平衡车只靠MPU6050传感器反馈的位置角度是无法保持静止平衡的，必须结合电机的速度反馈才能实现稳定的自平衡。电机测速一般采用霍尔编码器和光电编码器两种方式。后者的精度要比前者高，价格也贵些。我这里使用的是霍尔编码器。（配图）

两个电机有两个编码器，分别接到树莓派的gpio12，gpio16，gpio20和gpio21上。（图）

编码器有两相：A相和B相。它们在波形的输出上相位正好相差90度，我们就是利用这个相位差来判断电机的正反转的。如下图是电机正转的两相波形图。（图）

图中A相领先B相，A上升沿时B都是低电平，B上升沿时A是高电平，A下降沿时B是高电平，B下降沿时A是低电平。也就是说符合上面组合的都表明此时电机是正转。

下图是反转的两相波形图。（图）

刚好和上面的相反，这里就不罗嗦了，大家自行分析。

通过AB相配合可以实现4倍频测速，即在每一个跳变沿都计一次数，这样在一个周期内可以有4次计数，这样可以提高测速的精度和速度。当然也可以使用2倍频，既AB分别计数。有些单片机处理速度不够快，可以采用2倍频。

为了提高响应的及时性，我采用GPIO中断来捕获编码器的信号。Linux系统的中断是由内核管理的，在应用层无法使用，所以必须在内核编程。这相当于是编码器的驱动程序，采用内核模块很合适。关于GPIO中断和内核模块的内容可以查看我的另一篇专栏。

我在树莓派上采用4倍频测速，代码的逻辑跟上面描述的一样，如果判断出是正转就将编码器计数变量加1，反转就减1。

得到编码器的值后需要从内核层传到应用层，这里我创建了一个字符设备来完成这个功能。在应用层我们只需定期打开并读取这个字符设备就能得到编码器的值。关于字符设备的内容可以查看我的另一篇专栏。

编码器的计数与电机的转速是正比关系，所以我们不需要非得算出电机的具体转速值。这个计数值就可以直接作为PID控制器的速度反馈量。

编码器这块儿涉及到Linux两个重要的内容——中断和字符设备，还需要了解内核模块和内核编程，对于开发者还是有一定要求的，许多小的知识点需要掌握。不过现在有许多关于这些内容的书，网上也有很多资料，相信是难不住大家的。

OK，关于编码器就说到这里吧，希望对大家有所帮助，喜欢的小伙伴就三连支持一下吧。

限于个人水平，难免有错误和遗漏，欢迎大家交流指正。