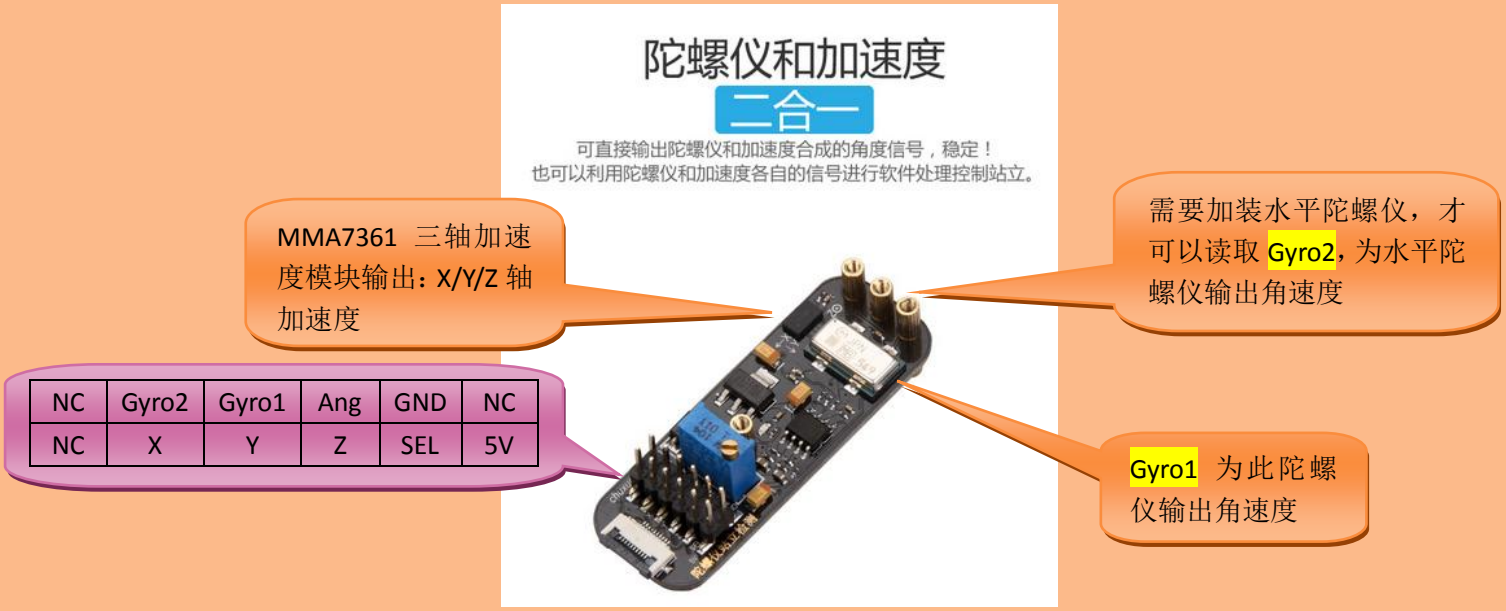
野火平衡组站立调试指南——中级篇

一．整体思路

采集 ENC-03 陀螺仪角速度，利用加速度计 MMA7361 Z 轴角度和陀螺仪角度软件算法融合，消除陀螺仪飘移，利用 PD 算法实现直立。



初级篇的教程方法仅适合练习使用，本教程的目的是进一步学习相关算法，实现车模直立。此算法可用于正式的飞思卡尔智能车比赛中。

二．模块引脚

与 初级篇相比，没了角度管脚！

此例程需要用到的 模块管脚有：Z 轴，角速度。

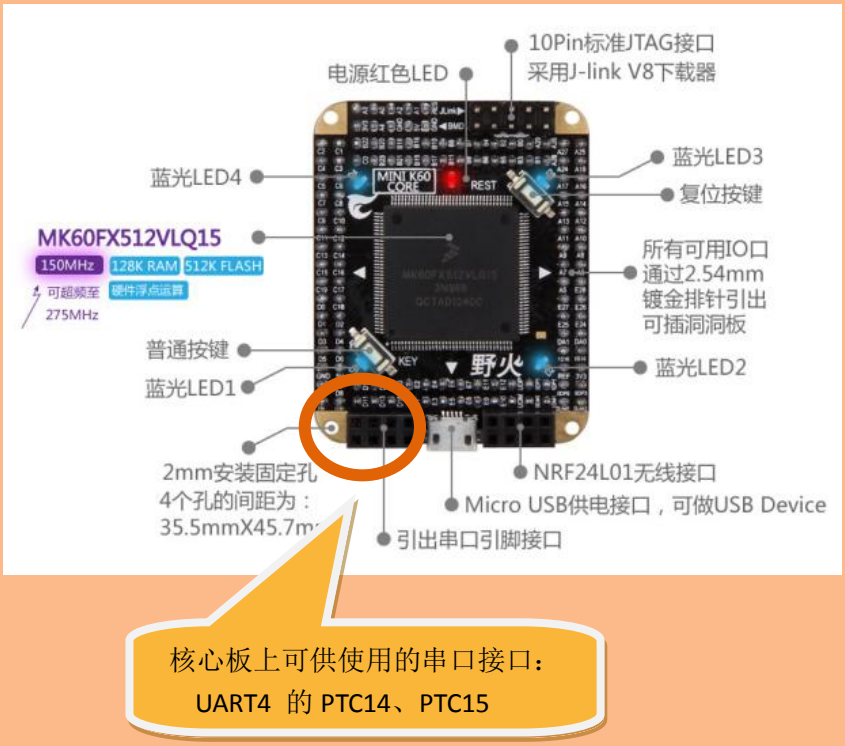
三．搭建调试环境

此章节与初级篇一模一样！！

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：

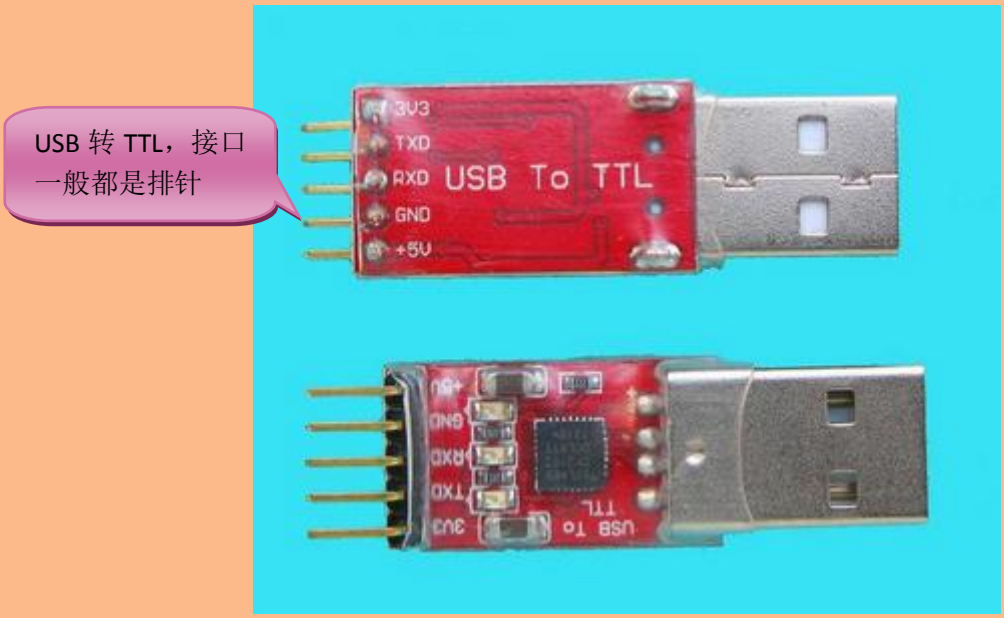
5v　GND　RXD　TXD

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 。

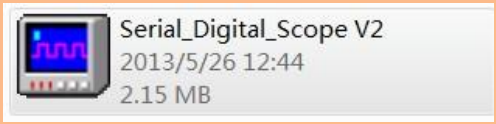
A．使用方法

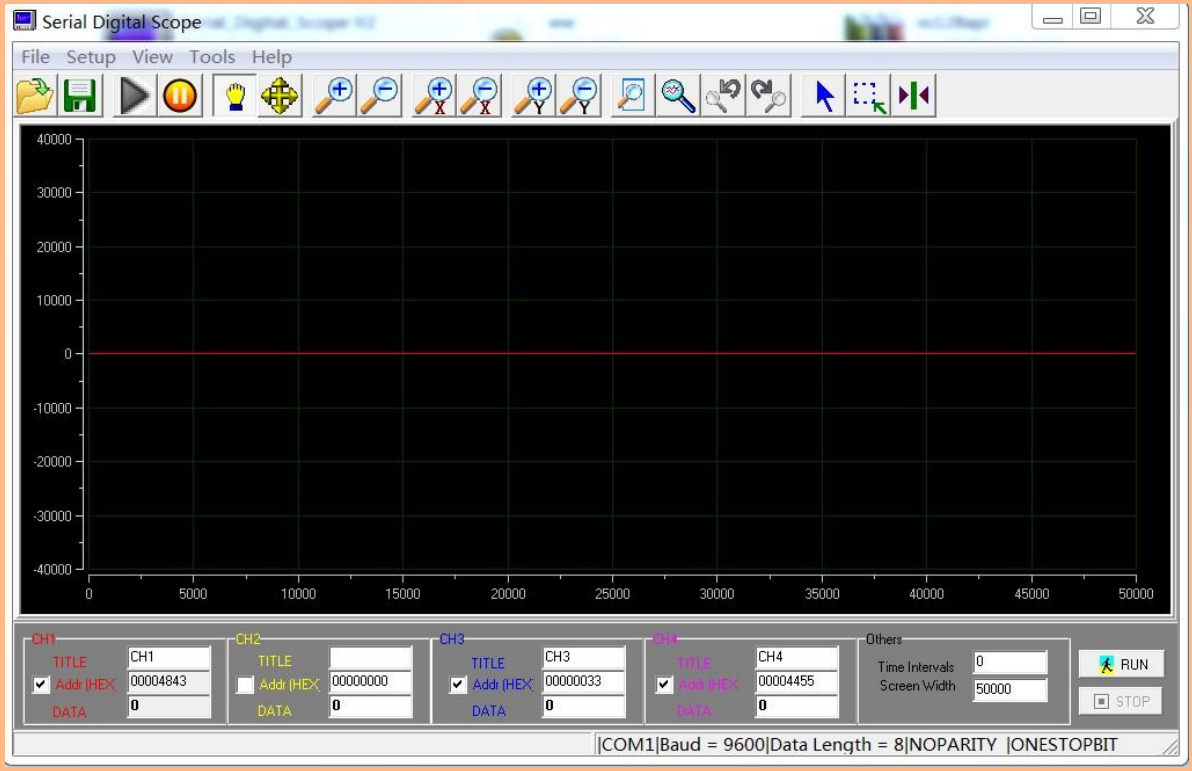
首先准备串口通信工具后，插入电脑，安装完 USB 转串行口驱动之后，在我的电脑--设备管理器--端口处找到 Prolicfic USB-to-Serial Comm Port(COM1)。

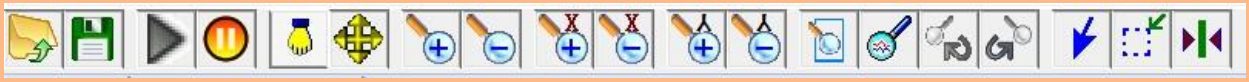
如果不是 COM1，点击串口驱动，右键-属性-端口设置-高级-COM 端口号修改为 COM1.

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

与初级篇相比，少了硬件融合后的角度

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

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

//由于使用软件滤波，因此不再用 硬件输出的角度
//adc_init (Ang);         //角度
```

采用软件滤波，不需要硬件融合后的角度信号

2. 定时器的初始化

与初级篇相比一模一样！

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

与初级篇一模一样！

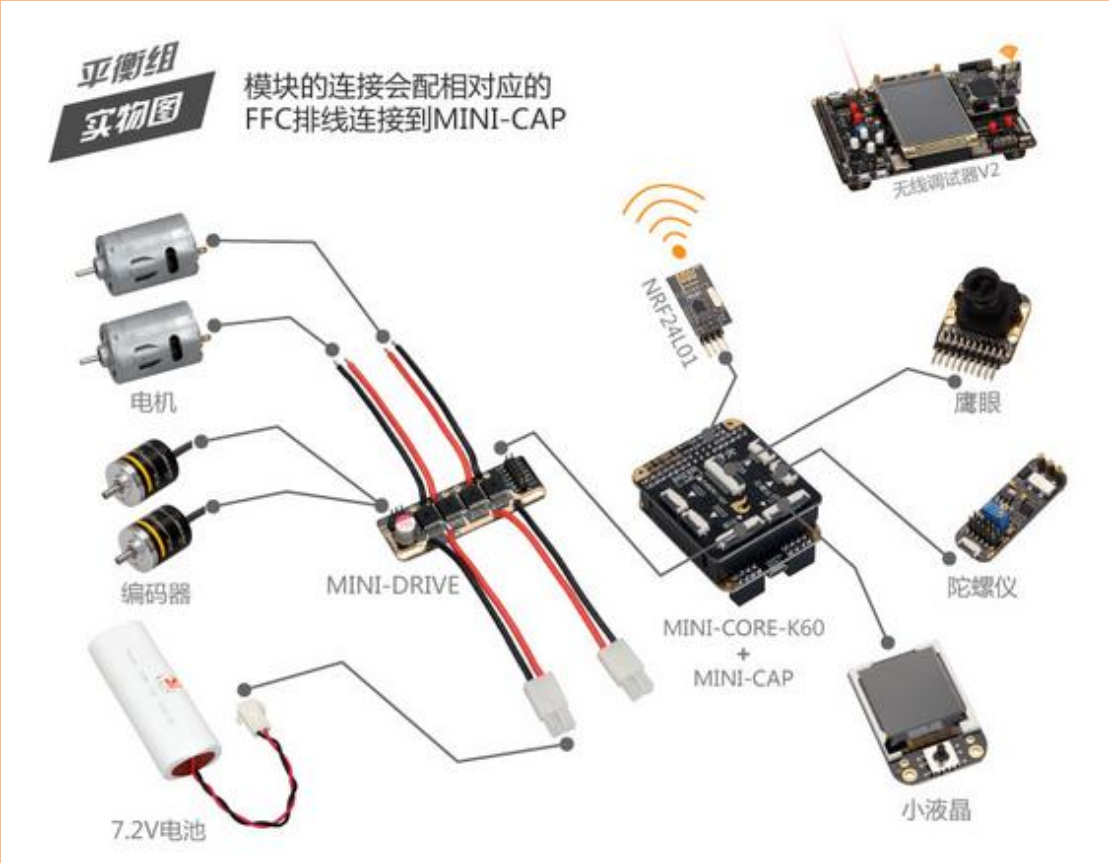
野火的电机驱动模块带 4 路半桥，可实现 2 路全桥。我们例程里就是用了 2 个全桥接入 2 个电机驱动模块。电机驱动模块带了 使能端 进行使能控制。

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);
```

修改了占空比的精度

```
    //开启使能端
    gpio_init(PTD15,GPO,0);
    gpio_init(PTA19,GPO,0);
    gpio_init(PTA5 ,GPO,0);
    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。

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

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();

    Gyro_Now = (GYRO_VAL - ENC03 ) * Gyro_ratio;
                          //陀螺仪采集到的角速度归一化
    angle_offset_vertical=(MMA7361_vertical - MMA7361)*MMA7361_ratio ;
                  //将加速度计采集到的角度归一化,乘上 0.375 是为了归一化到 0~90°

    if(angle_offset_vertical > 90)angle_offset_vertical = 90;
                             //防止加速度角度溢出
    if(angle_offset_vertical < -90)angle_offset_vertical = -90;

    //计算融合后的角度
    QingHua_AngleCalaulate(angle_offset_vertical,Gyro_Now);
                                          //清华角度滤波方案


    /*******************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] = ENC03;
    OutData[1] = MMA7361;//Gyro_Now;
    OutData[2] = angle_offset_vertical ;
    OutData[3] = g_fCarAngle;
    OutPut_Data();
#elif  0
    OutData[0] = angle_dot;
    OutData[1] = Gyro_Now;
    OutData[2] = angle_offset_vertical ;
    OutData[3] = angle;
    OutPut_Data();
#endif
}
```

此处代码为：清华角度滤波方案。

附带初级篇硬件滤波的代码（对比差异）：

```
    g_fCarAngle=(float)(real_angle-
                        real_angle_vertical)*ratio;
    //归一化角度=(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
```

与初级篇一样

```
1.  /*
2.  *  功能说明：直立速度计算
3.  *  参数说明：angle                融合后最终角度
4.  *          angle_dot             陀螺仪角速度
5.  *
6.  *  函数返回：无符号结果值
7.  *  修改时间：2013-2-10
8.  * 备注：参考清华源码
9.  */
10. void Speed_Calculate(float angle,float angle_dot)
11. {
12.     /*****************************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

由于理论值与实际值有误差，传感器的安装位置与环境也影响实际测量值的取值，因此需要对测量值进行线性归一化处理。

1．加速度 Z 轴归一化

（1）测量不同位置的值

红色加粗表示代码中用到

这里提供的是一种思路，不一定是正负 90 度。

①零偏数值：**MMA7361_vertical**
车模直立，Z 轴在水平方向的 AD 数值；
②+90 度数值：MMA7361_forward
车模向前水平 90 度，即 Z 轴在垂直方向 AD 数值(+1g)；
③-90 读数值：MMA7361_backward
车模向后水平 90 度，即 Z 轴在垂直方向 AD 数值(-1g)；

此时测量出来的 AD 数值和理论值存在一定的偏差，这是传感器本身存在的问题，所以需要线性归一化进行误差校正，而且每款传感器都需要根据安装位置及环境进行测量。

（2）归一化法测量

比例因子：**MMA7361_ratio**

MMA7361_ratio=　180　/　（ MMA7361_forward-MMA7361_backward ）;
或者：
MMA7361_ratio=　90　/　（ MMA7361_forward-MMA7361_vertical）;
或者：
MMA7361_ratio=　90　/　（ MMA7361_vertical-MMA7361_backward）;

实测输出模拟值：　　**MMA7361**
角度：**angle_offset_vertical**
angle_offset_vertical = (MMA7361_vertical - MMA7361) * MMA7361_ratio ;

这样计算出来的最终的近似角度计算在（+-90 度内），进行相关的限幅就可以了。

2．陀螺仪角速度

陀螺仪比例：　　**Gyro_ratio**
1 首先设定重力加速度角度补偿时间常数为 **GRAVITY_ADJUST_TIME_CONSTANT** 为 2

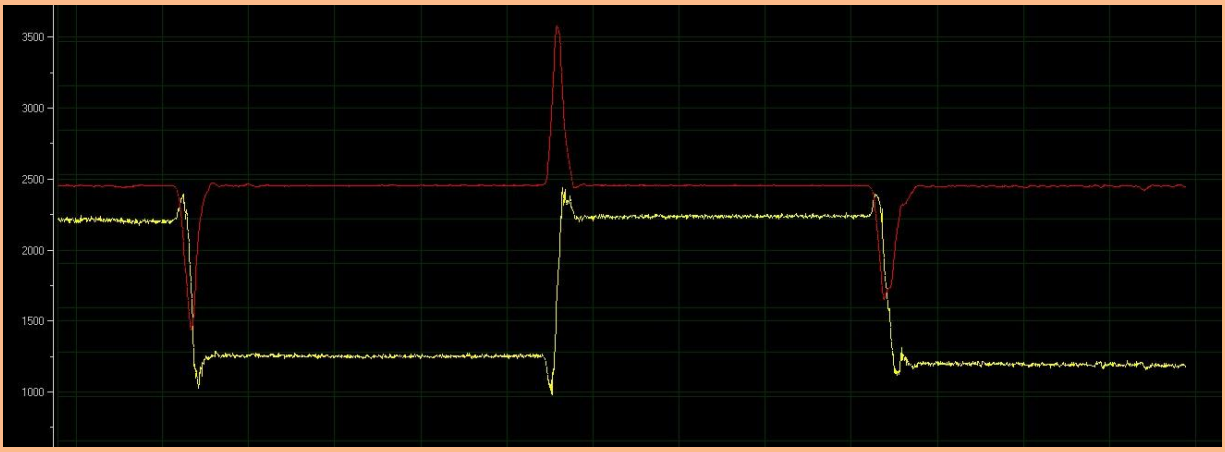
2 选择不同 的陀螺仪比例参数 **Gyro_ratio**：
车模向前向后分别倾倒 60 度。
同时观察波形图，选择角度数值刚好不过冲的条件下，为 Gyro_ratio 最佳值。

3 理论计算：
Rgyro=Rad ÷ Rg ÷ K
Rad 为 AD 转换因子 3.3v/4096
Rg 为陀螺仪比较因子 0.67mv*deg/sec 即单位时间内变化 1 度为 0.67Mv
K 为模块放大电路的放大倍数，为 10.
Rgyro=Rad ÷ Rg ÷ K=3.3*1000/4096/0.67/10=0.12.
具体如何选择合适的参数，可以参考官方的视频介绍和文档说明。

可以不用管它，或者看官方的文档说明

视频:直立车模调试指南:http://www.chuxue123.com/forum.php?mod=viewthread&tid=1245

4 这里介绍如何结合 K60 工程修改相应的参数快速实现直立。
A 采集陀螺仪静止输出 **GYRO_VAL** ， 7361 零偏数值 **MMA7361_vertical** 数值。
车模向前向后分别倾倒 60 度。
红色为陀螺仪 AD 输出
黄色为 Z 轴 AD 输出



*分析：陀螺仪 AD 输出，在静止的时候为中间数值，快速转动的时候，根据 ENC-03 向前向后，数值变化，可以看出模块反冲数值极小。Z 轴 AD 输出根据向前向后转动，数值也在中间数值变化。注意方向性。

比如 GYRO_VAL =2430 ， MMA7361_vertical = 1710

B 计算陀螺仪角速度 Gyro_Now 和 Z 轴归一化角度 angle_offset_vertical；

主要修改调节参数 Gyro_ratio 和 MMA7361_ratio。

代码：

```
1.    Gyro_Now = (GYRO_VAL - ENC03 ) * Gyro_ratio;
2.                                    //陀螺仪采集到的角速度归一化
3.    angle_offset_vertical = (MMA7361_vertical - MMA7361) * MMA7361_ratio ;
4.                          //将加速度计采集到的角度归一化，乘上0.129是为了归一化到0~90°
5.    if(angle_offset_vertical > 90)angle_offset_vertical = 90;
6.    if(angle_offset_vertical < -90)angle_offset_vertical = -90;
7.                                     //防止加速度角度溢出
```

*其中 MMA7361_ratio.为 0.127 是我测试模块计算出来的。实际要自己测。

比如 2420 1710 1000

90/（2420-1710）=0.1267

*Gyro_ratio 的确定：

理论计算的数值不太准确，可以按照官方方法测试。D 步骤介绍调试方法。

C 计算出融合的角度 g_fCarAngle 和角速度 Gyro_Now；

主要修改参重力补偿系数 GRAVITY_ADJUST_TIME_CONSTANT 和积分时间系数 DT。

代码：

```
1.    void QingHua_AngleCalaulate(float G_angle,float Gyro)
2.    {
3.     float fDeltaValue;
4.     g_fCarAngle = g_fGyroscopeAngleIntegral;    //最终融合角度
5.     fDeltaValue = (G_angle - g_fCarAngle) / GRAVITY_ADJUST_TIME_CONSTANT;
6.     //时间系数矫正
7.     g_fGyroscopeAngleIntegral += (Gyro + fDeltaValue) * DT;    //融合角度
8.    }
```

*积分时间系数 DT 主要决定定时器的定时时间，假设定时 5MS，则 DT=0.005

*重力补偿系数 GRAVITY_ADJUST_TIME_CONSTANT，首先确定某个数值，实现直立之后，再设置为不同参数，然后观察参数影响效果，选择最优参数。

比如 DT=0.005

GRAVITY_ADJUST_TIME_CONSTANT= 2

D 以下按照比较 Z 轴计算的角度趋势和最终的融合角度趋势，反复调整 Gyro_ratio 的数值。可以先代入理论计算值，从小往上调节，找到合适的曲线。

原则是：最终的融合角度曲线能快速跟踪 Z 轴计算的角度曲线，同时不至于超调。

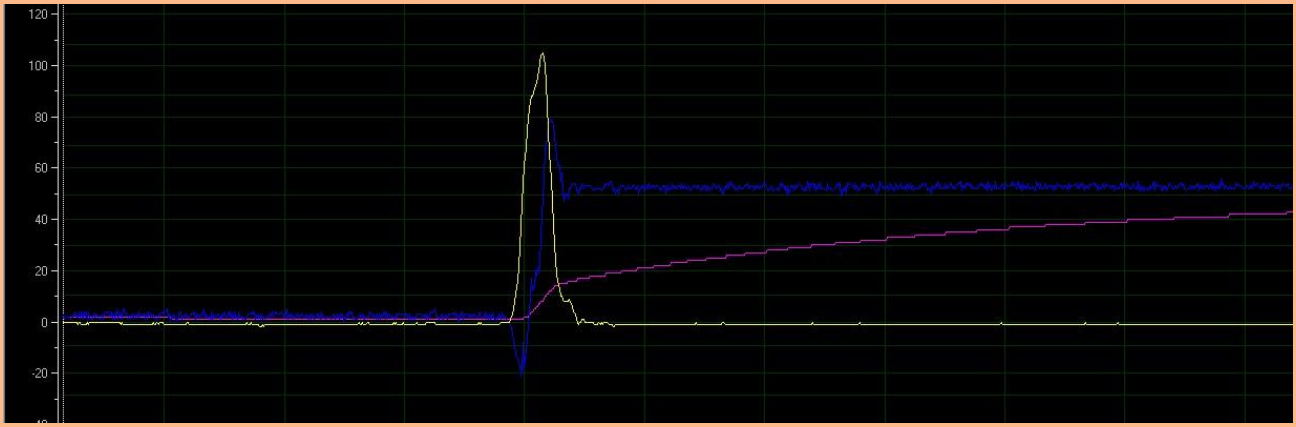
如下图：

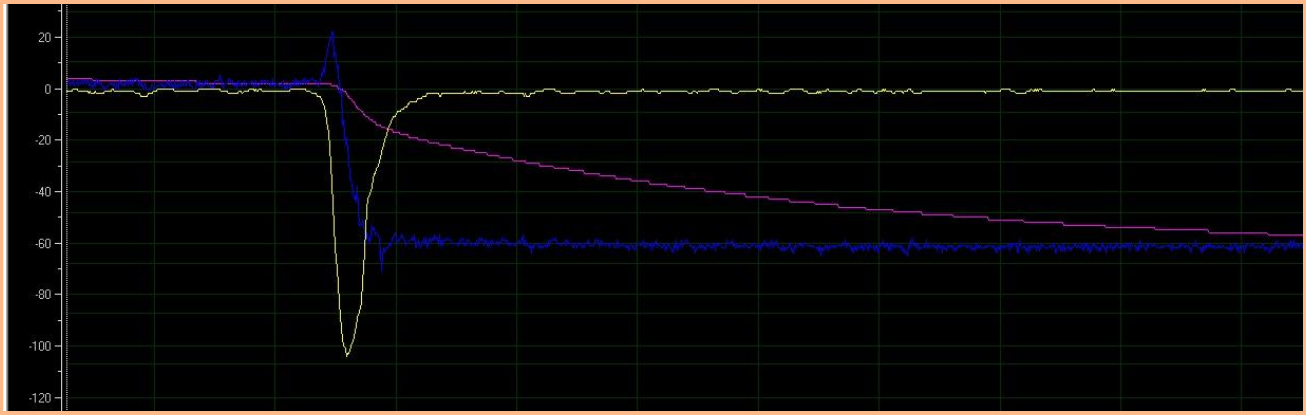
Gyro_ratio= 0.2

黄色　角速度*比例之后的数值

蓝色　 Z 轴计算的角度

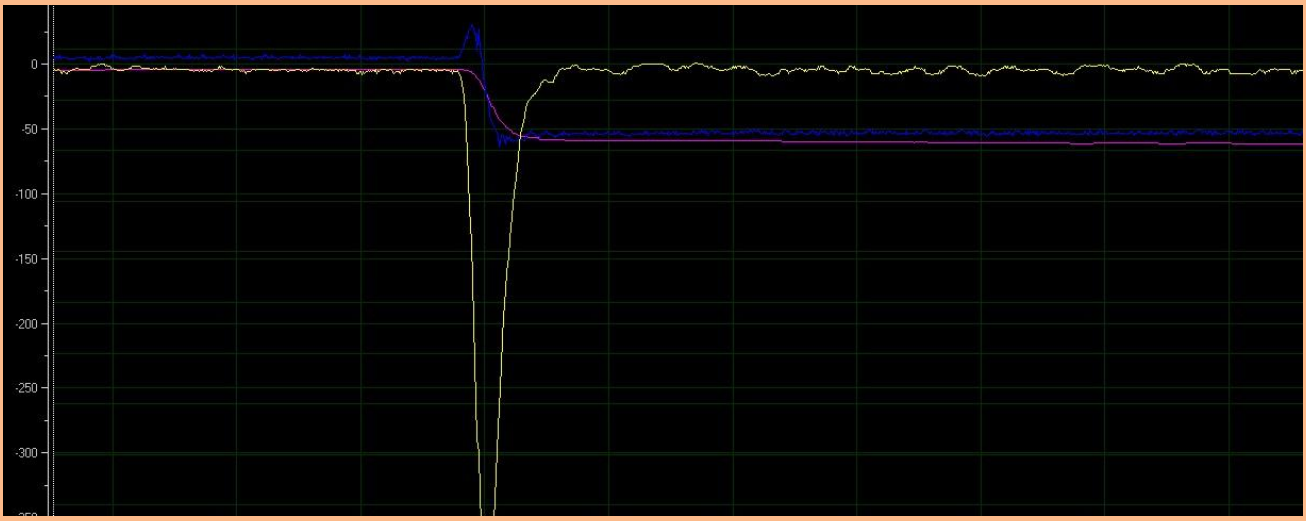
粉色　最终的融合角度





分析：此时参数过小，最终的融合角度曲线跟踪不上 Z 轴

Gyro_ratio= 0.7

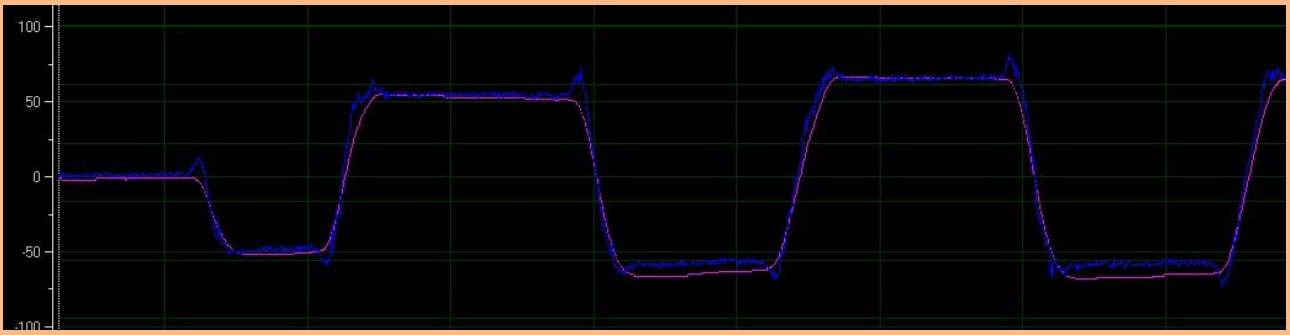

0
-50
-100
-150
-200
-250
-300


80
60
40
20
0
-20
-40
-60
-80
-100


200
150
100
50
0
-50
-100
-150
-200

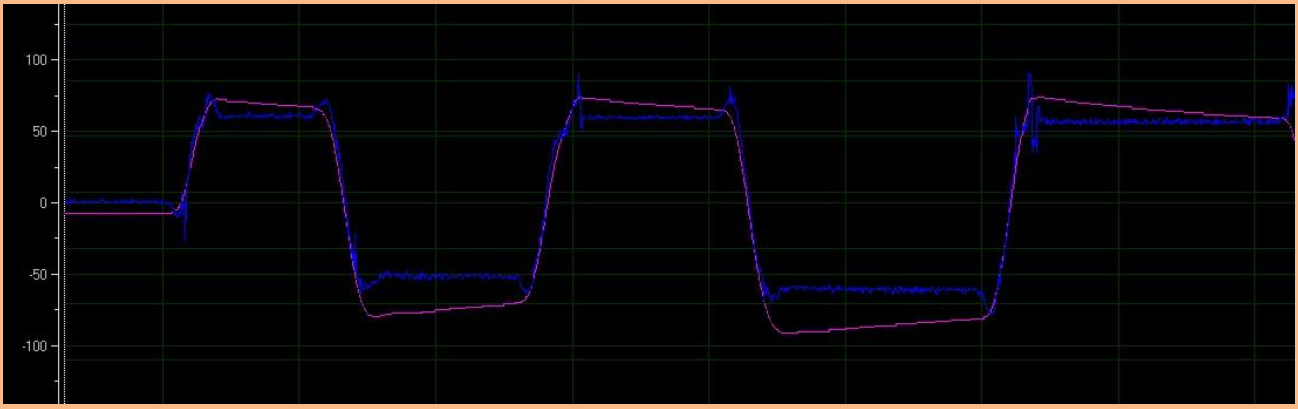
分析：此时参数基本吻合，

Gyro_ratio=0.8



分析：参数吻合，进行下一步 PD 参数调节

Gyro_ratio= 1



分析：此时参数过大，超调

六．控制参数的整定

1 角度控制，主要修改比例参数 P_ANGLE 和微分参数 D_ANGLE

代码：

```
1. void Speed_Calculate(float angle,float angle_dot);
2. speed_Start = angle * P_ANGLE  + angle_dot * D_ANGLE ;
```

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          50
2.    #define D_ANGLE          1.5
```

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，客服震荡，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。

隔离保护电路：

K60 输出的 PWM 难以提供足够的驱动能力驱动 4 个 7960，而且电机驱动历来都是单片机杀手，因而需要加入保护隔离电路，可选择 MOS 管、者反相器、同相器。

野火用的是 MOS 管，PWM 信号经过隔离保护电路后，会产生反相作用（等效于反相器），即原先 K60 PWM 输出 20% ，经过 MOS 管保护电路后，变成 100% - 20% = 80%。因而下面的 PWM 数值需要反转。

取值范围：

FTM 初始化为 1000（精度值为 1000，所以占空比=1000/1000=100%，经过保护电路后变成 1-100%=0%），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 PD 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 车模往一个方向偏，为什么？

这是因为你目标角度为 0，然后陀螺仪中值测量不准确，导致计算出的角度和目标 0 度角有偏差，可以修改陀螺仪静止数值 GYRO_VAL，其中加大向后，减少向前。

2 容易出现的一个错误是利用融合之后的角度判断电机正反转，实际是需要利用 P D 参数计算的最终数值计算正反。

3 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。

4 其余滤波算法？

教程利用清华的融合算法，有兴趣还可以利用工程代码中提供的卡尔曼滤波算法，方法一致，只需要修改卡尔曼相应参数即可。

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。

视频:直立车模调试指南

http://www.chuxue123.com/forum.php?mod=viewthread&tid=1245&extra=page%3D1