速度控制，尽量提高直道速度，适当降低弯道速度。

3 传感器信号获取与处理

传感器信号进入单片机后，由于环境因素，容易产生干扰信号，造成小车行进不稳定。所以应该处理输入信号，以减小环境噪声的干扰。在小车在赛道上行进时，受到的干扰信号有两类，一类时偶然干扰信号，这类干扰可以通过多次对信号值取平均来去除。由于传感器的灵敏度限制与赛道本身的一些问题，也会出现另一类干扰，这类干扰信号与普通的信号同时存在，并且持续时间也与正常信号相当，前面的方法对于这种干扰的消除效果并不太好。所以这里结合实际情况，抽象出干扰信号的特点，形成一套规则，通过逻辑判断把不符合这套规则的信号剔除，达到消除干扰的目的。针对我们使用的赛道，在使用现有的传感器布局时，除了在起始线处，不可能出现三个以上传感同时检测到信号的情况，另外如果有两个传感器同时检测到信号，则一定是相邻的两个，否则就可以判断输入信号中存在干扰，调用相应函数进行处理。

4. 控制策略设计

4.1 控制策略概述

本项目中，被控对象有两个：舵机的偏转角和电机的转速。分别由反射式红外传感器和速度传感器获取，形成两个闭环回路。

根据第二章对于赛道的分析，小车的速度控制需要依靠经验进行调整，以满足在不同路况下的稳定运行；小车的位置需要根据黑线与车体的关系进行调整。因此在本项目中，对速度采取条件控制，对于舵机偏转采用基于偏差量的 PID 反馈控制。

为了使小车的控制更加精确，能对具体的路况。例如，大弯道，小弯道，连续 S 形弯道）作出更有针对性的调整，本项目通过专门的信号处理和识别模块来识别小车所处的状态，并根据经验条件对不同情况下的算法组合进行调配与组合，使得小车根据不同情况智能选择行驶策略的功能。

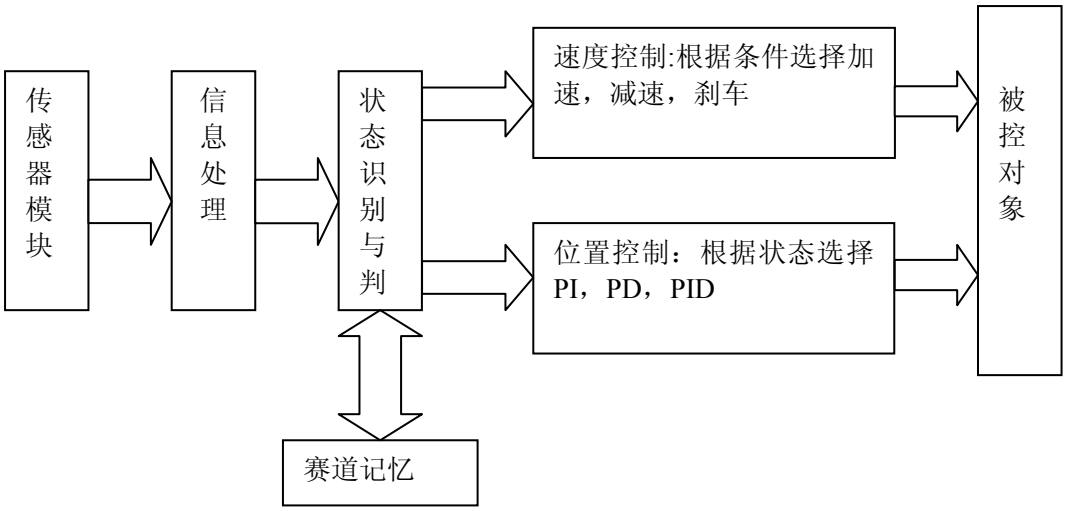

图 2 控制算法功能图

4.2 PID 控制算法

4.2.1 PID 算法原理

比例，积分，微分（PID）是建立在经典控制理论基础上的一种控制策略。PID 控制器作为最早实用化的控制器，已经有五十多年的历史，现在仍然是最广泛的工业控制器。PID 控制器最大的特点是简单易懂，使用中不需要精确的系统模型等先决条件，因而成为应用最广泛的控制器。

系统偏差信号为 e(t)=r(t)-y(t)。在 PID 调节作用下，控制器对误差信号 e(t)分别进行比例(P)，积分(I)，微分运算(D)，其结果的加权和构成系统的控制信号 u(t)，送给被控对象加以控制。

PID 控制器的数学描述为式 4-1

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{t}] \qquad \text{式 4-1}$$

式中，Kp 为比例系数，Ti 为积分时间常数，Td 为微分时间常数。

比例环节的主要作用是：Kp 的值增大时，系统的响应速度加快，闭环系统响应的幅值增加。当达到某个 Kp 值，系统将趋于不稳定。

当增加积分时间常数 Ti 的值时，系统超调量减小，而系统的响应速度将变慢。因此，积分环节的主要作用是消除系统的稳态误差，其作用的强弱取决于积分时间常数 Ti 的大小。

4.2.2 PID 算法改进

（1）降低微分误差

有许多求离散时间微分的方法。最简单的方法是

$$D(n) = K_D \frac{E(n) - E(n-1)}{\Delta t} \qquad \text{式 4-2}$$

实际上，这种一阶微分方程，对于噪声十分敏感。图 3 给出了一个带有噪声的 E（n）序列。通过该图我们可以看到，若在这种情况下使用公式 4-2 进行微分运算，会产生很大的误差。

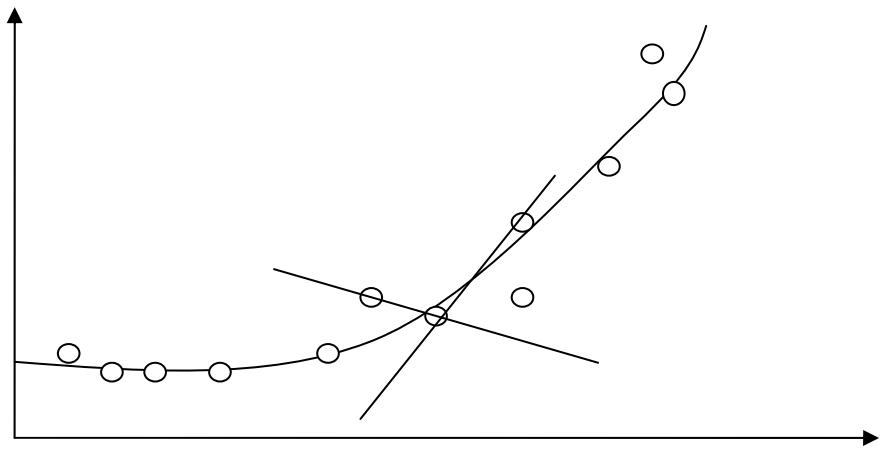

图 3 噪声对求离散微分的影响

为了减小这样的误差，可以通过计算两个不同时间跨度（甚至是多个不同时间跨度）的微分平均值来实现。

$$D(n) = K_D[\frac{1}{2}\frac{E(n) - E(n-3)}{3\Delta t} + \frac{1}{2}\frac{E(n-1) - E(n-2)}{\Delta t}] \qquad \text{式 4-3}$$

公式 4-3 可以化简为如下形式:

$$D(n)=K_D\frac{E(n)+3E(n-1)-3E(n-2)-E(n-3)}{6\Delta t} \qquad \text{式 4-4}$$

（2）合理调度积分

另外，在积分控制中，在开始或停止工作的瞬间，或者大幅度地给定量时，由于偏差较大，积分项的作用将会产生一个很大的超调，影响车辆调整的稳定，甚至使车辆偏出跑道。

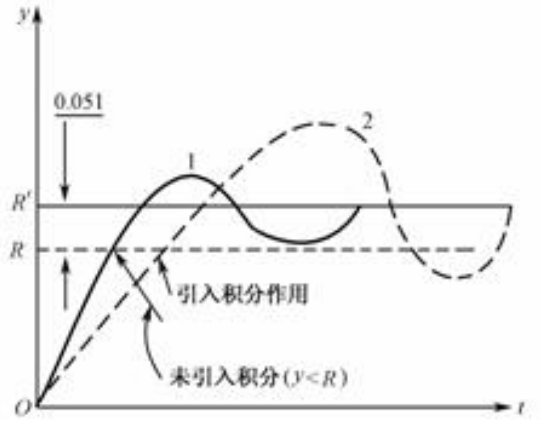

图 3 引入积分的超调效应

为此，可以采用积分分离手段，即在被控制量开始跟踪时，取消积分作用，直到被控制量接近新的给定值时，才可以在 PID 算式中，引入如下的算法逻辑功能。

$$u(k)=X_p e(k)+K_i T_i\sum_{i=0}^{k}e(i)+T_d[e(k)-e(k-1)] \qquad \text{式 4-4}$$

在式 4-4 中，Ki 为引入的逻辑系数。

4.2.3 PID 算法程序实现

PID算法在参数设置合理的情况下可以达到很好的控制效果。通过理论分析，加入积分环节可以消除稳态误差，加入微分可以提高响应速度。通过实际对小

车的试验中，积分环节效果十分明显，即便在积分系数I很小的时候，依然会造成舵机响应的缓慢，使得小车难以及时对赛道的变化作出响应；同时在比例控制下，小车在直道行驶过程中已经基本没有稳态误差存在，因此加入积分环节意义不大[5]。

在实际系统中，小车的 PID 参数是根据路况来进行动态调整的，尽量最大限度的扬长避短。

小车系统中，PID 控制器以比例（P）控制为主，对于积分（I）和微分（D）采取有条件的选用，具体操作方法如下：当小车进入弯道的时候，加入微分环节，以使舵机迅速反应，克服弯道扰动；当进入直道时，加入积分控制，使小车保持平稳运行，但积分系数（I）取值微小，避免由于反应过慢而引起的小车运行不稳。

图 4 中的振荡分析模块对于控制过程至关重要，既是采用 PI 或者 PD 控制的依据，同时也是系统对于采用其他辅助控制算法的依据。该模块通过计算近一段时间（50 次传感器采样）内偏差量的均方差来得出小车运行路线同实际路线的离散程度即判断小车是否运行稳定，具体细节将在 5.1 节中进行介绍。

根据振荡分析模块的输出结果，程序会判断是采用 PI 控制还是 PD 控制或者单纯的 P 控制，使得小车控制更具针对性，效果更好。

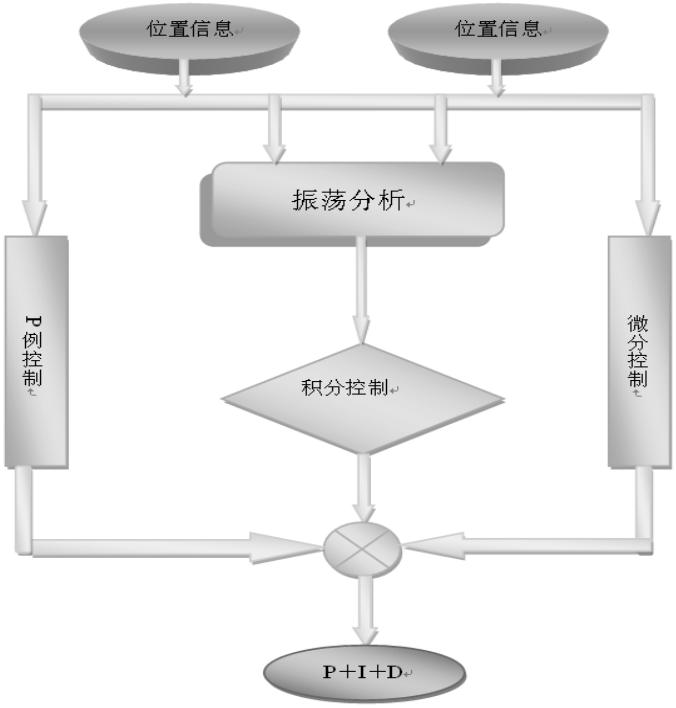

图 4　PID 算法示意图

4.3 速度控制算法

小车运行中另外一个关键的物理量是速度。由于小车在运行过程中有较大的时延和惯性，从控制信号输出到舵机响应到位有一段时间延迟，同时小车运行本身的惯性也使得车体难以按照理想状况立即调整完毕。时延加惯性的系统特性对于小车通过弯道时，影响尤其明显。

进入弯道后，速度过快容易冲出赛道；而速度过慢可能会导致动力不足卡停在弯道处。同时小车进入直道时，需要尽可能提升速度，并保持在一定范围内，避免因为速度过快影响直道稳定性。如果没有速度闭环，仅仅依靠对电机的开环控制这是无法实现的。在小车实际运行中，电机接受到控制信号后，从初始转速提升到期望转速，需要经历 3－5 秒的加速过程，延迟是相当大的。因此，除了使用PID对于位置偏差量进行控制外，本项目对小车速度也进行闭环控制[6][7]。

在速度控制中，如何获取小车当前速度，以及如何对小车进行减速是两个关键问题。

首先研究一下小车的减速方法，以下对三种可能的方案进行了比较。

一．通过机械接触小车轮或轴实现减速。第一种方案使用的范围很广，但是机械加工复杂，需要对车体作改造，另外可靠性与可控性不太好，不适合单片机的控制。

二．使电机短时间停机。第二种方案可控性强，适合于轮轴自锁的电机，如减速电机与步进电机。

三．使电机瞬时反转来快速降低小车速度。第三种方案与第二种类似，适合于直流电机，在一般条件下减速效率优于前者。

本项目中由于采用普通直流电机作为驱动源，综合成本和可靠性因素，采用第三种方案即电机反转的方式进行快速刹车。

为了能对速度进行检测，引入了速度传感器。该传感器采用 ST188 红外对管作为传感器件，装配在小车驱动轮，后轮的上方。同时在后轮贴上黑白相间的

标识。当小车行驶时，红外传感器通过检测黑白标识，来获取运行速度。

以下程序示意出了本项目采用的速度控制方案。

```
if( BrakeTime != 0)              // BrakeTime 表示减速时间
{
    BrakeTime--;                 //冷却时间处理
    减速功能子函数
}
else
{
    if(小车偏离赛道比较严重 and 小车刹车就绪)
        启动减速功能，给定刹车强度，减速冷却开始
    else
        根据小车状态与当前速度决定小车速度
}
```

程序设计中通过调整减速时间（BreakTime）以及减速功能子函数里设定的速度参数来控制减速的程度。通过实验，在良好识别弯道的前提下，可以达到迅速降低电机转速满足过弯要求。

5 小车智能控制

5. 1 振荡检测

本项目中小车对于行驶策略的选择都是依据小车对振荡情况的判断。该振荡检测模块就是实现判别小车振荡剧烈程度（即稳定性系数）的功能。

对于振荡检测，通过分析小车最近一段时间的振荡状况来进行判断。该算法使用一个链表来存储位置偏差信息，记录最近 50 次的偏差值，在第 51 次传感器信息到来时，把最开始第一次的数据整个存储空间中弹出，以保证链表中始终保存最近 50 次的位置信息。将这 50 次偏差的均方差和标准差，作为稳定性系数。每次采样结束，程序进行一次链表的操作，进行一次均方差和标准差的

计算，并对稳定系数进行判断，是否进入振荡状态。

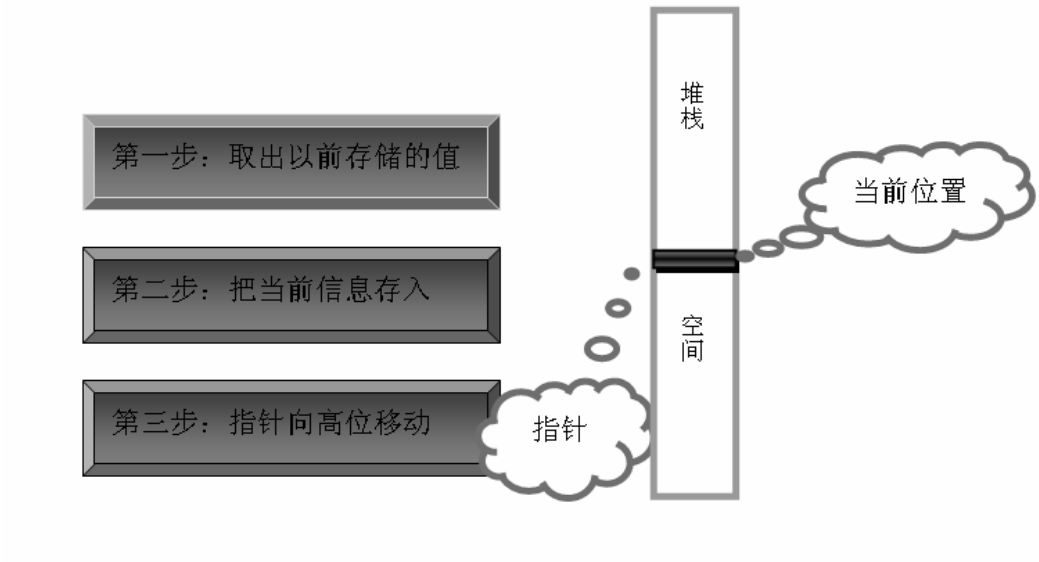

图 5 振荡检测的信息存储示意图

5.2 赛道记忆与回放

在正式比赛中，小车在赛道中要连续运行数圈，为了获取尽可能多的信息量，最大限度提高单圈最好成绩，可以采用记忆算法，即完成前面完整一圈的过程中，将赛道的信息尽记录下来，为后面的几圈的运行提供提前参考，以便提前做出判断，使运行更加稳定精确[9]。

该方法只适合于小车在同一个赛道反复行驶多次的情况，并且赛道上有明可供识别的标志，以判断赛道起始位置。具体工作程序如下：

小车在首圈运行中，记录特定路况的数量，出现时间和位置（例如弯道数量）[8]。如果每 1/25 秒存储用一次，每个指令为 2 字节，则数据量为 50 字节/秒。若直接存储，5 分钟的指令需要 15KB的内存空间。为了尽量减少冗余，采用简易微损压缩算法，将同一时间段内相似的指令合并，加一个项控制项，指示所包含的项数，对数据进行压缩。据估计这种算法的压缩比在 10－80％范围内，实际操作中能达到 6：1 以上的压缩比，即 5 分钟的指令大约只要 2～3KB

的存储空间。当然，具体压缩效率由指令变化频繁程度而有所不同。

当小车首圈运行结束，通过判断赛道的起始线来确定是否调用记忆的路况。则在以后数圈内，小车运行到赛道的特定部位时，可以不依赖传感器，而根据经验来进行更有效的控制。本项目中，小车通过计算进入直道的时间，可以在弯道到来之前进行减速；在连续小幅弯道到来时时，可以直接从道路中心穿越而不必完成连续拐弯的动作。

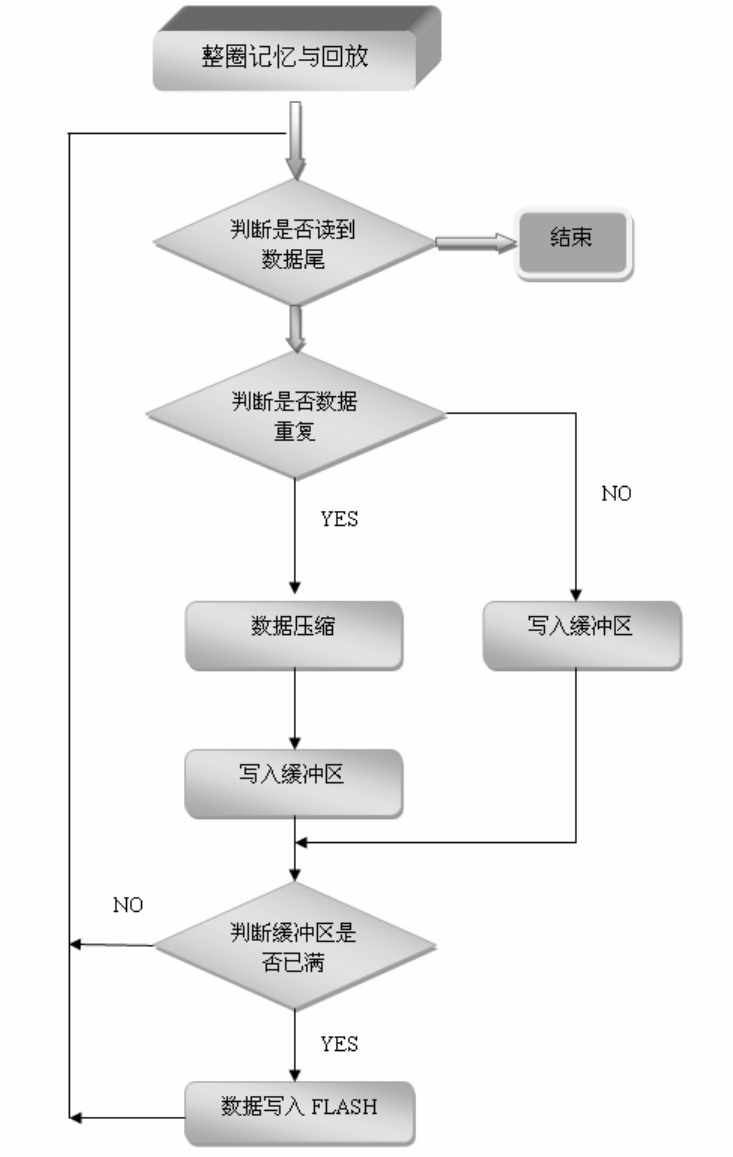

图 6 赛道记忆流程图

以下为信息采集、数据压缩与据存储的示意代码：

```
Instr          //存储空间首地址
InstrAddr      //存储空间地址指针

int ReadData()                          //读取存储的控制数据信息
{
    int data, counter;                  //定义临时变量
    data=Instr[InstrAddr];              //读取数据
    if(data&0xf0 = = 0)                 //判断是否读到数据尾
    {
        StopCar();
    }
    else if(data&0xf0 == 1)             //判断是否为控制信息
    {
        InstrAddr=0;
        counter=1;
    }
    else                                //否则为数据信息
    {
        counter=data&0x0f;
        InstrAddr=InstrAddr+2;          //将存储地址指针往高地址移动
    }
    counter--;                          //以下代码是对地址进行操作
    if(counter= =0)
    {
        counter=1;
        InstrAddr=InstrAddr+2;
    }
    //获取方向和速度数据
```

```
    //返回读到的控制信息
}
```

6．模块的调度与配合

该模块在整个系统中起到决策算法的作用，对不同的控制算法进行合理安排与调度。在前面介绍的几个功能函数中，涉及到，P 控制模式，PI 和 PD 控制模式，反转减速，直到加速，弯道运行，直到加速，直到限速，弯前减速，快速刹车。调度模块依据路况为每种算法分配优先权，优先权通过经验测试来进行确定。

以减速过程为例：反转减速时函数有启动的寿命，反转减速函数在运行时也会抢占电机正常调速函数的运行，当反转减速过程完成以后，还要经历冷却时间才能相应下一次减速，冷却控制函数的运行从上一次减速结束时开始，到冷却时间结束后中止，在此函数运行过程中，反转减速函数的控制权被剥夺，而正常调速函数取得控制权。

参照uC/OS-II的核心代码[10]，给出适合于单片机综合处理的一种调度方案

```
unsigned char priority;                    //优先级调度控制单元 8 位有效
unsigned char FuncReady;                   //函数就绪表，为1表示函数就
绪

char const UnmapTable[] = {                //优先级查找表
    0, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,     // 0x00 to 0x0F
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,     // 0x10 to 0x1F
    …… …… …… …… …… ……                    …… …… ……
    …… …… …… …… …… ……                    …… …… ……
    5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,     // 0xE0 to 0xEF
    4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0      // 0xF0 to 0xFF
```

```
};

void Scheduler(void)                                    //功能函数调度器
{
    switch(UnmapTable[priority] & FuncReady)    //优先级与就绪表取与
    {
        case 0:                                         //表示没有功能函数就绪
            break;
        case 1:                                         //最高优先级
            //功能函数
            break;
        case 2:
            //功能函数
            break;
        …… …… ……
        case 7:                                         //最低优先级
            break;
    }
}
```

参考文献

[1] 董克,刘明锐.机器人与人工智能发展.上海，上海交通大学出版社，2004

[2] 陈懂，刘瑢，金世俊.智能小车的多传感器数据融合.现代电子技术，2005，第六期

[3] 张宏希.红外接收组件原理及其应用.石河子大学学报，2005.12

[4] MC9S12DB128B Device User Guide V1.09.Motorola,Inc

[5] Jonathan W.Vavlano 著 李曦等译.嵌入式微计算机系统 实时接口技术[M].北京.机械工业出版社,2003

[6] PWM_8B8C Block User Guilde V01.16.Motorola Inc.Document number:S12PWM8B8CV1/D

[7] 王宜怀 刘晓升.嵌入式应用技术基础教程[M] 北京清华大学出版社 2005

[8] 邵贝贝.单片机嵌入式应用的在线开发方法[M].北京.清华大学出版社,2004

[9] 吕广明，孙立宁.自动引导车轨迹偏差的智能控制.哈尔滨工业大学学报.2002 年 35（12）:1466-1467

[10] Jean J.Labrosse 著 邵贝贝等译 .嵌入式实时操作系统 uC/ OS-II（第二版）[M].北京.北京航空航天大学出版社,2003.5

附录 B：程序清单

小车控制程序清单

```
/*
*********************************************************************************
*
* 工程名称：SmartCar
* 功能描述：结合飞思卡尔16位单片机MC9S12DG128B完成小车自动寻迹，沿黑线行驶功能
* IDE环境： Metrowerks CodeWarrior 4.1
* 组成文件：
*          main.c
*          SmartCar.c/PID.c/LCD1620.c/Test.c
* 说明： 本版本为智能小车程序早期版本，还有待更进一步完善
* 日期：2006-5-6
*                                          (c) Copyright 2006,Zhao Cheng
*                                                All Rights Reserved
*
*
* By     : Zhao Cheng
*********************************************************************************/

/*
*********************************************************************************
**                                              main.c
*
*                                          (c) Copyright 2006,Zhao Cheng
*                                                All Rights Reserved
*
* By     : Zhao Cheng
*********************************************************************************/

#include <hidef.h>                  /* common defines and macros                */
#include <mc9s12dg128.h>            /* derivative information                   */

#pragma LINK_INFO DERIVATIVE "mc9s12dg128b"

#define HIGHSPEED 11500             /* 速度参量，此处未使用测速模块             */
#define LOWSPEED0 12500             /* 0-24000 数值越大，速度越慢               */
#define LOWSPEED1 12000             /* used in CarMain()                        */
#define STABMAX 50
#define StopCar()    PORTK |= 0x80  /* stop the motor                           */
#define StartCar()   PORTK |= 0x04  /* start the motor                          */
#define BrakeCar()   PORTK &= 0xfb  /* slow the speed of the SmartCar           */


unsigned int SYSCLOCK=0;            /* update in INT_Timer0()                   */
/*
*********************************************************************************
*
*                                          FUNCTION PROTOTYPES
*********************************************************************************/
```

```
/*                    write in "SmartCar.c"                                  */
void Init_INT_RTI(void);              /* initiate Real Time Interrupt        */
void Init_INT_Timer(void);            /* INT_Timer0 initiate                 */
void Init_PWMout(void);               /* initiate   PWM output               */
void PWMout(int, int);                /* output PWM                          */

/*                    write in "PID.c"                                       */
void Init_PID(void);                  /* initiate PID parameter              */

int CalculateP(void);                 /* calculate parameter P               */
float CalculatePID(void);             /* calculate PID                       */
int SignalProcess(unsigned char);     /* Process the signal from the sensors */

/*                    write in "Test.c"                                      */
void IOtest(void);                    /* Test I/O                            */
void PWMtest(void);                   /* Test PWM output                     */
int SignalTest(void);                 /* Test the sensors                    */


/*                    write in local file                                    */
void Init(void);                      /* initiate parameter                  */
void ProtectMoto(void);               /* the function protecting the Motor */
void CarMain(void);                   /* SmartCar main function              */
```

```
/*
*********************************************************************************
*
*                                                主程序
*
* 程序描述: 完成智能小车系统的初始化，通过按键可选择工作模式，有I/O测试，PWM 输出测试
*             传感器测试，以及小车正常工作模式
*
* 硬件连接：PORTB 接传感器
*             PWM 输出口 （1）接舵机 （2）接电机驱动芯片MC33886
*
* 说明： 无
*********************************************************************************/

void main(void)
{
    Init();

    DDRB = 0x00;

    switch(PORTB)
    {
        case 0x80:
            IOtest();
            break;
        case 0x40:
            PWMtest();
            break;
        case 0x20:
            SignalTest();
            break;
        default:
            DDRA = 0x00;
            DDRB = 0xff;
            DDRK = 0xff;
            PORTB = 0xff;
            CarMain();
            EnableInterrupts;
            for(;;);
            break;
    }
}
/*
*********************************************************************************
*
*                                小车寻迹行驶函数
*
* 程序描述: 通过传感器采集数据，并对其进行处理，通过PID算法得出小车稳定行驶所需的参数，进而调用PWM输出函数
*             控制舵机与电机的工作
*
* 注意： 这个函数调用了 SignalProcess(unsigned char)，BrakeCar()，PWMout(Direction, Velocity)
```

```
*
* 说明：无
*******************************************************************************/

void CarMain(void)
{
    static int Direction=0, Velocity;
    static unsigned char signal;
    static unsigned int BrakeTime = 0, BrakeControl = 0;
    static unsigned int Stability=0, Stab[STABMAX], PStab=0, StabAver;
    int i;

    signal = PORTA;
    PORTB = ~signal;

    Direction = SignalProcess( signal );

    /*    稳定性系数的计算                                                    */
    Stability -= Stab[PStab];
    Stab[PStab] = (unsigned int)Direction/100;
    Stability += Stab[PStab];
    PStab++;
    if(PStab >= STABMAX)        PStab=0;

    StabAver = 0;
    for(i=0;i<STABMAX;i++)
    {
        if(Stability > Stab[i])
            StabAver += Stability - Stab[i];
         else
            StabAver += Stab[i] - Stability;
    }

    if( BrakeTime != 0)
    {
        BrakeTime--;
        BrakeCar();
    }
    else
    {
        StartCar();
        if(BrakeControl>0)
            BrakeControl--;

        if(Direction < -4000 || Direction > 4000 )
        {
        Velocity = LOWSPEED0;
        if(BrakeControl == 0 && StabAver/STABMAX<22)
        {
            BrakeTime = 20;
            BrakeControl = 120;
        }
        }
    else
    {
```

```
                if(Direction < -2500 || Direction > 2500 )
                    Velocity = LOWSPEED1;
                else
                    Velocity = HIGHSPEED;
        }
        }
        PWMout(Direction, Velocity);
}
```

```
/*
*********************************************************************************
*
*                                          系统初始化函数
*
* 程序描述: 初始化了系统时钟，FLASH 和 EEPRO的工作频率，PWM输出口，定时器，以及PID算法中的有关参数
*
* 注意： 这个函数调用了 Init_PWMout()   nit_INT_Timer()   nit_PID()
*
* 说明： 无
*********************************************************************************/
void Init(void)
{

    REFDV=0x01;                     /*   initiate PLL clock                    */
    SYNR =0x02;                     /*    system clock    24M
    */
    while (!(CRGFLG & 0x08)){}      /*   wait untill steady                    */
    CLKSEL=0x80;                    /*   选定所相环时钟                        */

    FCLKDIV=0x49;                   /*   使FLASH 和 EEPROM                     */
                                    /*   的擦除操作工作频率在200HZ左右         */
    ECLKDIV=0x49;

    Init_PWMout();                  /*   01:50Hz   45:1kHz                     */

    Init_INT_Timer();               /*   initiate ETC(Enhanced Capture Clock)  */

    Init_PID();                     /*   initiate PID caculating process       */

    DDRK |= 0x80;                   /*   Start Car -- stop car                 */
    PORTK &= 0x7F;
}
```

```
/*
*********************************************************************************
*
*                                          SmartCar.c
*
*                                  (c) Copyright 2006,Zhao Cheng
*                                          All Rights Reserved
*
* By      : Zhao Cheng
* Data : 2006_5_6
* Note : Don't change this file if possible.
*********************************************************************************/

#include <hidef.h>
#include <mc9s12dg128.h>

extern SYSCLOCK;                    /*  引用全局变量,系统时钟                      */
void CarMain(void);

/*
*********************************************************************************
*
*                                          PWM初始化函数
*
*********************************************************************************/

void Init_PWMout(void)
{
   PWME = 0x22;                     /*01:50Hz    45:1kHz                         */
   PWMPOL = 0x22;
   PWMCTL = 0x50;

   PWMCLK = 0x02;
   PWMSCLA = 4;
}
```

```
/*
*********************************************************************************
*
*                                    PWM输出函数
* 程序描述：输入参数为方向，速度
*           方向：－45～45
*           速度：0～24000
*********************************************************************************/

void PWMout(int Direction, int Velocity)
{
  Direction = Direction/3 + 4500;
  if(Direction<3000) Direction=3000;
  if(Direction>6000) Direction=6000;
  PWMPER01 = 60000;              /*   Center 1500ms*3                          */
  PWMDTY01 = Direction+93;       /*   设置舵机角度                             */

  if(Velocity>24000) Velocity=24000;
  PWMPER45 = 24000;              /*   1kHz ( <10kHz )                          */
  PWMDTY45 = Velocity;           /*   设置电机速度                             */
}

/*   initiate Real Time Interrupt    1.0                */
void Init_INT_RTI(void)
{
  RTICTL = 0x74;
  CRGINT |=0x80;
}

/*   Real Time Interrupt       1.0                      */
interrupt void INT_RTI(void)
{
    CRGFLG |= 0x80;              /*   clear the interrrupt flag                */

}

/*                  INT_Timer0 initiate 1.0          */
void Init_INT_Timer(void)
{
    TSCR2 =0x07;                 /*   128Hz at 16M bus clok                    */
                                 /*   128Hz * 2/3    at 24m bus clock          */
                                 /*   in fact it is a little more than it.  */
    TIOS |=0x01;                 /*    I/O select                              */
    TIE  |=0x01;                 /*    Interrupt Enable                        */

    TSCR1|=0x80;                 /*    TSCR1_TEN=1 //Timer Enable              */
}

/*                  INT_Timer0        1.0                                      */
interrupt void INT_Timer0(void)
{
  SYSCLOCK++;
  CarMain();
```

```
    TC0 = TCNT + 1874;              /*   1875-1 :100Hz                          */
                                    /*   F = Fosc / (TC*128)                    */
    TFLG1 |=0x01;                   /*   clear interrupt flag                   */
}

/*    not finished EEPROM                                                       */
void EEPROM(void)
{
      ECLKDIV = 0x4F;
      while(!(ECLKDIV&0x80))        /*   wheather                               */
      {}
      while(!(ESTAT&0x80))          /*   wheather the command buffer is empty   */
      {}
      while(!(EPROT&0x80))          /*   wheather the eeprom is enabled to      */
      {}

      ECMD = 0x41;
      ESTAT |= 0x80;
      while(!(ESTAT&0x80))          /*   wheather the command buffer is empty   */
      {}
}
```

```
/*
*********************************************************************************
*
*                                                PID.c
* Description: This file includes some basic calculation function of PID
*                                            (c) Copyright 2006,Zhao Cheng
*                                                All Rights Reserved
*
* By      : Zhao Cheng
* Data : 2006_5_6
* Note : Don't change this file if possible.
*********************************************************************************/

#include <mc9s12dg128.h>            /* derivative information */

/*
*********************************************************************************
*
*                                            宏定义
*********************************************************************************/

#define STABMAX 50
#define SENSORNUM 8
#define SAMPLETIMES 5

/*
*********************************************************************************
*
*                                          FUNCTION PROTOTYPES
*********************************************************************************/

int CalculateP(void);
float CalculatePID(void);

/*********************************    PID控制程序    *******************************/
struct CARSTATE
{
      int E0;
      int E1;
      int E2;
      int E3;
      float Integral;
}CarState;


/*
*********************************************************************************
*
*                                          初始化PID参数
*********************************************************************************/

void Init_PID()
{
      CarState.E0 = 0;
      CarState.E1 = 0;
```

```
    CarState.E2 = 0;
    CarState.E3 = 0;
    CarState.Integral = 0;
}

/*
*********************************************************************************************
*                                                信号处理函数
*
* 程序描述: 对传感器采集过来的数据进行处理，得到一些基本的计算参数
*
* 说明： 无
*********************************************************************************************/

int SignalProcess( unsigned int signal )
{
    const int BitValue[8] = {43,26,12,6,-6,-12,-26,-43};   //MAX:28
    int i,CurrPoint=0,LastPoint=0,BitNum=0;
    unsigned char SignalBit[8];


    for(i=0;i<8;i++)
    {
        SignalBit[i] = signal & 0x0001;
        BitNum += SignalBit[i];
        signal >>= 1;
    }

    switch(BitNum)
    {
        case 1:
            for(i=0;i<8;i++)
                if(SignalBit[i] != 0)
                    CurrPoint += BitValue[i];
            CarState.E0 = CurrPoint;
            break;

        case 2:
            for(i=0;i<8;i++)
                if(SignalBit[i] != 0)
                    CurrPoint += BitValue[i];
            CurrPoint >>= 1;
            CarState.E0 = CurrPoint;
            break;

        default:
            CarState.E0 = CarState.E1;
            break;
    }

    return CalculateP()*100;
}

/*
*********************************************************************************************
```

```
*
*                                                    PID计算函数
*
* 程序描述: 计算P参数
*
* 说明： 无
*********************************************************************************/
int CalculateP(void)
{
    CarState.E1 = CarState.E0;
    return((int)CarState.E0);
}
```

```
/*
*********************************************************************************************
*
*                                                PID计算函数
*
* 程序描述: 对传感器采集过来的数据进行处理，得到一些基本的计算参数
*
* 说明： 无
*********************************************************************************************/

float CalculatePID(void)
{
    float P, I = 0, D;
    /*          parameter const                                                 */
    float Kp = 1.0, Ki = -0.0002, Kd = -0.0002;
    /*          P parameter                                                     */
    P = CarState.E0 * Kp;
    /*          I parameter                                                     */
    if(P+I<2)
    {
     CarState.Integral += Ki * CarState.E0;
     I = CarState.Integral;
    }
    /*          D parameter                                                     */
    D = Kd * ( CarState.E0 + 3*CarState.E1 - 3*CarState.E2 - CarState.E3 )/6.0;

    CarState.E3 = CarState.E2;
    CarState.E2 = CarState.E1;
    CarState.E1 = CarState.E0;

    return (P+I+D);
}
```

```
/*
*********************************************************************************
*
*                                                Test.c
* Description: This file includes I/ O function for test, the PWM outputs function for test, function
*                       testing sensors.

*                                                (c) Copyright 2006,Zhao Cheng
*                                                All Rights Reserved
*
* By      : Zhao Cheng
* Note : Don't change this file if possible.
*********************************************************************************/

#include <hidef.h>
#include <mc9s12dg128.h>

#define HIGHSPEED 8000
#define LOWSPEED 11000                  /*    速度变量，0-24000  数值越大，速度越慢          */
void PWMout(int, int);
                                        /*    24000-20000                                   */
void IOtest(void)
{
      static unsigned char i=0,j=0x01,k;

      DDRB = DDRA = 0xFF;
      PORTB = 0xf0;

      for(;;)
      {
            k=(~j)&0x7f;
            PORTA = PORTB = k;
            while (TCNT != 0x0000);
            while (TCNT == 0x0000)
            {
                  if(i>9)
                  {
                        j=j<<1;
                        i=0;
                  }
                  i++;
            }
            if(j>=0x80)
                  j=0x01;
      }
}


void PWMtest(void)
{
      int counter=-4500;

      DDRB = 0xff;
      PORTB = 0xff;
```

```
    TSCR1 = 0x80;                   /*    enable timer TCNT                    */
    TSCR2 = 0x00;                   /*    TCNT prescaler setup                 */

    for(;;)
    {
        while (TCNT != 0x0000);
        while (TCNT == 0x0000);
            counter=counter+30;
        if(counter >= 3000)
        {

            counter = 0;
            PWMout(4500, LOWSPEED);
        }
        if(counter == 1500)
        {
            PWMout(-4500, LOWSPEED);
        }
        PORTB = (char)(counter/100);
    }
}


void SignalTest(void)
{
    unsigned char signal;
    int Direction, Velocity;

    Direction = 0;
    Velocity = LOWSPEED;
    DDRA = 0x00;
    DDRB = 0xff;
    signal = PORTA;
    PORTB = ~signal;
    switch(signal)
    {
        case 0x08:                  /*    0001 1000                            */
        case 0x10:
            Direction = 800;
            Velocity = HIGHSPEED;
            break;
        case 0x04:                  /*    0010 0100                            */
        case 0x20:
            Direction = 1500;
            Velocity = HIGHSPEED;
            break;
        case 0x02:                  /*    0100 0010                            */
        case 0x40:
            Direction = 2800;
            Velocity = HIGHSPEED;
            break;
        case 0x01:                  /*    1000 0001                            */
        case 0x80:
            Direction = 4000;
            Velocity = LOWSPEED;
```

```
            break;
        case 0x3c:              /*  0011 1100       over start line                         */
        case 0xff:              /*  1111 1111       over crossing line                      */
        case 0x00:              /*  0000 0000       go straight     not need changed state  */
        default:
            break;
    }
    if(signal > 0x0f)
        Direction = -Direction;

    PWMout(Direction, LOWSPEED);
}
```

```
/*
*****************************************************************************
*
*                                                    LCD1620.c
* ICC-AVR application builder : 2006-1-8 21:43:48
* Target : M8
* Crystal: 4.0000Mhz
*
* Note : Don't change this file if possible.
*****************************************************************************/

#define CMD_CLEAR 0x01
#define CMD_RESET 0x02


#include <iom8v.h>
#include <macros.h>

#define LCD_DATA          0xff
#define LCD_EN            0x01                  //PORTC 0
#define LCD_RS            0x02                  //PORTC 1
#define LCD_RW            0x04                  //PORTC 2
#define LCD_DATAPORT    PORTB
#define LCD_ENPORT      PORTA
#define LCD_RSPORT      PORTA
#define LCD_RWPORT      PORTA

void lcd_init(void);
void lcd_write_cmd(unsigned cmd,unsigned data);
void lcd_setxy(unsigned char x,unsigned char y);
void lcd_write_string(unsigned char X,unsigned char Y,unsigned char *str);
void delay_nus(unsigned int n);
void delay_nms(unsigned int n);

void lcd_init(void)
{
    DDRB |= LCD_DATA;
    DDRA |= LCD_EN | LCD_RS | LCD_RW;
    LCD_RWPORT&=~LCD_RW;
    LCD_DATAPORT=0x30;                  //控制字规则：5：8bit,4:16x2,3:5x7
    LCD_ENPORT|=LCD_EN;
    delay_nus(1);
    LCD_ENPORT&=~LCD_EN;

    delay_nus(40);
    lcd_write_cmd(0,0x38);              //8bit test
    lcd_write_cmd(0,0x0c);              //显示开
    lcd_write_cmd(0,0x01);              //显示清屏
    lcd_write_cmd(0,0x06);              //显示光标移动设置
}

void lcd_write_cmd(unsigned cmd,unsigned data)
{
    if(cmd==0)
```

```
            LCD_RSPORT&=~LCD_RS;
        else
            LCD_RSPORT|=LCD_RS;
        LCD_DATAPORT&=0x00;
        LCD_DATAPORT=data;
        LCD_ENPORT|=LCD_EN;
        delay_nus(10);
        LCD_ENPORT&=~LCD_EN;
        delay_nus(10);
}

void lcd_setxy(unsigned char x,unsigned char y)
{
        unsigned char addr;
        if(y==0)
            addr=x+0x80;
        else
            addr=x+0xc0;
        lcd_write_cmd(0,addr);
}

void lcd_write_string(unsigned char X,unsigned char Y,unsigned char *str)
{
        lcd_setxy(X,Y);
        while(*str)
        {
            lcd_write_cmd(1,*str);
            str++;
        }
}

void delay_1us(void)                        //1us延时函数
{
        asm("nop");
}
void delay_nus(unsigned int n)              //N us延时函数
{
        unsigned int i=0;
        for (i=0;i<n;i++)
        delay_1us();
}
void delay_1ms(void)                        //1ms延时函数
{
        unsigned int i;
        for (i=0;i<1140;i++);
}
void delay_nms(unsigned int n)              //N ms延时函数
{
        unsigned int i=0;
        for (i=0;i<n;i++)
        delay_1ms();
}
//call this routine to initialize all peripherals
void main(void)
```

```
{
    lcd_init();
    while(1)
    {
        lcd_write_cmd(0,0x01);          //清屏
        delay_nms(2);
        lcd_write_string(0,0,"happy new year");
        delay_nms(100);
        lcd_write_string(0,1,"LCD successful!");
        delay_nms(100);
    }
}
/*******************************          程序结束     *******************************/
```

附录 C：红外传感器参数说明

该红外传感器（ST188）的特点是采用高发射功率红外光电二极管和高灵敏度光电晶体管组成，采用非接触式检测方式。ST188 的检测距离很小，一般为 4--15mm，因为小于 4mm 是它检测的盲区，而大于 15mm 则容易受到干扰，而且再大就检测不出来了。ST168 传感器很容易受到外界红外线信号的干扰，因此我们把它安装在距离检测物表面 10~12mm。ST188 传感器的外部检测电路很简单。 R5 限制发射二极管的电流。发射管的电流大则发射功率大，但不能超过它的极限电流，它的极限输入正向电流为 50mA。在输出端加一个非门整形，使输出满足 TTL 逻辑电平，能够直接与单片机连接。可以看出，若被检测物表面为白色，发射光全部反射，光电三极管导通，经反相后输出

表 C－1 极限参数（Ta=25℃）

项目		符号	数值	单位
输入	正向电流	IF	50	mA
	反向电压	Vr	6	V
	耗散功率	P	75	mW
输出	集-射电压	Vceo	25	V
	射-集电压	Veco	6	V
	集电极功耗	Pc	50	mW
工作温度		Topr	-20∽65	℃
储存温度		Tstg	-30∽75	℃

表 C－2 光电特性（Ta=25℃）

项　目		符号	测试条件	最小	典型	最大	单位
输入	正向压降	V_F	I_F=20mA	-	1.25	1.5	V
	反向电流	IR	V_R=3V	-	-	10	μA
输出	集电极暗电流	Iceo	Vce=20V	-	-	1	μA
	集电极亮电流	IL	Vce=15V I_F=8 mA	0.40	-	-	mA
	饱和压降	VCE	I_F=8mA Ic=0.5 mA	-	-	0.4	V
传输特性	响应时间	Tr	I_F=20mA Vce=10V IRc=100Ω	-	5	-	μs
		Tf		-	5	-	μs

注：集电极亮电流 IL、饱和压降 VCE、响应时间是在红外光电传感器前端面与亮检测面距离 4mm 处测得，其数值受亮检测面的表面光洁度及平整度影响。

附录 D：配件清单

飞思卡尔智能小车元器件清单

名称	型号	数量	单价	总价
单孔通用板	20cm×35cm	1 块		
插针	40×2*	4 排		
	40	1 排		
插座	40	2 个		
排线	16 线	1.0 米		
排线接插套件	同上	3 套		
红外对管	ST168	2 对		
	ST188	12 对		
	普通对管	5 对		
运算放大器	LM324	2 个		
精密电位器	102	10 个		
	202	4 个		
稳压管	LM7805	2 个		
	LM7806	2 个		
可调稳压管	LM317	1 个		
坦电容		4 个		
单孔通用板	20cm×35cm	1 块		
插针	40×2*	4 排		
	40	1 排		
插座	40	2 个		
排线	16 线	1.0 米		
排线接插套件	同上	3 套		
红外对管	ST168	2 对		
	ST188	2 对		
	普通对管	5 对		
运算放大器	LM324	2 个		
精密电位器	102	10 个		
	202	4 个		
稳压管	LM7805	2 个		
	LM7806	2 个		
可调稳压管	LM317	1 个		
坦电容		4 个		
单孔通用板	20cm×35cm	1 块		
插针	40×2*	4 排		
	40	1 排		