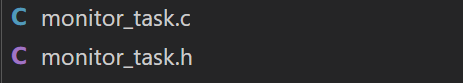
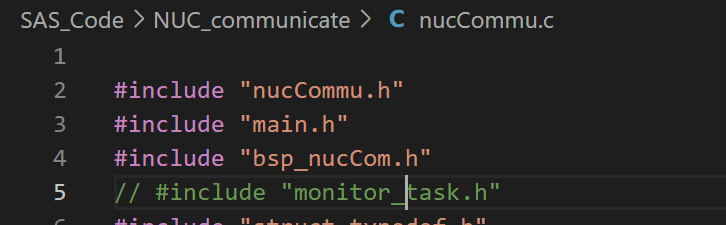
30.2版本，灯不亮了。Orz。找一个能亮的版本，复制成30.3，把30.2的文件一个个增进去。看到底哪里有问题。

1、删除monitor\_task





修改nuc com里的数据



编译，灯亮。

2、新增gimbal task

只增加必要的运行相关代码，调试的先不管

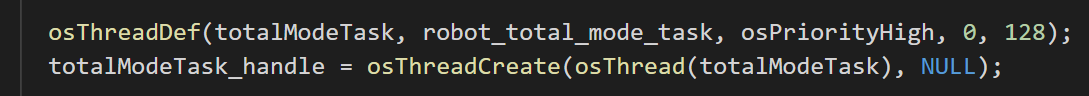
只增加头文件，移植相关包含文件后，灯亮

增加所有define，灯亮

增加全局变量，灯亮

3、增加mode相关定义和任务

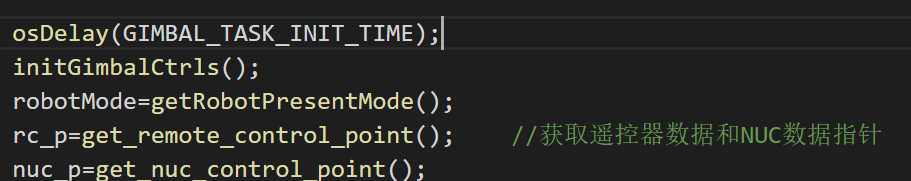
Freertos里手动增加任务



Uvision里增加文件。灯亮

增加gimbal所有函数定义，灯亮

任务函数增加指令

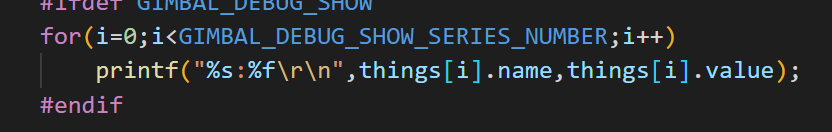


灯亮

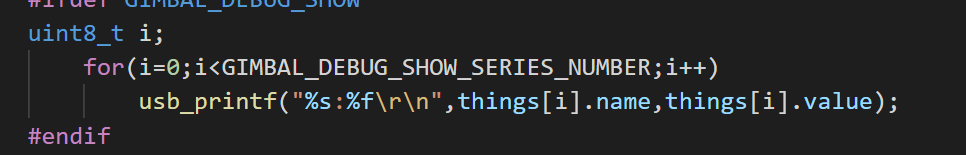
增加调试数据

傻瓜错误：

Usb里用了printf



改:

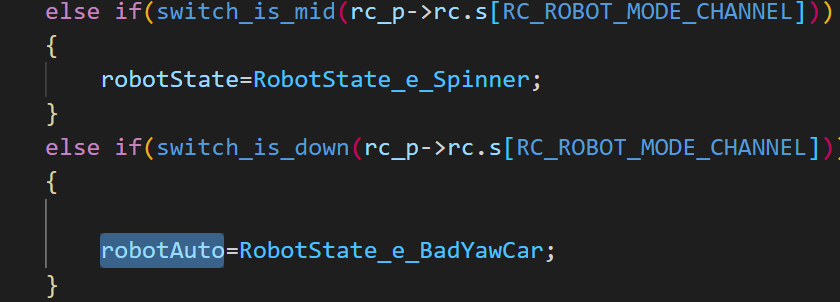


灯亮

新问题，磁力计掉线

而切换到空机器人v30.版本程序上，却是上线，怀疑是内存不够，发生了内存覆盖。把usb、gimbal的栈空间倍增了，解决了问题。

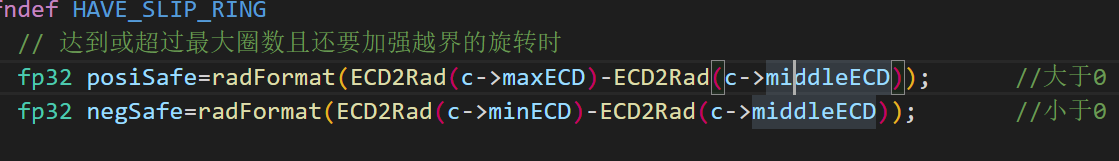
还有这种变量名写错了的



新问题，按下z键机器人会死机

不管重启

为什么这个地方写了中值？



啊这，改写当前值

Debug很久，无果。

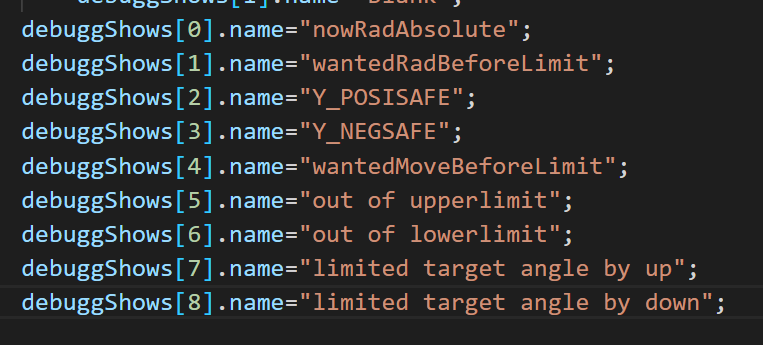
头晕，先睡一睡吧

睡醒了，在纸上写下想要看的值。而后根据这个目标，编写usb输出程序。最终得到

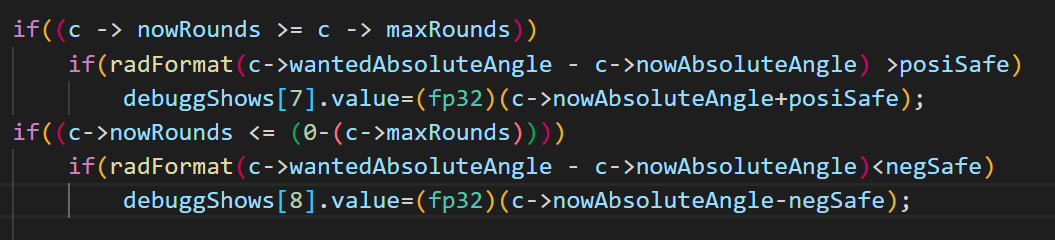




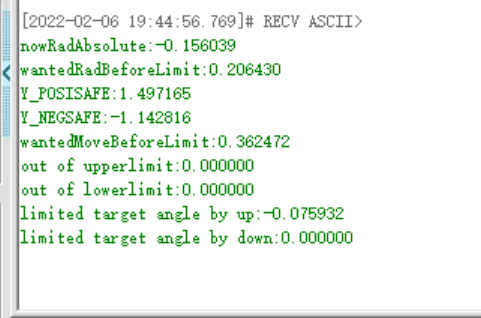
这里，似乎limit操作直接将范围限制到了边界上。



不知道出啥问题了，等价地把那个条件判断写成了



然后输出就符合预期了。



真的奇怪。不知道它这编译器怎么一回事。还是写一写常见、短小、简单的程序吧。毕竟不能和编译器硬杠。我们要和编译器合作，做出一些东西。

这也说明了我们测试的目的：是为了验证我们“合作”的结果对不对，而不只是我做出的东西对不对。因为编译器或者程序有时候会出问题。Orz

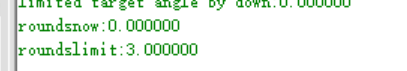
好吧，这就OK了。

好像去年在比赛现场，不知道为什么一个判断语句，人工看代码判断根本不可能进去，而机器它执行的时候就进入了。好像也是有一个双&在那里。

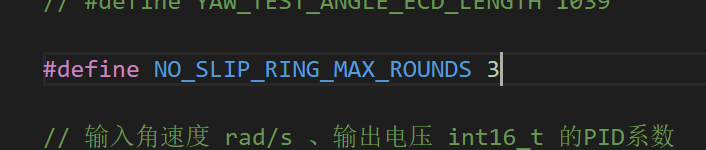
啊这，我们已经进入工业界了吗？

还是不对，为什么没有限制？

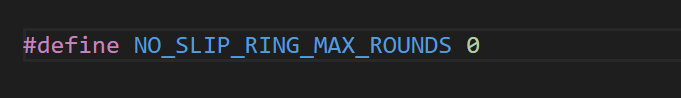
再次输入变量的值，得到：



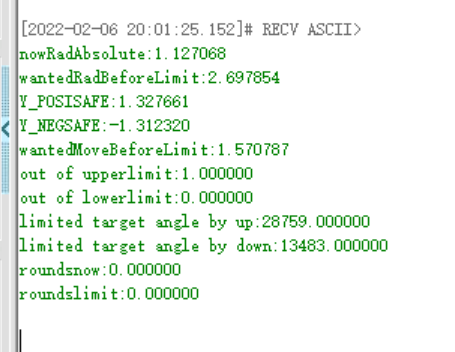
哇，我以为自己已经把它改成0了，orz



好，改



好了，进入的时候我们让它显示时间，那么就ok了



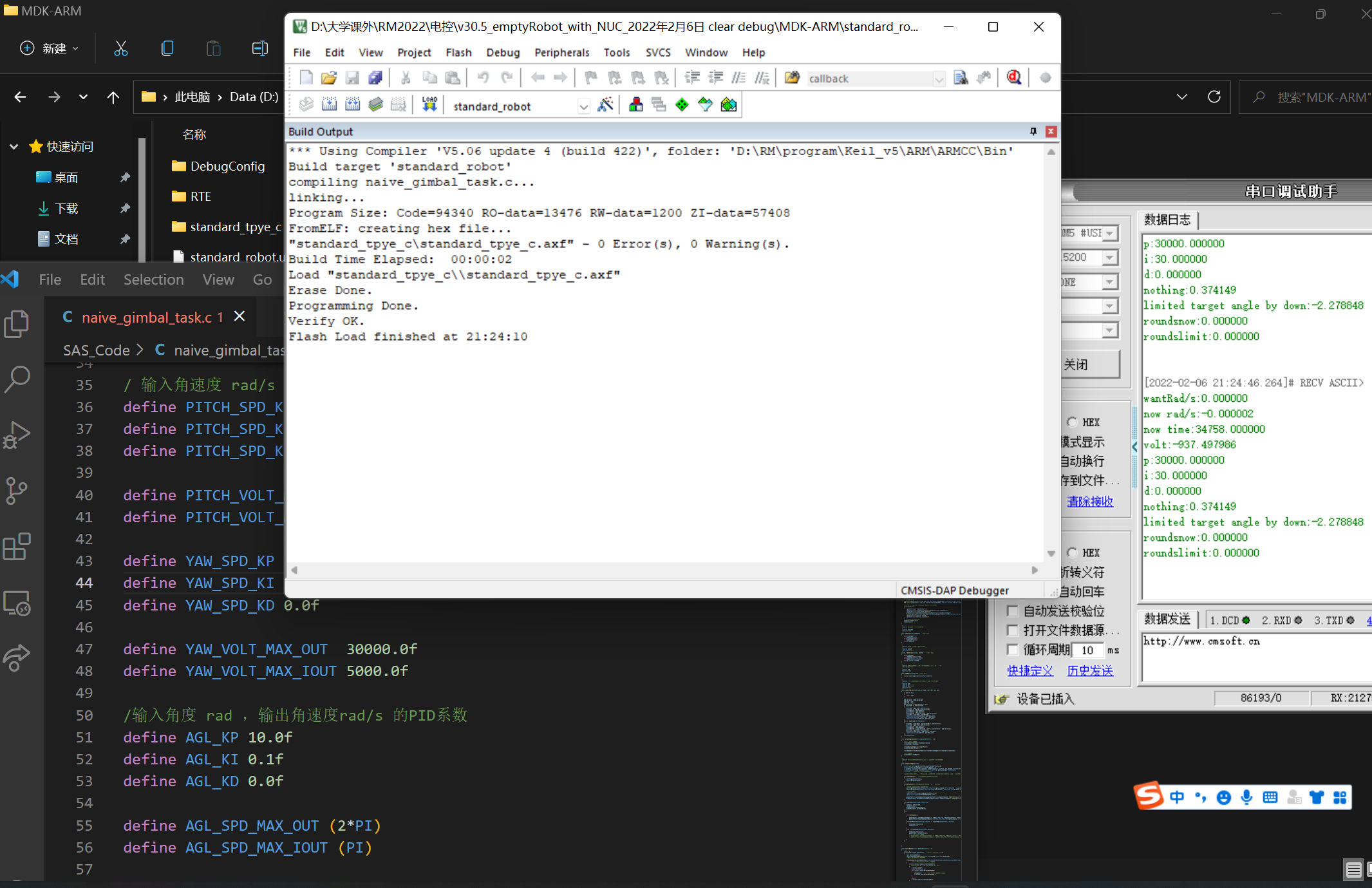
好，赶紧存档一下。

越懒会花越多时间。好好把每个变量打印出来看一看，不就好了吗？

大概角度限制在这里，还有很多问题。

看看云台电压控制了

调PID用的过程和方法



还挺好用，先调速度环，而后调角度环

结果暂定这样

要是轴承阻力小一些会更好一些

// 输入角速度 rad/s 、输出电压 int16\_t 的PID系数

#define PITCH\_SPD\_KP 30000.0f

#define PITCH\_SPD\_KI 300.0f

#define PITCH\_SPD\_KD 0.0f

#define PITCH\_VOLT\_MAX\_OUT  30000.0f

#define PITCH\_VOLT\_MAX\_IOUT 5000.0f

#define YAW\_SPD\_KP 60000.0f

#define YAW\_SPD\_KI 30.0f

#define YAW\_SPD\_KD 0.0f

#define YAW\_VOLT\_MAX\_OUT  30000.0f

#define YAW\_VOLT\_MAX\_IOUT 5000.0f

是用遥控器的一个摇杆作为目标输出量。

结果后来调yaw角度时一卡一卡的，于是觉得应该是调得少了，来大点

// 输入角速度 rad/s 、输出电压 int16\_t 的PID系数

#define PITCH\_SPD\_KP 300000.0f

#define PITCH\_SPD\_KI 30.0f

#define PITCH\_SPD\_KD 0.0f

#define PITCH\_VOLT\_MAX\_OUT  30000.0f

#define PITCH\_VOLT\_MAX\_IOUT 5000.0f

#define YAW\_SPD\_KP 1500000.0f

#define YAW\_SPD\_KI 30.0f

#define YAW\_SPD\_KD 0.0f

#define YAW\_VOLT\_MAX\_OUT  30000.0f

#define YAW\_VOLT\_MAX\_IOUT 5000.0f

巨大参数150000！可是我们的机械太卡，只好这么大

再来角度环，这次先把角度弄小一点

设置了宏，定义时调yaw，不定义时调pitch

云台有问题，当云台自己达到限度时，会无视越界提示，直接越界。需要限制。

（1）

（2）云台功率限制算法：当大电流超过1s的时候，也就是卡住的时候，让PID控制器清零，陀螺仪清零

安装时，避免ECD的中心处有0存在。需要用检测程序提前检查。

重新加了个限幅

#define PITCH\_TEST\_RAD\_MIN (-0.43f)

#define PITCH\_TEST\_RAD\_MAX  (0.3f)

void limitAnglesSecond(void)

{

    // if(radFormat(gimbalPitchCtrl.wantedAbsoluteAngle)>radFormat(ECD2Rad(gimbalPitchCtrl.maxECD)-ECD2Rad(gimbalPitchCtrl.nowECD)))

    //     gimbalPitchCtrl.wantedAbsoluteAngle=radFormat(ECD2Rad(gimbalPitchCtrl.maxECD)-ECD2Rad(gimbalPitchCtrl.nowECD))-0.05f;

    // if(radFormat(gimbalPitchCtrl.wantedAbsoluteAngle)<radFormat(ECD2Rad(gimbalPitchCtrl.minECD)-ECD2Rad(gimbalPitchCtrl.nowECD)))

    //     gimbalPitchCtrl.wantedAbsoluteAngle=radFormat(ECD2Rad(gimbalPitchCtrl.minECD)-ECD2Rad(gimbalPitchCtrl.nowECD))+0.05f;

    if(radFormat(gimbalPitchCtrl.wantedAbsoluteAngle)>PITCH\_TEST\_RAD\_MAX)

        gimbalPitchCtrl.wantedAbsoluteAngle=PITCH\_TEST\_RAD\_MAX-0.05f;

    if(radFormat(gimbalPitchCtrl.wantedAbsoluteAngle)<PITCH\_TEST\_RAD\_MIN)

        gimbalPitchCtrl.wantedAbsoluteAngle=PITCH\_TEST\_RAD\_MIN+0.05f;

}

效果还可以。能避免大电流烧电机。当卡住时，功率达到了80W

但是未来可能遇到陀螺仪零漂的问题。需要继续解决。

最终结果：调试了PID参数，增加了限幅，收获了几个程序v30.5,v30.6,未来换了轴承后可以用来再次调试这个车上的云台PID。

最终，把获取的参数放到v30.7里，增加二次限幅，OK。

对了，还有一个特别重要的

if(rc\_p->rc.s[1]==3)

{

    PID\_clear(&(gimbalYawCtrl.spd\_pid));

    PID\_clear(&(gimbalYawCtrl.agl\_pid));

    PID\_clear(&(gimbalPitchCtrl.spd\_pid));

    PID\_clear(&(gimbalPitchCtrl.spd\_pid));

}

刚开始一段时间，pid参数中有nan存在。需要想办法给PID参数清零，而后才能计算出电压并控制电机。