圆梦小车 Step by Step 之五

—— 让小车在“路”上走

前一篇讲述了如何使小车“**走路**”，本篇教小车在如何在“**路**”上行走！

在多数情况下，车是在路上跑的，我们的圆梦小车也不例外，只是它能认识的“路”与现实世界不同，因为它的眼睛与人不同，只能分辨颜色的深浅，所以它的路只能是在浅底色上的深色道路或者反之，我们通常称之为“轨迹”。

为了能辨别出轨迹，必须给小车配上能检测出深、浅颜色的“眼睛”—— 轨迹传感器，首先分析一下圆梦小车的轨迹传感器是如何构成的，又是如何工作的。

一、识别“路”的眼睛 —— 轨迹传感器

圆梦小车所用的轨迹传感器由反射式光电采样器、信号缓冲、背景光消除电路组成，将处理后红外光反射强弱模拟信号送入单片机，由单片机的 A/D 转换为数字信号后，再由软件来判断是否在轨迹上。

之所以没有直接使用比较器得出逻辑值，一是为了增加灵活性，可以通过软件方便的改变判断阈值，二是可以增加用途，如作为地面灰度检测。同时也增加学习的内容，如 A/D 的使用、逻辑值的转换，还可以用软件构成“施密特触发器”，增加采样的可靠性。

首先分析信号的预处理部分。

因为反射式光电采样最大的干扰就是环境光的变化，小车在行走时会有时顺光、有时逆光，虽可以通过遮光方式来改善之，但是如果能在电路上加以处理，会更加可靠。所以在设计时考虑了这个需求。

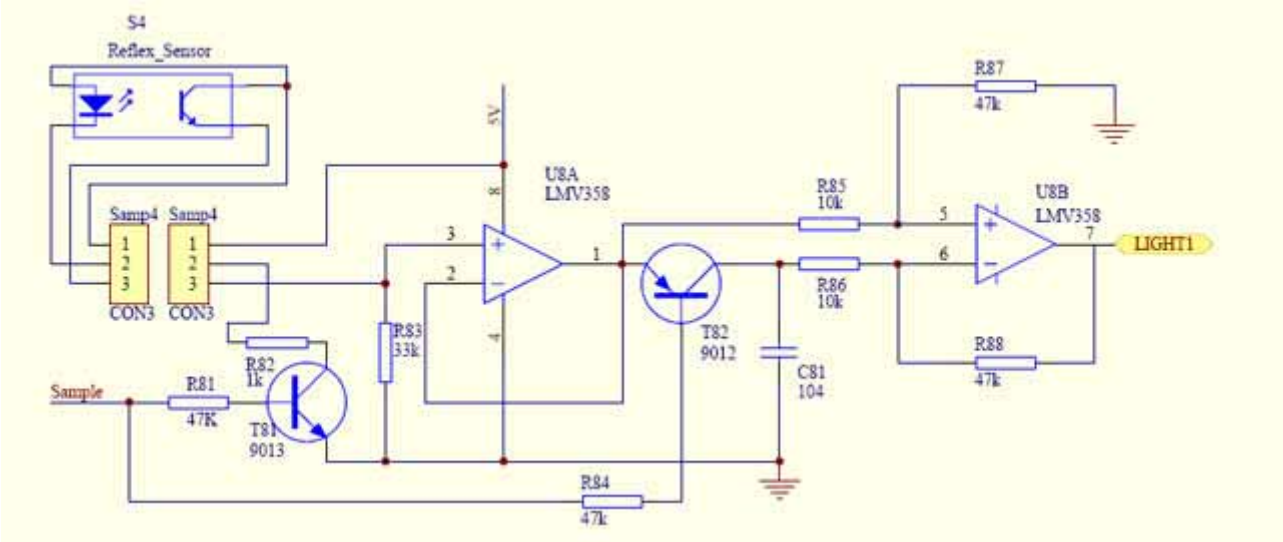
也许有人会问：既然已经将模拟量直接送到单片机中了，为何不通过软件处理来消除

呢？原因有二：

其一，不知道何时是顺光、何时是逆光？所以无法通过修正判断阈值来消除之；

其二，放大器的信号范围有限，有可能会饱和。

所以使用模拟电路来实现此功能，目前所用的电路如下：



图中，S4 为反射式光电采样器，Sample 端为单片机输出的控制信号，LIGHT1 端为反射光强弱模拟信号输出。

第一级运放构成同相缓冲器，一方面隔离后级电路对输入信号的影响，另一方面增加驱动能力，使电容 C81 能够快速跟踪输入信号，达到记忆背景光强度的目的，此电容可称之谓记忆电容。

第二级为减法器，实现对（总输入信号 – 背景光信号）的放大。

从图中可以看出，Sample 端控制着两个三极管，其中 T81(9013)控制反射采样器的红外发射管，T82（9012）控制第一级缓冲器输出给记忆电容 C81 的充电。

当 Sample 为低电平时，T81 截止，采样器的红外发射管熄灭，此时采样器输出的为背景光信号，此时 T82 导通，使记忆电容的电压等于背景光信号。第二级减法器的输入此时相等，LIGHT1 输出为“ 0 ”。

当 Sample 为高电平时，T81 导通，点亮红外发射管，此时采样器输出的为（背景光+反射光）信号，但是此时 T82 截止，将第一级输出与记忆电容隔开，记忆电容上仍保持着原来背景光的信号。第二级减法器的输入则为：

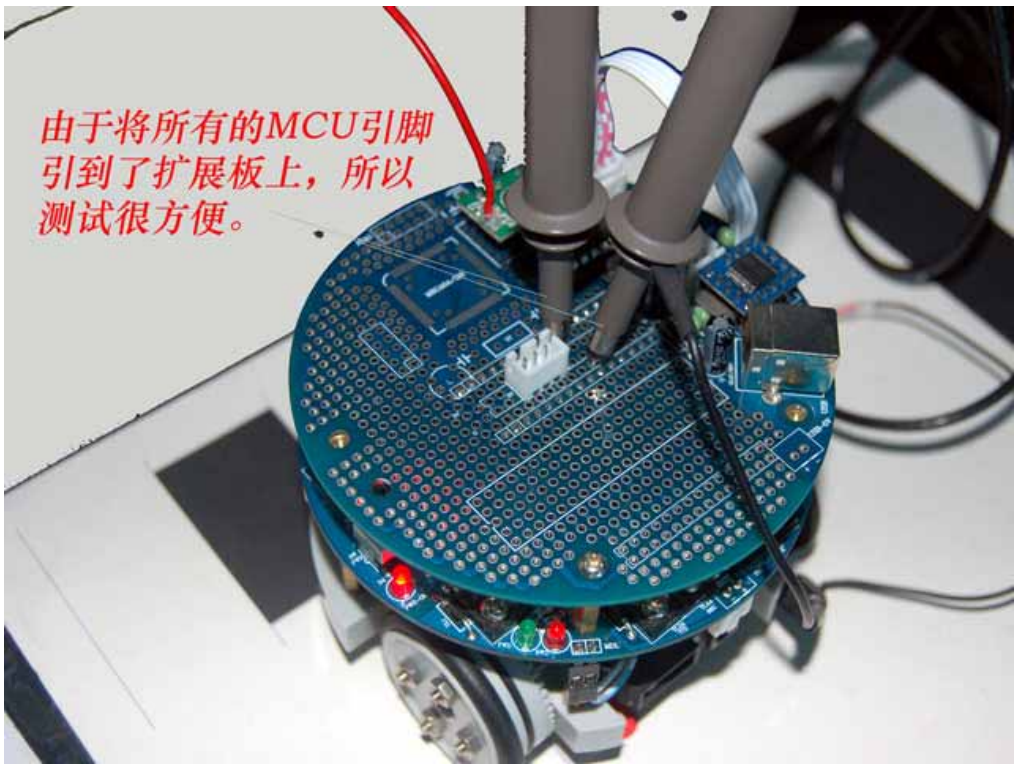
（反射光 + 背景光）— 背景光 = 反射光

从而达到消除背景光的目的，此时 LIGHT1 输出为放大后的反射光信号。

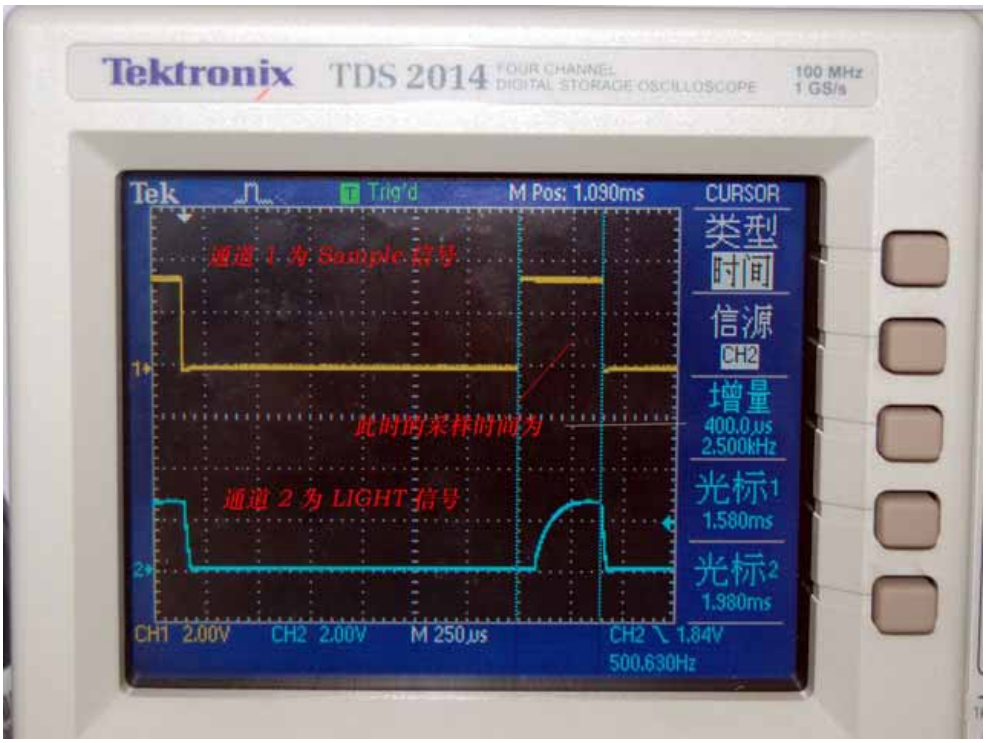
电路的工作原理大致如此，因为是以学习为目的，所以忽略了一些因素，如作为记忆电容充电控制的开关使用的是普通晶体管，读者可自行分析一下由此带来的不足。

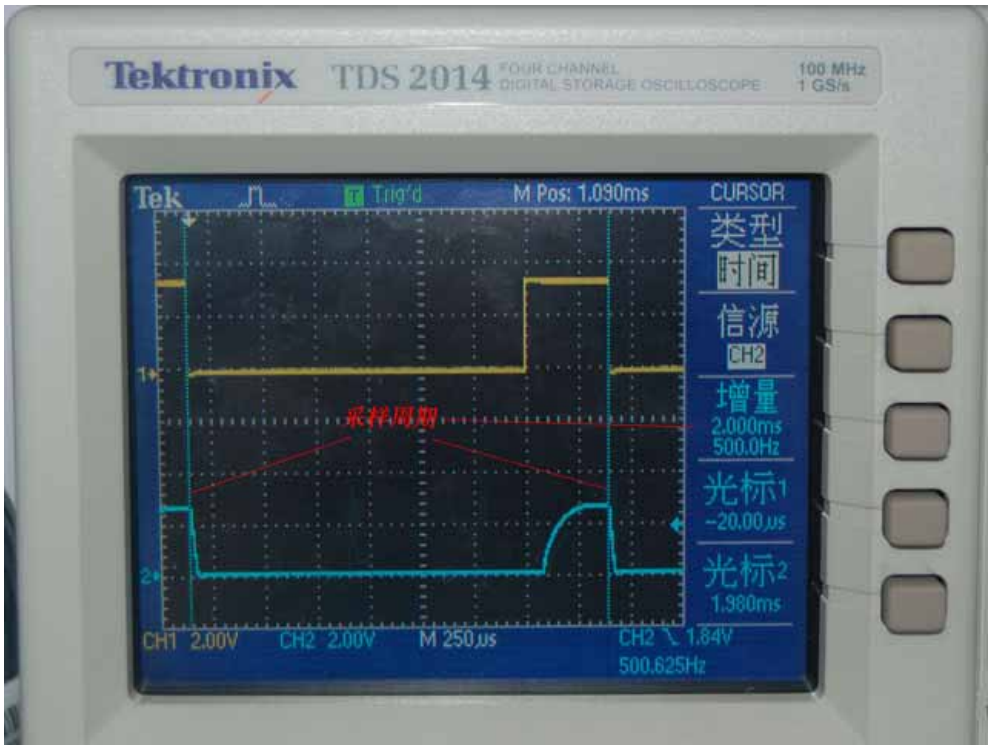
背景光的采集时间以及信号的采集时间如何确定，读者如果有条件可以使用二踪示波器检测 Sample 控制信号和 LIGHT 输出信号，用 Sample 作为触发信号，检测 Sample 上升沿后 LIGHT 输出的波形，应该是一个典型的一阶飞升曲线，Sample 保持时间至少应使此曲线达到水平区域，这个时间也就是启动 Sample 信号后的 MCU 读取延时值。

此时可以发现将 MCU 的所有引脚引到扩展板上的好处了，Sample 信号为 MCU 的第 9 脚，选择第一路测试，LIGHT1 为第 21 脚，直接在扩展板上就可以测量：



得到的波形为（运行我所提供的 Step5.hex）：





从上面两幅图中可以看出，信号采集延时为 400 us，采样的周期为 2ms。

注意图中的通道 2 波形，可以看出，在 Sample 信号将红外发射管点亮后，需要经过一段时间才能使采到的反射信号达到稳态。

如果没有手段，可大致按如下参数：

背景采集时间：〉700 us

信号采集延时： 300 ~ 400 us （此值不能太大，因为时间长了，记忆的背景光数值会变化）

此电路要与软件配合工作，否则不能输出正确的信号！！！

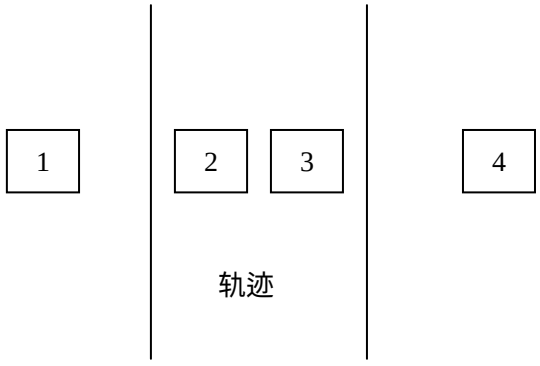
二、轨迹信号的处理

从轨迹传感器得到与反射强弱成正比的模拟信号后，需要交给 MCU 的 A/D 转换处理，再通过程序转换为该传感器是否在轨迹上的逻辑值，完成到这一步，才算是轨迹传感器，前面描述的硬件部分准确的说，应该只是轨迹采样器！

选择这款 MCU 的一个主要原因就是它带有 8 路 10 位 A/D 转换器，精度够了，转换速度也不低，在目前的设计中，每通道转换时间约 10 us。

使用内置的 A/D 转换器可大大简化电路，对于一些要求不是十分高或者没有特殊要求的场合，应该是首选方案，但是如果对于精度要求很高、信号需要隔离的场合，独立的 A/D 转换还是需要的，因为数字信号的隔离远比模拟信号容易。

圆梦小车设计了四路轨迹采样，是基于这样的考虑，用中间两个控制轨迹，两侧的用于检测道路的分支、直角弯、交叉口等，布置如下：



此方式并非最好，仅供参考。

4 路信号输入到 P1.0 – P1.3 ，使用 A/D 转换的前 4 个通道，对应关系见程序中的说明。

硬件和传感器的布置已交待清楚，下面该讨论如何将采到的模拟量转换为逻辑值了。

首先需要设立 4 个变量保存 A/D 的结果，因为轨迹采样要求不高，为提高速度，减少计算复杂性，使用 8 位模式（详见 STC12C5410AD 手册说明），为便于程序循环处理，用数组 ga_ucSampleVal[5] 保存，之所以增加一维，是为了 PC 读取时方便，将转换后的逻辑值作为第五单元。

为了兼顾 4 路轨迹采样的差异性，将判断阈值也设计为一个数组 ga_ucThreshold [4]，每个采样器有单独的判断阈值（我所附的程序就因为第 2 路的信号特别强，故提高了判断阈值，读者在使用时应注意根据自己的小车调整；也可以调整放大电路使 4 路性能接近），同时设计了一个上电初始化用的默认值 DEFAULT_THRES[4]。

调试时，可以利用写内存命令修改 ga_ucThreshold ，当调整合适后再修改程序中的默认值 DEFAULT_THRES[4]。

在所附的程序中，用软件模拟了一个“ 施密特触发器 ”的功能，设计了一个回差值 g_ucThresDelta，在根据 A/D 结果判断是否在轨迹上时，要根据原来的状态：

- 如果原来在轨迹上，则只有 A/D 结果大于 ga_ucThreshold + g_ucThresDelta 时才算脱离轨迹。
- 而如果原来不在轨迹上，则只有 A/D 结果小于 ga_ucThreshold - g_ucThresDelta 时才算进入轨迹。

这样处理后就不会出现在轨迹边缘频繁翻转状态的情况了。注意，上述讨论是基于轨迹为深色（黑色）！

回差也可以使用写内存的方式改变，合适后作为默认值 DEFAULT_THRES_DELTA。

通过上述处理将 4 路 A/D 结果转换为 4 个逻辑值，一个对应一位，存放在 ga_ucSampleVal[5] 的低 4 位。走轨迹的控制根据这个逻辑值实现。

详细的转换处理见程序中的 lineSamp_proc 函数。注意，此函数使用了参数传入和数据返回，这才是比较规范的函数编写方式，尽量不在函数中直接使用全局变量！特别是修改全局变量。

此函数中虽然还是使用了 ga_ucSampleVal、ga_ucThreshold、g_ucThresDelta 几个全局变量，但这几个只与此函数有关，如果不是为了调试，A/D 结果 ga_ucSampleVal 都可以作为局部变量。这样编写，主要是为了让初学者有个参数传递的概念。并未过于讲究，等以后尝试 RTOS 时，需要关注函数的可重入性时再作深究。

三、如何走在“路”上

因为本篇的主要目的是将小车的轨迹采样部分演示给读者，并非刻意追求走轨迹的效果，所以此处只是对轨迹采样逻辑值的一个简单应用示范，为了简化，只考虑 2、3 号传感器（见前面的传感器布置图），根据这两个传感器的值控制小车走在轨迹上。

初步的控制逻辑是：

- 如果 2、3 都在轨迹上（逻辑值为 0x06）,则小车直行，左右电机 PWM 值相同；
- 如果 2 不在轨迹上，则表示小车偏左，将右侧电机刹车；
- 如果 3 不在轨迹上，则表示小车偏右，将左侧电机刹车；

此逻辑基本是套用前一篇走直线的控制，只是控制变量由码盘的数值差变为了轨迹逻辑值。

此控制详见程序 YM1_Prog_5.C 中的 run_Online 函数。但测试后发现，效果不佳，分析后改进为 YM1_Prog_5A.C ，读者自己琢磨一下两者的差异，看看问题出在哪里。

这次只是实现了一个最基本的走轨迹功能，日后会研究改进之，达到 http://elm-chan.org/works/ltc/report.html 中小车行走的效果，也望读者中有哪位高人能尽早实现之，那才算是的控制！

四、PC 机侧程序的相应改进

为了配合走轨迹功能的调试，需要增加两部分内容：

- 轨迹采样部分的调试
- 走轨迹控制

轨迹采样部分的调试主要需求是读取相应的数据，如 4 个采样器的 A/D 结果、转换后的逻辑值等，以判断信号采集是否正确，程序处理是否合理。这部分不需要增加通讯命令，利用原来的读内存功能即可，只是为了方便，一次将所有需要的数据读回，并以合适的方式显示即可，所以这部分没有增加单片机侧的程序，只是在 PC 侧增加了一些显示和专用的操作按钮。

走轨迹控制与走直线类似，所以在 PC 侧将其合并，只是增加了一个运行模式选择，但单片机侧需要特殊的处理，所以增加了一个通讯命令：

小车走轨迹命令

命令字 —— 0x05

数据域 —— 行走距离（2 字节）PWM 基准值（2 字节，先低后高）

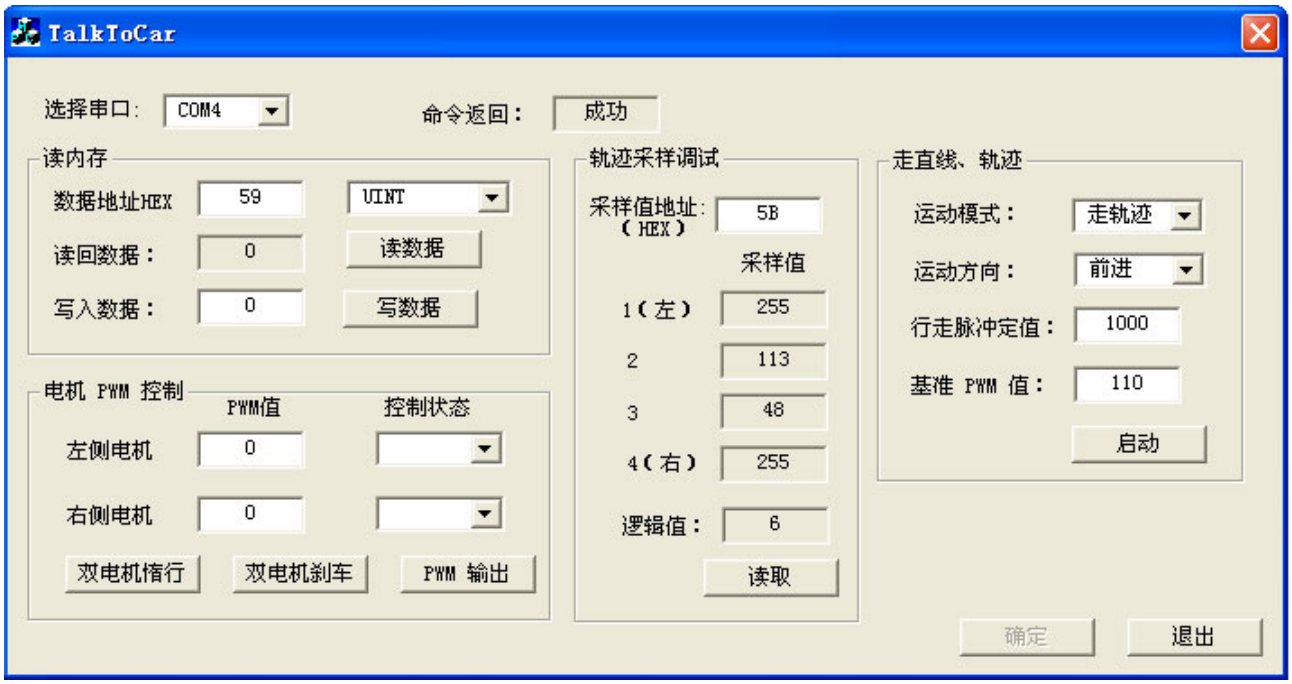
其中行走距离为左侧车轮的脉冲计数值，按目前的几何尺寸，最大值 65535 应该可以走 655 圈，约合 86 m。为 “ 0" 时连续行走。

返回数据帧：

帧头 发送方地址 自己的地址 帧长 命令 校验和

与走直线命令完全相似，但是不支持后退，因为小车的控制逻辑要变化，当然读者也可以尝试一下，可否让小车双向走轨迹？

修改后的 PC 界面如下：



注意，上部增加了一个“命令返回”状态指示，主要是为了判断是否可靠的得到了小车的数据，因为在读采样值时，有时数据不变，不能直观的判断是读回的数据没变，还是小车没有返回。

这个控制界面越来越丰富了，而且也越来越人性化，不知道使用哪种 LCD 屏可以如此随心所欲的添加需要的内容？又有何种方式能如此方便？我想读者也逐渐体验到 PC 机的好处了吧？

五、结语

本篇详细介绍的轨迹采样的原理，以及如何使用 STC12LE5412AD 单片机的 A/D 转换功能，示范了如何用软件实现“施密特触发器”，并在轨迹采样结果的基础上实现了基本的走轨迹功能。

走轨迹是一个值得深入探讨的问题，可以在采样方式上深入，如使用线阵 CCD；也可以在控制逻辑上深究，如 PID 控制，这些都有待读者去尝试、去攻克。

在第一轮基础问题都结束后，也许我们也会去尝试提高一个层次，将实现的内容从“有无”层面上升到“性能”，那时希望是个群言堂，而非我一人在此“忽悠”^_^

下一篇将给小车装上第三只“眼睛”，使它可以看清前面有无障碍，以免撞得“头破血流”!

Hanker

2007 年 7 月 17 日星期二

附件：

1、增加了轨迹采样的单片机程序

2、增加了轨迹采样调试的 PC 机程序

（注意，随着程序的升级，前面程序中的 bug 也有所暴露，希望读者能根据最新的程序予以修改，以免浪费自己的精力！）

3、小车走轨迹视频片断

参考资料：

1、STC12C5410AD 数据手册