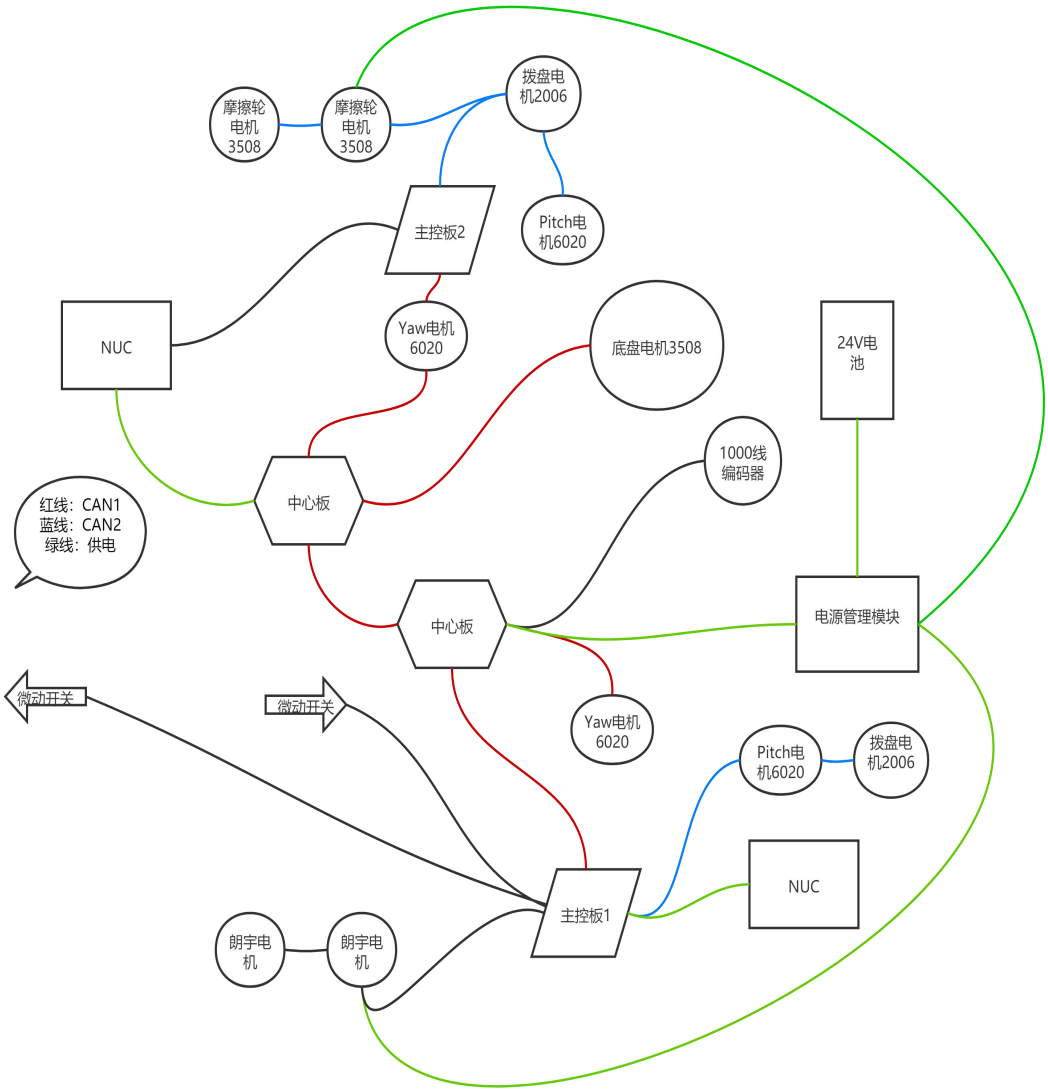
一、 硬件系统框图

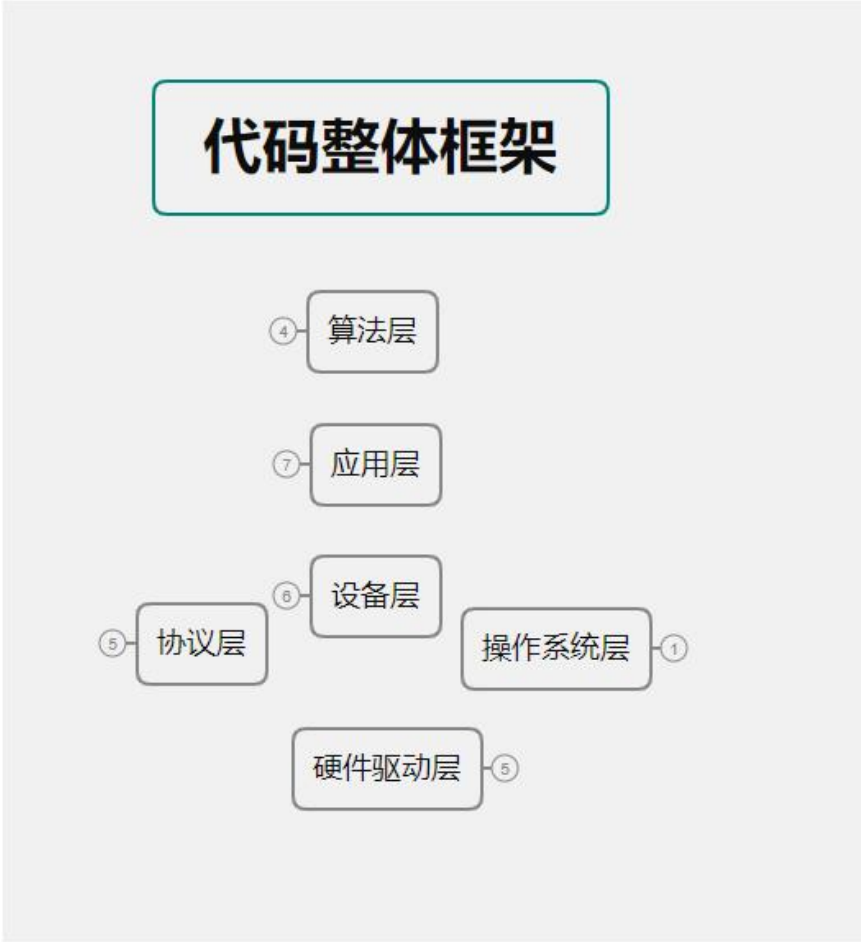
含主要布线和具体位置



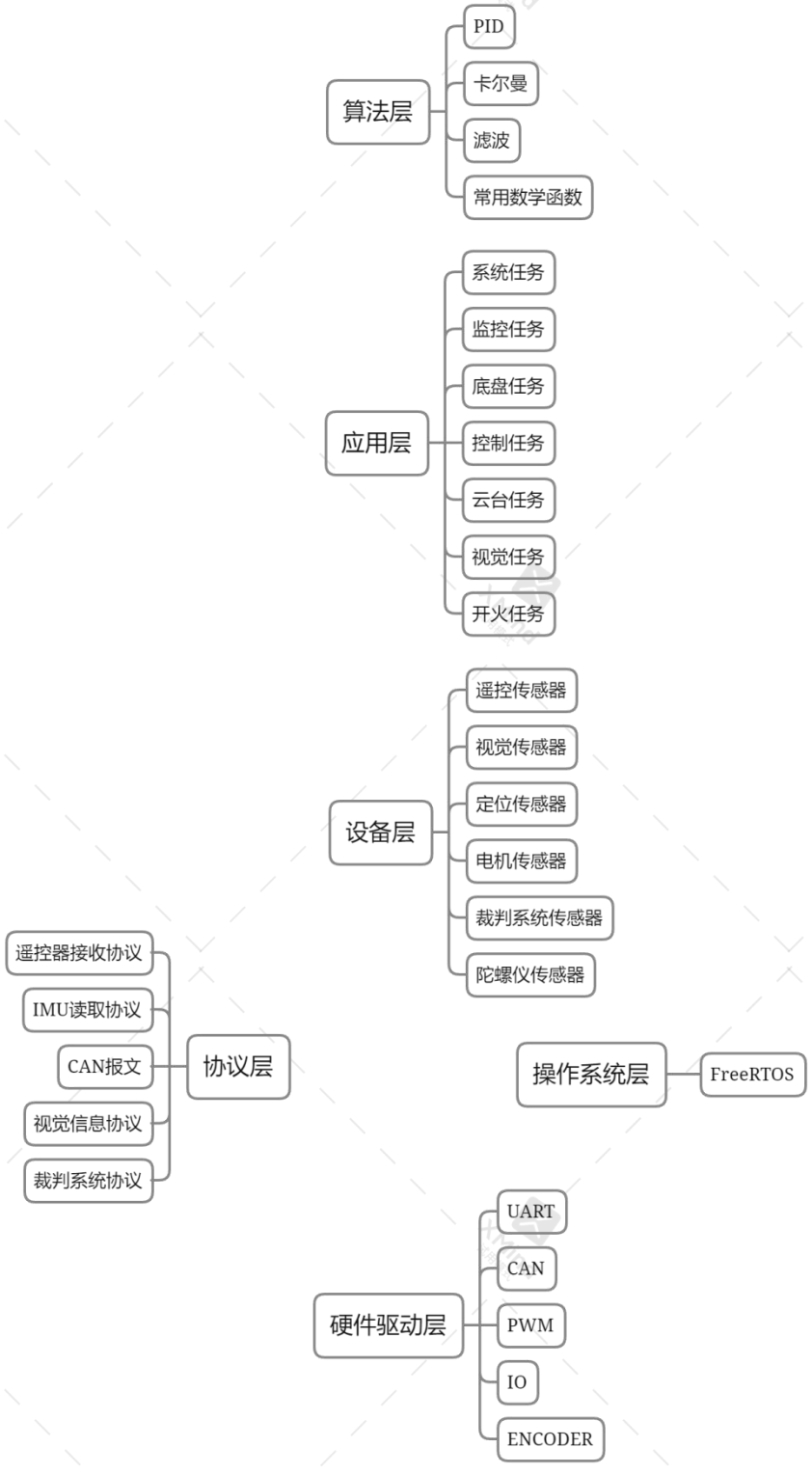
二、 嵌入式程序设计

1. 代码设计思想及框架

代码整体遵循分层设计思想，分为算法层、应用层、中间层（包含设备层、协议层、操作系统层）和硬件驱动层，层与层间不跨层调用，低耦合，高移植性，稳定性好。具体如下:

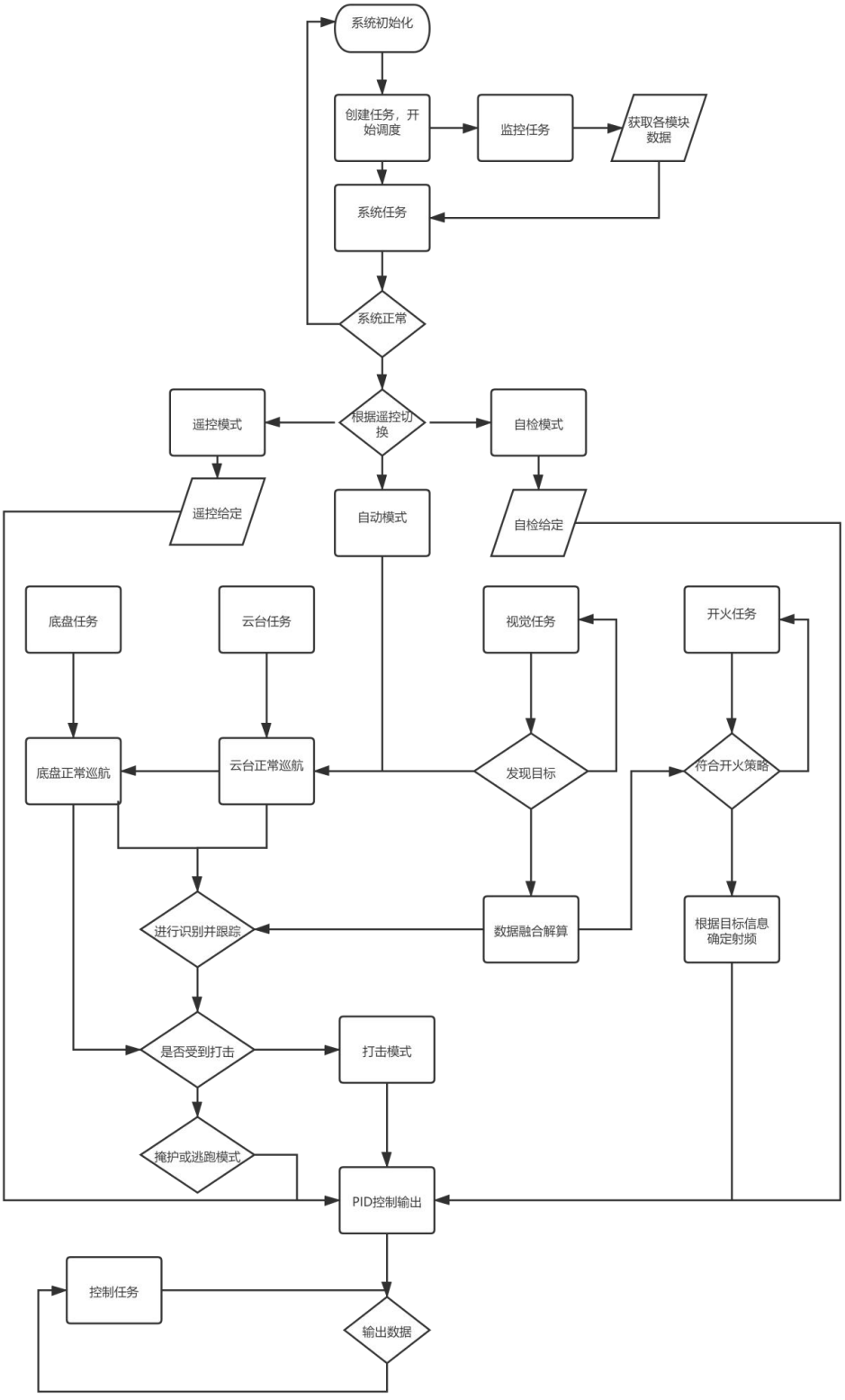


代码整体框架


算法层
PID
卡尔曼
滤波
常用数学函数
应用层
系统任务
监控任务
底盘任务
控制任务
云台任务
视觉任务
开火任务
设备层
遥控传感器
视觉传感器
定位传感器
电机传感器
裁判系统传感器
陀螺仪传感器
协议层
遥控器接收协议
IMU读取协议
CAN报文
视觉信息协议
裁判系统协议
操作系统层
FreeRTOS
硬件驱动层
UART
CAN
PWM
IO
ENCODER

2. 程序逻辑框图

主要包含各任务主要功能及逻辑，其次，工程中主要的部分也会分部分详细介绍



3. 双云台控制系统

在双云台哨兵中，为了实现更好的自瞄效果，我们上下云台都分别配置了 NUC 和摄像头。因此，会有两块独立的主控板分别安装在上下云台，来对整车进行协调控制。

在我们的双云台控制系统中，由于下云台主控可以连接裁判系统端读取信息，所以我们把下云台主控板作为双控中的主控，负责底盘运动定位、裁判系统端数据接收和下云台自瞄打击功能。而上云台主控板作为双控中的从控，独立负责上云台自瞄打击功能，并获取视野回传给主控。

要想双云台控制有良好的效果，上下主控板间一定要有良好的通信交互，在此，我们采用 can 路实现上下通信。为了不使一条 can 路上的 ID 过多，干扰程序正常运行，所以布线需要特意设计。如上面硬件系统框图所示，上下主控通信 can1 路上挂载了 3 个电机。同时为了不占用额外资源，避免 can 发送阻塞，此时需要把上下云台通信内容尽量精简，插入到已有 can 发送函数中的空闲位置，不在 can 路上增加额外的通信 ID。

```
if(sys.state == SYS_STATE_NORMAL)
{
    Mode_Data = RP_SET_BIT(Mode_Data,1); //遥控正常
    if(sys.remote_mode == RC)
    {
        Mode_Data = RP_SET_BIT(Mode_Data,2);
        Mode_Data =RP_SET_BIT(Mode_Data,3);  //遥控模式 11  3
    } else if(sys.remote_mode == AUTO)
    {
        Mode_Data =RP_SET_BIT(Mode_Data,2);
        Mode_Data =RP_CLEAR_BIT(Mode_Data,3);//自动模式  01  1
    } else if(sys.remote_mode == INSPECTION)
    {
        Mode_Data =RP_CLEAR_BIT(Mode_Data,2);
        Mode_Data =RP_SET_BIT(Mode_Data,3);//自检模式    10  2
    }
```

```
typedef __packed struct
{
    uint8_t online : 1;//遥控连接位
    uint8_t mode : 2;//控制模式位
    uint8_t attack_colour :2;//识别颜色位
    uint8_t friction_now :1;//摩擦轮开启位
    uint8_t dial_now : 1;//拨盘开启位
    uint8_t fire_stop :1;//强制停止位
}master_mode_t;
```

如图所示，发送和读取都按位进行，极大节省通信资源，提高传输效率。

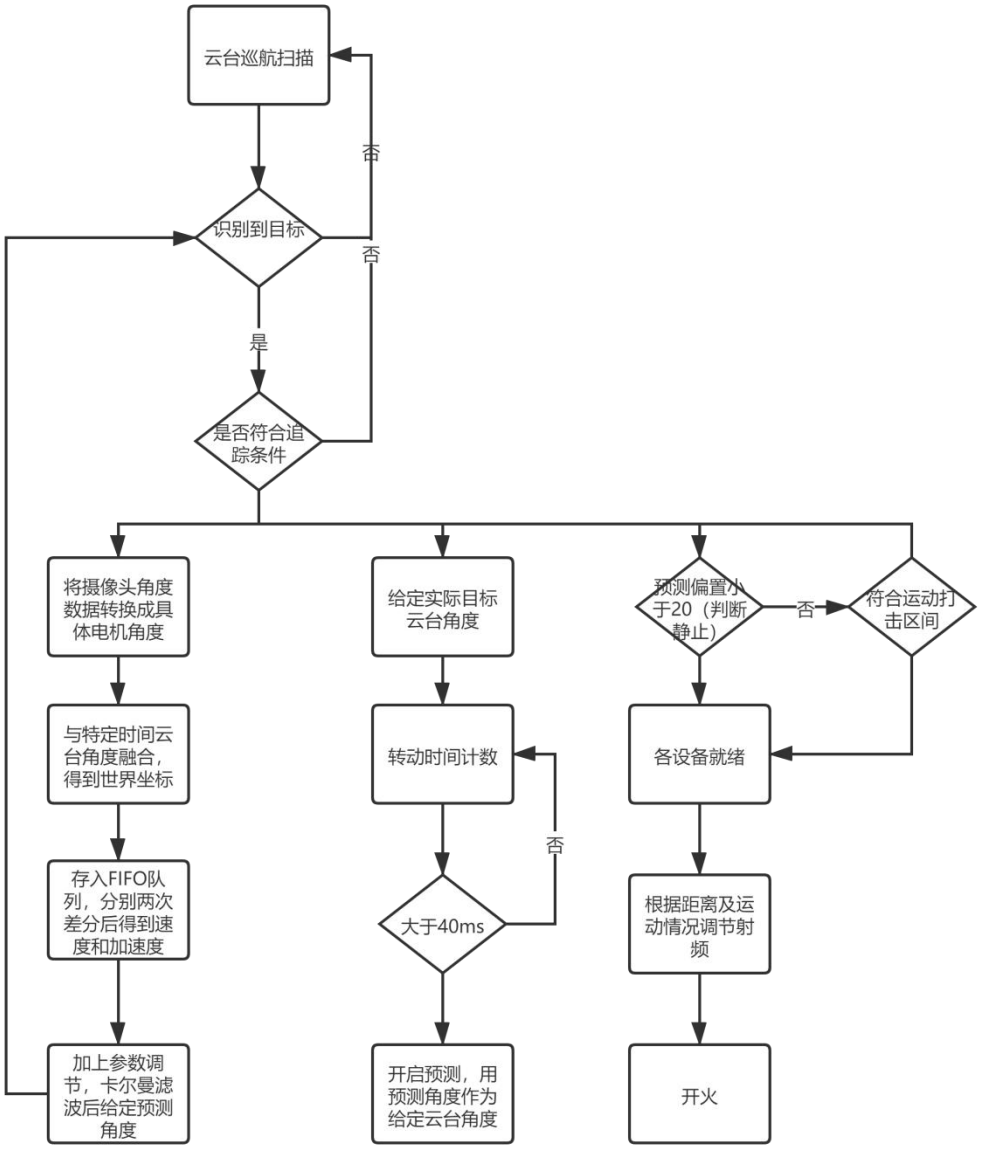
利用这套系统，可以做到只用一个遥控器就完整控制双云台，实现良好的调试控制效果，也方便比赛检录。此外，在通信链路或某个主控程序失灵的情况下，也能保证另一个云台正常工作，不至于完全瘫痪。

4. 自瞄开火系统

如下图所示，我们完整的自瞄打击逻辑是先由视觉方面处理图像，得到目标相对角度后回传给主控，在由主控程序实现自动开火打击。具体先要把相对角度转换为电机角度，用一阶卡尔曼滤波平滑曲线，再与特定时间点云台电机坐标融合，得到目标装甲板在世界坐标系下的确切坐标。此处需要将主控程序和电脑程序做精确的时间测量与校准，利用时间戳测出图像拍摄时间与主控接收时间有两帧的时间差，所以我们在融合坐标的时候回溯两帧前的云台角度，才能更接近真实数据。

预测处理方面，需要将世界坐标存入一个 FIFO 的队列中，新进数据与队列平均做差分，得到一个相对的速度角度。队列长度与速度实时性成反比，太长延迟过大，太短又容易受波动影响。经过具体测试，存储 3 到 6 帧的队列比较合适，在视觉数据在 120 帧以上时，可以保证数据的实时性和可靠性。同理可得目标的加速度，进一步提高自瞄的动态性能。

为了提高响应，在刚识别到目标时可以直接给定目标角度为云台目标位置，此时云台运动速度较快。避免云台过快对图像和预测数据产生干扰，此时应关闭预测，只使用真实角度。云台运动到一定区间时，即可进入开火策略判断，根据预测偏置大小和云台 PID 误差来判定目标装甲板是否相对静止或云台已经实现运动超前和进入开火区间。各方面准备就绪后就可以根据距离具体调节射频，直接开始打击。



具体命中率测试如下

距离	打移动目标（12 射频）	打陀螺目标（12 射频）
2m-5m	60%-80%	40%-60%
6m-8m	40%-70%	30%-60%

5. 双环底盘跑轨

为了尽可能提高哨兵跑轨速度，用以躲避弹丸打击，我们很多时候需要将速度拉满，但同时也会带来风险，比如容易耗尽缓冲能量、过于匀速以及刹不住车强行撞柱等问题，效果不好且容易造成机械损伤。因此，传统地面机器人纯给定速度来进行移动控制的方式不够适合挂载在轨道上的哨兵机器人，需要进行改进。

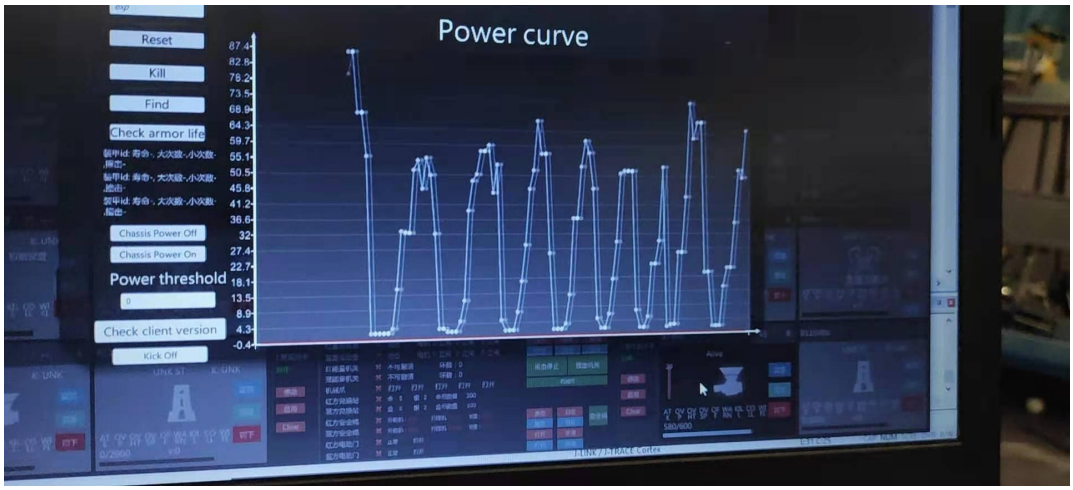
通过观察数控机床，可以借鉴其运行方式到跑轨运动中来。我们利用哨兵两端缓冲装置上的微动开关，以及添加一个 1000 线的编码器和从动轮连接。初始化阶段进行一

段全轨往返，通过两边微动开关的触碰和关于编码器的记录，就可以得到轨道上每点位置确定的坐标，精度可达到毫米级，多次往返误差可忽略不计。

与此配套，将移动方式改为定点移动，通过程序给定的坐标信息，送入双环 PID 控制器，外环给定坐标，输出期望速度，内环是传统的电机速度环控制，以此来不断动态调节速度，快速达到目标位置。

下面是进行全轨往返测试的数据，以及改进后的变向功率曲线。在双环跑轨控制下，跑轨速度得到很大的提升，而且撞柱前也可以自行调节，十分安全。变向时会根据坐标距离动态调节速度，在起步就把功率拉满，接近时再调节速度进行下一次变向。

控制方式	全轨往返时间（秒）
单环	5 - 5.3
双环	4.2 - 4.5



三、 团队配合设计

在 RoboMaster 的赛场上，多兵种协同作战才是主流。哨兵机器人作为全自动机器人，也需要和队友进行配合，在今年的赛场上，利用裁判系统端的车间通信功能，雷达站和云台手都可以给哨兵提供场上信息，哨兵主要通过以下四个功能和团队产生互动配合，执行战术。

1.强制退出自瞄：即使识别目标到也不打弹，达到欺骗作用，同时也防止工程车来吸引火力。

2.下云台向后侦察：正常情况下下云台的扫描范围是轨道前方，转动 140 度左右。但如果有敌方移到轨道后面盲区，可以通过雷达站或云台手可以给哨兵信息，下云台马上向后扫描，进行反击。这个功能有欺骗作用，当别人以为你不行的时候，其实你是真的行。

3.强制进入逃跑模式：在危险时刻，强制进入最高速度且不断变向跑轨。

4.移到轨道公路端：防止敌方车辆飞坡过来追杀，可以进行快速响应进行反击。

四、 开发调试工具

1.MDK5 + STM32CubeMX 开发
2.Zigbee 无线调试模块和上位机
3.自制 J-Link 和 J-Scope 软件
4.EventRecoder 组件