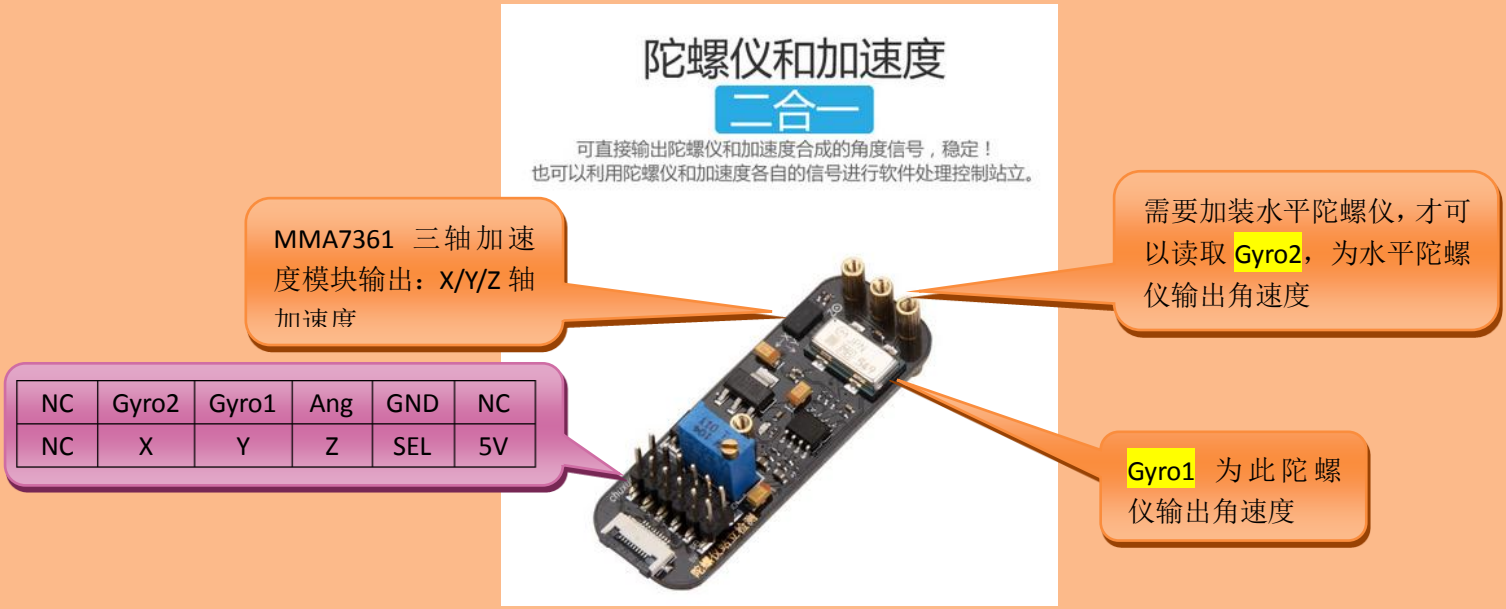
# 野火平衡组站立调试指南——初级篇

## 一．整体思路

采集 ENC-03 陀螺仪角速度，和模块硬件积分之后的角度，利用 PD 算法实现直立。
*教程适合新手利用模块快速入上手，实现车模直立。




注：此方法不能用在智能车比赛上，而是给大家练习使用（中级篇的方法可用在智能车比赛上，但难度比这个高，因此先用这个入门）。

## 二．模块引脚

此例程需要用到的 模块管脚有：ENC03 角速度， 角度 ， Z 轴

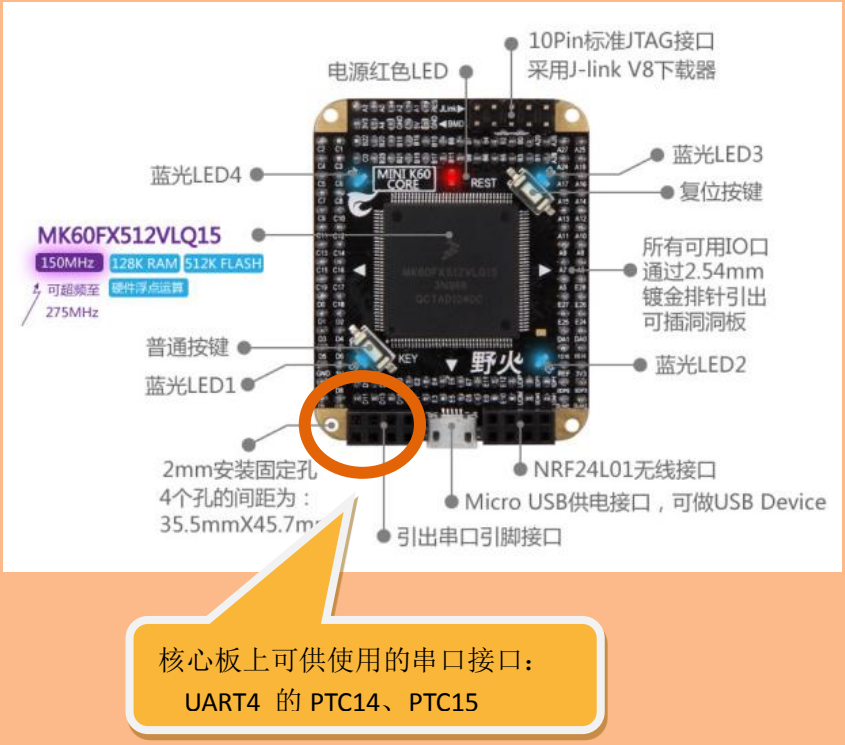
## 三．搭建调试环境

要实现直立，必须把单片机采集的参数发送到上位机，观察变化的趋势，才能更好的调节整个动态过程。

## 串口通信工具

可通过无线蓝牙，或者 TTL 转串口线来把单片机数据发送给电脑上位机，从而在上位机

上进行处理。




## A．无线蓝牙

分为发送端和接收端。可采用 2 个主从蓝牙模块搭配，也可通过主从模块和蓝牙适配器组合使用。

使用方法相当于“无线”的串口，分为主机和从机。

引脚接法：

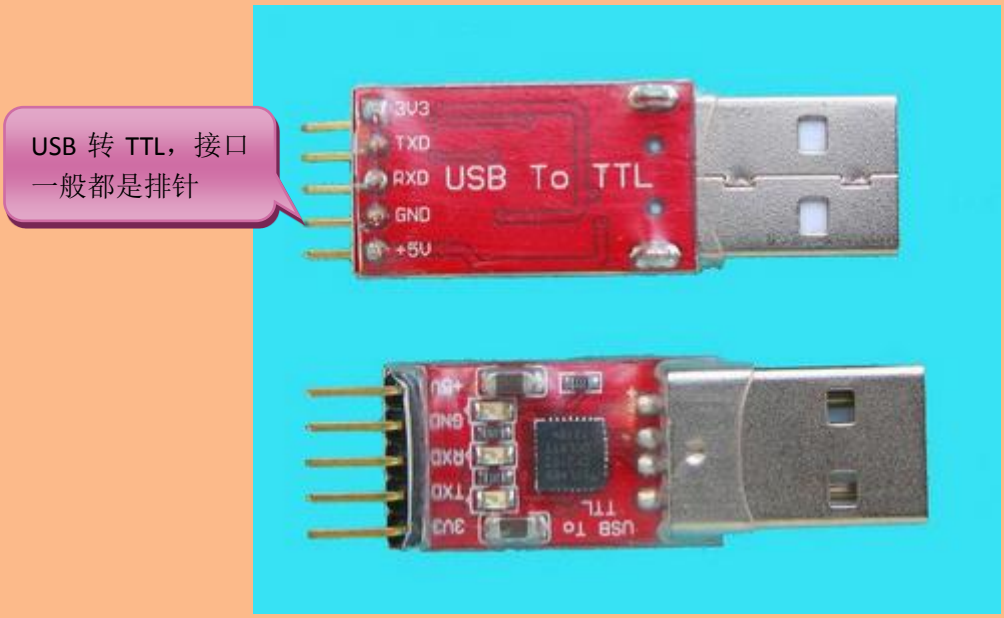
5v GND RXD TXD

注意:

蓝牙的 RXD 和单片机的 TXD 对应链接，蓝牙的 TXD 和单片机的 RXD 对应链接。

## B．USB 转 UART

通过 USB 转 TTL，直接把单片机的数据发送给上位机。




当然，也可以采用 USB 转 RS232，但 RS232 信号不是单片机能识别的，因此还需要 MAX3232 芯片 把 RS232 信号转为 TTL。


通过带 USB 接口的 RS232 模块，把单片机的 TTL 信号（0-5v）发送电脑串口端，通过上位机程序显示和接收。

# 上位机调试工具 Serial_Digital_Scope V2

我们采用的上位机是 Serial_Digital_Scope V2 。

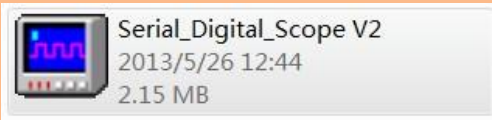
## A．使用方法

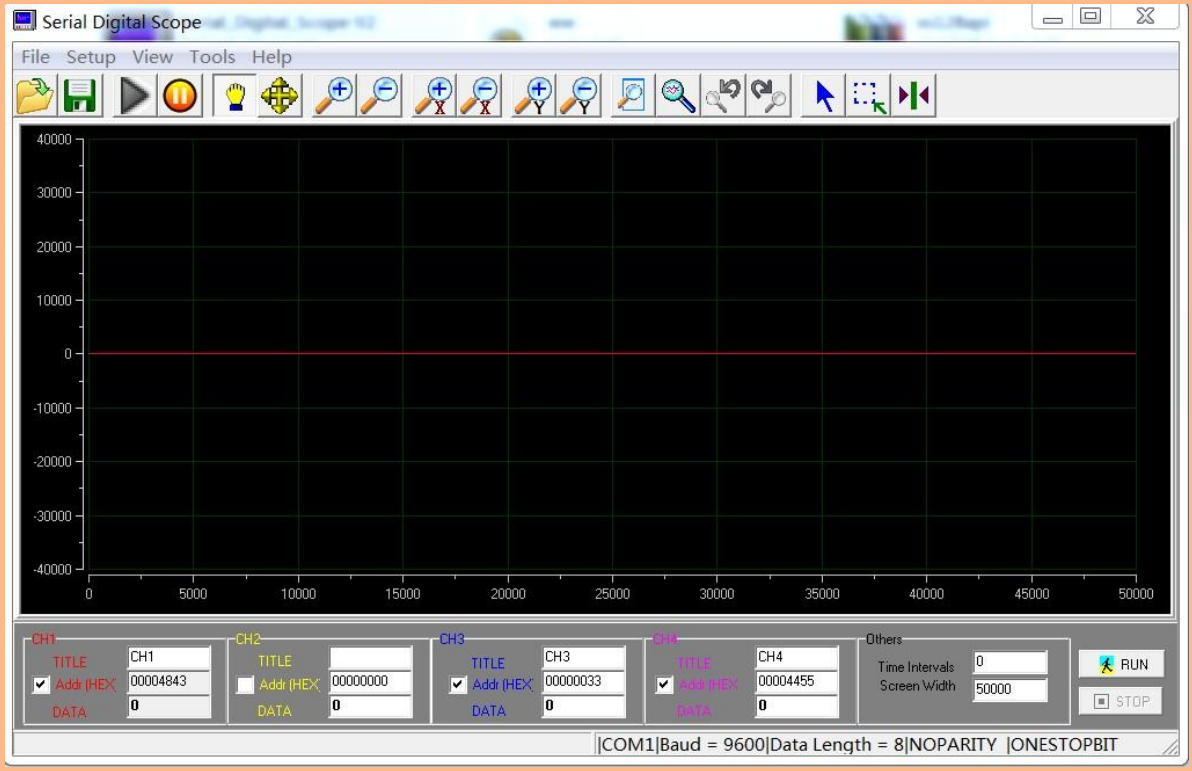
首先准备串口通信工具后，插入电脑，安装完 USB 转串行口驱动之后，在我的电脑--设备管理器--端口处找到 Prolicfic USB-to-Serial Comm Port(COM1)。

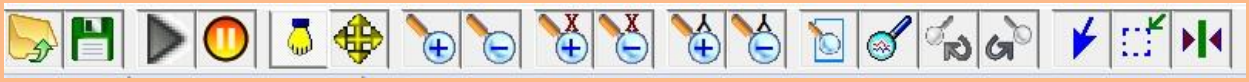
如果不是 COM1，点击串口驱动，右键-属性-端口设置-高级-COM 端口号修改为 COM1.

这个软件只兼容 COM1.


接着 轮到 Serial_Digital_Scope V2 上场：





依次为：打开文件夹，保存文件，恢复，暂停，移动，调焦，放大，缩小，X 轴调节，Y 轴调节，缩小到合适图像，缩放到最大等功能。


一共四个通道，使能相应的通道，打上勾。点击 RUN 即可显示数据。

## B．代码的移植

首先简单调用 K60 库函数 UART 相应函数是无法使用这个上位机的，需要在对 UART 发送的字节进行 CRC 校验，封装成相应的协议传输才行。具体参考代码工程。

## 步骤 1：移植上位机 API 函数

在 软件\Serial_Digital_Scope V2\xs128api.rar 压缩包里有 这软件的 下位机 API 接口函数。

把相应的 API 加入工程之后，需要修改 void OutPut_Data(void)函数，改成 K60 对应的代码：

```
1.      /*
2.      *  功能说明：SCI 示波器发送函数
3.      *  参数说明：
4.                OutData[]   需要发送的数值赋予该数组
5.      *  函数返回：无符号结果值
6.      *  修改时间：2013-2-10
7.      */
8.      void OutPut_Data(void)
9.      {
10.         int temp[4] = {0};
11.         unsigned int temp1[4] = {0};
12.         unsigned char databuf[10] = {0};
13.         unsigned char i;
14.         unsigned short CRC16 = 0;
15.         for(i=0;i<4;i++)
16.         {
17.
18.             temp[i]  = (int)OutData[i];
19.             temp1[i] = (unsigned int)temp[i];
20.
21.         }
```

```
22.
23.         for(i=0;i<4;i++)
24.         {
25.             databuf[i*2]    = (unsigned char)(temp1[i]%256);
26.             databuf[i*2+1] = (unsigned char)(temp1[i]/256);
27.         }
28.
29.         CRC16 = CRC_CHECK(databuf,8);
30.         databuf[8] = CRC16%256;
31.         databuf[9] = CRC16/256;
32.
33.         for(i=0;i<10;i++)
34.         {
35.             uart_putchar (UART4,(char)databuf[i]);
36.         }
37.     }
```

修改了此处代码

### 步骤 2：发送数据

Serial_Digital_Scope V2 支持 4 个通道发送，需要发送的 4 个数据分别写入到数组 OutData[]即可，接着调用 OutPut_Data() 函数 完成 发送。

```
1.     //假设需要发送的内容 real_angle 、 g_fCarAngle 、 ENC03 、Gyro_Now
2.     OutData[0] = real_angle;
3.     OutData[1] = g_fCarAngle;
4.     OutData[2] = ENC03 ;
5.     OutData[3] = Gyro_Now;
6.
7.     //执行发送函数
8.     OutPut_Data();
```

# 四．平衡组代码初始化

## 1．三轴加速度和陀螺仪初始化

野火采用的是 三轴加速度模块 MMA7361 和 陀螺仪 ENC-03MB ，两者都是输出模拟信号，初始化就是对 K60 ADC 管脚进行初始化，从而 ADC 采集信号。

```
1.     //定义野火 三轴加速度和陀螺仪模块的 ADC 通道
2.     #define XOUT     ADC1_DM0
```

```
#define YOUT    ADC0_SE16
#define ZOUT    ADC0_SE17

#define Gyro1   ADC1_SE16
#define Gyro2   ADC1_DP0
#define Ang     ADC0_SE18

//初级篇仅仅需要使用 Z 轴加速度 、 ENC-03 陀螺仪角加速度、融合后的角度
adc_init (ZOUT);          //MMA7361 Z 轴
adc_init (Gyro1);         // ENC03 角速度
adc_init (Ang);           //角度
```

## 2. 定时器的初始化

由于站立控制算法需要定时采集信号，调整站立姿势，因此需要加入 定时器来定时中断处理。

这里采用 PIT0 定时器，定时时间为 5ms：

```
pit_init_ms(PIT0, 5);                       //初始化 PIT0，定时时间为： 5ms
set_vector_handler(PIT0_VECTORn ,PIT0_IRQHandler);
                                  //设置 PIT0 的中断复位函数为 PIT0_IRQHandler
enable_irq (PIT0_IRQn);                                   //使能 PIT0 中断
```

## 3. 初始化电机驱动模块

野火的电机驱动模块带 4 路半桥，可实现 2 路全桥。我们例程里就是用了 2 个全桥接入 2 个电机驱动模块。电机驱动模块带了 使能端 进行使能控制。

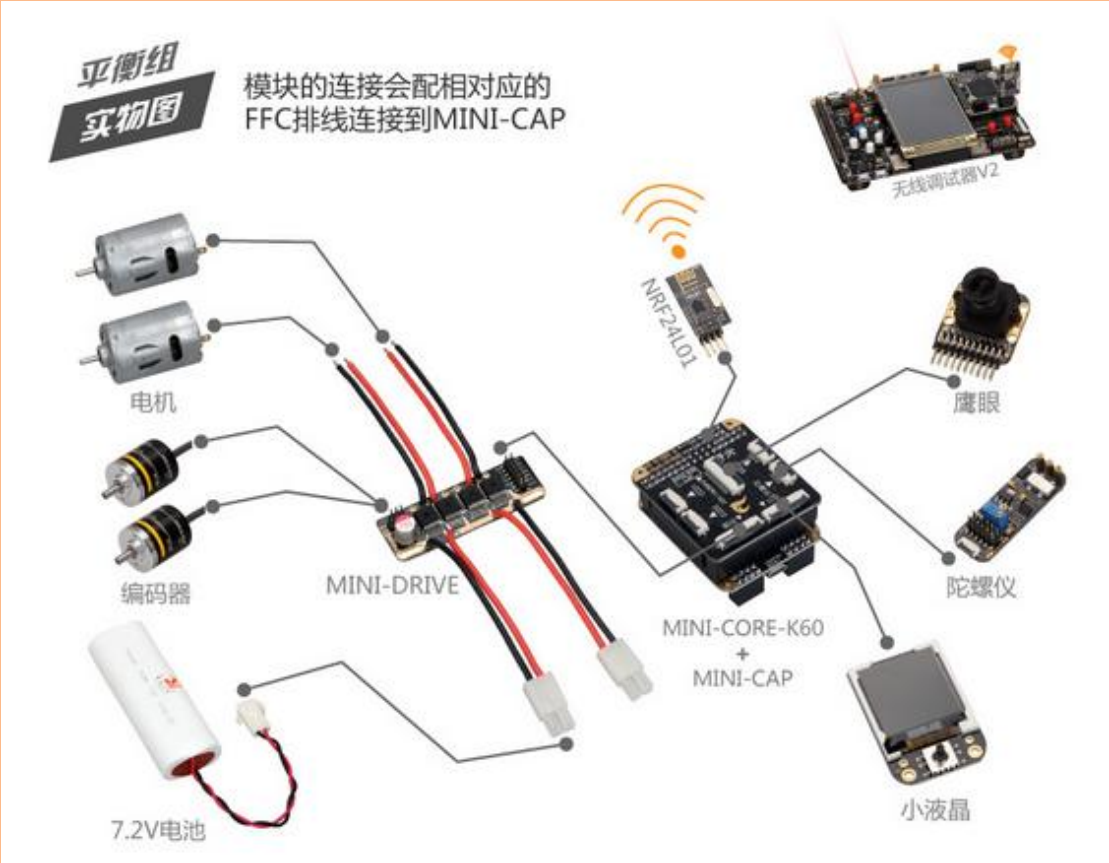
我们的小底板上选择的接线为：FTM0_CH3~ FTM0_CH6，带 使能端控制。

```
//初始化 PWM 输出
//FTM 的管脚 可在  fire_port_cfg.h
//宏定义 FTM0_PRECISON   改为  1000u
//PWM 数值反转。
FTM_PWM_init(FTM0, FTM_CH3,10000,1000);
FTM_PWM_init(FTM0, FTM_CH4,10000,1000);
FTM_PWM_init(FTM0, FTM_CH5,10000,1000);
FTM_PWM_init(FTM0, FTM_CH6,10000,1000);

//开启使能端
gpio_init(PTD15,GPO,0);
```

修改了占空比的精度

```
12.     gpio_init(PTA19,GPO,0);
13.     gpio_init(PTA5 ,GPO,0);
14.     gpio_init(PTA24,GPO,0);
```




## 4. 配置中断服务函数

前面讲过，我们采用 PIT0 定时 5ms 来定时获取姿态，从而调整姿态。

定时中断服务函数里，主要是进行 AD 采集，ENC03 角速度和模块输出角度的换算，然后通过 PD 参数的相乘，计算出 PWM 数值，输出到电机驱动，实现直立。

```
1.    void PIT0_IRQHandler(void)
2.    {
3.        AD_Calculate();                                  //直立角度，角速度计算
4.        Speed_Calculate(g_fCarAngle,Gyro_Now);           //直立速度计算
5.                        // g_fCarAngle 模块输出角度    ,Gyro_Now    ENC03 角速度
6.        PIT_Flag_Clear(PIT0);                            //清中断标志位
7.    }
```

AD_Calculate 和 Speed_Calculate 都是 清华方案 现成代码，初学者 无需纠结于如何实现，而是先跳过理解，懂得如何调用这函数即可，等小车站立好后，再去理解也不迟。

在这里，野火直接贴修改后的代码：

```
/*
*  功能说明：直立角度计算
*  参数说明：

*  函数返回：无符号结果值
*  修改时间：2013-2-10
* 备注：参考清华源码
*/
void AD_Calculate(void)
{
    Rd_Ad_Value();                          //采集 AD

    g_fCarAngle=(float)(real_angle-real_angle_vertical)*ratio;
                    // 归一化后的角度 = (AD 采集的角度 - 角度中值)/归一化比例
                    // g_fCarAngle 为归一化到-90 +90 内的角度

    Gyro_Now = (int)(GYRO_VAL - ENC03 );
                      // 陀螺仪新值 = 陀螺仪中值 - AD 采集的陀螺仪角度
                     //Gyro_Now 减去中值后的角速度

    //上诉两个参数都是 传递进去 Speed_Calculate 函数 进行 速度控制

    /**********************串口看波形（选择使用）**********************/
#if 0                               //宏条件编译 选择是否使用 虚拟示波器
    OutData[0] = real_angle;
    OutData[1] = g_fCarAngle;
    OutData[2] = ENC03 ;
    OutData[3] = Gyro_Now;
    OutPut_Data();
#endif
}
```

```
/*
*  功能说明：直立速度计算
*  参数说明：angle                融合后最终角度
*          angle_dot              陀螺仪角速度
*
*  函数返回：无符号结果值
*  修改时间：2013-2-10
* 备注：参考清华源码
*/
void Speed_Calculate(float angle,float angle_dot)
{
    /***************************速度计算**************************/
```

```
13.         speed_Start = angle * P_ANGLE  + angle_dot * D_ANGLE ;
14.                                                         //直立时所要的速度
15.           //P_ANGLE  P_GYRO  宏定义 直立所需要的 PD 参数
16.
17.         Speed_L = speed_Start;//左轮总速度
18.         Speed_R = speed_Start;//右轮总速度
19.         /**************将最大速度限制在 985 个 PWM 内*********************/
20.         if(Speed_L > 985)  Speed_L = 985;
21.         if(Speed_L < -985)   Speed_L = -985;
22.         if(Speed_R > 985)  Speed_R = 985;
23.         if(Speed_R < -985)   Speed_R = -985;
24.
25.         /*******因为驱动部分对信号进行反相，所以需对速度进行一个最终的处理******/
26.         if(Speed_L > 0)      //因为加了反相器，所以 PWM 要反过来添加
27.             Speed_L_Last = 1000 - Speed_L;
28.         else
29.             Speed_L_Last = -1000 - Speed_L;
30.
31.         if(Speed_R > 0)      //因为加了反相器，所以 PWM 要反过来添加
32.             Speed_R_Last = 1000 - Speed_R;
33.         else
34.             Speed_R_Last = -1000 - Speed_R;
35.
36.         /***********用所得到的对应角度的速度进行 PWM 控制*****************/
37.         if(Speed_L >= 0)     //angle 大于 0，向前，小于 0，向后
38.         {
39.             FTM_PWM_Duty(FTM0,FTM_CH3,1000);
40.             FTM_PWM_Duty(FTM0,FTM_CH5,(uint32)(Speed_L_Last - MOTOR_DEAD_V
AL_L));                                                 //加入死区电压
41.         }
42.         else
43.         {
44.             FTM_PWM_Duty(FTM0,FTM_CH5,1000);
45.             FTM_PWM_Duty(FTM0,FTM_CH3,(uint32)(-Speed_L_Last - MOTOR_DEAD_
VAL_L));                                                //加入死区电压
46.         }
47.
48.         if(Speed_R >= 0)     //angle 大于 0，向前，小于 0，向后
49.         {
50.             FTM_PWM_Duty(FTM0,FTM_CH6,1000);
51.             FTM_PWM_Duty(FTM0,FTM_CH4,(uint32)(Speed_R_Last - MOTOR_DEAD_V
AL_R));                                                 //加入死区电压
52.         }
53.         else
```

```
54.         {
55.             FTM_PWM_Duty(FTM0,FTM_CH4,1000);
56.             FTM_PWM_Duty(FTM0,FTM_CH6,(uint32)(-Speed_R_Last - MOTOR_DEAD_
VAL_R));    //加入死区电压
57.         }
58.
59.     }
```

# 五．传感器参数的整定

## 1．采集 Z 轴输出和角度输出

### 1．整定的目的

主要调节滑动变阻器，使硬件角度和 Z 轴输出趋势一致。其中滑动变阻器改变角速度的比例。


### 2．整定的方法

从模块正面看，滑动变阻器,顺时针旋转到顶端，然后逆时针开始旋转。

首先硬件滤波中间角度从 0 开始上升，旋转到一定程度，角度的波形与 Z 轴的趋势 一致，当继续逆时针旋转，出现数值跟踪缓慢和超调等现象。反复调整滑动变阻器，找到合适的状态。

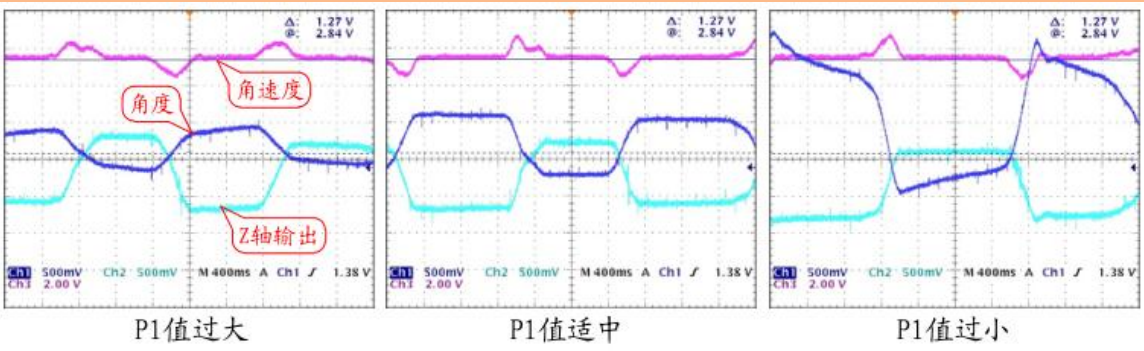



图 3- 24 不同 P1 阻值对应的输出波形

具体操作方法：向前转动 45 度，静止，然后向后转动 45 度，静止。

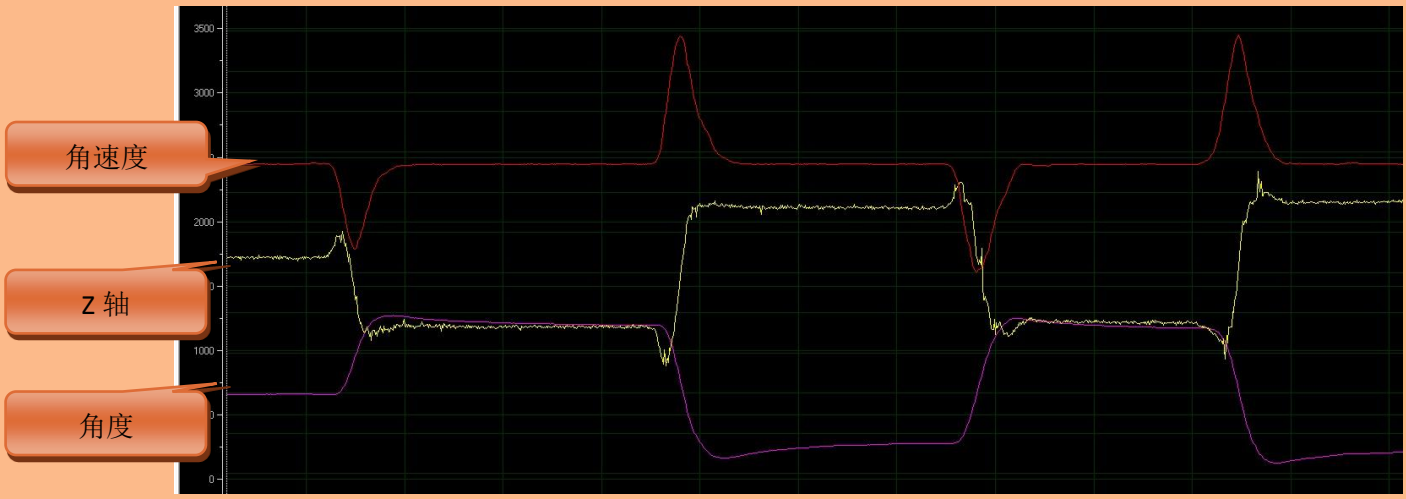
如下图所示就是采集回来的波形分析：

红色 为 ENC03 角速度数值

黄色 为 Z 轴数值

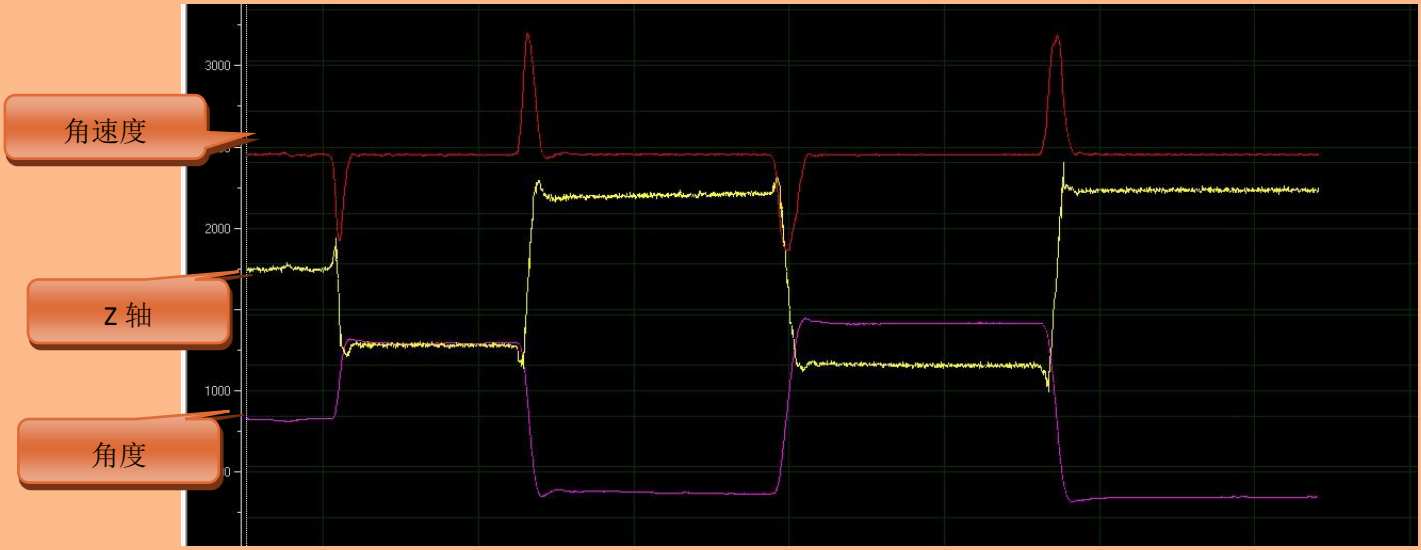
粉色 为 硬件滤波后的角度

## 超调现象分析



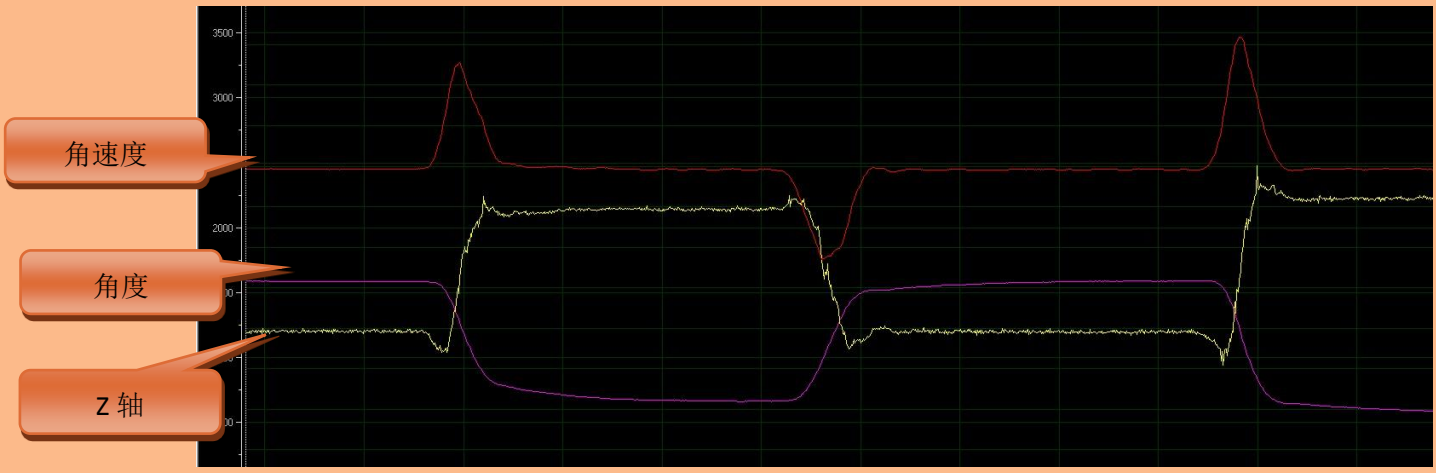

分析：从模块正面，逆时针旋转滑动变阻器到一定程度，角度输出相对于 Z 轴超调。

### 正常现象分析




分析：从模块正面，继续逆时针旋转滑动变阻器到一定程度，此时角度和 Z 轴不超调，跟踪不缓慢，其中角度输出和角速度方向相反。

### 跟踪缓慢分析




分析：从模块正面，继续逆时针旋转滑动变阻器到一定程度，此时角度输出比较缓慢。

## 2．计算陀螺仪的角速度和融化后角度

陀螺仪的角速度和融合后的角度 ，都不能直接拿来 计算，而是需要 经过处理。在 AD_Calculate 函数内进行下面的换算：

```
Rd_Ad_Value();                              //采集 AD(Z 轴、角度、陀螺仪角速度)

g_fCarAngle=(float)(real_angle-real_angle_vertical)*ratio;
                        // 归一化后的角度 = (AD 采集的角度 - 角度中值)/归一化比例
                        // g_fCarAngle 为归一化到-90 +90 内的角度

Gyro_Now = (int)(GYRO_VAL - ENC03 );
                                // 陀螺仪新值 = 陀螺仪中值 - AD 采集的陀螺仪角度
                                //Gyro_Now 减去中值后的角速度
```

## 陀螺仪中值的测量

保持车模静止，测出 ENC03 静止的数值 GYRO_VAL
假定测得 ENC03 静止的 值 为 2430
那么在 control.h 里面 修改 GYRO_VAL 的值：

```
#define GYRO_VAL            2430        //陀螺仪中值
```

一般加大向后，减少向前（与安装在车上位置相关）
如果偏左偏后转动，就需要调陀螺仪位置。

## 角度中值的测量

保持车模站立平衡，测量出车模实际的直立角度输出 real_angle_vertical

假定测得角度的输出值为 920
那么在 control.h 里面 修改 real_angle_vertical 的值：

```
#define real_angle_vertical 920         //角度
```

## 角度归一化系数的确定

* ratio 的确定，向前水平放置车模，测量出角度输出数值 real_angle__forward，
或者向后水平放置车模，测量出角度输出数值 real_angle_backward.
ratio=90/(real_angle__forward- real_angle_vertical)

那么在 control.h 里面 修改 ratio 的值：

```
#define ratio               0.118       //归一化比例
```

*不设置 ratio 也可以，直接进入下一步骤。只要调节比例参数 P_ANGLE 即可。

# 六．控制参数的整定

此部分主要修改比例参数 P_ANGLE 和微分参数 D_ANGLE

涉及的代码：

```
1.    void Speed_Calculate(float angle,float angle_dot);    //速度计算
```

这个函数内部会调用如下代码：

```
1.    speed_Start = angle * P_ANGLE  + angle_dot * D_ANGLE ;//直立时所要的速度
```

具体操作的方法：

先设定 D = 0，逐步增加 P 参数，当车模出现震荡，加入 D，当出现抖动，增加 P，反复调节 P，D 即可实现直立。

如果需要长时间静止，还需要加入编码器，双闭环，才行。

比如：

```
1.    #define P_ANGLE            30
2.    #define D_ANGLE            0.5
```

分析：

比例控制能迅速反应误差，从而减小误差。但不能消除误差，微分控制可以减小超调量，克服震荡，使系统稳定性提高，加快动态响应速度，较小调整时间，从而改变动态性能。

取值范围：

FTM 初始化为 1000，所以要根据融合的角度 g_fCarAngle 和角速度 Gyro_Now 的大概取值范围确定 P、D 参数，主要结果不能太小，这样输出的 PWM 太小，也不能超出 1000.

# 七．调节过程中出现的问题

## 1．需要注意传感器的正负方向

当角度向前倒下，小车电机向前，当角度向后倒下，小车电机向后，最终融合的角度和角速度大小和变化趋势，根据上图趋势。原则是当融合的角度为正，角速度也需要按照同方向增大。角速度相当于正向反馈。否则当 2 者方向判断相反，小车不能直立。

## 2．增强电机驱动能力

K60I/O 口同时驱动 4 个 7971，驱动能力不足。野火的电机驱动加入了 MOS 管进行隔离反相，提高了驱动能力。

由于 PWM 经过反相（单片机输出占空比为 20%，电机驱动实际输入占空比为 80%），因此单片机输出 PWM 占空比的时候也需要反转。

## 3．车模往一个方向偏，为什么？

首先确定安装模块是否水平，先尝试反复调整位置，观察是否能静止。

接着可以试图修改车模实际的直立角度输出 real_angle_vertical。

这是因为你目标角度为 0，然后中值测量不准确，导致计算出的角度和目标 0 度角有偏差，也可以修改 ENC03 静止的数值 GYRO_VAL。

注明：文档参考官方文件《电磁组直立车模参数整定与调试指南 1.0》结合 K60 处理器编写。