自平衡三棱柱调试笔记

目录

一、 说明

自平衡三棱柱是我在学校时的创新项目独轮三棱柱的一个衍生品，在这里将一套廉价解决方案和调试的过程经验与心得分享给大家，理论什么的不说太多，很多是一些调试经验，有些地方可能写的不是很准确，各位大佬若发现一些问题可以直接向我们淘宝小店纤毫电子反馈，感谢！

二、 三棱柱倾角测量

理想情况下，我们只需测量其中一个方向上的加速度值，就可以计算出三棱柱的倾角。比如使用 X 轴。三棱柱静止的时候，只存在重力加速度，没有运动加速度，此时 X 轴输出零。当三棱柱有倾角之后，g 会在 X 轴有重力加速度分量，而且该轴倾斜的角度和重力分量的大小相关。

Angle_ Y=atan2(AcceL_ Y,Accel Z)*180/PI (1)

Angle_ X=atan2(Accel X.AcceL Z)*180/PI (2)

atan2(y,x)是表示 X-Y 平面上所对应的(x,y)坐标的角度，返回的是原点至点(x,y)的方位角，即与坐标系的 x 轴的夹角，输出值要通过* 180/PI 转化为角度。

Accel_ X、Accel_ _Y、 Accel_ Z 分别表示重力加速度在该轴的分量。

由于运动加速度的累加也会影响角度测量的精度，MPU6050 里面也集成了三轴陀螺仪，因为陀螺仪输出的是角速度，我们可以对角速度进行积分得到角度，而积分会导致累积误差，且随着时间增加越来越大。加速度计测量的角度信号在受到外界干扰的情况下有很大的噪

声，我们通过算法融合两者角度值最终得到角度值。

一阶互补滤波：

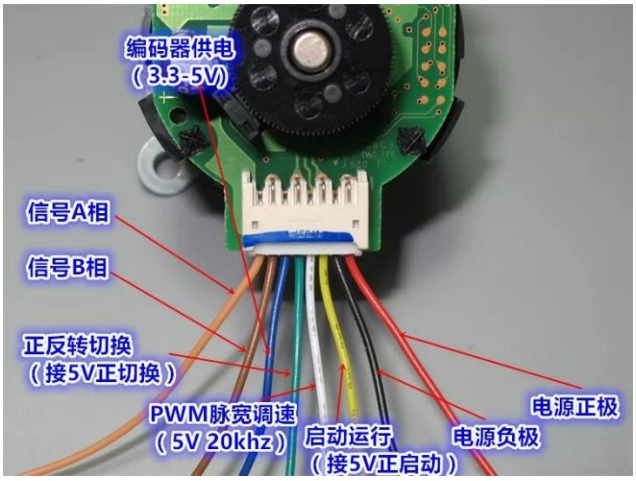
angle = K1 * angle_ m+ (1-K1)* (angle + gyro_ m* dt)，

angle 是融合后的角度值，angle_ m 是加速度测量得到的角度，angle +gyro_ m* dt 是陀螺仪积分得到的角度，dt 为采样周期，单位是 s，在我们的代码中是 0.003. K1 是滤波器系数，这边我们选取 0.02，一阶互补滤波也可以看做是加权平均。

三、 无刷电机简述

1. 标准版无刷电机

该独轮平衡三棱柱中我们使用日本万宝至无刷伺服电机，内置驱动，支持正反转，PWM 调速，并且带有 100 线编码器 AB 相双通道信号输出。



该电机接线图如上图所示，

1.信号 A 相和信号 B 相为编码器脉冲输出端；

2.正反转切换的线我们直接用单片机的引脚 3.3V 电平控制，是完全没有问题的；

3.编码器供电接 3.3V；

4.PWM 接单片机的 PWM 输出，启动运行我们接单片机 IO 口，在电机初始化时置为高电平；

5.电源负极接 GND，电源正极接 12V。

6.一定要注意，控制信号的地和供电的地一定要是一个地，这个共地的概念虽然十分基础，但我们还是发现有不少同学没有共地!

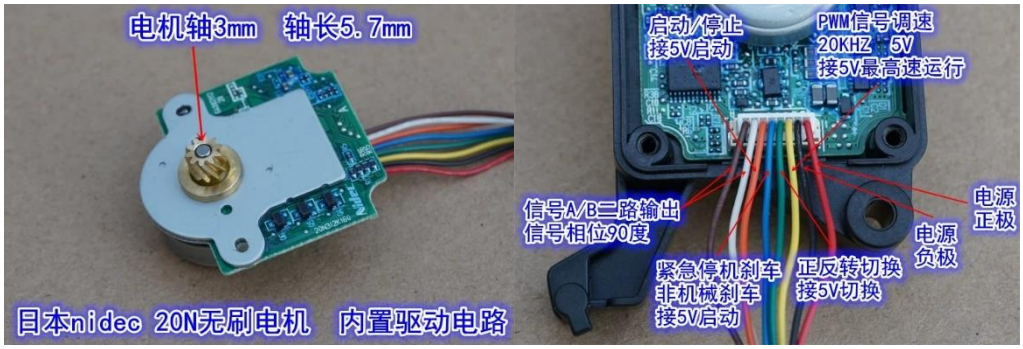
无刷电机参数：

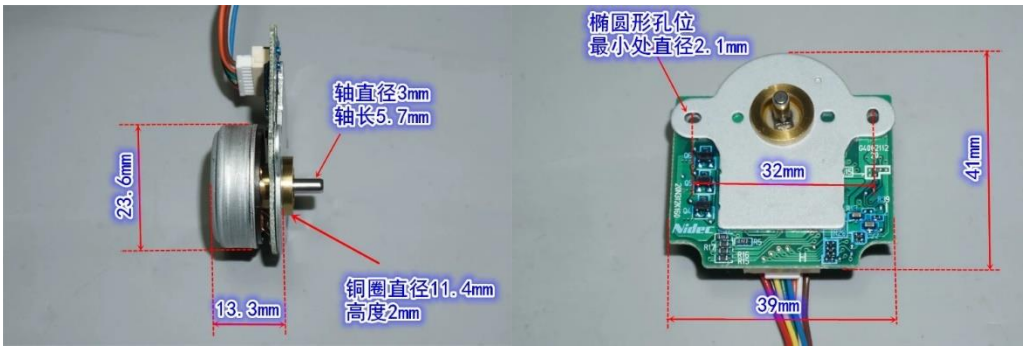
驱动系统	3 相双极性方波驱动 PWM 控制系统
最大功率	10W
最大扭矩	0.0385N*m
电机驱动电压	12-24V
控制系统电压	5V（经实验 3.3V 也支持）
编码器	100 线
尺寸	直径 42mm，长度 39mm
重量	140g

作为一款三棱柱套件，拆机无刷电机即可满足我们的使用，至于全新的该款无刷电机，我们经过长时间寻找调研，发现这是一款用在打印机上的电机，无法找到销售渠道，目前只能用拆机的电机，虽然使用的是拆机电机，但无刷电机的特性决定其运行平稳、安静、精准、寿命长。

2. 迷你版无刷电机

对于迷你版三棱柱，它使用的无刷电机的尺寸和接线图如下：





四、 侧向平衡调试

在进行侧向调试前先讲一下为什么没有用 DMP，在我调试三棱柱侧向平衡时，过程比较曲折，一直运行失调，起先讲错误归咎于电机，但换了两次电机，不管是直流还是无刷都没有什么效果，加低通滤波算法无果，请教大佬也没有好的建议，但发现使用电位器反馈侧向角度可以正常运行，但陀螺仪反馈不能成功，于是讲问题锁定在陀螺仪上，发现 mpu6050 的 DMP 滤波器低通特性过强，使角速度反馈高频段严重失真，而这几乎肉眼无法辨别，抛弃 DMP，读取原始数据，使用互补滤波算法，修改滤波参数，终完成由陀螺仪反馈的侧向平衡。可能是个人水平有限，如果各位大佬能使用 DMP 搞出来的，还望请教。

根据角动量守恒原理，三棱柱往左倾时惯量轮轮要往左运动，三棱柱往右倾时惯量轮要往右运动。由于三棱柱到达平衡的位置之后因为惯性会往另外一个方向倒，所以不仅需要在电机上施加和倾角成正比的回复力，还需要增加和角速度成正比，与运动方向相反的阻尼力。为简化控制，加速度控制反映为 PWM 占空比。

调试技巧：

1.比例控制是引入了回复力；

2.微分控制是引入了阻尼力；

3.微分系数与转动惯量有关；
4.三棱柱质量一定，重心位置增高，比例控制系数下降，微分控制系数增大；
5.重心位置一定， 质量增大，比例控制系数增大，微分控制系数增大。
平衡三棱柱侧向直立环使用 PD (比例微分)控制器，下面是直立 PD 控制的代码:

```
int balance_x(float Angle,float gyro)
//倾角 PD 控制 入口参数：角度 返回值：倾角控制 PWM
{
    float Balance_KP=230,Balance_KI=0,Balance_KD=-1.6;
    float Bias;                                  //倾角偏差
    static float D_Bias,Integral_bias;           //PID 相关变量
    int pwm;                                     //PWM 返回值
    Bias=Angle-zhongzhi;                         //求出平衡的角度中值 和机械相关
    Integral_bias+=Bias;
    if(Integral_bias>30000)Integral_bias=30000;
    if(Integral_bias<-30000)Integral_bias=-30000;
    D_Bias=gyro;                                 //求出偏差的微分 进行微分控制
    pwm=Balance_KP*Bias+Balance_KI*Integral_bias+D_Bias*Balance_KD;
     //计算倾角控制的电机 PWM
    return pwm;
}
```

调试过程包括确定独轮平衡三棱柱的机械中值、确定 kp 值的极性和大小、kd 值的极性和大小等步骤。

在调试直立环的时候，要在定时中断服务函数里面屏蔽速度环：

```
Balance_Pwm_x=balance_x(Angle_x,Gyro_x);    //角度 PD 控制
//velocity_Pwm_x=velocity_x(Encoder_x);     //速度 PI 控制
```

1. 确定三棱柱侧向的机械中值

把平衡三棱柱（此时其不会前后倾斜，各位客官老爷想想办法哈）放在地面上，关闭电机 pwm，记录能让三棱柱接近平衡的角度，一般在 91° 附近（三棱柱左右并非对称），所以就是 zhongzhi=91；

2.确定 kp 值的极性与大小(令 kd=0)

首先我们估计 kp 的取值范围。我们的 PWM 设置的是 7200 代表占空比 100%，假如设定 kp 值为 360，那么三棱柱在±5 度的时候就会达到最大输出。我们大概估算一下感觉还可以，首先给个试探值 kp：-100，可以观察到，三棱柱往哪边倒，电机会往那边加速让三棱柱倒下，说明 kp 值的极性反了，接下来设定 kp=100,这个时候可以看到三棱柱有直立的趋势，具体的数据接下来再仔细调试。

我们将 kp 值增加，直到出现大幅度的低频抖动，最终确定 KP=230；

3.确定 kd 值的极性与大小(令 kp=O)

通过观察 MPU6050 角速度原始数据，最大值不会超过 4 位数，再根据 7200 代表占空比 100%，所以估算 kd 值在 10 以内，先设定 kd 值为-1，kd 值的作用在此表现为施加阻尼

力阻碍三棱柱的左右摆动，但此时表现的时加速三棱柱的摆动，说明了 kd 的极性反了。接下来设定 kd=1，当我们左右摇摆三棱柱的时候，惯量轮旋转产生阻碍三棱柱运动的力量，可以确定 kd 的板性是正的，具体的数据接下来再仔细调试。

我们让 kd 一直增加，直到三棱柱出现高频抖动，这时我们把 kp 值也增加进去，最终取值为 kp：230，kd：1.6。

4.确定 kp 与 ki 值的极性与大小

在前面调节直立控制后，我们虽然看到惯量轮能提供反扭力让三棱柱维持直立一段时间，但是，惯量轮最终会不断加速，到电机的速度极限后，电机不能进一步加速提供反扭，三棱柱最终还是倒了。

接下来关闭已经调试好的直立控制部分：

```
//balance_Pwm_x=balance_x(Angle_x,Gyro_x);    //角度 PD 控制
velocity_Pwm_x=velocity_x(Encoder_x);          //速度 PI 控制
```

速度 PI 正反馈控制函数：

```
int velocity_x(int encoder)
//位置式 PID 控制器 入口参数：编码器测量位置信息，目标位置  返回  值：电机 PWM
{
    float Position_KP=25,Position_KI=0.015,Position_KD=0;
    static float Pwm,Integral_bias,Last_Bias,Encoder;
    Encoder *= 0.65;                                    //一阶低通滤波器
    Encoder += encoder*0.35;                            //一阶低通滤波器
    Integral_bias+=Encoder;                             //求出偏差的积分
    if(Integral_bias>20000)Integral_bias=20000;
    if(Integral_bias<-20000)Integral_bias=-20000;
    Pwm=Position_KP*Encoder+Position_KI*Integral_bias+Position_KD*(Encoder-
Last_Bias);          //位置式 PID 控制器
    Last_Bias=Encoder;                                  //保存上一次偏差
    return Pwm;                                         //增量输出
}
```

积分项由偏差的积分得到，所以积分控制和比例控制的极性是相同的。以上代码实现的效果是：使三棱柱在保持侧向平衡的同时惯量轮速度为零。调试过程包括计算速度偏差、确定 kp 和 ki 值的极性(也就是正负号)与大小。

我们通过 STM32 定时器的编码器接口模式对编码器进行四倍频，并使用 M 法测速(每 6ms 的脉冲数)得到三棱柱惯量轮的速度信息，这里的速度控制是正反馈的，当三棱柱侧向倒下的时候，我们要让三棱柱回正，惯量轮需要更大的反扭力去矫正，所以这是一个正反馈的效果。

怎么判断速度控制是正反馈还是负反馈？先设定 kp=20，ki=0.005，拿起三棱柱，旋转三棱柱惯量轮，惯量轮会加速,直至电机的最大速度，这是正反馈，也是我们期望看到的。至此，我们可以确定 kp，ki 的符号应该是正的。

5.确定 kp 与 ki 的大小(开启直立控制)

下面我们进行三棱柱惯量轮速度控制 kp 与 ki 值的整定，此时需要打开直立环，因为我们需要结合直立环观察速度环对直立环的影响：

```
Balance_Pwm_x=balance_x(Angle_Balance_x,Gyro_Balance_x);     //角度 PD 控制
velocity_Pwm_x=velocity_x(Encoder_x);                        //速度 PI 控制
```

调试的理想结果应该是:三棱柱保持侧向平衡的同时，惯量轮速度接近于零。实际上我们想像一下，这当然无法实现！三棱柱会受到扰动，惯量轮需要一直旋转来提供反扭力使三棱柱平衡，不过我们速度控制的目的在于让惯量轮的速度不要太大，超出 3000 转每分钟，电机就没有办法再加速提供反扭力了。

根据工程经验，ki 值大约为 kp 值的 200 分之一，所以大家可以以这个为参考。

首先，设定 kp=10,ki=0.005；三棱柱的惯量轮速度控制比较弱，很难让惯量轮速度维持在较小值。

设定 kp=15,ki=0.01；三棱柱的速度惯量轮控制的响应有所加快，但是速度来回摆动还是有点大，还是不足以让惯量轮速度维持在较小值。

设定 kp=25,ki=0.015；三棱柱已经性能很完美了，接下来加大 kp 值看一下效果。

设定 kp=50,ki=0.02；这个时候我们可以看到速度正反馈过于强，惯量轮速度已经发散。

以上内容就是我在调试三棱柱时的一些心得了，时间紧张，一方面我加紧继续完善，另一方面有不足之处欢迎各位大佬批评指正。