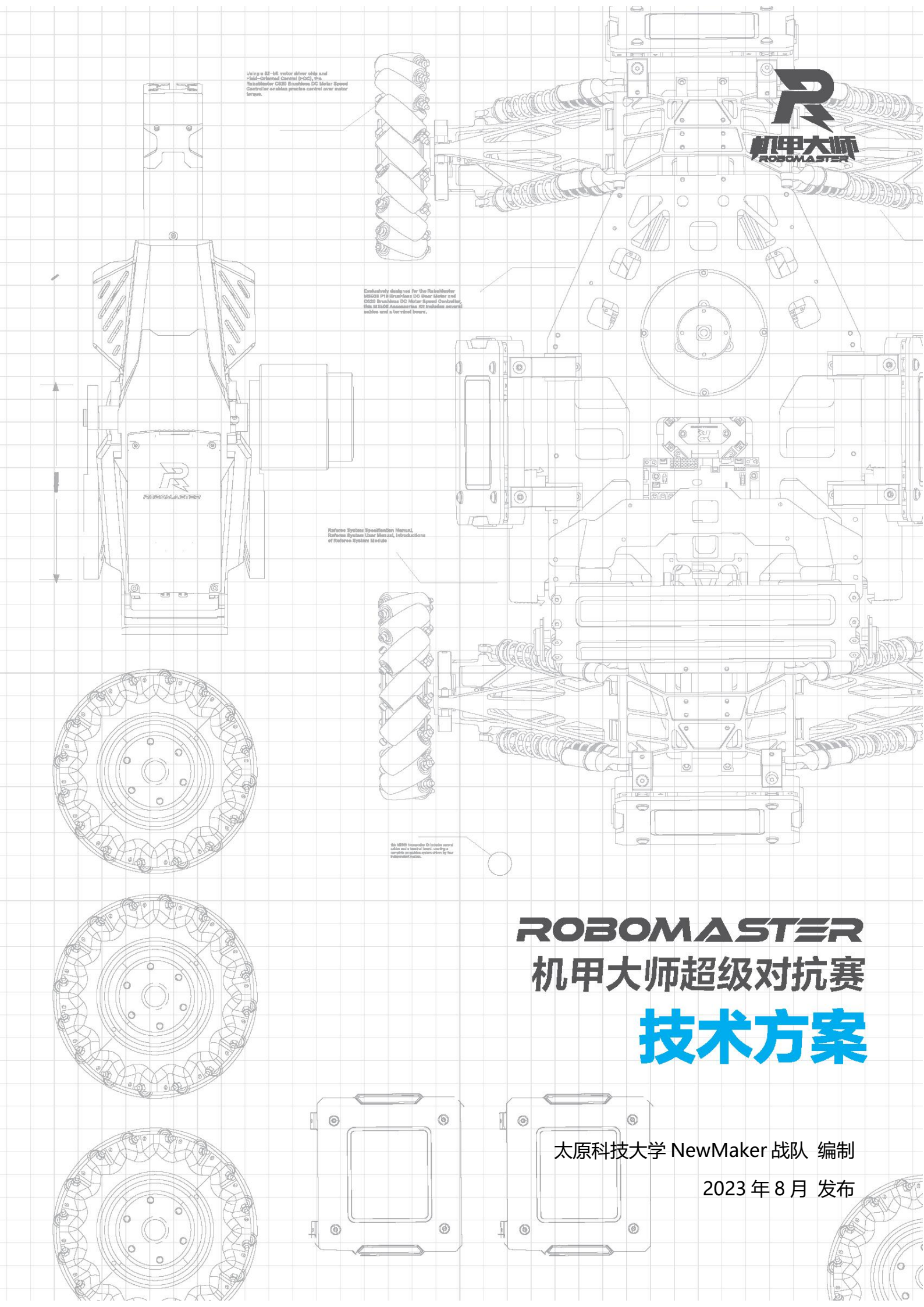


ROBOMASTER
机甲大师超级对抗赛
技术方案

太原科技大学 NewMaker 战队 编制

2023 年 8 月 发布

前言

本文档由太原科技大学 NewMaker 战队编制，适用于 RoboMaster 2023 机甲大师超级对抗赛。主要撰写人员包括：

模块	撰写人员 1	撰写人员 2
机械	吴嘉琪	胡佩泽
硬件	邢磊	
软件	许佳奔	
算法	郝威程	景佳柱
其他	喻文平	

目录

1. 概述

1.1 背景与目标

1.1.1 背景

哨兵机器人（以下简称“哨兵”）是 RoboMaster 机甲大师赛中非常重要的一环，在往届的比赛当中，哨兵运动在一维的轨道之上，运动方式相对单一，全自动的决策相对来说较少，而在 2023 赛季当中，哨兵脱离了一维的轨道限制，能够像步兵、英雄机器人一样在地面进行全方位移动，对参赛队伍的要求更是提高了一个档次，并且由于哨兵机器人没有操作手的原因，这就需要写出一套哨兵自己的决策方式，相当于让哨兵自主决策自己下一步要做什么。哨兵的下地让比赛变得更具多样性，也让比赛变得更加精彩、有看点，如果己方的哨兵不论是机械结构还是软件控制，亦或者是决策以及自瞄算法做到完善，那将会给己方的战术增加更多的可能，甚至决定比赛的走向。以上便是本技术方案的背景。

1.1.2 目标

NewMaker 战队（以下简称“我队”）在本赛季的哨兵技术方案当中有以下几个主要目标：

1) 我队计划在机械结构方面，采用双云台的机械结构，能够扩大搜索范围，同时也能够进行集火打击，更快速消灭敌方目标；底盘采用全向轮结构，在“小陀螺”模式下更加的稳定，同时具备全向移动的能力。

2) 我队在超级电容方面，采用我队自研的超级电容模组以及控制板，在规定范围内最大化利用电容，为哨兵提供更高的功率。

3) 我队在控制方面，首先最基本的就是功率、射速等限制，其次就是双头配合以及建图与自主路径规划，计划做到基础在巡逻区进行巡逻，如果研发时间足够，尝试哨兵根据时间等判断因素进行自主巡逻，在规定的时间达到我队的战略目标。

4) 我队在算法方面，基础做到自瞄跟随，但面对移动目标时，需要对其进行位置预测，因此我队计划在算法当中加入预测部分，以及分辨不同的机器人种类，实现对我们指定的目标进行打击。

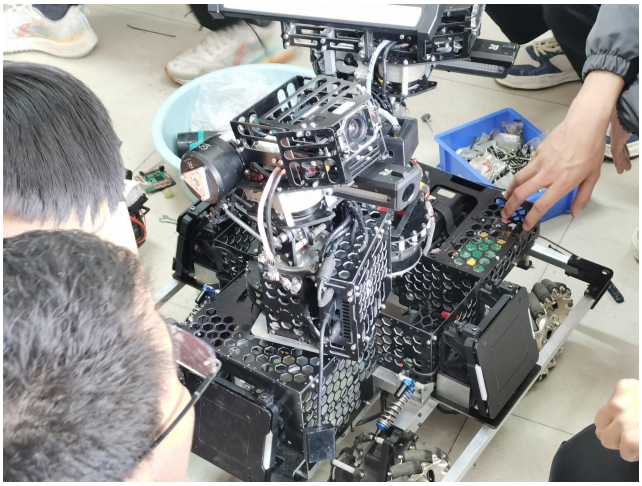
1.2 其它学校机器人分析综述

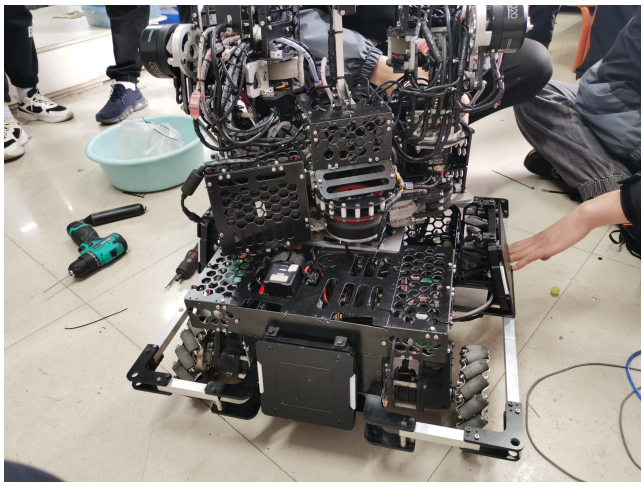
由于今年哨兵第一次下地，是一个较为新的点，因此开源等资料相对其他兵种还是少很多的，下面仅拿几个作为例子来分析。

1.2.1 底盘

1.2.1.1 麦轮底盘

麦轮底盘是 RM 传统的底盘设计，但其体积与重量可能成为一定的缺点，而体积大的显著缺点是小陀螺速度慢。但是，大面积的铝管作为车架是非常简洁可靠的方案，自适应悬挂或者是独立悬挂在结构上更简单、更轻，虽然其联轴器可能会导致同心度不好，但确实是轻便可靠的方案。性能的体现主要还是看各家设计者的想法以及测试，如果没有设计好的话可能就会产生轮子外八，对减震和小陀螺都会产生一定的影响。以下图示为太原工业学院哨兵机器人。

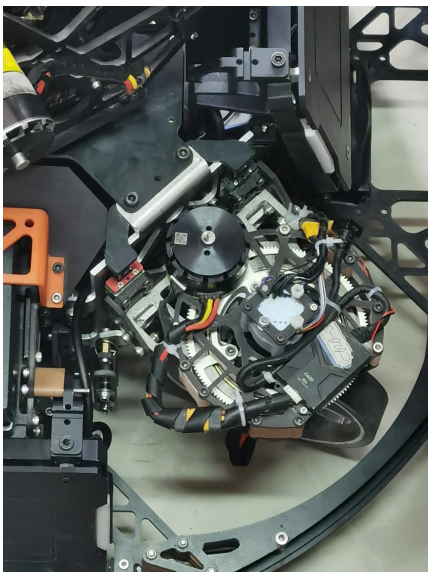


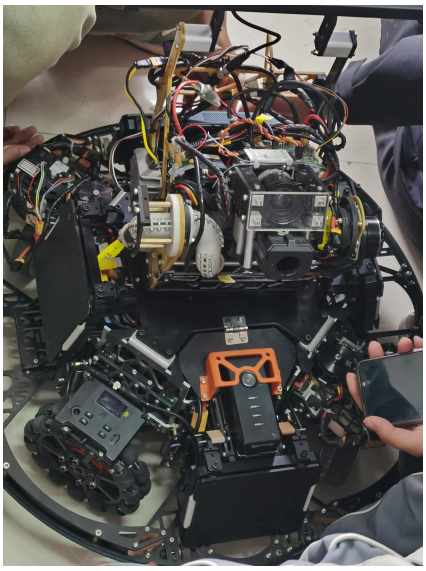


1.2.1.2 舵轮底盘

舵轮底盘是华南理工大学在 21 赛季带来的方案，当时那个赛季就有很多学校做出了许多优秀的舵轮步兵。

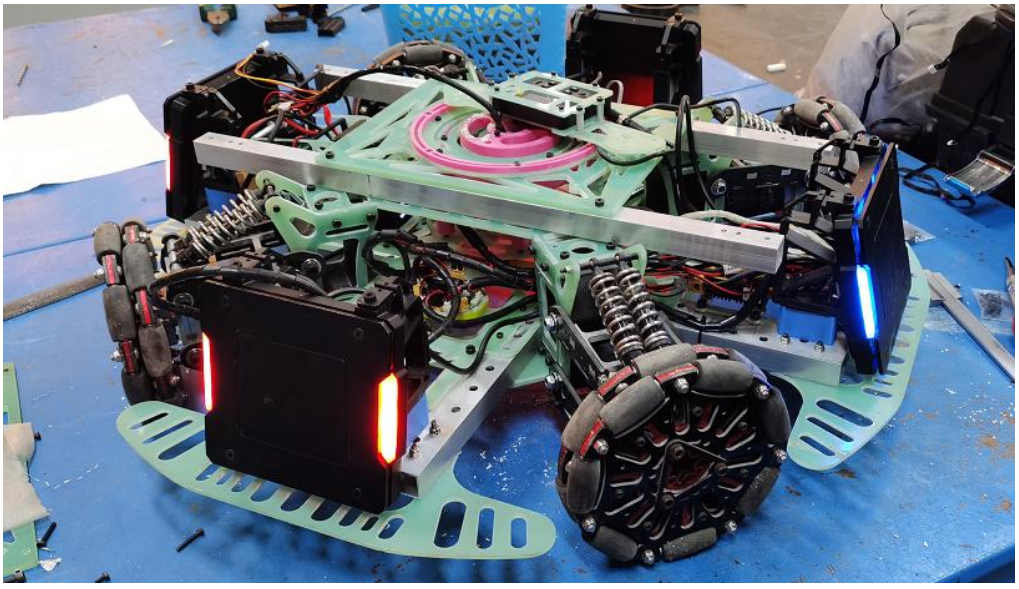
悬挂一直是舵轮底盘一个很大的不同，华南理工大学和深圳大学的开源中使用的是纵向排布的平行四边形悬挂，其中深圳大学采用的是自适应悬挂。广东工业大学和华南农业大学使用的是舵下的悬挂，有点事簧下质量很小，缺点是会导致轮轴距的变化。下图为太原理工大学的舵轮哨兵。





1.2.1.3 全向轮底盘

全向轮底盘是 21 赛季哈尔滨工业大学（深圳）提供的技术方案，其优点是小陀螺时转速很快，机动性很强，但显而易见的缺点是在“X”排布下前进的时候会有速度损失，如果是“十”字的排布下前进虽然没有损失，但前方横着一个轮子，过盲道或者上坡时通过性较差。



1.3 机器人功能定义


哨兵
全向轮底盘
形状，尺寸
圆形底盘 —— 灵活性高，小陀螺不易被逼停
大尺寸 —— 配合双云台，大底盘，重心稳定
通过性 —— 可上12°梯形高地
全向移动
能够以自身位置为原点，向任意方向移动且停止时的刹车距离低于普通四麦轮步兵的1.5倍
全速移动时，底盘功率平均值达到当前功率上限的 95%，方差为5% ，且在未启用超级电容的情况下全程不会因底盘功率超限而掉血
小陀螺功能
原地进行小陀螺时自身中心位置不发生明显移动
底盘抖动程度与普通四麦轮步兵相同
轮组及悬挂
152mm全向轮轮组
减震性能好，即使通过中央起伏路段，也能保证云台稳定性
云台
供弹链路 —— 鹅颈链路
链路采用板材拼接的方式，同时在链路中增加轴承提升链路的顺滑度
yaw、pitch轴采用滑槽过渡，在实现功能的同时简化滑槽复杂度
发射器限位 —— 采用弹性限位以保证发射顺畅且简化鹅颈中滑槽的设计难度
子云台 —— 两子云台在保证有一定交叉角的前提下，其间距尽可能缩小以减少底盘主yaw轴电机驱动云台的所需扭矩
弹舱 —— 空间利用率高 —— nuc位于弹舱底板与拨弹盘顶面之间的夹层
射击系统
中心供弹拨弹盘
射击稳定性高
连续射击1000发弹丸的卡弹率为0
最大射速稳定接近最高速26m/s
射速方差不大于2
雷达通信
发送 —— 发送追踪敌方机器人位置信息，为哨兵提供全局视野，辅助哨兵决策与路径规划
接收 —— 接受我方其他机器人信息并反馈给哨兵
超级电容
功率控制板
双向同步整流 BUCK-BOOST
主控采用STM32H743VIT6芯片，具有高分辨率定时器（Hrtim）
软件方面主控进行升降压变换器控制、采样监测、错误保护和通信等功能，采用Pi调节点电压环和电流环实现功率自适应和功率输出闭环控制，达到底盘功率利用最大化
电容组 —— 采用被动均衡成熟且稳定的方案，由9个2.7V法拉电容串联组成
其他功能
便于维护
小模块损坏1分钟内可修复
大模块损坏3分钟内可修复
稳定性高，不易损坏
整体外形美观

1.4 机器人核心参数

名称	参数
重量、重心	28.4kg，320mm
尺寸（长宽高）	640mm*640mm*663mm
主要传感器型号、参数、数量	1xLivox Mid-360
电路功耗、所有电容总容量、工作时电压范围等	静态功耗 3W，总容量 1968.3J，范围 7-24.3V
执行器件（电机、气缸等）用途与数量说明等	底盘：4*3508 YawPitch：5*6020 摩擦轮：2*3508 无减速箱 拨弹盘：2*2006
机器人其它核心性能参数，例如：车体最大移动速度、爬坡角度，云台自由度等	爬坡角度：35° 云台自由度：-45.3°——+25° 最大移速：3.768m/s

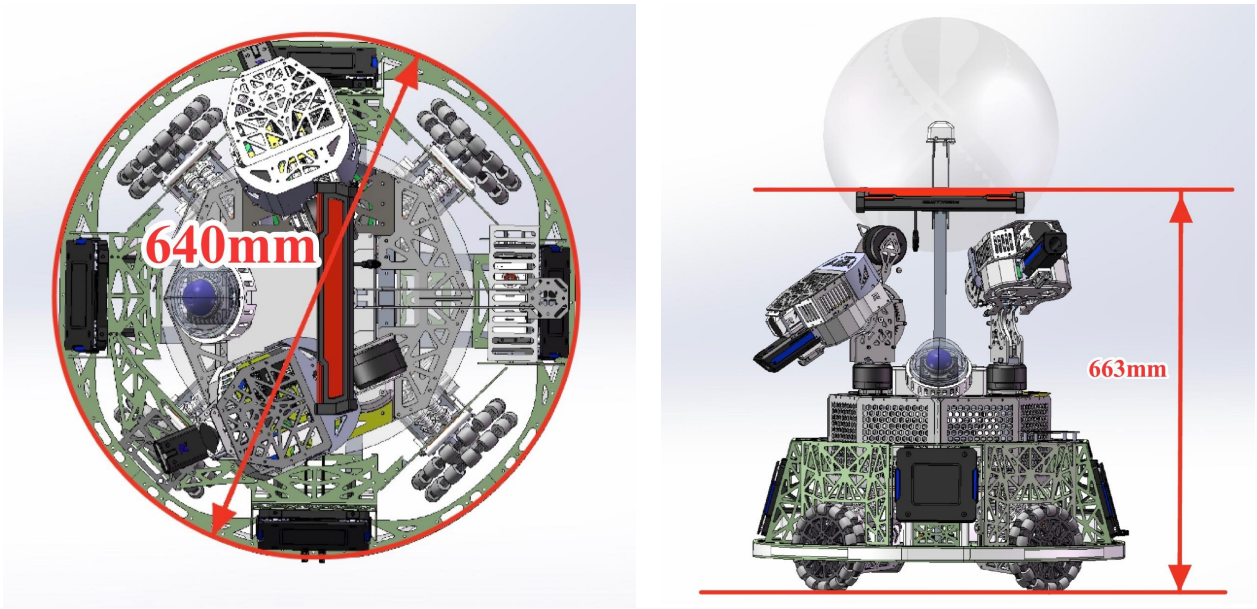


图 1：机器人长宽高及重心示意图

1.5 设计方案

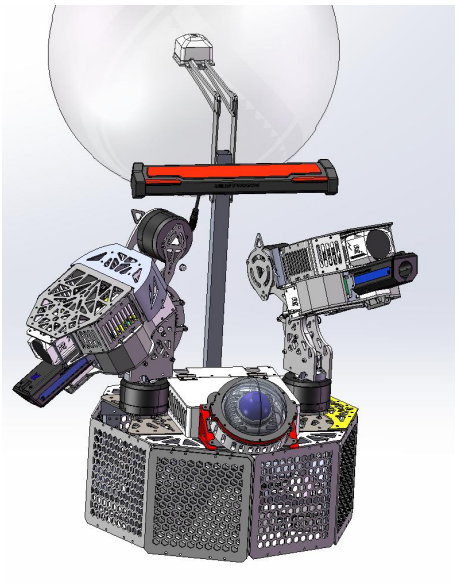
1.5.1 机械结构设计

1.5.1.1 整体机械结构

图 2：哨兵完整机械结构

本赛季哨兵为——双云台鹅颈链路中心供弹全向轮哨兵。

1.5.1.2 云台部分机械结构设计



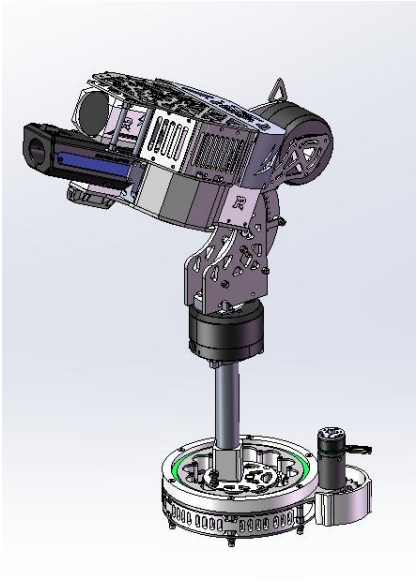


图 3、图 4：完整云台和单云台结构

设计方案：

云台部分由可 360°自由旋转的主 yaw 结构和两个可 120°自由旋转子 yaw 结构复合而成，通过主 yaw 子 yaw 配合转动可实现枪管 360°无死角打击，打击范围明显优于普通单枪发射和双枪发射。且可以同时击打多个目标或集中火力击打某一个目标，形成强力的火力压制。

如图所示，云台子 yaw 轴设计采用 GM6020 作为主要驱动机构，pitch 轴同样选用 GM6020 电机，设计简洁且可靠。

链路设计：

云台链路由铝管和板件链路、过渡滑槽滑片组成。

板材链路：此段链路宽度为 18mm，两侧安装型号为 2-5-2.5 的小轴承，轴承伸出量为 0.4mm，经实物验证弹丸可顺畅通过。

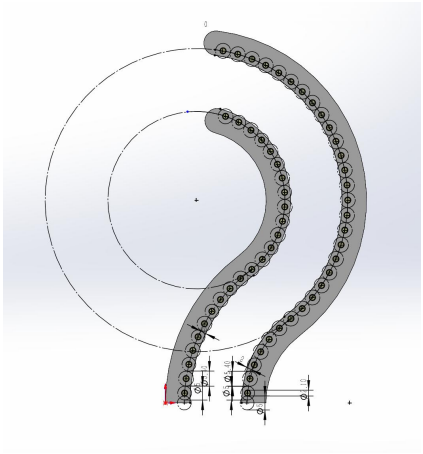


图 5：链路

铝管链路、过渡滑槽滑片：此段链路为连接板材链路和发射枪管的结构。在板材链路与 p 轴顶针座连接处设计滑槽、滑片，通过螺栓连接至槽口，一端与在槽口中滑动另一端绕顶针座转动。选用此种方式保证了 p 轴转动过程中，弹丸依旧可以顺滑通过。

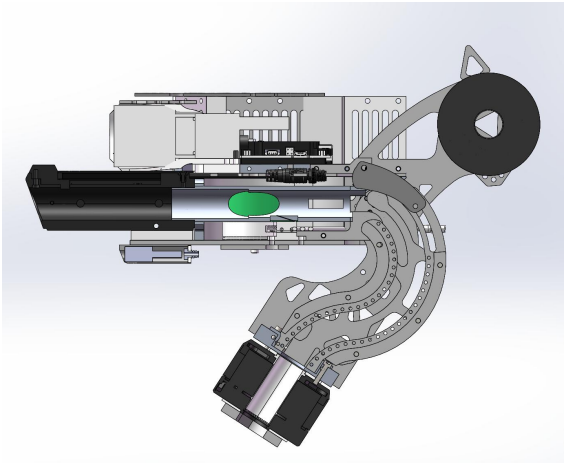


图 6：小 yaw

yaw 轴基座设计：为适配 6020 电机直连驱动方式，自主研发了一个 yaw 轴基座，通过基座本身累榫卯结构与四个六面体螺母的配合，减小了仅通过普通角铝的定位误差，保证了 yaw 轴板材链路与 6020 电机中空内壁的同轴度。

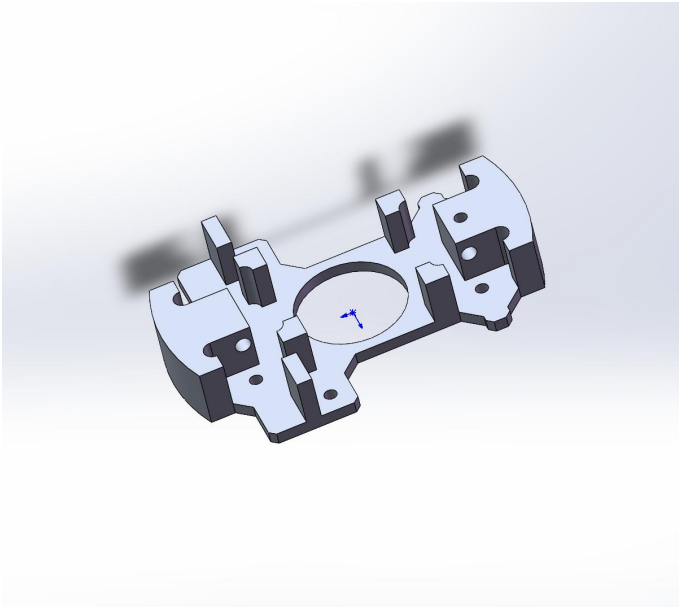


图 7：小 yaw 基座

1.5.1.2.1 中心供弹拨弹盘

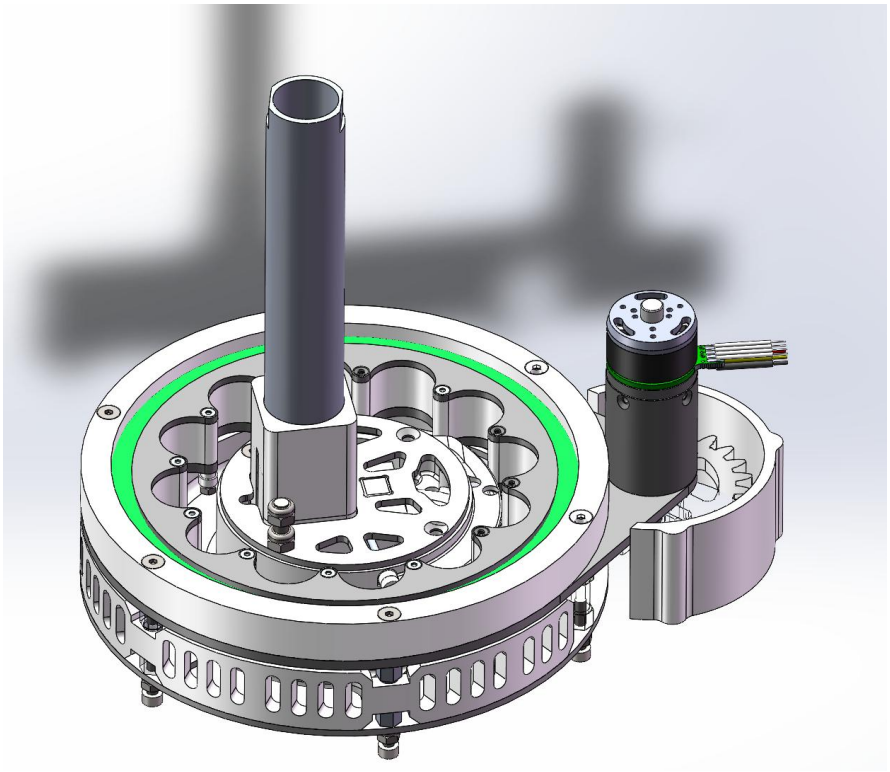


图 8：中心供弹拨弹盘完整机械结构

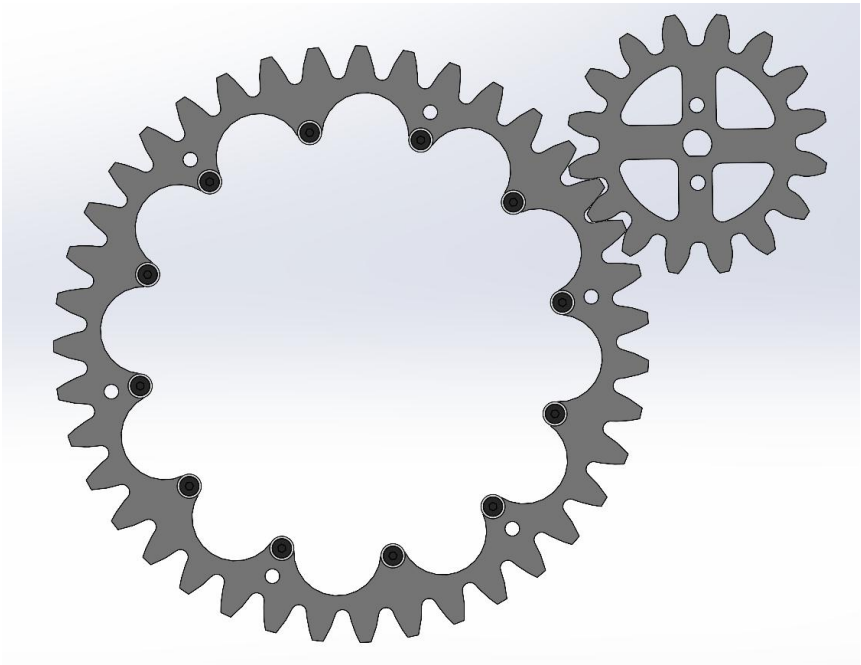
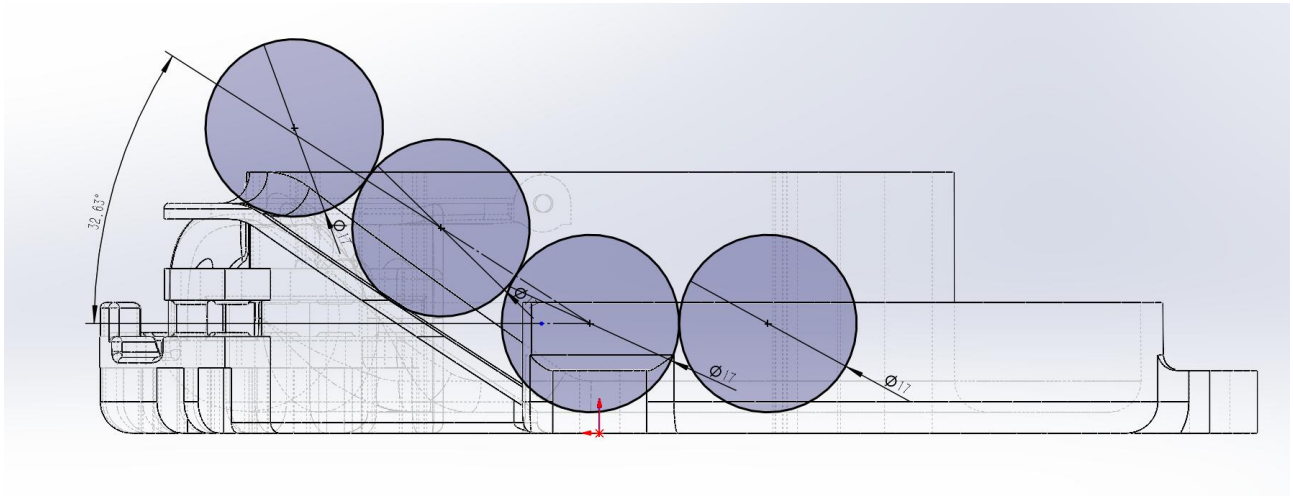
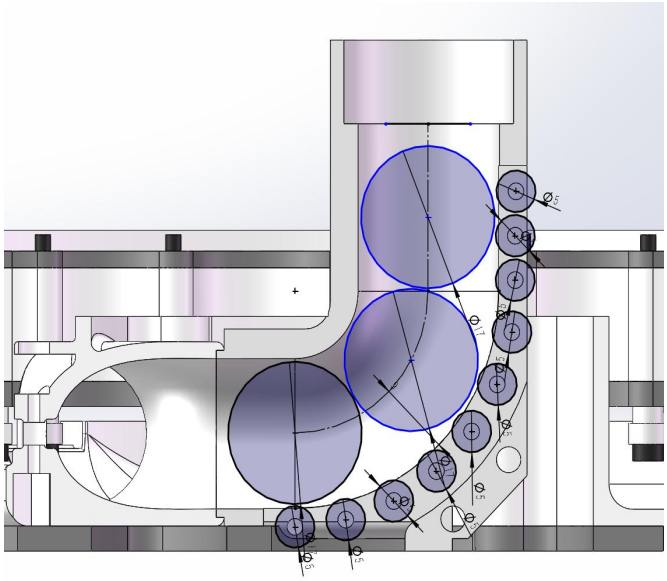


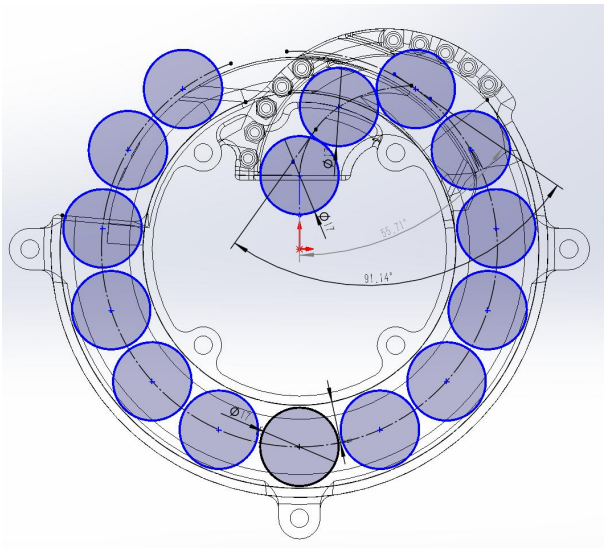
图 9：齿轮传动比 39:16



（1）



（2）



（3）

图 10：以上为模拟弹丸路径曲线（1）（3）为弹丸向心路径（2）为弹丸抬升路径

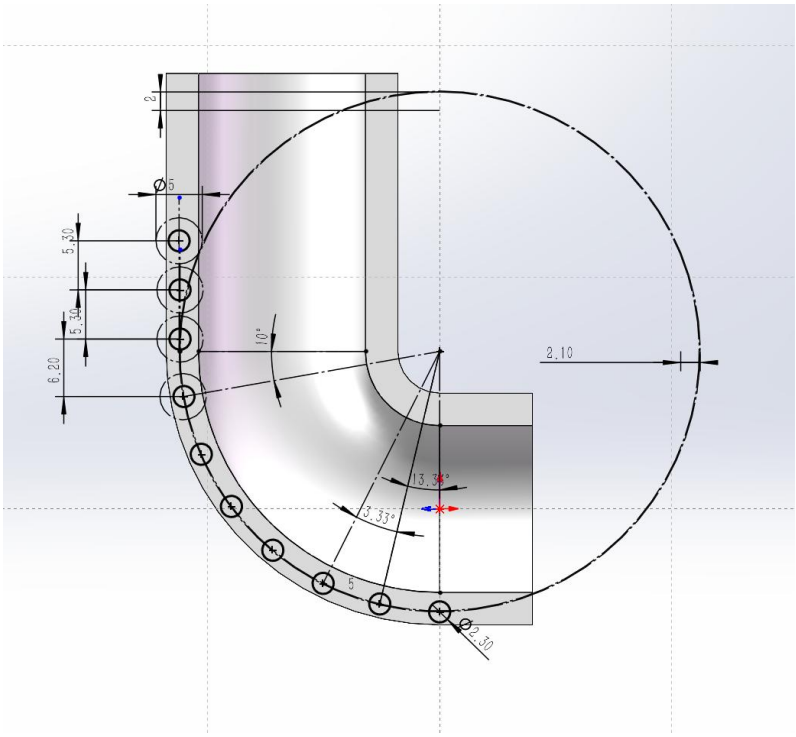


图 11：抬升路径最终具体设计参数

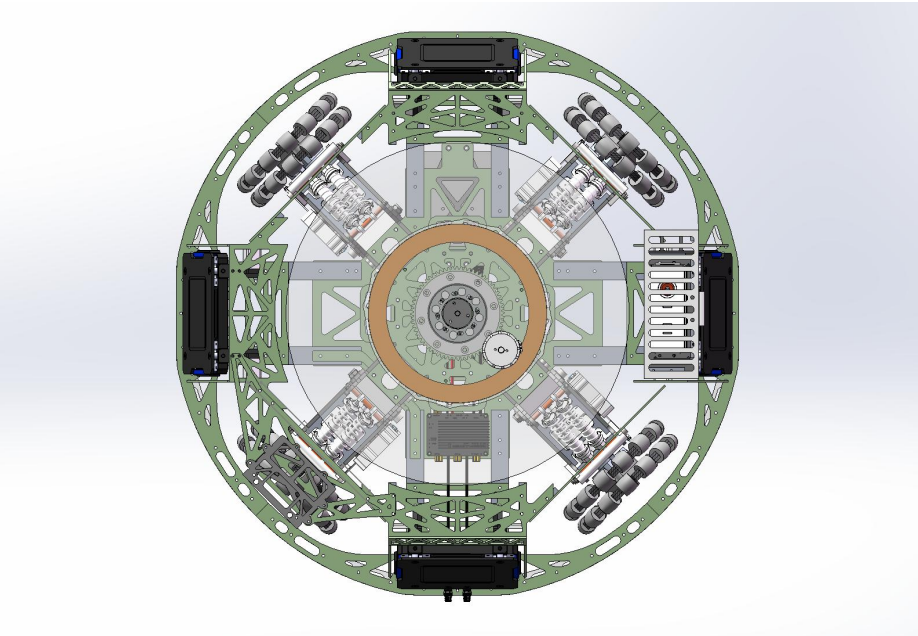
根据哨兵的双云台设计，为了减少因链路的转弯次数而卡弹的概率，提高其通过性，哨兵选用中心供弹方案。中心供弹拨弹盘则是该方案的重中之重，小弹丸的中心供弹拨盘最早是出现在哈工大 21 赛季的哨兵机器人上，北信科也在 2022 赛季后做出开源，他们的小弹丸中心供弹拨盘采用了分段式设计，同时我们也参考了北信科、北理工的大弹丸中心供弹，综合这些开源，我们以此为参考，设计了我们的 12 齿中心供弹拨弹盘。

本次设计的中心供弹拨弹盘分为拨轮和拨盘。拨齿在内，依靠外环齿轮进行传动，通过深沟球轴承配合打印件进行限位及运行，组成拨轮结构。

拨盘则分为螺旋弹路与抬升弹路。拨弹盘通过螺旋弹路将弹丸送至中心位置，通过抬升弹丸送上云台。螺旋弹路、抬升弹路的角度均经过大量修改测试得出。

因为本赛季哨兵拥有 750 发弹丸发射量，弹舱的设计在底盘和云台之间“类下供”设计，拨弹盘到鹅颈链路则经过一根铝管垂直抬升，所以与大弹丸中心供弹拨盘的向心-抬升复合弹链不同，小弹丸的中心供弹完全拆开了向心和抬升 90°，抬升弹链作为一个单独打印件方便装配。但目前弹丸的路径曲线，卡死点依然存在一定的问题，还需要继续优化。

1.5.1.3 底盘设计



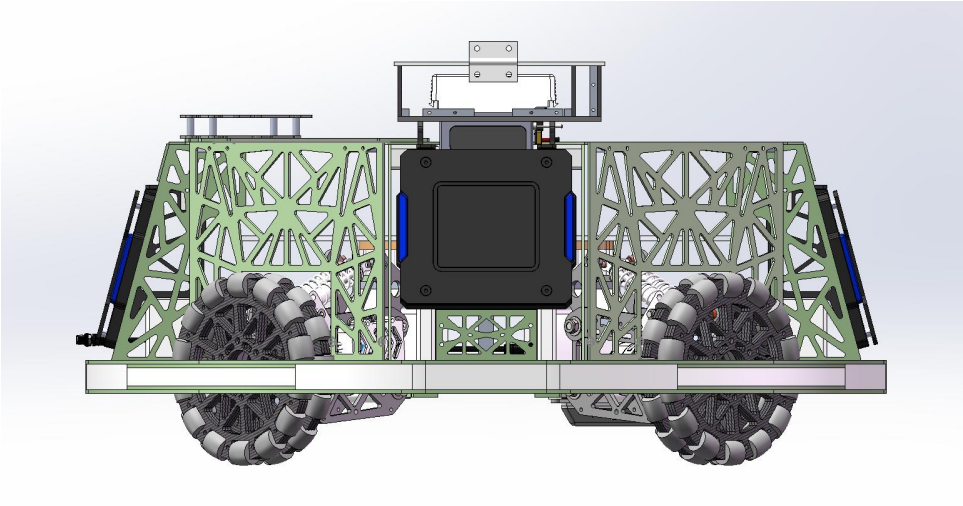


图 12、13：底盘完整机械结构图

1.5.1.3.1 底盘框架

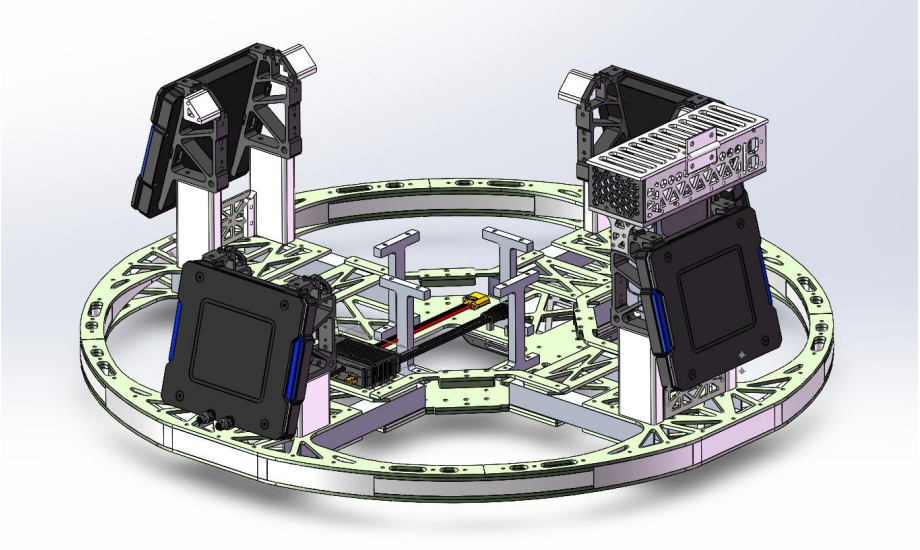


图 14：底盘框架完整机械结构

联盟赛和分区赛均选用 20*40*1.5 大截面铝管组成，经过赛场检验，强度冗余，重量偏重，且占用空间较大，国赛经过重新设计，缩小尺存至底盘直径 640mm，选用 20*20*1.2 铝方管作为主体框架。

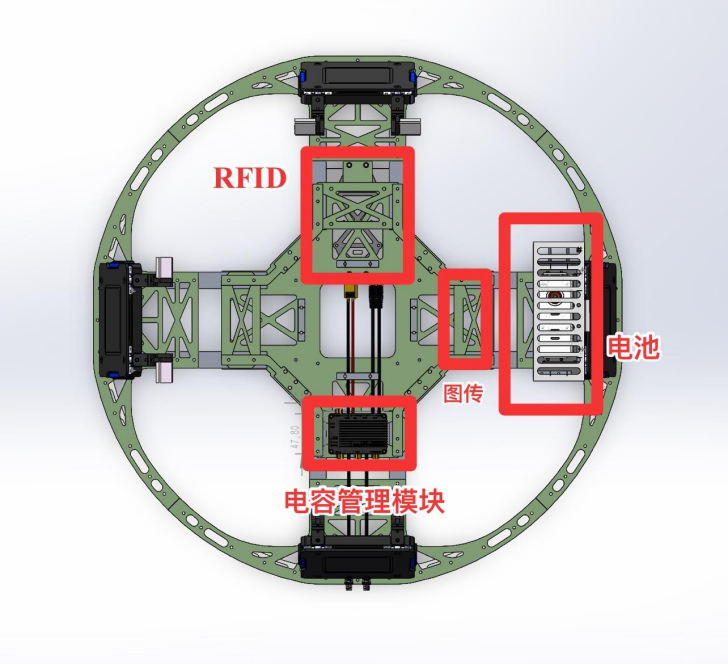


图 15：电子器件布局

底盘的裁判系统与电控元件，布局于此。分区赛时，因 RFID 识别问题，造成基地护甲展开，特将 RFID 上方不放置任何电控原件和裁判系统，避免上方经过大电流，高频信号线等造成的电磁干扰，影响场地交互。同时降低 RFID 距地面高度，保证更良好的识别率。由于全向轮底盘空间的分割性，且为保证弹舱容积满足至少 750 发的要求，将电池放于装甲板后方。

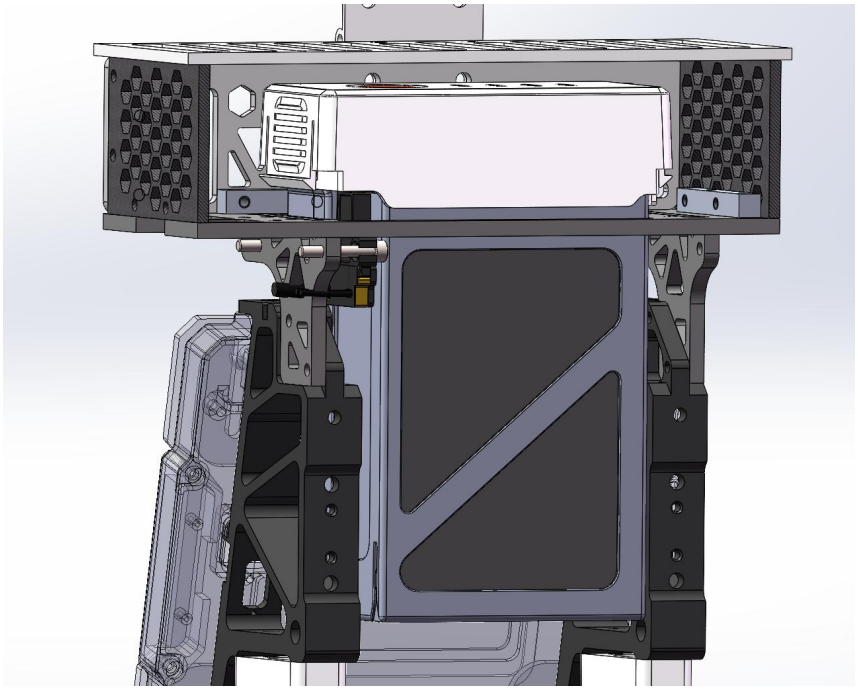


图 16：自研电池架

联盟赛期间我们用合页做抽屉式设计来更换电池，朝向由外指向中间，目的是避免小陀螺时，由于转速过快，离心力将电池甩出。但是此方案更换电池极为不方便，浪费时间，我们参考太原科技大学论坛开源-无人机电池架减重二次设计，结合自身需求，设计符合哨兵的自制电池架。

1.5.1.3.2 轮系设计

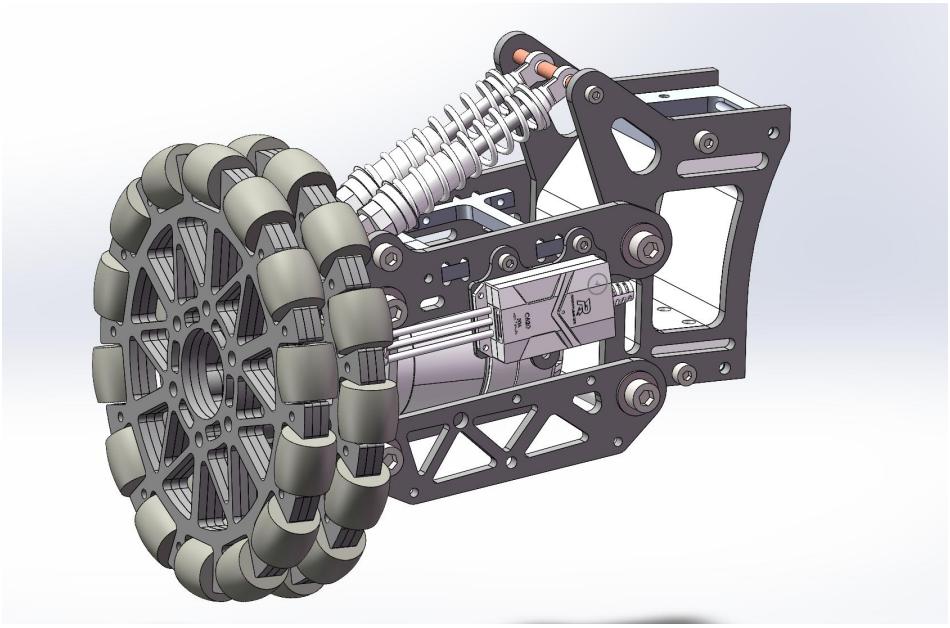


图 17：全向轮轮组完整机械结构

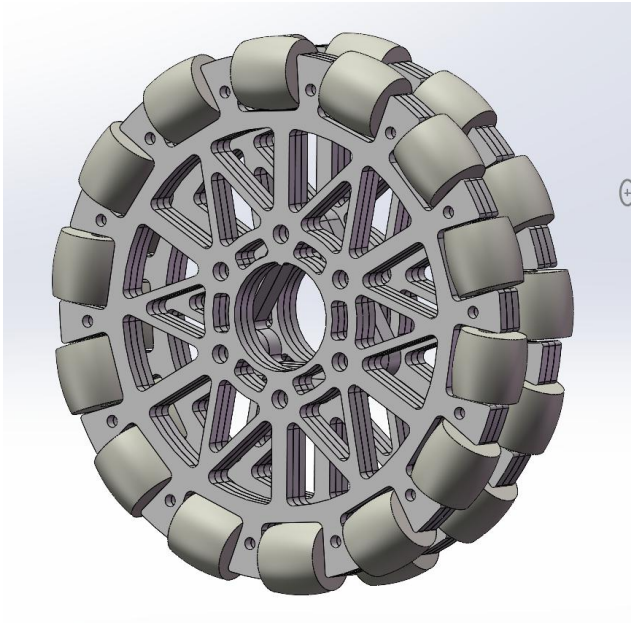


图 18：改装全向轮

该全向轮由航发全向轮滚子和自制玻纤板拼接而成，从航发官方全向轮整体重量 0.64kg 降到 0.4kg，保证了抓地力的同时减轻了重量。

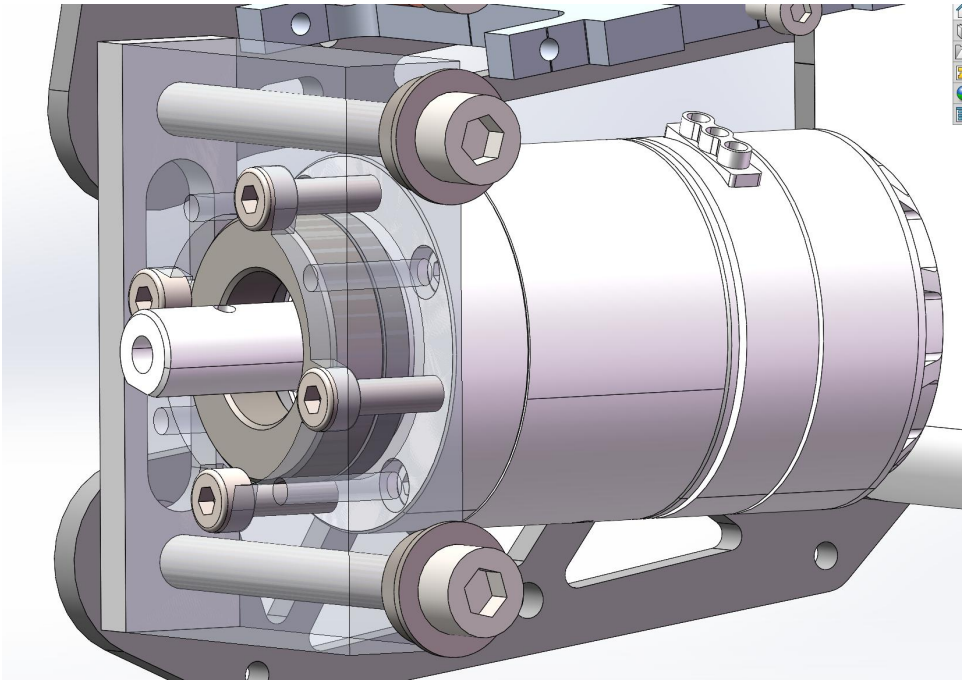


图 19：电机座及电机安装

电机座主体由一块经过 CNC 加工的铝块组成，内部则由四根螺栓压住 17-30-7 轴承，以此来减少联轴器直接作用在电机座的径向力和摩擦力。

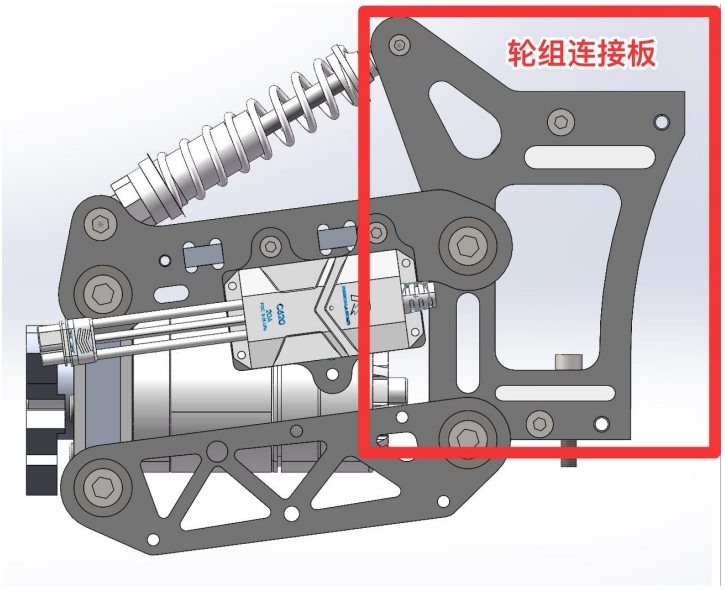


图 20：平行四边形悬挂模块

轮组为平行四边形悬挂，并采用模块化设计，及保证了强度，也方便装配。根据我校对哨兵的战场定位和实际需求，本版本的悬挂采用独立悬挂，使用溪地减震器，但后续通过换两侧轮组连接板则可增加自适应。

1.5.1.3.3 底盘大 yaw

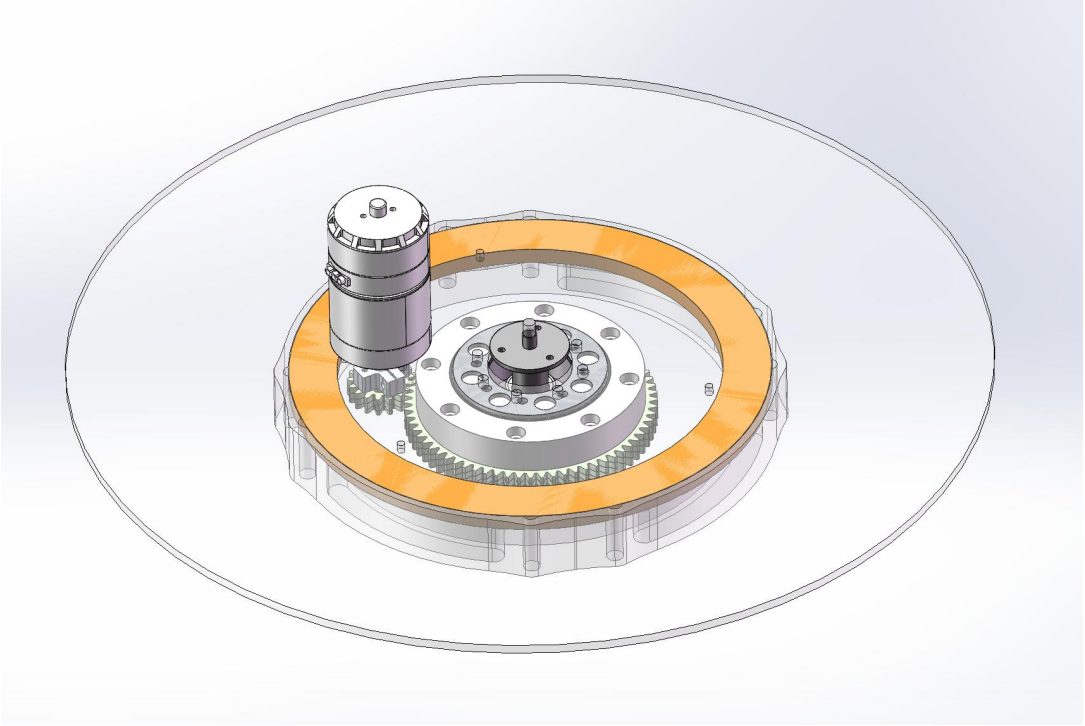


图 21：底盘大 yaw 完整机械结构

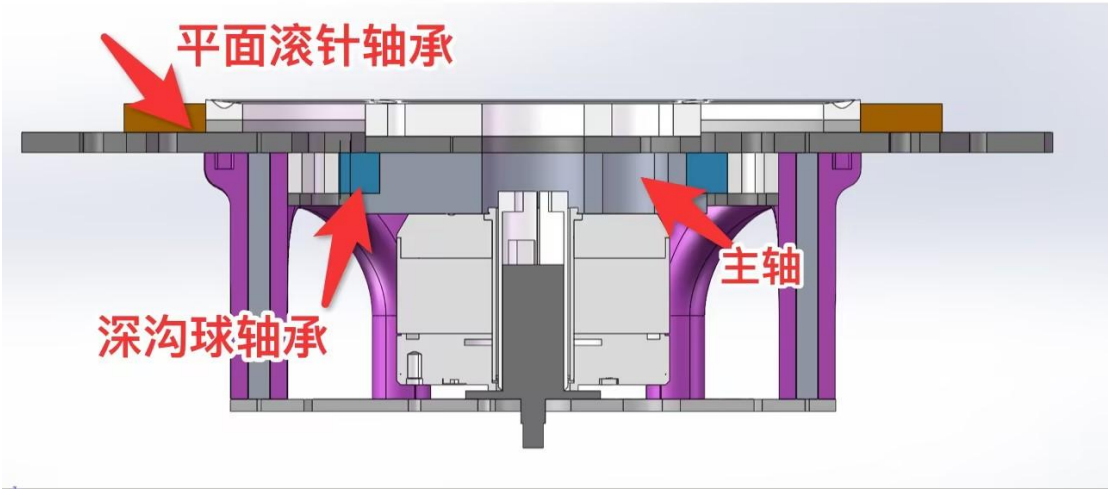
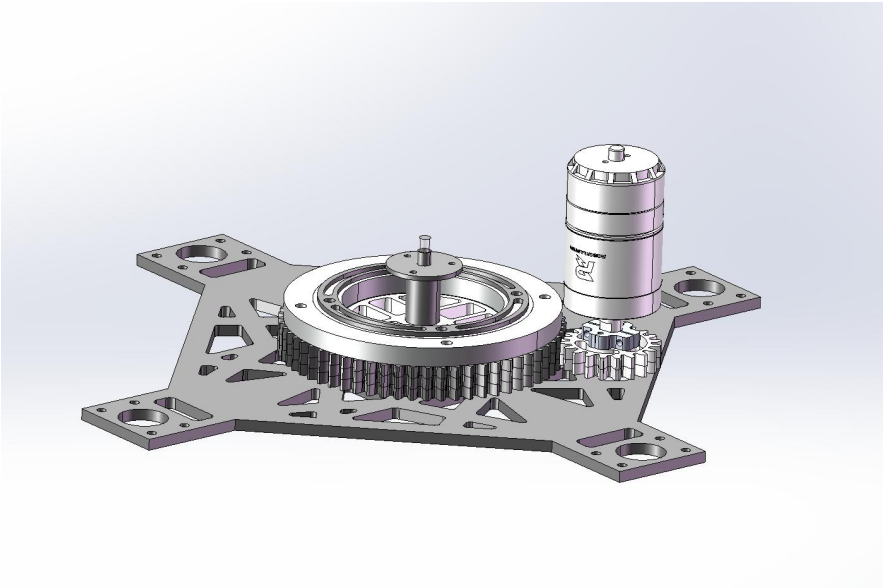


图 22：联盟赛大 yaw 剖面图



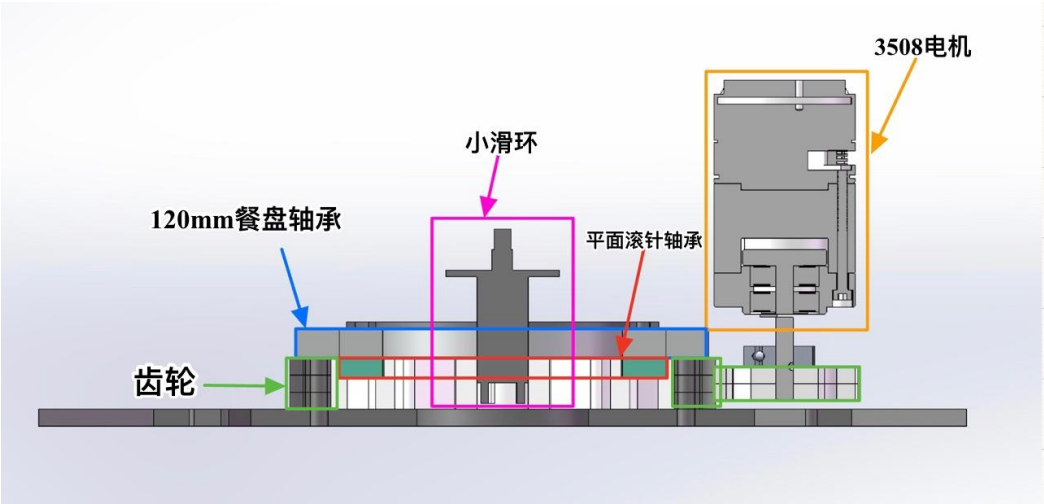
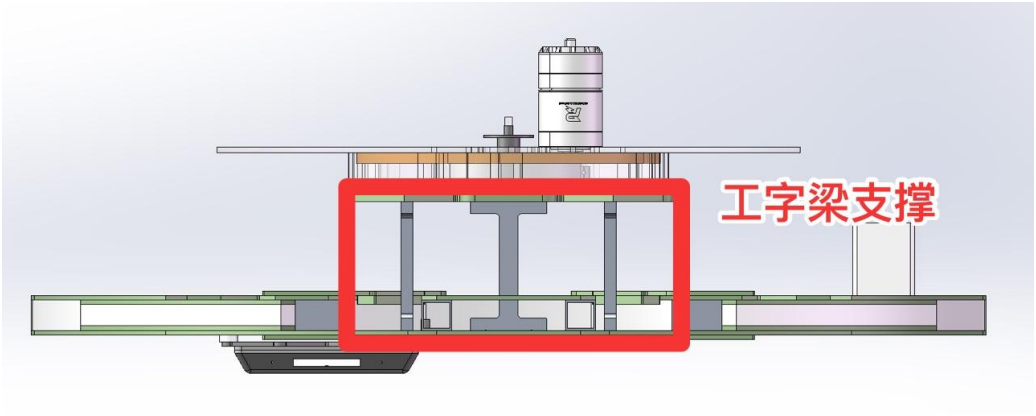


图 23、24：分区赛大 yaw 完整机械结构和剖面图



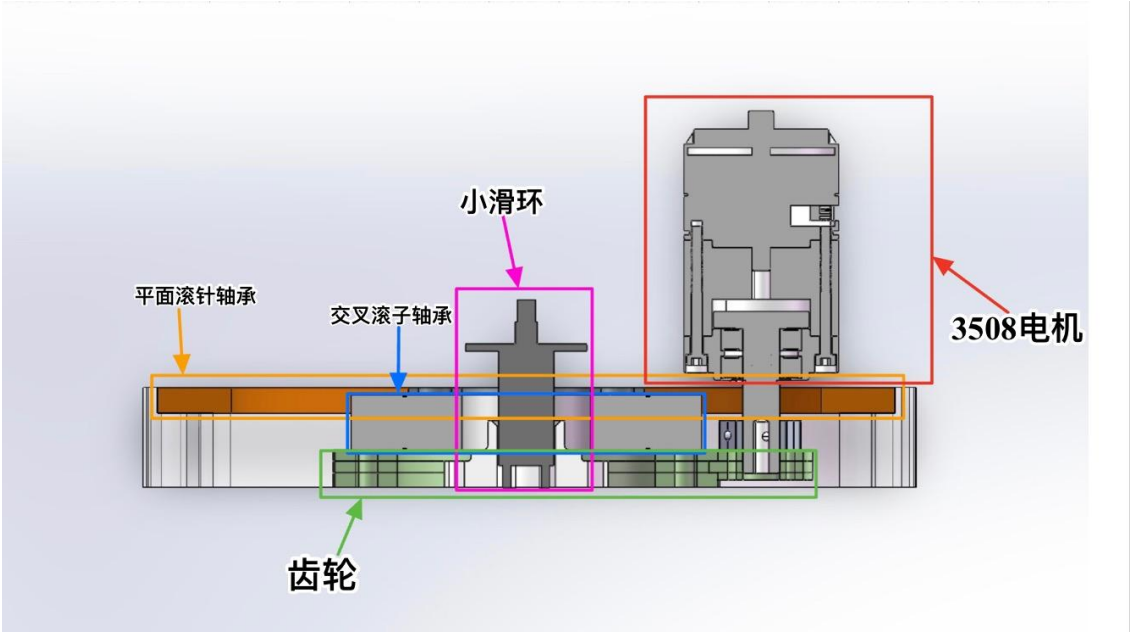


图 25、26：国赛底盘大 yaw 完整机械结构和剖面图

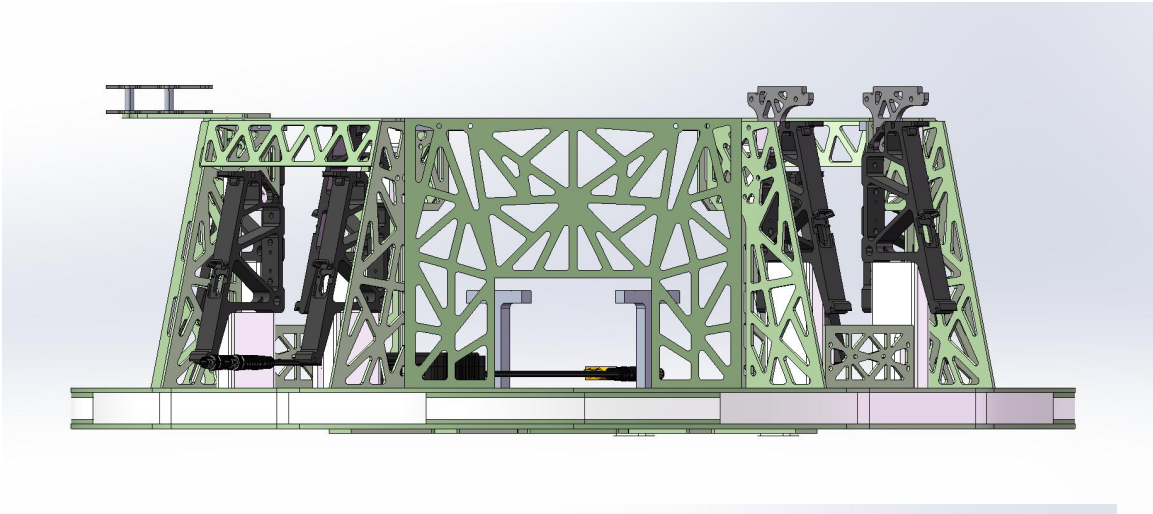
底盘的稳定性决定了双云台发射的稳定性，因此底盘的 yaw 轴设计，为整个赛季哨兵的重中之重。

联盟赛版本 yaw 轴，为本赛季哨兵第一代 yaw 轴，选用 6020 电机直连方式，由于云台加上 750+弹丸后重量超过 6020 电机直连所能承受的负载，电机在超过 120° 后，电机对角度的控制逐渐失灵，时间久将导致云台底盘一起旋转，成为了当时一大问题。

分区赛经过分析，选用 3508 电机带减速箱，通过齿轮进行传动的方案。在轴承方面，考虑到经济性的原因，选用 120mm 餐盘轴承，下方加入 95mm 平面滚针轴承，对餐盘轴承进行保护和降低摩擦阻力，并提高其承载能力，该与过去步兵复杂的轴系不同，设计新颖，装配简便，易于维修和更换轴承。但是根据分区赛的表现，我们依然发现，由于餐盘轴承间的游隙大，会在小陀螺旋转时，被云台放大化，导致云台起伏大，影响发射和视觉识别，同时餐盘轴承材料并非轴承钢，后期磨损十分严重。

国赛我们基于进行了再一次迭代，轴承换成交叉滚子轴承，内外径在动载荷下游隙小于5μm，承载能力达到300kg，加入200mm平面滚针轴承，既提高其承载面积，也可以保护交叉滚子轴承。同时考虑到分区赛yaw轴下方无支撑，负载导致的玻纤板形变也会降低其稳定性，国赛版本特意加入四根工字梁，增加稳定性。最后其赛场表现，效果优于前几代yaw轴。

1.5.1.3.4 底盘防护



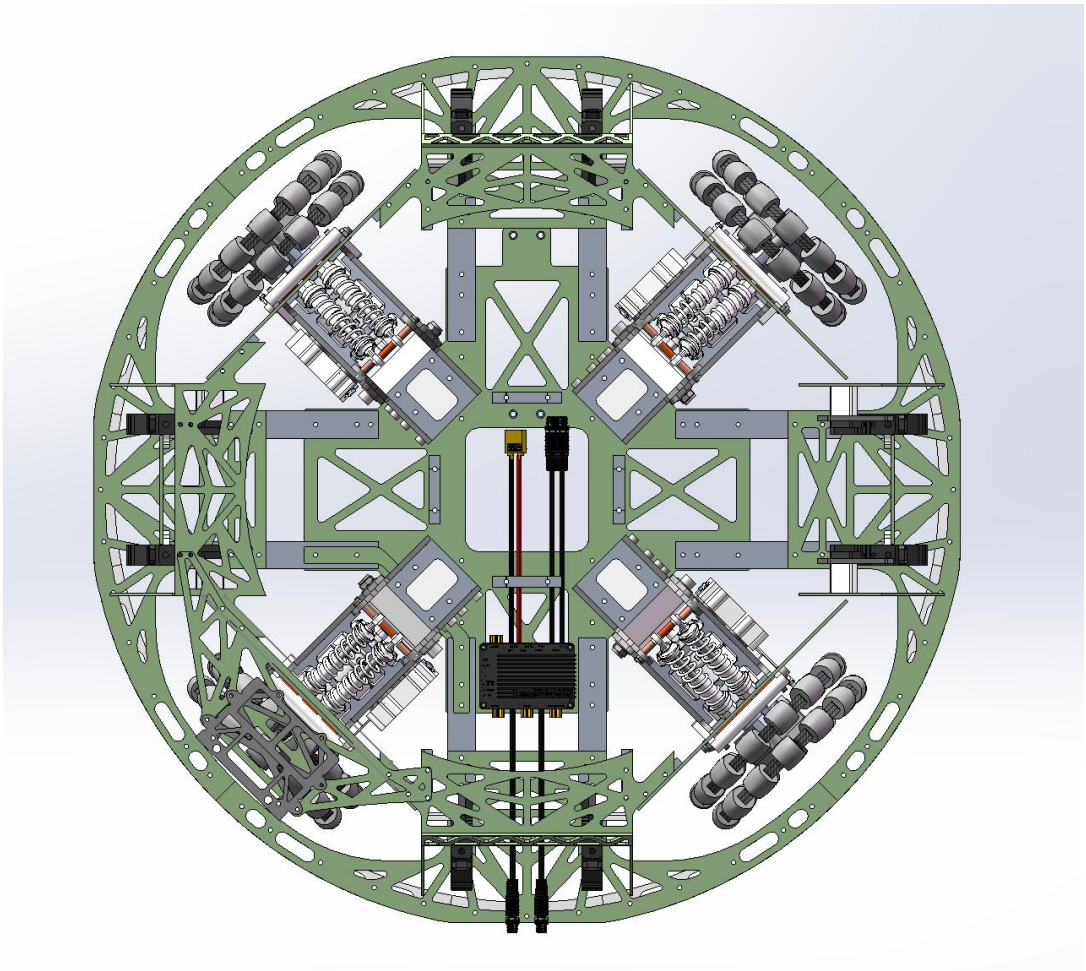


图 27、28：底盘防护完整机械结构

联盟赛和分区赛两代哨兵，因重量等原因，均采用塑料围栏网为四周防护，防止弹丸进入卡住底盘 yaw 轴，打坏线材和电控元器件，底盘框架则采用半包围式防护，保护轮组。经过赛场检验，这些防护依然存在很多缺陷，防护不足的情况，国赛则全部改为 2mm 玻纤板防护，框架设计为全包围式，较比原先防护重量增加 0.5kg，但防护效果大大优于之前的版本。

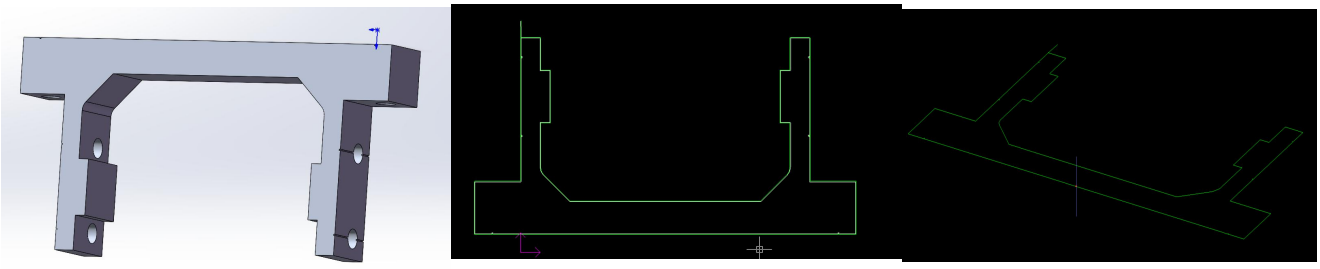
1.5.1.4 工艺选择

在哨兵机器人整体的设计过程中，综合机械制造的合理性，再结合本学校的加工资源情况，采用了比较多的用 2D 雕刻加工玻纤板结构设计和线切割加工铝件的设计，一些必要的铝件上则使用雕刻机和铣床，结合 3D 打印件与部分车削的轴类零件作辅助。

1.5.1.4.1 线切割件

可一笔画成的铝轮廓件、铝柱铜柱等导电柱材、铝管等。

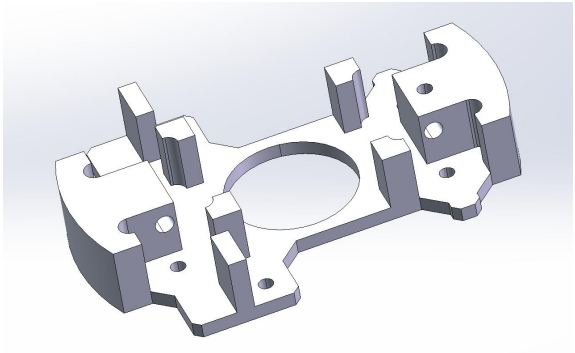
例如：测速固定铝块，使用 10mm 厚的铝板，用线切割机床加工。

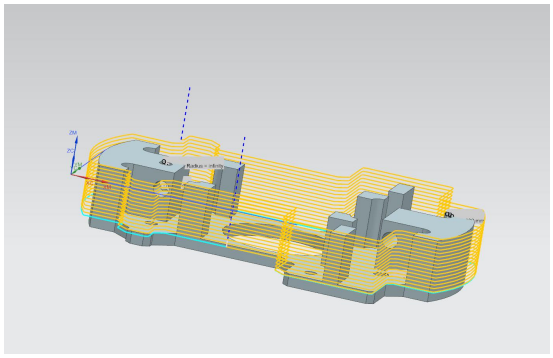


1.5.1.4.2 三轴雕刻机

铝板、玻纤板、碳板、精度定位打孔。

例如：yaw 轴基座，使用 10mm 厚度铝板，用三轴雕刻机加工。



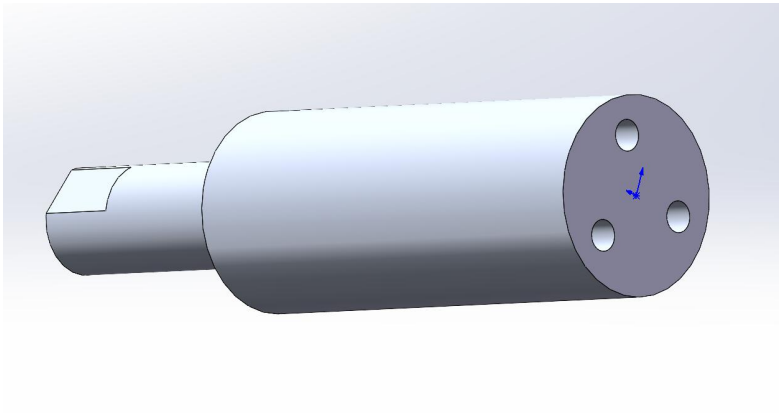


1.5.1.4.3 氩弧焊

车架连接。

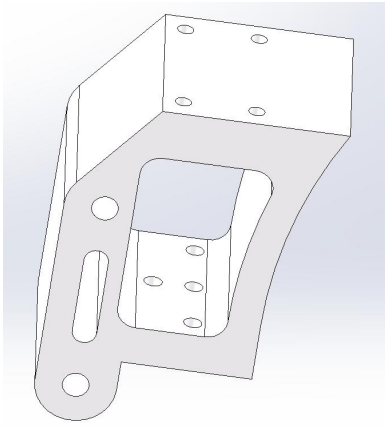
1.5.1.4.4 数控车床

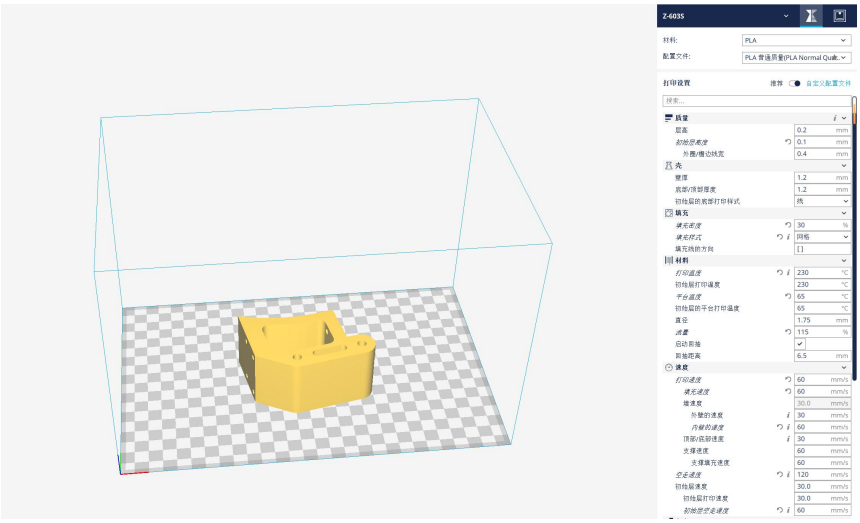
铝金属棒料车轴。



1.5.1.4.5 3D 打印

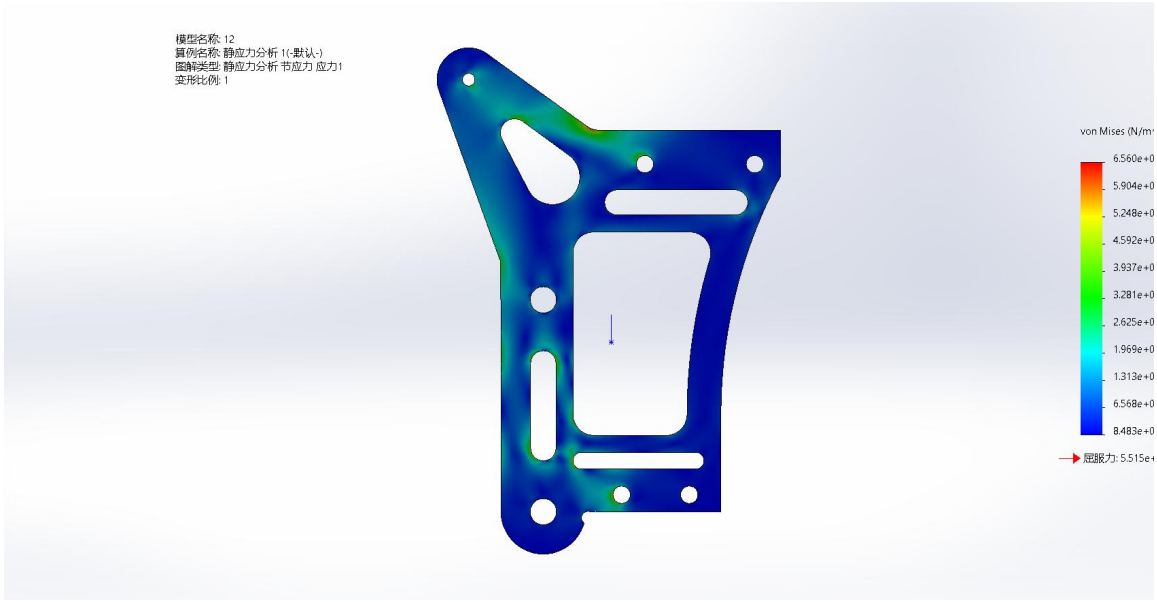
复杂难以加工且强度要求低的零件。





1.5.1.5 核心零件有限元分析、静力学分析

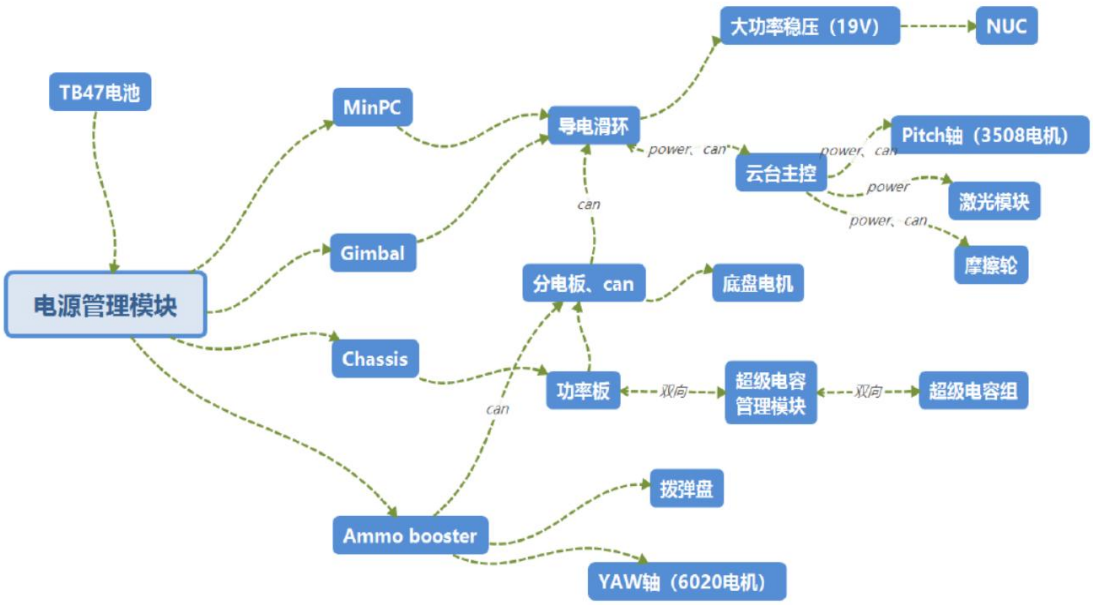
分析案例为 轮组连接板，在设定约束条件为 4 个螺栓固定圆柱面，载荷情况为 49N 单向力。得出的如第一张图所展示，考虑实际状况时工况比仿真载荷要低很多，可得该零件的强度已经很满足真实应用场景。



仿真结果的最大位移量为 0.005842mm，可见其零件刚性也十分满足真实应用场景。

1.5.2 硬件设计

1.5.2.1 整机硬件方案框图



1.5.2.2 硬件主要用到主控板、功率板和其他小功能硬件模块

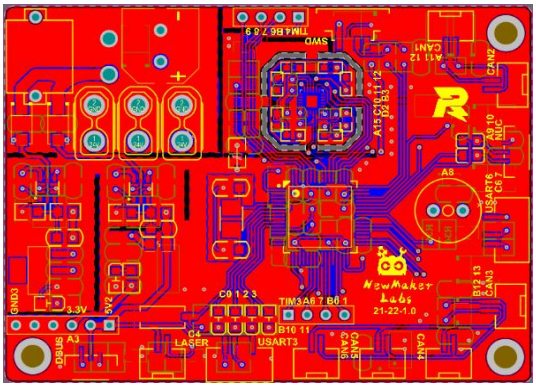
主控板：

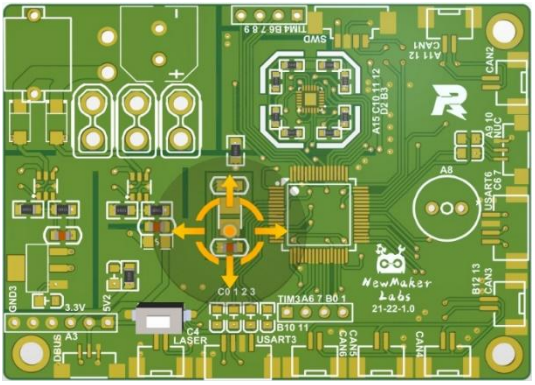
主控板是控制的核心，根据比赛需求，我们设计了具有较高的运算性能、较多的通信接口和较小的稳定主控。

主控芯片采用 STM32F405RGT6，其提供了工作频率为 168 MHz 的 Cortex™-M4 内核（具有浮点单元）的性能，64 引脚，4 x 4.2 mm 的小体积封装。具有高性能、高集成度等优点。STM32F405RGT6 功能如下：

- 通信接口多达 15 个（包括 6 个速度高达 10.5 Mb/s 的 USART、3 个速度高达 42 Mb/s 的 SPI、3 个 I²C、2 个 CAN 和 1 个 SDIO）。
- 模拟：2 个 12 位 DAC、3 个速度为 2.4 MSPS 或 7.2 MSPS（交错模式）的 12 位 ADC。
- 定时器多达 17 个：频率高达 168 MHz 的 16 和 32 位定时器。
- 可以利用支持 Compact Flash、SRAM、PSRAM、NOR 和 NAND 存储器的灵活静态存储器控制器轻松扩展存储容量。

PCB 设计过程中对开关电源和陀螺仪的布局单独处理，整体遵循分块铺铜和单点接地的设计理念，保留了较为完整的地回路，具有较高的稳定性。

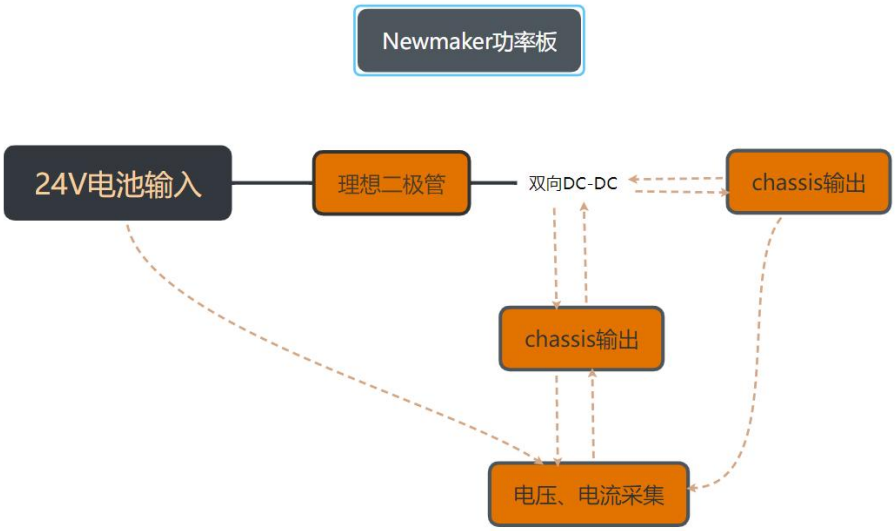




功率板：

超级电容利用多组超级电容器将能量以电场能的形式储存起来，当能量紧急缺乏或需要时，再将存储的能量通过控制单元释放出来，准确快速地补偿系统所需的有功和无功，从而实现电能的平衡与稳定控制。超级电容功率控制板是控制超级电容充放电的装置，其设计的好坏决定了底盘功率利用率的高低。

功率板设计目标是高集成度、高效率和大功率。最简单的超级电容模块一般为电池输出全部经过一个降压变换器供电容充电，再将电容电压升压给底盘供电，这个方案因电池的输出都要经过一次降压和一次升压变换，所以效率很低，且体积过大，不满足高效率的设计目标。参考大连理工大学超级电容方案，进行 PCB 双层板设计。对数字、模拟、功率信号进行分块布局，提高稳定性。双向同步整流 BUCK-BOOST 变换器能够很好的满足需求，相对于单纯的 BUCK 电路或 BOOST 电路，不仅能实现能量的双向流动，还能在同一方向实现升降压功能。



双向 BUCK-BOOST 拓扑简介

双向 BUCK-BOOST 电路拓扑是由同步 BUCK 电路和同步 BOOST 电路级 联而成。传统的 BUCK 电路和 BOOST 电路中由二极管续流，但在低压大电流场 合，由于二极管上存在导通压降，会引起较大的导通损耗。利用 MOS 管代替传 统的 BUCK 电路和 BOOST 电路中的续流二极管，由于 MOS 管开通时的管压降 相对较低，能够显著降低电路中半导体的导通损耗，这种方式称为同步整流。

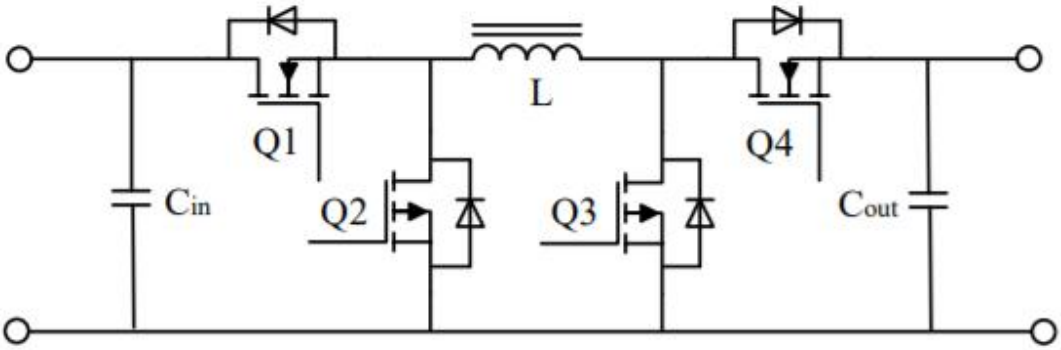


图 1.1 双向 BUCK-BOOST 电路

即如上图 1.1 所示，双向同步整流 BUCK-BOOST 电路是由同步 BUCK 电路 和同步 BOOST 电路构成，其中 MOS 管 Q1，Q2 和电感 L 构成同步 BUCK 降压 变换器电路，MOS 管 Q3，Q4 和电感 L 构成同步 BOOST 升压变换器电路。 由于同步整流 BUCK-BOOST 拓扑左右完全对称，即 MOS 管 Q3，Q4 和电 感 L 也可以构成同步 BUCK 降压变换器电路，MOS 管 Q1，Q2 和电感 L 也可以 构成同步 BOOST 升压变换器电路。因此同步整流 BUCK-BOOST 电路在任何一 方向上均可以实现升降压功能，即电路能量可以双向流动。 以左侧为输入，右侧为输出为例子，分析电路工作原理。当 Q4 常闭，Q3 常 开，Q1 与 Q2 以特定占空比互补导通，则电路工作于 BUCK 模式，如图 1.2 所 示；当 Q1 常闭，Q2 常开，Q3 和 Q4 以特定占空比互补导通，则电路工作于 BOOST 模式，如图 1.3 所示；当 Q1 和 Q2，Q3 和 Q4 均特定占空比互补导通， 则电路工作于 BUCK-BOOST 模式（MIX 混合模式），如图 1.1 所示。

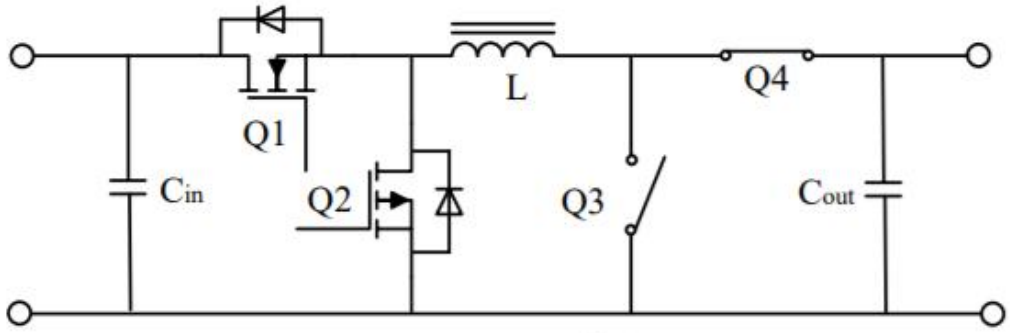


图 1.2 BUCK 模式

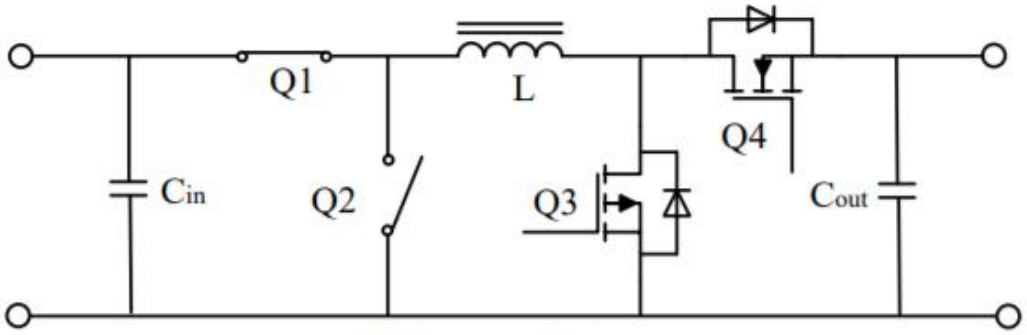


图 1.3 BOOST 模式

BUCK 工作模式

当输出电压显著小于输入电压时，电路工作在降压区（BUCK 模式），此时 Q1 和 Q2 以特定占空比互补导通，Q4 常闭合 Q3 常开，电路等效于同步 BUCK 电路。在实际应用中，Q1 与 Q4 的为上管，通常采用自举升压的驱动方式，即对于 Q4 而言，在一个开关周期内，Q3 需要有特定的导通时间，否则当 Q4 的自举电容能量损耗完时，Q4 将截止。即该应用场合，可以使得 Q3 以很小得占空比导通，即 Q4 以接近满占空比导通。 当 Q1 开通 Q2 关断时，其等效电路如图 1.4 所示，输入 Vin 通过 Q1 为电感 L1 储能并为负载供电，电感两端的电压为 Vin-Vout，电感电流线增加，电感储能，输出电容储能。

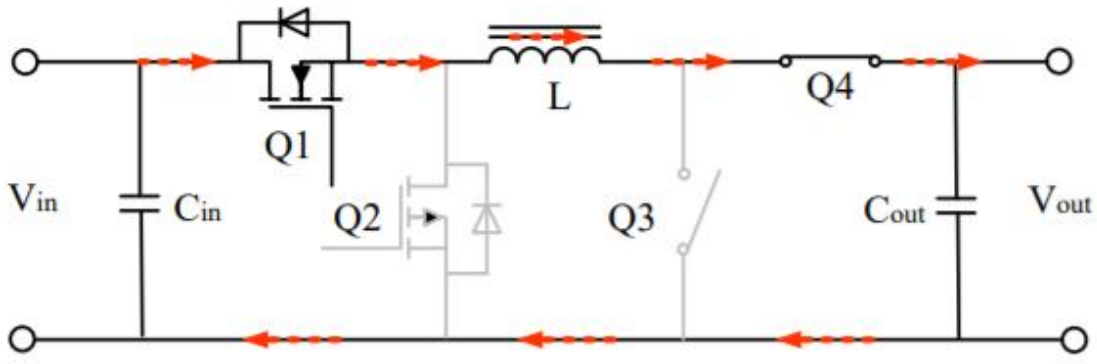


图 1.4 BUCK 模式电感储能阶段

此时，输入输出电压，电感电流的关系为：$V_{in} - V_{out} = L\frac{di_L}{dt}$

当 Q1 关断 Q2 开通时，其等效电路如图 1.5，所示由于电感电流不能突变，电感电流方向不变，通过 Q2 进行续流，电感两端的电压为-Vout，电感电流线性减少，电感放能。

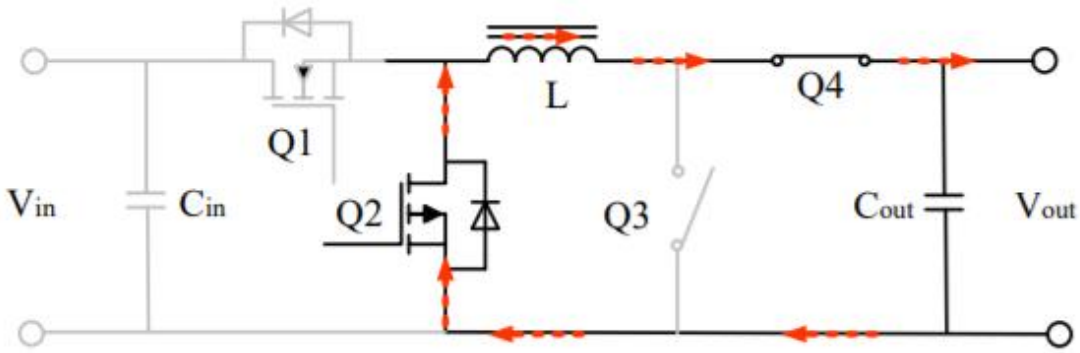


图 1.5 BUCK 模式电感放能阶段

此时，输出电压与电感的关系为：$V_{out} = -L\frac{di_L}{dt}$

Q1 和 Q2 驱动和电感电流波形如图 1.6 所示。

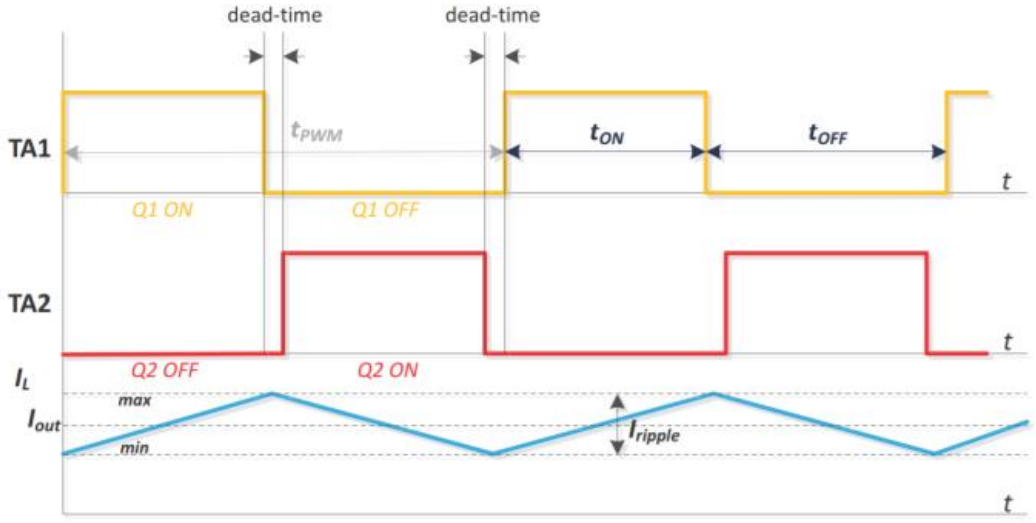


图 1.6 BUCK 模式工作波形

在一个开关周期内，Q1 导通的时间为 ton，Q2 导通的时间为 toff，PWM 的开 关周期时间为 TPWM，定义占空比为 D，则：

$$tON + tOFF = T_{PWM}$$

$$D = \frac{tON}{T_{PWM}}$$

tor=(1-D)×Twm

根据电感能量在一个开关周期内平衡定理：

D×(V'in-Yout)+(1-D)×(-Yout)=0

则可计算工作于 BUCK 模式下输入输出电压和占空比得关系

$$\frac{V_{out}}{V_{in}} = \mathrm{D}$$

BOOST 工作模式

当输出电压显著大于输入电压时，电路工作在升压区（BOOST 模式），此时 Q3 和 Q4 以特定占空比互补导通，Q1 常闭合 Q2 常开，电路等效于同步 BOOST 电路。当 Q3 开通 Q4 关断时，其等效电路如图 1.7 所示，输入 Vin 通过 Q3 为电感 L 储能并为负载供电，电感两端的电压为 Vin，电感电流线增加，电感储能，输出电容为负载供电。

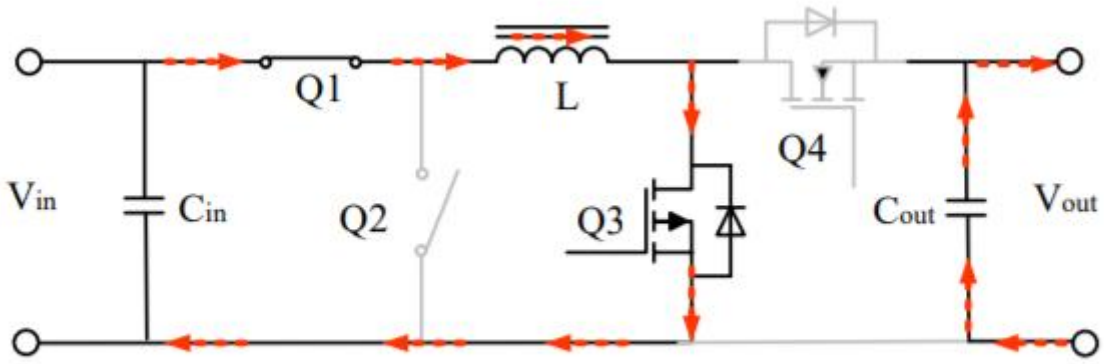


图 1.7 BOOST 模式电感储能阶段

此时：$V_{in} = L\frac{di_L}{dt}$

当 Q3 关断 Q4 开通时，其等效电路如图 1.8 所示，电感两端得电压为 VinVout，电感电流线性减少，电感放能，同时电感中得能不补充输出电容再上一阶 段损失的能量。

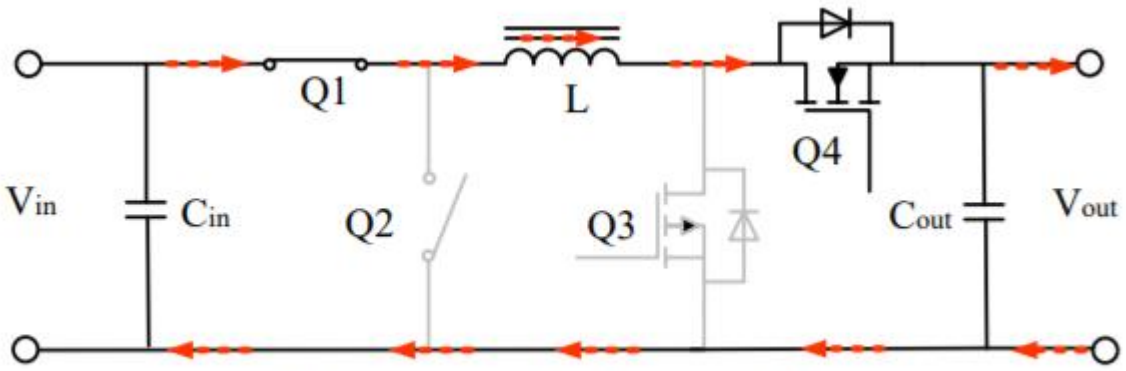


图 1.8 BOOST 模式电感放能阶段

Q3 和 Q4 驱动和电感电流波形如图 1.9 所示。

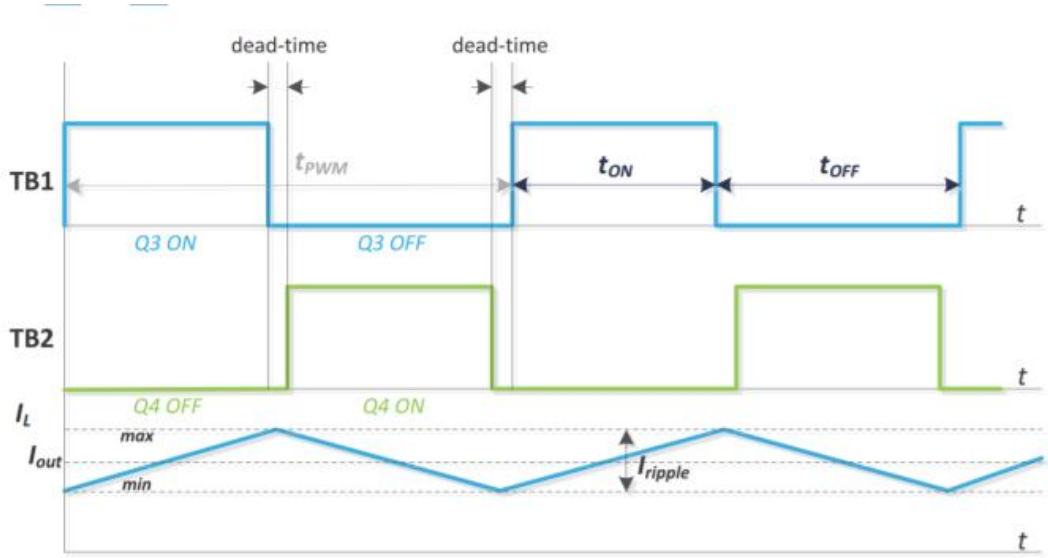


图 1.9 BOOST 模式工作波形

在一个开关周期内，Q1 导通的时间为 ton，Q2 导通的时间为 toff，PWM 的开关周期时间为 T_{PWM}，定义占空比为 D，则：

$$tON + tOFF = T_{PWM}$$

$$D = \frac{tON}{T_{PWM}}$$

$$toFF = (1 - D) \times T_{PWM}$$

根据电感能量在一个开关周期内平衡定理：

$$D \times Vin + (1 - D) \times (Vin - Vout)$$

则可计算工作于 BOOST 模式下输入输出电压和占空比得关系：

$$\frac{V_{out}}{V_{in}} = \frac{1}{1 - D}$$

BUCK-BOOST 模式（MIX 模式）

当输出电压和输入电压接近时，工作于 BUCK 模式和工作于 BOOST 模式 均不能满足输出电压的要求。这时需要 BUCK 和 BOOST 电路同时工作，即在一 个开关周期内，Q1 和 Q2 以特定占空比互补导通，Q3 与 Q4 以特定占空比互补 导通。 定义 BUCK 电路的占空比为 D1，BOOST 电路的占空比为 D2，BUCK 电路 输出的电压我 V0，即 BOOST 电路的输入电压为 V0，其等效电路如下图 1.10 所 示。

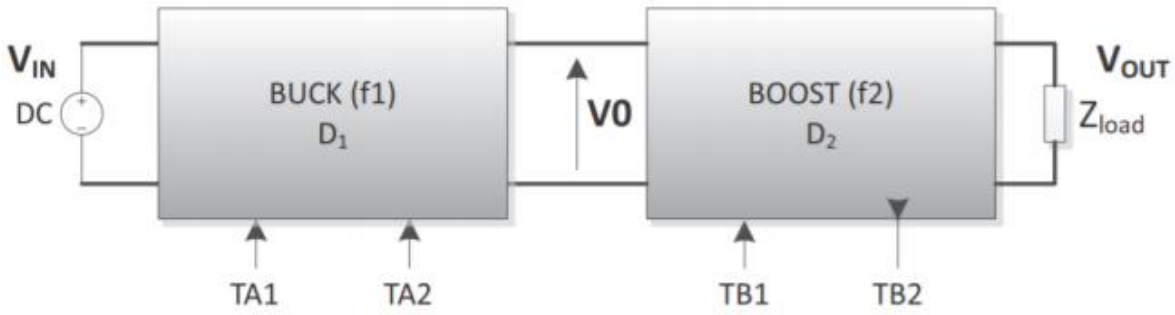


图 1.10 BUCK-BOOST 等效电路

对于 BUCK 电路输出：

$$V0 = f1(D1) \times Vin$$

对于 BOOST 电路输出：

$$Vout = f2(D2) \times V0$$

则：

$$\frac{Vout}{Vin} = f(D1, D2) = f1(D1) \times f2(D2)$$

即简化可得：

$$\frac{Vout}{Vin} = D1 \times \frac{1}{1 - D2} = \frac{D1}{1 - D2}$$

由上述公式可知，BUCK-BOOST 工作与 MIX 模式下时，输出电压同时受 BUCK 电路占空比 D1 和 BOOST 电路占空比 D2 影响。

由于工作与 BUCK-BOOST 模式下的输出电压和输入电压很接近，可设置 BUCK 的占空比为固定占空比 0.8，调节 BOOST 的占空比，即可调整输出电压。 结合上述分析的 BUCK 工作模式和 BOOST 工作模式，工作于该模式下的电流 波形和占空比如下图 1.11 所示。

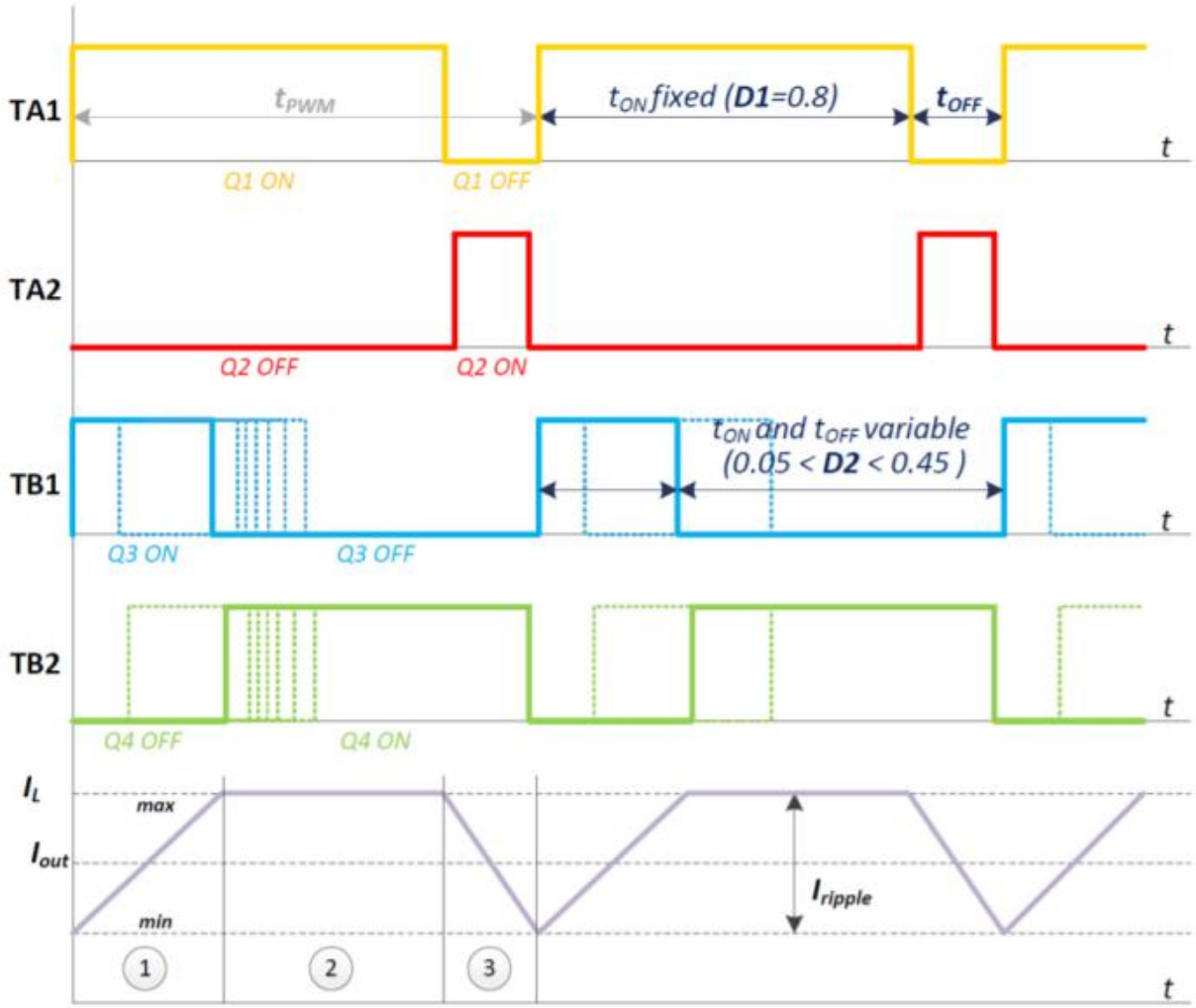
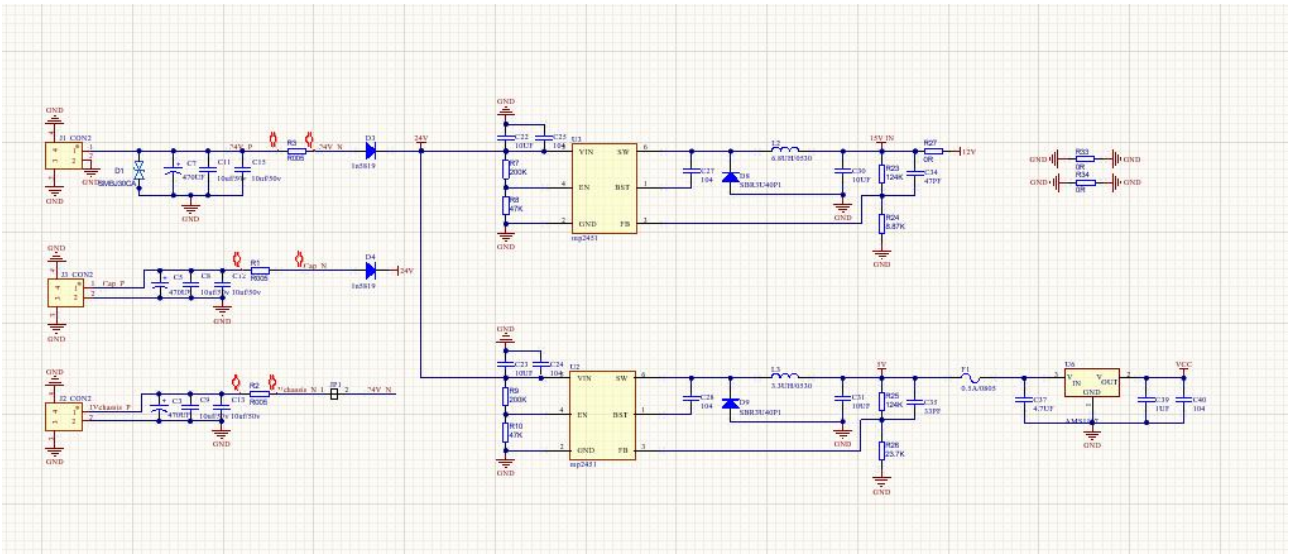


图 1.11 MIX 模式电路图

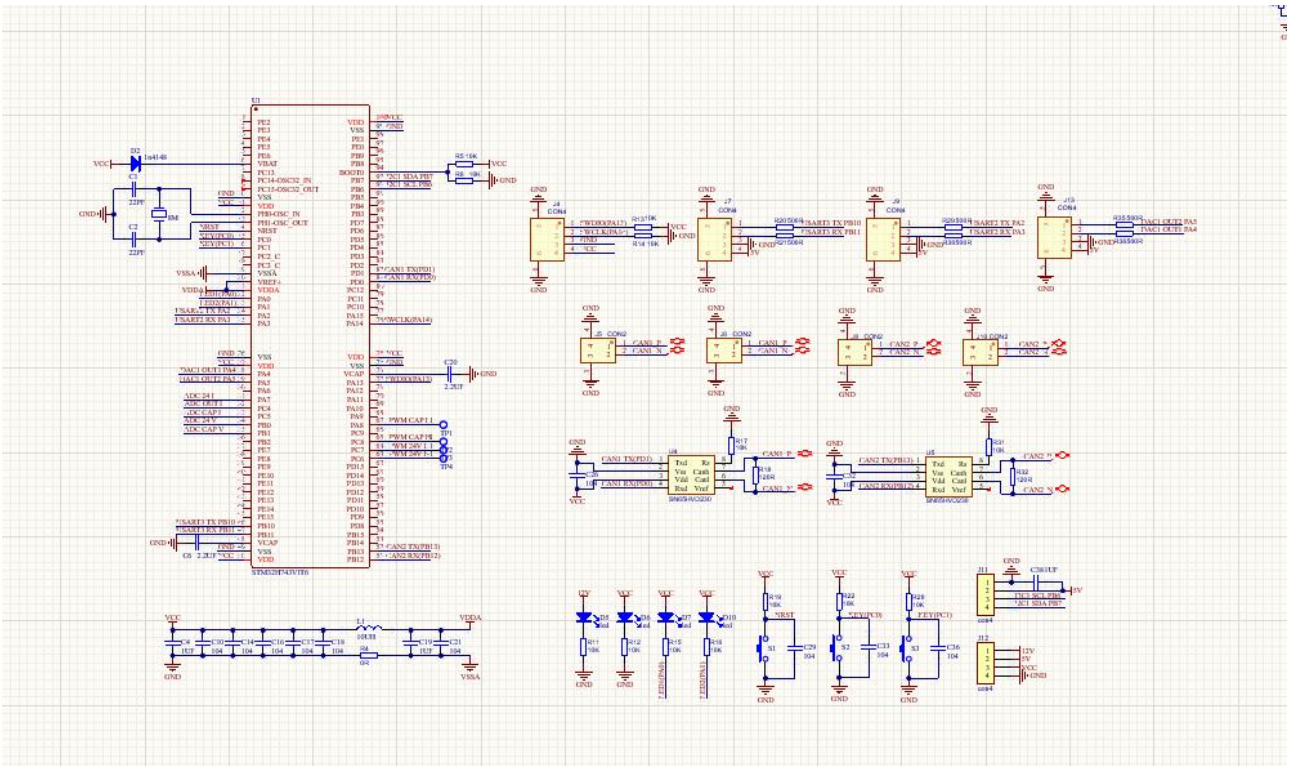
综合上述三种工作模式，分别检测输入电压和输出电压幅值，当输出电压小 于 0.8 倍的输入电压时，可使电路工作于 BUCK 模式；当电输出电压大于 1.2 倍 的输入电压时，可使电路工作于 BOOST 模式；当输出电压在 0.8 倍和 1.2 倍输 入电压范围内时，可使电路工作于 MIX 模式。

板载供电

控制级 24->5->3.3 ：使用 mp2451 芯片和 AMS1117 芯片做两级电源为主控和外设供电两级电源均作了充足的滤波处理保证纹波绝对合格芯片工作稳定。



最小系统



主控采用 STM32H743VIT6 芯片，STM32H743/753 系列产品集成了工作频率高达 480 MHz 的 Arm® Cortex®-M7 内核（具有双精度浮点单元）。具有高分辨率定时器（Hrtim），专门针对数字电源转换应用，例如 D-SMPS、照明、焊接、太阳能系统逆变器及无线充电器。STM32H743VIT6 功能如下：

- 多达 35 个通信接口包括 FD-CAN、USB 2.0 高速/全速、以太网 MAC、摄像头接口。
- 可利用带有 32 位并行接口或双模 Quad-SPI 串行闪存接口的灵活存储控制器轻松扩展存储器容量。
- 模拟外设：12 位 DAC，快速 16 位 ADC。
- 16 位高精度定时器上的多个 16 位和 32 位定时器运行频率高达 480 MHz。

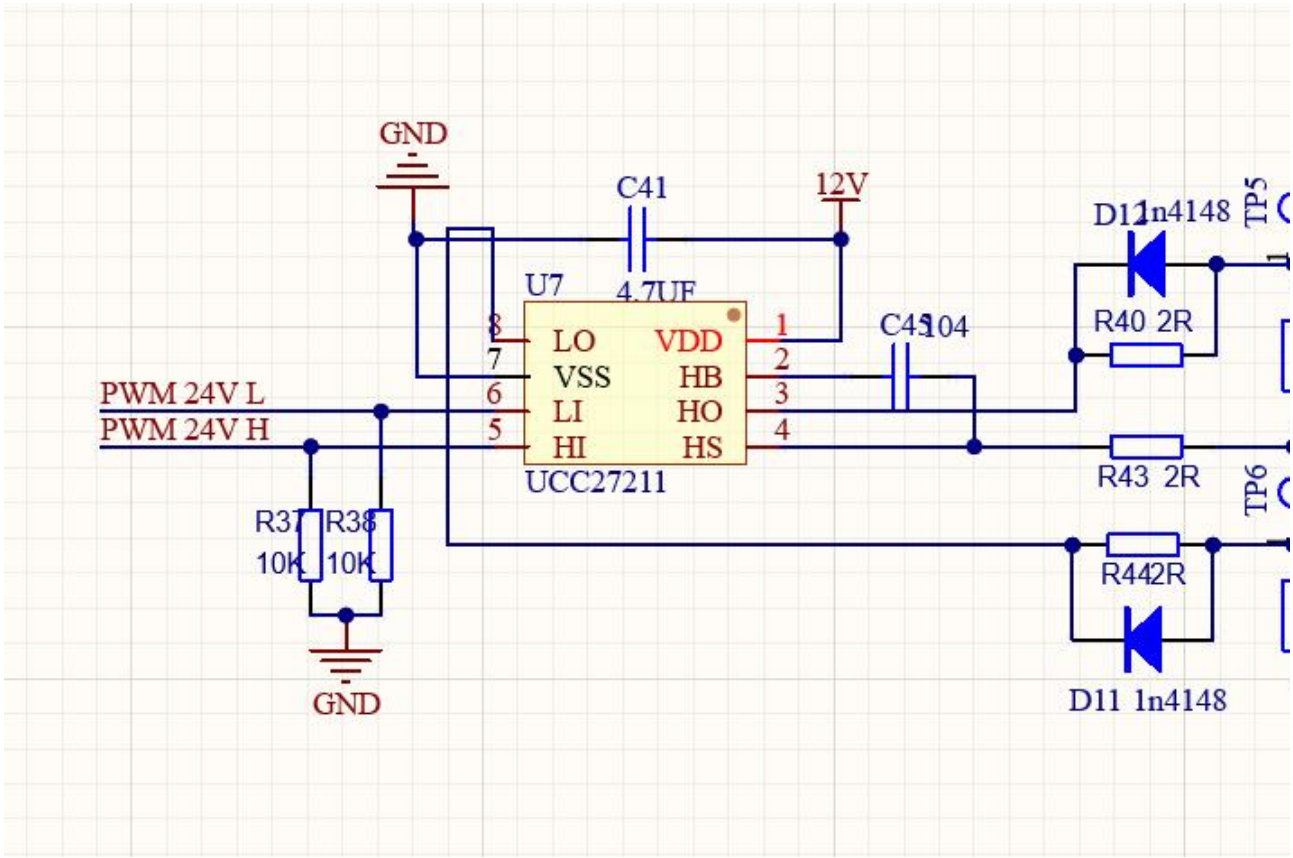
简洁的最小系统常用功能串口 IIC FDCAN 通信 HRTIM 均有单片机 ADC 的供电直接耦合到 AMS1117 的 3.3V 供电并未使用基准源因为 AMS117 的滤波已经做的足够彻底了基准源的价格高节省成本。

其中各个端口配置如图中所示：

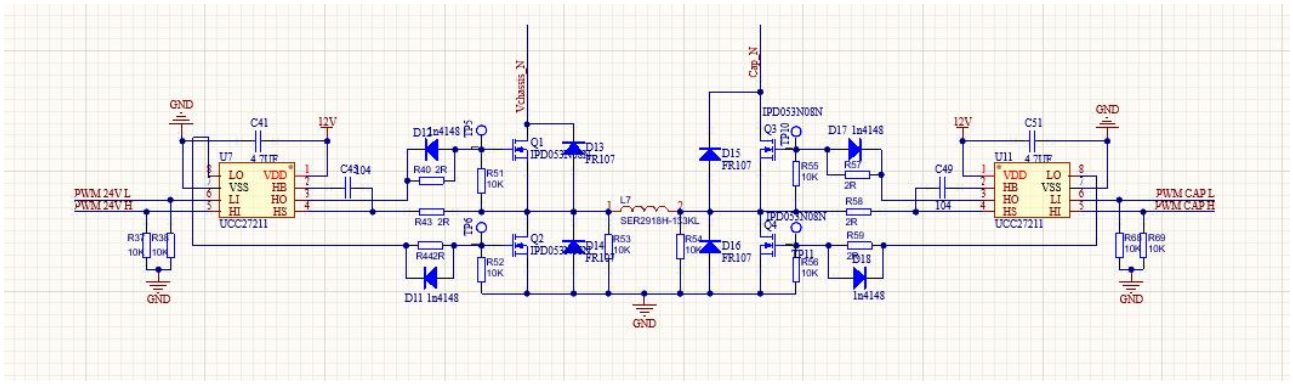
分类	引脚名称	对应信号	说明
PWM 信号	PA8	PWM1H	BUCK 上桥臂驱动
	PA9	PWM1L	BUCK 下桥臂驱动
	PA10	PWM2H	BOOST 上桥臂驱动
	PA11	PWM2L	BOOST 下桥臂驱动
ADC 信号	PA0	ADC_Vin	输入电压检测
	PA1	ADC_Iin	输入电流检测
	PA2	ADC_Vout	输出电压检测
	PA3	ADC_Iout	输出电流检测
	PA4	ADC_VADJ	滑动变阻器电压检测
串口通信	PB6	USART1_TX	USART1 发送
	PB7	USART1_RX	USART1 接收
程序下载接口	PA13	SWDAT	SWD 仿真接口
	PA14	SWCLK	
LED 指示灯	PB0	LED_G	绿灯
	PB1	LED_Y	黄灯
	PB2	LED_R	红灯
OLED	PB8	I2C1_SCL	I2C 通信时钟
	PB9	I2C1_SDA	I2C 通信数据

半桥驱动

选用 UCC27211 芯片 输入输出隔离 在 mos 或者驱动出问题的时候可以防止前级单片机引脚击穿造成毁灭性的打击芯片自带死区时间 减少 mos 同时导通带来烧毁 mos 的严重后果严格按照手册设计，实测驱动波形无尖峰保证 mos G 极波形正确工作稳定。

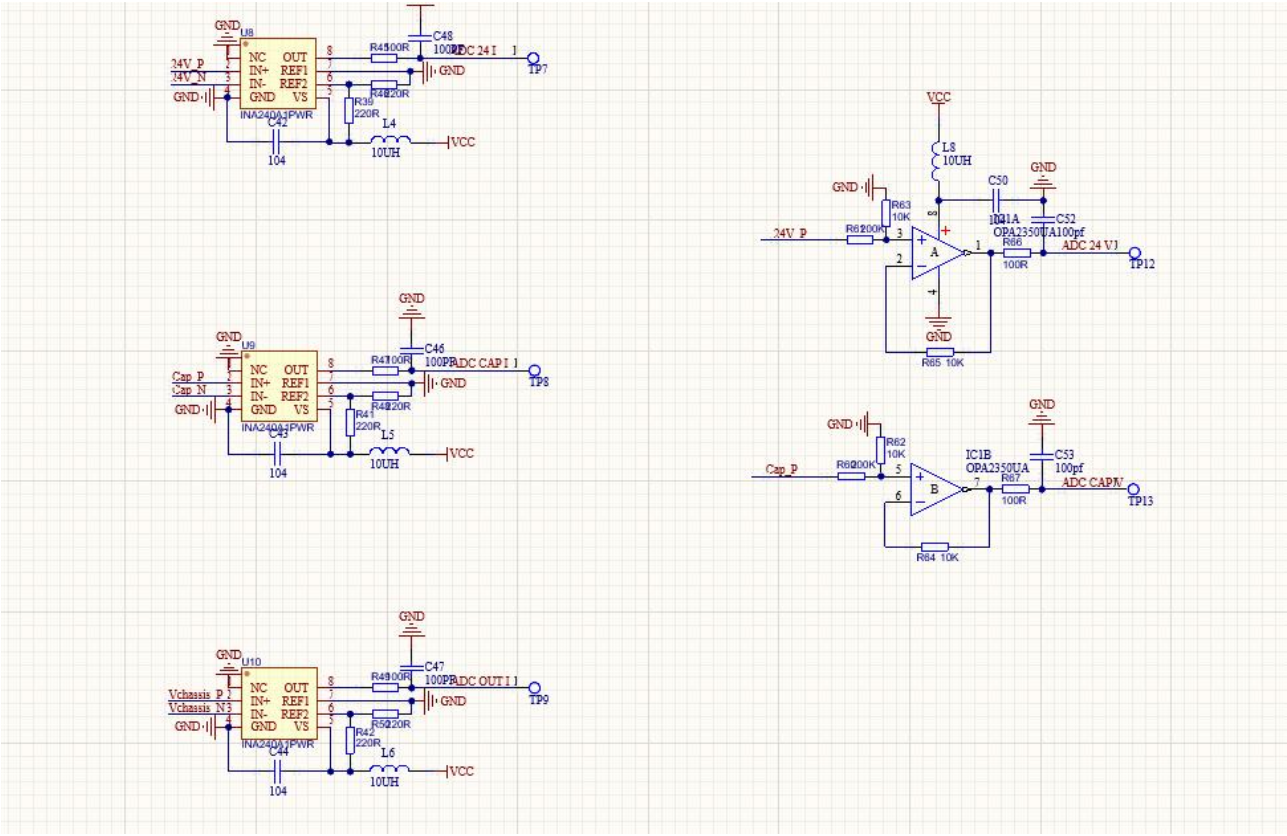


主拓扑



采样电路

INA240 双向电流检测芯片 线性度高 检测精准 三电流均可测得正负 实时检测电流流向是否异常 OPA2350 精密运放 采集电压波动值实测可以控制在 0.2V 以内检测精准



超电组方案参数以及计算

采用 bw6106 的被动均衡方案 电路简单保证鲁棒性和可维修性 均衡速率和精确度都有保证

我们选用的是 9 个 2.7v 60F 的电容单体

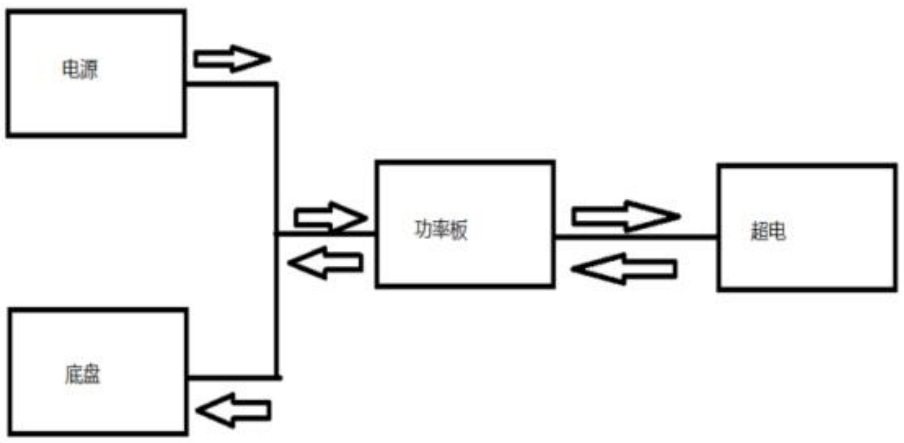
总电压=2.7*9=24.3V

单个电容容量 E=C*U*U/2=218.7j

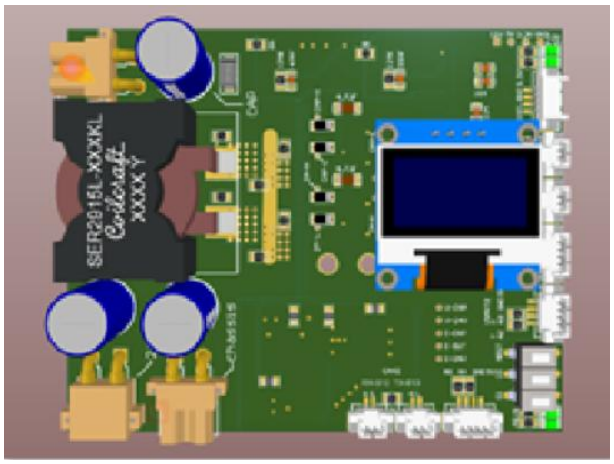
电容总容量 Em=9*E=1968.3j

实测电容总容量：1860-1940j

充电框图

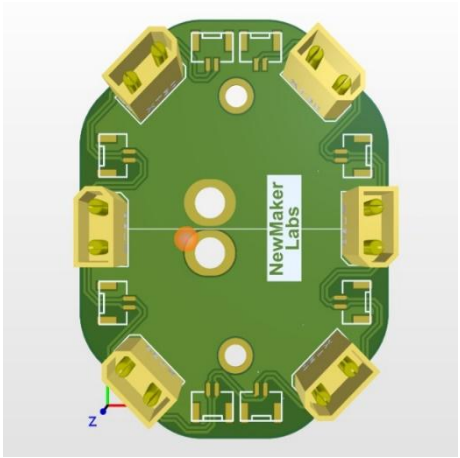


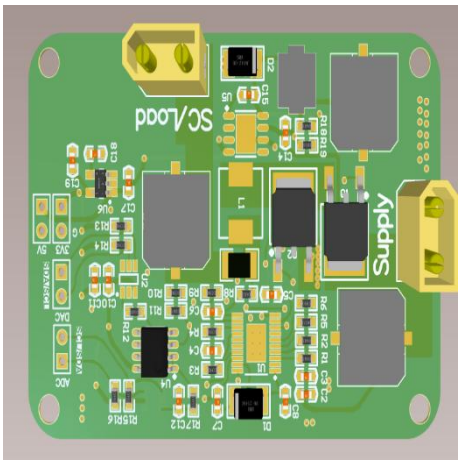
成品展示



硬件其他模块设计

比赛中经常用分电板，和各种小功能硬件模块，如 USB 转串口、jlink、可调稳压、大功率隔离稳压等，自行设计的这些模块可以更好地适配机械结构和专用的通信协议。





1.5.3 软件设计

1.5.3.1 整体架构

底盘系统的主控芯片为 STM32F427IGH，其承载哨兵机器人的底盘操作任务，接受云台系统的控制指令，完成运动分解、底盘跟随、底盘速度控制、大 YAW 的旋转、超级电容以及裁判系统通信等等功能，是机器人移动的控制中心。

云台系统主控芯片全为 STM32F427IGH，其承载哨兵机器人的云台操作任务，包含两个云台分别的 Yaw,Pitch 轴电机、拨弹电机、摩擦轮等电机控制，负责枪管的朝向的调整、射击任务逻辑控制等哨兵机器人整体控制中心。

两大系统都使用了 FreeRTOS 操作系统来规划调度各个任务的运行，同时保证任务的完成与大量信息的通信的流畅，合理利用单片机资源。

在调试过程中，皆以 Jlink 作为调试器，使用 Jscope 实时显示数据波形，远程时则使，利用 Keil5 的 debug 功能调整参数与逻辑。利用 Keil5 编译完成代码，通过 Jlink 下载器下载入单片机芯片内，上电即可运行任务。另外依靠正点原子的无线调试器来调整底盘运动中功率的变化和缓冲能量的使用率。我们还将程序分为初始化硬件模块与逻辑应用模块,实现分层设计,避免两者相互干扰,并且便于后续调试。

程序主体采用 FreeRTOS 即时多线程操作系统，主要分为个 7 个线程：

```
/* definition and creation of MonitorTask */
osThreadDef(MonitorTask, StartMonitorTask, osPriorityNormal, 0, 256);
MonitorTaskHandle = osThreadCreate(osThread(MonitorTask), NULL);

/* definition and creation of ControlTask */
osThreadDef(ControlTask, StartControlTask, osPriorityHigh, 0, 256);
ControlTaskHandle = osThreadCreate(osThread(ControlTask), NULL);

/* definition and creation of SystemTask */
osThreadDef(SystemTask, StartSystemTask, osPriorityNormal, 0, 128);
SystemTaskHandle = osThreadCreate(osThread(SystemTask), NULL);

/* definition and creation of GimbalTask */
osThreadDef(GimbalTask, StartGimbalTask, osPriorityHigh, 0, 256);
GimbalTaskHandle = osThreadCreate(osThread(GimbalTask), NULL);

/* definition and creation of FireTask */
osThreadDef(FireTask, StartFireTask, osPriorityHigh, 0, 256);
FireTaskHandle = osThreadCreate(osThread(FireTask), NULL);

/* definition and creation of VisionTask */
osThreadDef(VisionTask, StartVisionTask, osPriorityHigh, 0, 256);
VisionTaskHandle = osThreadCreate(osThread(VisionTask), NULL);
```

监控线程：检测电机，串口，以及各通信的收发情况，检测设备状态，确保在设备在线的情况下对数据进行操作。

控制线程：确定各设备初始化的完成，以及在板间通信，电机数据发送方面起着重要的作用。

系统线程：确保在各系统正常的情况下进行主要程序的操作和做好遥控器模式切换的操作

云台线程：初始化部分：在每次数据重连或者模式切换下进行云台回正的操作，在主体程序中包括遥控，比赛和自检模式下的云台，遥控模式下的云台，做到对云台YAW轴和PITCH轴的控制，比赛模式下，包括巡航和击打模式的切换，在自检模式下，开启云台摩擦轮，关闭拨弹盘做到对装甲板的预测。

开火线程：在比赛模式下通过裁判系统得知比赛开始开启摩擦轮，在进行预测后根据视觉发来的角度偏差，确保在可击打的范围内开启拨弹盘，遥控模式下，通过左拨杆，做到对摩擦轮和拨弹盘的控制。

视觉线程：包括检测是否收到视觉发来的数据，检测视觉发来数据的正确与否，以及给NUC 的发送和接受 NUC 的数据。

底盘线程：包括底盘跟随，小陀螺，底盘逆运动学结算。

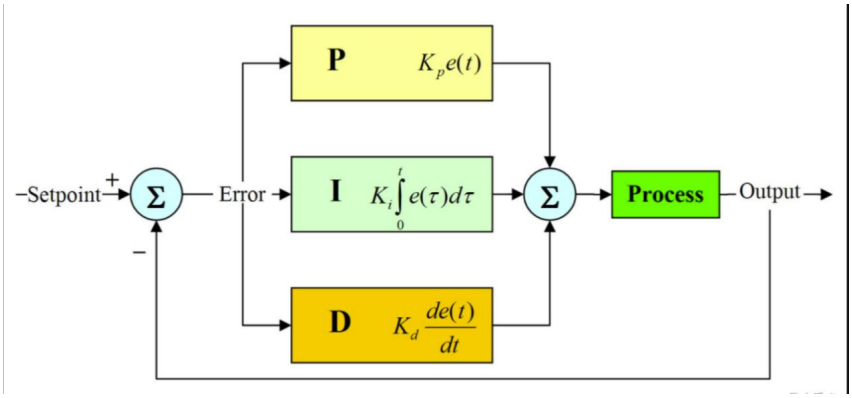
1.5.3.2 运行流程

以大 YAW 上的 C 板为主，大 YAW 上的 C 板负责底盘部分的移动与小陀螺部分，以及大 YAW 的 YAW 轴电机控制，并将裁判系统的数据、以及遥控器上的拨杆以及滚轴数据通过 CAN2 分别发送给小 YAW 上的 C 板，确保整车一体化控制。两个小 YAW 上的 C 板通过 CAN1 分别控制各自的拨弹盘电机，P 轴电机，摩擦轮电机等。

1.5.3.3 重点功能

底盘方案

我们的哨兵采用的是全向轮的四轮底盘方案，具有全向移动功能，相应的需要进行运动学解算，前后速度 x，左右速度 y 以及自旋速度 w 三个自由度运动的叠加分别赋予底盘四个电机，求出底盘四个电机后采用 PID 速度闭环计算电机电流并通过 can 发送至 C620 电调，以实现精准控速。



另外底盘需要进行功率限制，最大功率为 150w，在计算出四个电机所需电流之后还需进行电流限制，根据限制功率给定电流绝对值累加的最大值，如果超出这个范围以根号的形式缩小电流再发送至 C620 电调。

云台控制结构

为了使小陀螺时云台更稳定，我们引入了六轴陀螺仪控制，利用解算得出的欧拉角和三轴角速度来控制云台的 yaw 轴和 pitch 轴，采用陀螺仪的角度作为外环输入量，得出内环的输入量，陀螺仪的速度作为内环的输入量，最后输出电机电流，这是典型的 PID 双环结构，另外由于 pitch 轴存在限位，而陀螺仪的基准又可能存在偏差，为此我们通过电机实际角度与陀螺仪反馈的角度进行联合观测，可以计算出陀螺仪 pitch 轴的上下限幅，从而防止 pitch 轴摆动过小或者堵转的情况的发生。

另外由于赛场情况复杂，为了防止电机因为机械卡住而电机堵转而烧毁，我们引入了电机防堵转，当云台 yaw 轴和 pitch 轴电机长时间未到达目标位置时视为电机堵转，会将当前的实际陀螺仪角度附给目标角度，这样就能避免电机堵转了，另外由于调试时可能会对机器人进行无力操作，而操作会使云台位置改变，关闭无力时电机转动到之前的位置容易使调试人员受伤，而防堵转刚好解决了这个问题。

为了应对赛场激烈的战斗，我们设置了云台快速转向的按键，控制云台顺时针或逆时针旋转 90 度或 180 度，防止转动过快视野丢失，我们引入了斜坡，该斜坡遵循 sin 函数变化规则，使云台 yaw 轴转动时非常丝滑，视野也非常清晰不模糊。

为了满足三个 YAW 轴之间的协同配合通过 CAN 进行板间标志位，和数据的收发。

做好云台 PITCH 和 YAW 的限位，确保两头之间不会出现不必要的碰撞，在做好限位的情况下，做到巡航和击打的切换，避免在切换后出现两头之间的干涉，做到互不干扰，协同配合。

发射机构方案

发射为了弹道稳定我们采用无减速器的 3508 当作摩擦轮，拨弹盘采用十二格 2006 拨弹盘，3508 采用 PID 速度闭环，可以使摩擦轮转速稳定在正负 50/min，使弹道得到很大提升。关于枪口射速的读取，实时的枪口射速是通过服务器的日志来进行调试，我们通过多次测试选 择加入弹速修正的算法，实现了从裁判系统读取到的实际弹速与设定弹速的闭环控制，将弹速误差缩小在 ±0.5 米每秒以内。为了符合规则的要求，防止因枪口超热量而导致的机器人血量减少问题。我们通过对官方提供 的裁判系统通信附录的解读，成功获取到了精确的枪口热量并设置了枪口热量限制，枪口热量达到该限制值则进行低射频射击，同时根据枪口热量实时变化改变拨弹策略， 实现在击打时精确把控枪口热量。

为了防止拨弹盘意外卡弹，我们引入了卡弹检测，当拨弹盘电机较长时间未达到期望位置时且速度为零时则视为堵转，检测到堵转之后拨弹盘将自动退一发弹，然后再重新拨弹，经测试效果非常明显，也防止了拨弹盘电机的堵转烧毁。

板间通信

在双云台控制系统中，本赛季哨兵机器人一共有三块主控板分别位于两个小 YAW 云台，大 YAW 云台，其中大 YAW 云台负责接收遥控器信息，控制小 YAW 云台击打，底盘负责控制底盘运动，接收裁判系统信息，下云台负责下云台击打功能，要想双云台三主控要有良好的效果，三主控之间一定要有良好的通信功能，在此基础上，我们使用 can 路实现上-中，中-下两两主控间通信，上云台将遥控器信息，击打目标情况发送给底盘主控，底盘主控再发送给下云台主控，底盘主控将接收的裁判系统信息，例如机器人 ID，双枪口的弹速，枪口热量，车间交互等数据发送给上下云台，从而实现了只有一个遥控器，就能完整控制三个主控板执行相应功能。

```
    if(sys.state == SYS_STATE_NORMAL)
    {
        rc_data = RP_SET_BIT(rc_data,8);
/*给视觉发的打红打蓝*/
        if(judge_sensor.info->GameRobotStatus.robot_id == 7)                // 自身红色
        {
            rc_data = RP_CLEAR_BIT(rc_data,1);                            // 10   打蓝
            rc_data = RP_SET_BIT(rc_data,2);
        }
        else if(judge_sensor.info->GameRobotStatus.robot_id == 107)         // 自身蓝色
        {
            rc_data = RP_SET_BIT(rc_data,1);                          // 01       打红
            rc_data = RP_CLEAR_BIT(rc_data,2);
        }
        else
        {
            rc_data = RP_CLEAR_BIT(rc_data,1);                          // 00
            rc_data = RP_CLEAR_BIT(rc_data,2);
        }
/*比赛开始标志位*/
        if(judge_sensor.info->GameStatus.game_progress == 4)
        {
            rc_data = RP_SET_BIT(rc_data,3);
        }
        else
        {
            rc_data = RP_CLEAR_BIT(rc_data,3);
        }
/*模式选择位*/
        if(sys.remote_mode == RC)                                           //遥控模式  11
        {
            rc_data = RP_SET_BIT(rc_data,4);
            rc_data = RP_SET_BIT(rc_data,5);
        }
        else if(sys.remote_mode == AUTO)                                    //自动模式  01
        {
            rc_data = RP_SET_BIT(rc_data,4);
            rc_data = RP_CLEAR_BIT(rc_data,5);
        }
        else if(sys.remote_mode == INSPECTION)                              //自检模式  10
        {
            rc_data = RP_CLEAR_BIT(rc_data,4);
            rc_data = RP_SET_BIT(rc_data,5);
            /*左右云台选择*/
            if( abs(rc_sensor.info->thumbwheel) >=630 )
            {
                tum_cnt++;
                if(tum_cnt > 100)
                {
                    if(rc_sensor.info->thumbwheel >= 630)
                    {
                                rc_data = RP_SET_BIT(rc_data,6);        //左云台 01
                                rc_data = RP_CLEAR_BIT(rc_data,7);
                    } else if(rc_sensor.info->thumbwheel <= -630)
                    {
                                rc_data = RP_CLEAR_BIT(rc_data,6);      //右云台 10
                                rc_data = RP_SET_BIT(rc_data,7);
                    }
                    tum_cnt = 0;
                }
            }
            /*拨动单盘*/
            if(sys.gimbal_now == MASTER_UP)
            {
                dream = RP_CLEAR_BIT(dream,3);
                dream = RP_CLEAR_BIT(dream,4);
                if(rc_sensor.info->s1_siwtch_up)
                {
                    if((dream & 0x0001) != 0x0001)
                    {
                        dream = RP_SET_BIT(dream,1);
                    }
                    else
                    {
                        dream = RP_CLEAR_BIT(dream,1);
                        dream = RP_CLEAR_BIT(dream,2);
```

另外，为了不占用额外资源，在发送对应信息时将通信内容尽量精简，在进行例如颜色，是否开火的状态标志位时，发送和读取都进行按位进行的方案，极大的节省了 CPU 资源，大大的提高了通信效率。

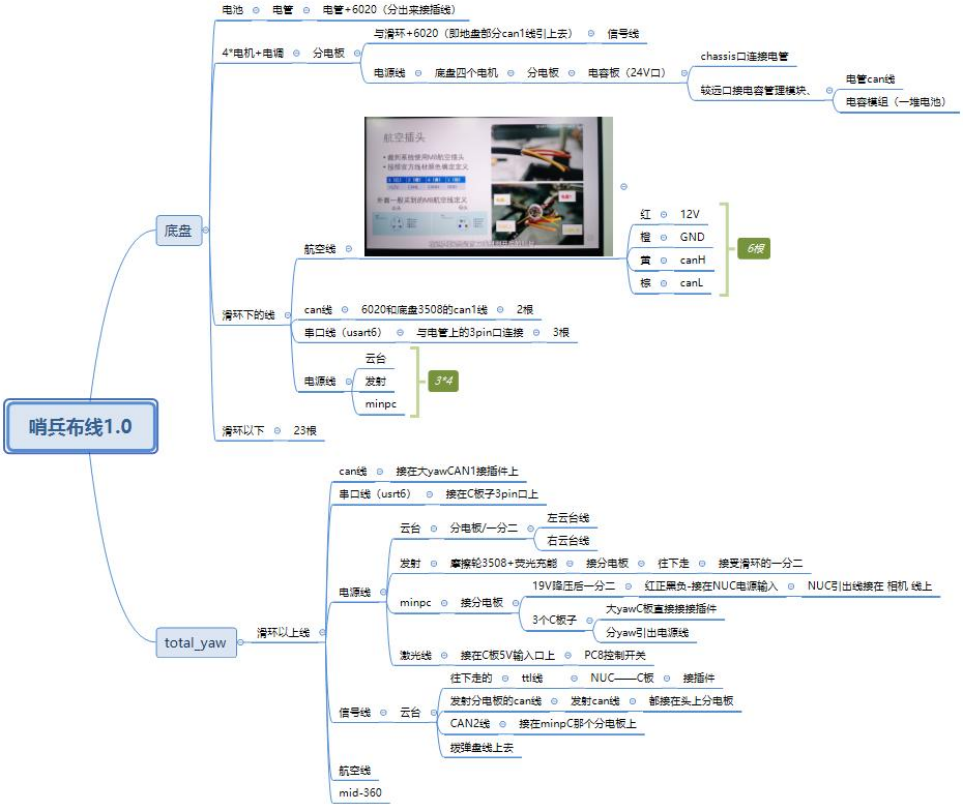
电容

引入功率环和能量缓冲环，进行并行 PID 计算，实现超级电容自动充放电。

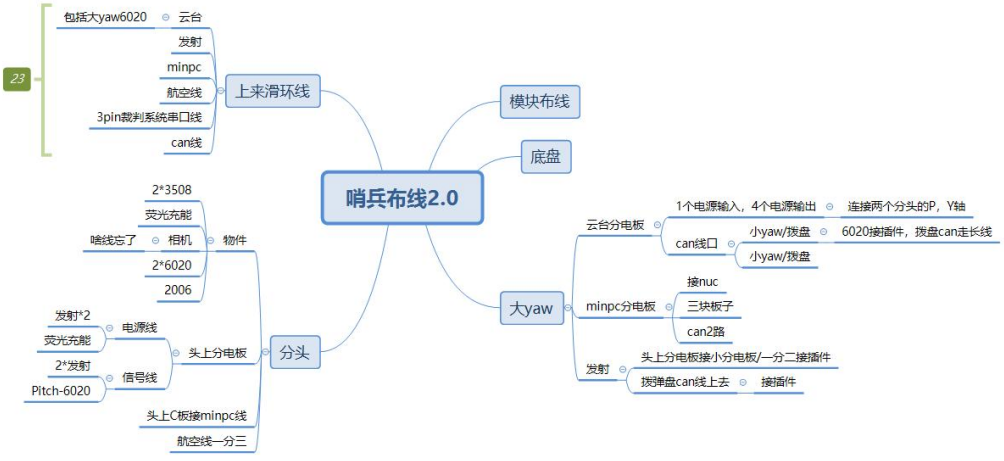
```
/*******************************************************************************
  * @brief                                  自动模式
  * @author                                 Zhu Beibei NewMaker
  * @param                                  加入限制功率环和缓冲能量环，
  * @param                                  充放电完全自动进行，无需手动开关
  * @retval                                 返回无
  *******************************************************************************/
float power_limit_t;
void Power_Limit_Loop()
{

//增量式
    //关闭电容
    if(can_data.Model==3)
    {
        hrpwm.Power_Out=0;
    }
    else
    {
            if(hrpwm.Loop_cnt>4||hrpwm.Loop_cnt<-4)//pid积分抗饱和，限制i
            {
                    //限制功率环
                    velocity_control_angle(&Out_Power_Loop,can_data.power_limit,adc.Power_t,Max_Power,35000.0f);
                    //缓冲能量环
                    velocity_control_angle(&Out_Buff_Loop,can_data.power_buff,50,Max_Power,20000.0f);
            }
            else
            {
                    //限制功率环
                    velocity_control_angle(&Out_Power_Loop,can_data.power_limit,adc.Power_t,Max_Power,260000.0f);
                    //缓冲能量环
                    velocity_control_angle(&Out_Buff_Loop,can_data.power_buff,50,Max_Power,80000.0f);

                  //只充不放模式下的pid积分抗饱和
                  if(Out_Power_Loop.iout<-20000&&can_data.Model==2)  Out_Power_Loop.iout=-20000;
                  if(Out_Buff_Loop.iout<-20000&&can_data.Model==2)   Out_Buff_Loop.iout=-20000;
            }

            //双环并行输出，取小
            if(Out_Power_Loop.out_angle_total>Out_Buff_Loop.out_angle_total)
                   hrpwm.Loop_Out=Out_Buff_Loop.out_angle_total;
            else
                   hrpwm.Loop_Out=Out_Power_Loop.out_angle_total;

            //充电模式不放电
            if(can_data.Model==2)
            {
                if(hrpwm.Loop_Out<0)     hrpwm.Loop_Out=0;
            }

            hrpwm.Power_Out=hrpwm.Loop_Out;
    }

//位置式
```

1.5.3.4 软件测试

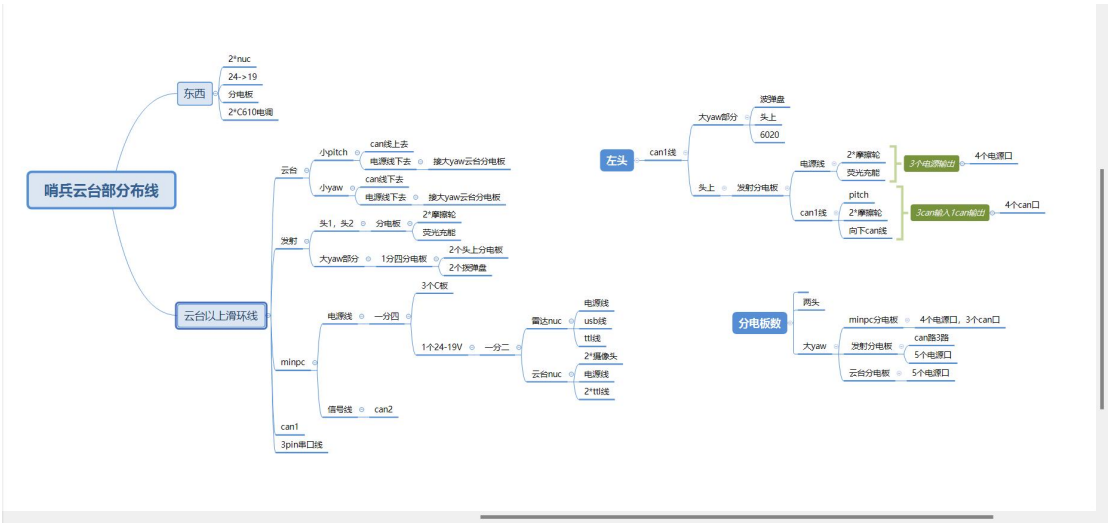
关于布线测试三次，最终确定为最后一板。



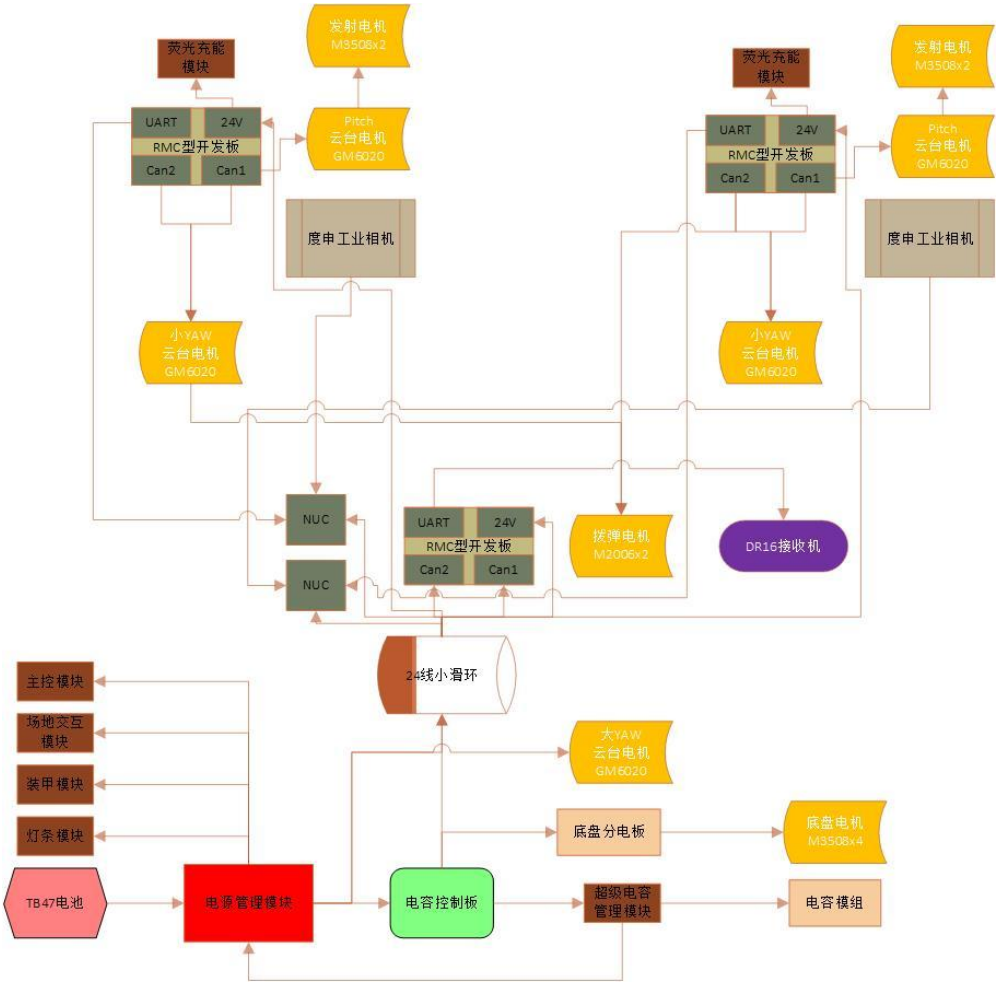
哨兵布线 1.0



哨兵布线 2.0

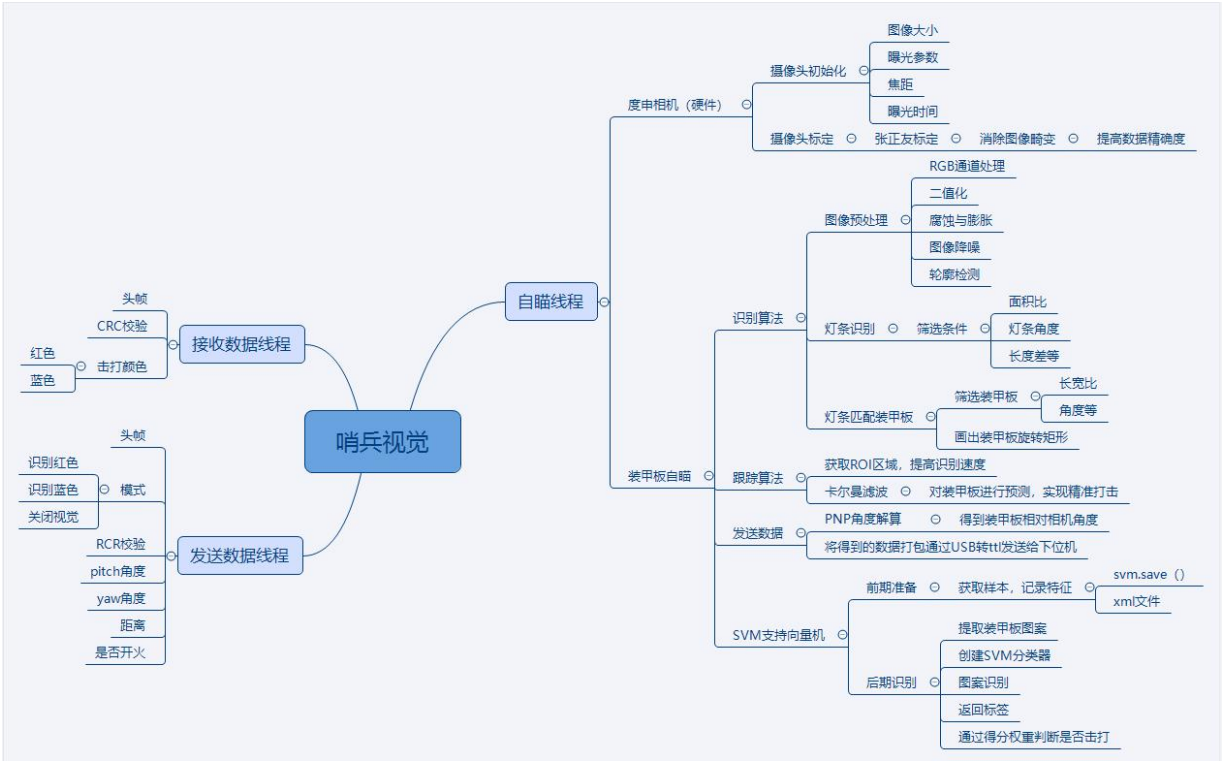


哨兵布线 3.0



哨兵布线 4.0

1.5.4 算法设计



1.5.4.1 识别思路

1. 通过电控发送的数据确定击打颜色，进行通道分离，阈值筛选，通道相减，二值化，膨胀等过程，为后期框选识别做准备。

```
void ImgProcessing::Deal(Mat &img)
{


    split(img,bgr_channels);              // 原来的图像处理
    inRange(img,Scalar(B,G,R),Scalar(255,255,255),bright_img);
    //imshow("bright_img",bright_img);
    switch (enemycolor)
    {
    case RED:
        subtract(bgr_channels[2],(min_r*0.001)*bgr_channels[1],sub_img);
        threshold(sub_img,sub_img,min,255,THRESH_BINARY);
        dilate(sub_img,sub_img,element);
        final_img=sub_img & bright_img;
        //imshow("sub_img",sub_img);
        break;
    case BLUE:
        subtract(bgr_channels[0],(min_b*0.001)*bgr_channels[1],sub_img);
        threshold(sub_img,sub_img,min,255,THRESH_BINARY);
        dilate(sub_img,sub_img,element);
        final_img=sub_img & bright_img;
        //imshow("sub_img",sub_img);
        break;
        //imshow("final",final_img);
    }
```

2. 对处理完的图像进行轮廓检测，并计算最小包围矩形，根据灯条尺寸，装甲板两边尺寸等条件，筛选出灯条轮廓。为后续筛选装甲板，PNP 角度解算做准备。

```
ArmorBag::ArmorBag(Mat &img, const LightBag & l_light, const LightBag & r_light) {//用左右灯条初始化ArmorBag
    this->l_light = l_light;
    this->r_light = r_light;

    armorAngle = (l_light.angle + r_light.angle) / 2;


    Size L_Size1(float(l_light.lightRect.size.width), float(l_light.lightRect.size.height));//灯条尺寸
    Size R_Size1(float(r_light.lightRect.size.width), float(r_light.lightRect.size.height));
    Size L_Size2(float(l_light.lightRect.size.width), float(1.8*l_light.lightRect.size.height));//装甲板两边尺寸
    Size R_Size2(float(r_light.lightRect.size.width), float(1.8*r_light.lightRect.size.height));
    RotatedRect LLight(l_light.center, L_Size1, this->armorAngle),  LSide(l_light.center, L_Size2, this->armorAngle);
    RotatedRect RLight(r_light.center, R_Size1, this->armorAngle),  RSide(r_light.center, R_Size2, this->armorAngle);
    Point2f     ll[4],rr[4],                                        lside[4],rside[4];
                LLight.points(ll);RLight.points(rr);                LSide.points(lside);RSide.points(rside);
    Point2f     l_up = ll[2], l_low = ll[3],                        ls_up = lside[2], ls_low = lside[3];
    Point2f     r_up = rr[1], r_low = rr[0],                        rs_up = rside[1], rs_low = rside[0];
    light_Vertices.emplace_back(l_up);   armor_Vertices[0]=ls_up;
    light_Vertices.emplace_back(r_up);   armor_Vertices[1]=rs_up;
    light_Vertices.emplace_back(r_low);  armor_Vertices[2]=rs_low;
    light_Vertices.emplace_back(l_low);  armor_Vertices[3]=ls_low;

    armor_area = (int)(light_Vertices[1].x-light_Vertices[0].x)*(light_Vertices[3].y-light_Vertices[0].y);

    armor_center.x=(r_light.center.x + l_light.center.x)/2;
    armor_center.y=(r_light.center.y + l_light.center.y)/2;

    center_diff=(armor_center.x-0.5*img.cols)*(armor_center.x-0.5*img.cols)+(armor_center.y-0.5*img.rows)*(armor_center.y-0.5*img.rows
```

3. 根据灯条平行度，中心连线水平度，灯条长宽比，左右灯条长度差比值等进行装甲板初步筛选。

```
float ArmorBag::lightAngleDiff() const// 灯条平行度——装甲板左右灯条角度差
{
    float light_angle_diff = abs(l_light.angle-r_light.angle);
    cout<<"1.lightAngleDiff"<<"  "<<light_angle_diff<<endl;
    return light_angle_diff;
}

float ArmorBag::lightCenterDeviation() const// 中心连线水平度——两灯条中心连线与水平线夹角
{
    float delta_x = r_light.center.x - l_light.center.x;
    float delta_y = r_light.center.y - l_light.center.y;
    float center_angle = abs(delta_y/delta_x);
    cout<<"2.center_angle"<<"  "<<center_angle<<endl;
    return center_angle;
}
/* 标准小装甲比--12.3/5.7=2.158      极限--   最小--6.00/5.7=1.05    最大--12.3/4.0=3.07
 * 标准大装甲比--23.1/5.8=3.982      极限--   最小--12.0/5.7=2.10    最大--23.1/4.0=5.77       理论值
 */
float ArmorBag::lengthDistanceProportion() const//灯条距离与灯条长度比——距离比/两灯条长度平均值
{
    float delta_x = r_light.center.x - l_light.center.x;
    float H_distance_proportion = abs(2*delta_x/(l_light.lightRect.size.height+r_light.lightRect.size.height));
    //cout<<"3.H_distance_proportion"<<"  "<<H_distance_proportion<<endl;
    return H_distance_proportion;
}

float ArmorBag::lengthDiffProportion() const// 左右灯条长度差比值
{
    float H_diff = abs(l_light.lightRect.size.height-r_light.lightRect.size.height);
    float H_diff_proportion = 2*H_diff/(l_light.lightRect.size.height+r_light.lightRect.size.height);
    cout<<"4.H_diff_proportion "<<"  "<<H_diff_proportion <<endl;
```

4. 通过透视变换处理图像后，进行数字识别，以确保识别无误，以便后续确定击打优先级。

```
void ArmorBag::getWrap(Mat &img, Point2f vertices[4])//对装甲板图案作透视变换，20*20的大小，给svm分类
{
    Mat rotation;
    Point2f dst_points[4];
    dst_points[0] = Point2f(0.0,    0.0  );
    dst_points[1] = Point2f(20.0,   0.0  );
    dst_points[2] = Point2f(20.0,   20.0 );
    dst_points[3] = Point2f(0.0,    20.0 );
    rotation = getPerspectiveTransform(vertices,dst_points);//透视方式（最好写透视变换矩阵，才不显的low）
    warpPerspective(img,armor_warp,rotation,Size(20,20));

    //gamma_correction(img, img, 300);      //gramma矫正使数字变亮
    //imshow("armor_warp",armor_warp);
    cvtColor(armor_warp,armor_warp,COLOR_BGR2GRAY);//以下为作二值化处理
    threshold(armor_warp,armor_warp,30,255,THRESH_BINARY_INV);
}
```

5. 通过以上操作，确定极大优先级，发送击打信息给电控，以完成自动开火。

1.5.4.2 重要算法原理阐述、公式推导

1. SVM 数字识别

数字识别是装甲板检测的最后一项工作，也是能否击打装甲板的关键，是体现鲁棒性最重要的一环。

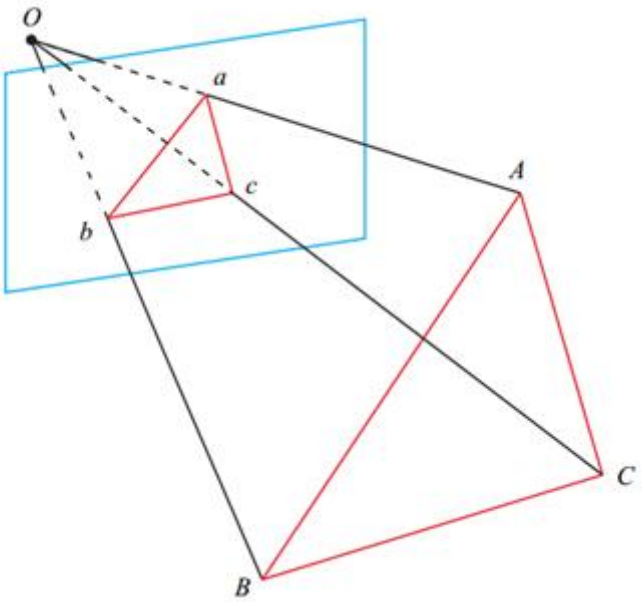
（1） 通过透视变换将各个角度的数字转换到正面，实现平衡性和共性比例不变性，提高分类效率。

（2） 通过加权来设置对步兵，英雄以及工程的击打优先级，以便更好的发挥哨兵的作用。

```
if(armors.size() != 0)//判断这次是否找到装甲板
{
    if(lastArmor.score == 0)//判断上次目标是否为空
    {
        for(uint i = 0 ; i < armors.size();i++)//遍历装甲板，附分，找到得分最高的
        {
            armors[i].score += (int)armors[i].armor_area/10;              //根据装甲板面积大小加分
            armors[i].score += (int)armors[i].armor_size*100;             //根据装甲板大小类型加分
            armors[i].score -= (int)armors[i].center_diff/100;            //根据装甲板中心距图像正中心距离减分
            if(armors[i].robotType == 2){armors[i].score -= 500*i;}       //若是工程，则减去对应得分
        }
        sort(armors.begin(), armors.end(),[](ArmorBag & a1, ArmorBag & a2)//根据装甲板得分从大到小排序
        {return a1.score > a2.score; });
        targetArmor = armors[0];//上次目标为空，赋值
        lastArmor = targetArmor;
    }
    else
    {
        for(uint i = 0 ; i < armors.size();i++)
        {
            armors[i].score += (int)armors[i].armor_area/10;              //根据装甲板面积大小加分
            armors[i].score += (int)armors[i].armor_size*100;             //根据装甲板大小类型加分
            armors[i].score -= (int)armors[i].center_diff/100;            //根据装甲板中心距图像正中心距离减分
            if(armors[i].robotType == 2){armors[i].score -= 500*i;}       //若是工程，则减去对应得分
        }
        sort(armors.begin(), armors.end(),[](ArmorBag & a1, ArmorBag & a2)//根据装甲板得分从大到小排序
        {return a1.score > a2.score; });
        if(abs(armors[0].armor_center.x-lastArmor.armor_center.x)+abs(armors[0].armor_center.y-lastArmor.armor_center.y)>60)
        {                                                                                                    //不必说
            if(lastArmor.robotType == armors[0].robotType)//判断这次目标装甲类型和上次是否一样
            {
                targetArmor = armors[0];//位置相差太大，但装甲板类型相同，则直接赋值
                lastArmor = targetArmor;//跳变过大，预测时需将这个视为异常数据
                T=2;
            }
```

2. PNP 角度解算

通过处理，判断得到最佳装甲板后，将其进行 PNP 角度解算，得到相机坐标系中目标装甲板的坐标和相应的姿态角。经过坐标系变换到云台坐标系，并再加一定的运算得到云台到目 标装甲板的 pitch，yaw，距离等信息。提取出装甲板，旋转臂，旋转中心等关键要素，分析 出目标形状位置及有效击打区域。



Z 轴：

= 2(21, 11)

Y 轴：

= 2(− 31, √ 32 2 + 33 2)

X 轴：

= 2(32, 33)

通过以上处理来解算出装甲板位姿，为后续准确击打做准备。

3. 卡尔曼滤波

简介：卡尔曼滤波（Kalman filtering）是一种利用线性系统状态方程，通过系统输入输出观测数据，对系统状态进行最优估计的算法。由于观测数据中包括系统中的噪声和干扰的影响，所以最优估计也可看作是滤波过程。数据滤波是去除噪声还原真实数据的一种数据

处理技术，Kalman 滤波在测量方差已知的情况下能够从一系列存在测量噪声的数据中，估计动态系统的状态。由于它便于计算机编程实现，并能够对现场采集的数据进行实时的更新和处理，Kalman 滤波是目前应用最为广泛的滤波方法，在通信，导航，制导与控制等多领域得到了较好的应用。卡尔曼滤波的应用领域非常广泛，在航空航天、信息技术等领域尤为广泛。在我的理解里，卡尔曼滤波就相当于一个带有权重的低通滤波，即调整观测值的权重(测量值)和估计值的权重，是更相信观测值还是更相信估计值的一个过程，观测值*权重+估计值*权重 = 修正值，即最优估计。

卡尔曼滤波五大公式

1. 向前推算状态变量

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

2. 向前推算误差协方差

$$P_k^- = AP_{k-1}A^T + Q$$

3. 计算卡尔曼增益

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R}$$

4. 由观测变量z_k更新估计

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

5. 更新测量误差

$$P_k = (I - K_k H)P_k^-$$

4. 自启动：

通过写脚本和用 Ubuntu 自带的‘Startup Applications’软件进行自启动。

5. 串口通信：

通过 USB 转 ttl 模块，变化为 4pin 线接入主控板，与嵌入式达成通信协议，将数据发送给下位机，同时开放串口权限，并通过调用函数收取下位机发送的数据。

6. 硬件：

Intel NUC11、度申工业相机、USB2ttl。

1.5.4.3 算法性能，优缺点分析优化方案

算法性能：

目前可以精准框选装甲板以进行击打，图像处理速度较快，可以很好的完成视觉的基本功能。

优点：

1. 本赛季新加了数字识别，使识别跟稳定，准确性更高，，以保证更强的鲁棒性。
2. 设置了击打优先级，在识别到多兵种的情况下，可通过优先级进行击打，将哨兵作用最大化。

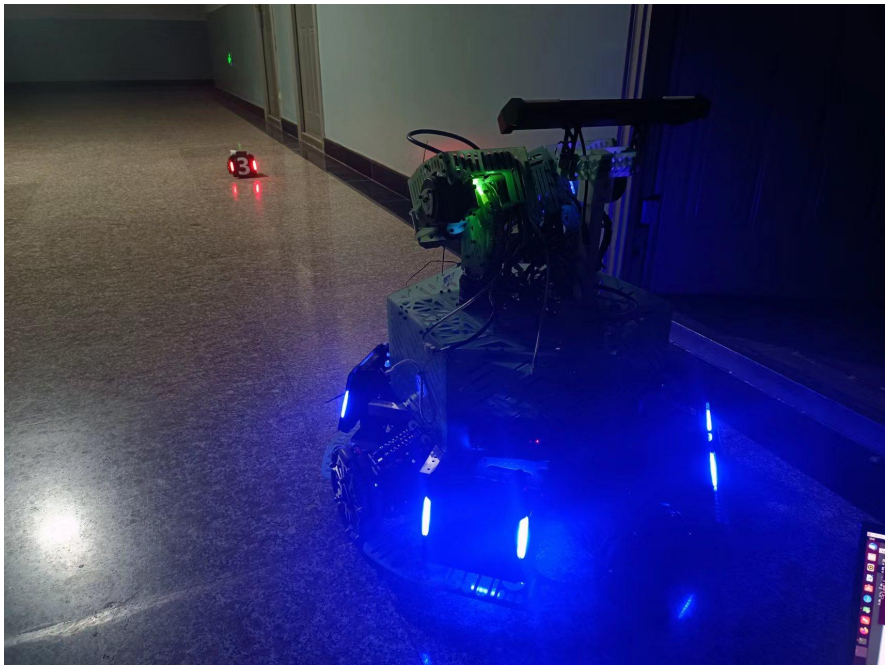
缺点：

1. 数字识别在光线较暗的情况下无法框选，有待提升。
2. 在装甲板移动过快的情况下会掉识别，距离较远时识别会抖动等。

优化方案：

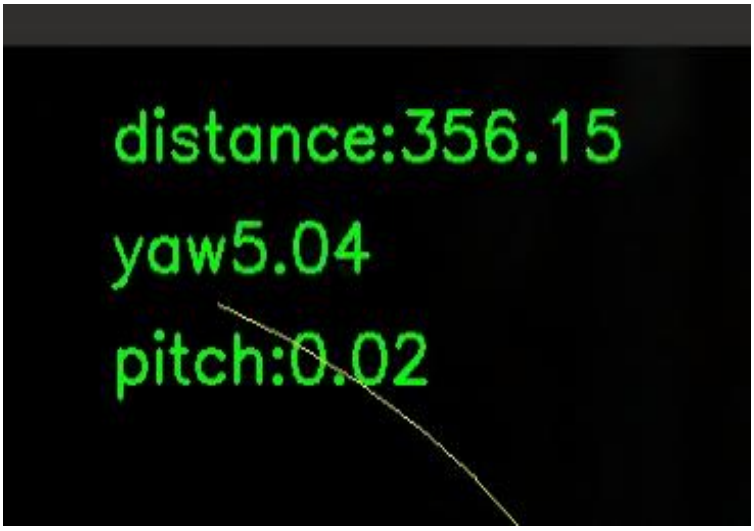
1. 前预测是嵌入式加的，打完省赛计划试试视觉预测，可以的话突破反陀螺。
2. 省赛后将实现单 NUC 控制双相机，为雷达单留一个 NUC 以最大化其性能。

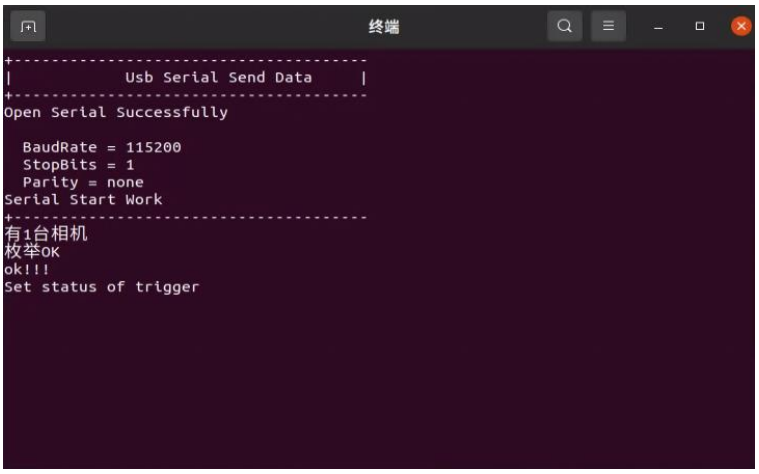
1.5.4.4 算法结果




Dpendil img
delay:22.1ms
distance:376.10
yaw5.03
pitch:0.02
num:3
target


num:3
target

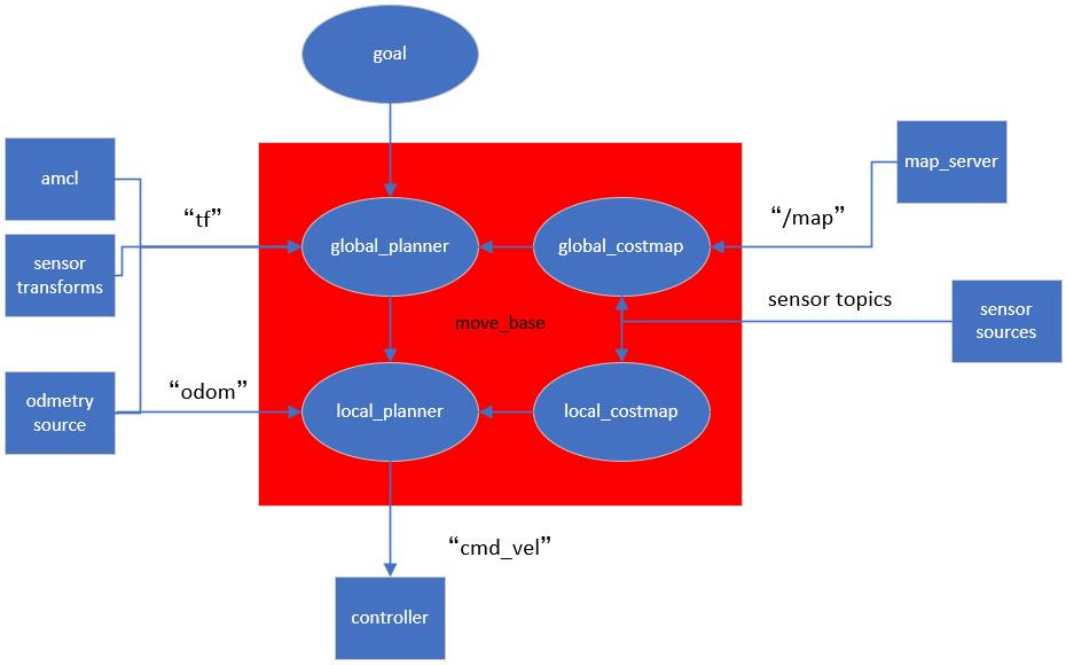

distance:356.15
yaw5.04
pitch:0.02


终端
+-------------------------------------
| Usb Serial Send Data |
+-------------------------------------
Open Serial Successfully
BaudRate = 115200
StopBits = 1
Parity = none
Serial Start Work
+-------------------------------------
有1台相机
枚举OK
ok!!!
Set status of trigger

1.5.5 其它

哨兵机器人在本赛季采用了 SLAM 技术，底层采用 ROS（Robot Operating System 机器人操作系统），在真实世界当中实现了机器人自主导航的功能。

1.5.5.1 导航模块简介



关键技术：

全局地图、自身定位、路径规划、运动控制、环境感知。

- 全局地图

SLAM，即时定位与地图构建，运用此技术实现哨兵机器人在未知环境中从一个未知位置开始移动，在移动过程中根据位置估计和地图进行自身定位，同时在自身定位的基础上建造增量式地图，以绘制出外部环境的完全地图。我们采用的是 ROS 较为常用的 gmapping 算法进行全局地图构建，能够做到边跑边建图，然后利用 map_server 来保存地图。在更换为 Livox Mid-360 激光雷达（以下简称 360 激光雷达）后采用了 Fastlio2 算法对全局静态地图进行构建。

- 自身定位

定位算法采用的是 ROS 当中的 amcl（自适应的蒙特卡洛定位），用于 2D 移动机器人的概率定位系统，使用粒子过滤器根据已知地图跟踪机器人的姿态。在更换为 360 激光雷达后采用的是用其自带 IMU 进行自身定位。

- 路径规划

  采用的是 ROS 中提供的 move_base 包来实现路径规划，主要分为两大规划器：

  1. 全局路径规划

     根据给定的目标点和全局地图实现总体的路径规划，使用 A* 算法进行全局路径规划，计算最优路线，作为全局路线。

  2. 局部路径规划

     实现动态避障，并选取当前最优路径以尽量符合全局最优路径。

- 运动控制

Move_base 会通过话题“cmd_vel”发布 geometry_msgs/Twist 类型的消息，传递的是运动命令，基于机器人的基座标系。串口通信通过订阅此话题将速度命令转换为电机命令并发送。

- 环境感知

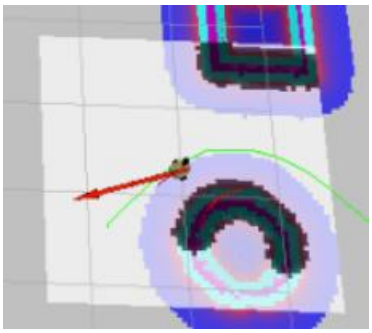
  通过激光雷达感知外界环境的深度信息。

1.5.5.2 导航具体顺序

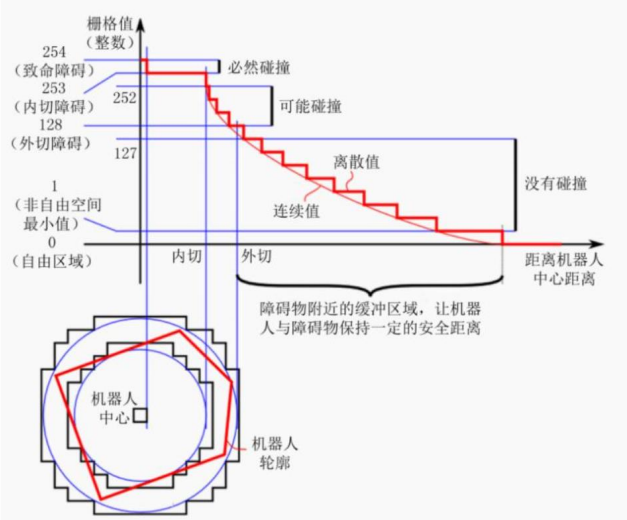
- SLAM 建图
  - Fastlio2：根据移动机器人里程计数据和激光雷达数据来绘制三维维的栅格地图。
  - 编写 launch 文件启动 360 激光雷达建图。
- 地图保存
  - map_server 功能包中提供了两个节点: map_saver 和 map_server，前者用于将栅格地图保存到电脑，后者读取电脑的栅格地图并以服务的方式提供出去。
- 定位
  - 定位就是推算机器人自身在全局地图中的位置，当然，SLAM 中也包含定位算法实现，不过 SLAM 的定位是用于构建全局地图的，是属于导航开始之前的阶段，而当前定位是用于导航中，导航中，机器人需要按照设定的路线运动，通过定位可以判断机器人的实际轨迹是否符合预期。在 ROS 的导航功能包集 navigation 中提供了 amcl 功能包，用于实现导航中的机器人定位。

- AMCL(adaptive Monte Carlo Localization) 是用于 2D 移动机器人的概率定位系统，它实现了自适应（或 KLD 采样）蒙特卡洛定位方法，可以根据已有地图使用粒子滤波器推算机器人位置。

- 路径规划
  - move_base 功能包提供了基于动作(action)的路径规划实现，move_base 可以根据给定的目标点，控制机器人底盘运动至目标位置，并且在运动过程中会连续反馈机器人自身的姿态与目标点的状态信息。
  - 代价地图
    - SLAM 构建的地图是静态地图，而导航过程中，障碍物信息是可变的，可能障碍物被移走了，也可能添加了新的障碍物，导航中需要时时的获取障碍物信息。
    - 在靠近障碍物边缘时，虽然此处是空闲区域，但是机器人在进入该区域后可能由于其他一些因素，比如：惯性、或者不规则形体的机器人转弯时可能会与障碍物产生碰撞，安全起见，最好在地图的障碍物边缘设置警戒区，尽量禁止机器人进入。
    - 组成
      - 静态地图层，SLAM 构建的静态地图。
      - 障碍地图层，传感器感知的障碍物信息。
      - 膨胀层，在以上两层地图上进行膨胀（向外扩张），以避免机器人的外壳会撞上障碍物。
      - 自定义 costmap。



- 碰撞算法
  - 致命障碍:栅格值为 254，此时障碍物与机器人中心重叠，必然发生碰撞。
  - 内切障碍:栅格值为 253，此时障碍物处于机器人的内切圆内，必然发生碰撞。
  - 外切障碍:栅格值为[128,252]，此时障碍物处于其机器人的外切圆内，处于碰撞临界，不一定发生碰撞。



  - 非自由空间:栅格值为(0,127]，此时机器人处于障碍物附近，属于危险警戒区，进入此区域，将来可能会发生碰撞。
  - 自由区域:栅格值为 0，此处机器人可以自由通过。
  - 未知区域:栅格值为 255，还没探明是否有障碍物。

1.5.5.3 雷达站

针对哨兵的特点，为哨兵开发了哨岗辅助模块。

该模块主要由目标检测和目标跟踪两部分构成，目标检测部分采用轻量化便于部署的 Yolov-little 模型，通过激光雷达与相机的配合使用，重建赛场地图，实时检测目标位置；目标跟踪部分主要基于 deepsort 算法进行修改，实现对场上机器人跟踪，再通过车端和哨岗的通信将哨岗信息同步到车端小地图，用于决策和小地图自瞄。

目标检测

通过激光雷达与相机联合标定，使用官方的软件得到的点云图制作深度图，将神经网络检测得到的目标坐标与深度图得到较为准确的目标位置与 ID。

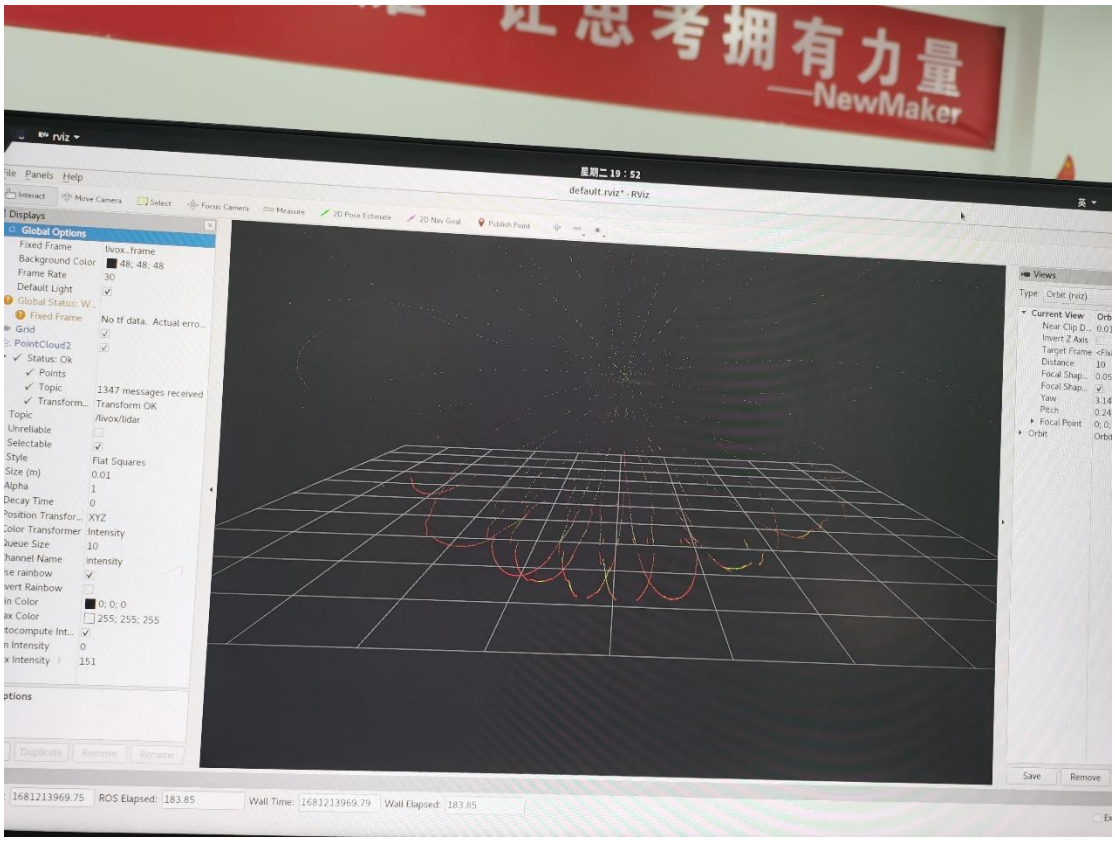
跟踪器

首先我们构造跟踪器，用 vector 保存目标对象的信息，包括上一/几帧中目标的大小/位置/速度等状态，以及最后一次 更新的时间戳等。再加上卡尔曼进行预测和更新，之后将两个相机的检测结果与轨迹预测结果根据 IOU 利用匈牙利算法进行匹配，再将初步匹配的结果与从裁判系统接收到的我方机器人 ID 与位置进行对比进行修正，同时为了提高跟踪的实时性，加入了检测 ROI，下一帧图像只在 ROI 上运行检测算法。

算法优缺点

中近距离情况下，检测目标较为准确，对于远端目标，由于清晰度的下降，机器人 ID 误识别较为严重，由于加入了 ROI，大大降低了目标跟踪算法的算力占用，提高了实时性。


map
ROBOMASTER
ROBOMASTER
1
2
1


—NewMaker
rviz
File Panels Help
default.rviz* - RViz
Displays
Global Options
Fixed Frame
Background Color
Frame Rate
Default Light
Grid
PointCloud2
Status: Ok
Points
Topic
Transform...
Unreliable
Selectable
Style
Size (m)
Alpha
Decay Time
Views
Current View
Save
Remove
ROS Elapsed: 183.85
Wall Time: 1681213969.79
Wall Elapsed: 183.85

1.6 研发迭代过程

1.6.1 测试记录

发射功能

测试环境：整车装配完成，750 发小弹丸填充完成

测试设备：自定义遥控器、大装甲板

测试结果：可以做到 5 米内全中

子云台巡航功能

测试环境：整车装配完成，750 发小弹丸填充完成

测试设备：笔记本

测试结果：符合最初设置巡航路径

平移功能

测试环境：整车装配、大理石底板

测试设备：自定义遥控器

测试结果：完成平移，有路径偏移现象

小陀螺功能

测试环境：整车装配、大理石地板、750 发弹丸

测试设备：自定义遥控器

测试结果：小陀螺稳定且转速超预期

1.6.2 版本迭代过程记录

版本号或阶段	功能或性能详细说明	完成时间
V1.0	新哨兵出图	2023.2.26
V1.1	底盘和云台加工并且装配完成	2023.3.26
V1.2	软件组布线完成，硬件组将超级电容和电容板安装完毕，实现框选装甲板	2023.4.1
V1.3	视觉组加入自瞄，开始自主运动和路径规划，实现角度解算	2023.4.5

V1.4	实现基础功能，多次训练不断优化，与嵌入式完成串口通讯，主 YAW 轴 3 电机方案测试	2023.4.6
V1.5	基本功能做稳定，加上线性弹频，小幅度大 yaw 巡航，基本实现单相机自瞄全部功能，3508 减速箱方案设计成型，并加工，分区赛底盘装配布线完成	2023.5.5
V1.6	解决拨弹盘疯转，去掉大 yaw 巡航，还是进行 3508 大 yaw 的研究，对功控进行迭代，开始研发一个 nuc 控制两个相机	2023.5.9
V1.7	实现 3508 大 yaw 的全部功能并开始配合导航进行控制，实现一个 nuc 控制两个相机	2023.5.15
V1.8	实现单云台大 yaw 跟随，实现串口重映射保证通信正常	2023.5.20
V1.9	实现大小 yaw 配合，双云台集火，优化几何筛选，数字识别，减少误识别	2023.5.26
V2.0	将小地图数据传给导航，实现云台手和哨兵的交互，增加大津法调阈值提高代码鲁棒性，分区赛哨兵实现所有基本功能	2023.6.1
V2.1	完成双枪哨兵的控制部分代码，解决补偿突变问题，国赛新 YAW 轴出图，轮组优化方案出图，国赛第一版哨兵底盘出图	2023.7.2
V2.2	解决双头哨兵大 yaw 不稳定的原因，提高对平步识别率，国赛第二版哨兵底盘出图	2023.7.10
V2.3	完成双枪哨兵的预测，解决近距离识别问题，底盘加工装配结束，开始布线，针对上一个版本尺寸大、重量吃紧、弹道散步差、云台响应慢、主 yaw 转动失稳等问题进行改进，通过重新设计空间布局到更新机械结构，对整车的核心性能实现了优化重整	2023.7.15
V2.4	三代哨兵对应代码，测试环高与平地补偿，整车出车，实现基础功能	2023.7.22
V2.5	解决大小 yaw 的 bug，做双枪哨兵预测，喷漆，贴哑光贴纸，更换辊子，满足赛场需求	2023.7.28
V2.6	完善了云台手按键部分，尝试双头哨兵预测	2023.8.2
V2.7	做双头哨兵的预测，增加录视频功能	2023.8.4

1.6.3 重点问题解决记录

序号	问题描述	问题产生原因	问题解决方案&实际解决效果	机器人版本号或阶段	解决人员
1	弹道扩散大	预置使用的微动开关弹片，并且发射枪管采用 18mm 内径的铝管，没有很好的起到调整弹丸姿态的作用	采用 17.5mm 枪管，通过缩小弹丸可活动范围来降低散射程度. 静止状态下，6m 内可控制在三分之二小装甲范围内，由于 6m 外不考虑击打，故未进行更远距离的测试	V1.3	机械工程师:吴嘉琪 胡佩泽
2	底盘框架性能未达预期，线路裸露严重	最初底盘选择焊接工艺，由于焊接时产生热变形，导致底盘框架未能很好的契合图纸，导致其性能差强人意，并且由于底盘、云台同心度差的缘故，导致在运动和击打过程中，整个机器人明显晃动，使击打效果明显变差	重新对底盘进行设计，并加强对称性的要求，以达到其内部将部分影响整车稳定性的部分原因，内部解决，比如受力时对称结构和上下云台同心，可抵消大部分，达到整车表现出来的受力效果很小。其中新设计的机械结构以及专门为哨兵设计的分电板，在重新布线后，整车基本没有裸	V1.1	机械工程师:胡佩泽 吴嘉琪

序号	问题描述	问题产生原因	问题解决方案&实际解决效果	机器人版本号或阶段	解决人员
			露线路，整车电气控制线路更加稳定，精简		
3	机器人无法实现对近距离目标的打击。	滑槽的上下区间的限制导致云台俯角变低	改动 P 轴位置，从而改变下滑槽的长度和弧度，在不影响链路丝滑性的前提下加长了下滑槽的长度。改变 P 轴位置从而对重心进行重新计算和质量配比	V1.4	机械工程师:吴嘉琪 胡佩泽
4	俯仰角变化时候电机负担过大。	云台前后重量配比没有配好	改变 P 轴位置从而对重心进行重新计算和质量配比	V1.2	机械工程师:吴嘉琪 胡佩泽
5	拨弹盘在连续拨弹时会出现概率性卡弹现象	由于拨弹盘是使用 PLA 材料 3D 打印而成，强度底，大量弹丸堆积在分弹层时，挤压螺栓，导致拨弹盘内壁碎裂；同时拨弹盘堵时，软件加入反转程序，尝试解决，却因为分弹层坡度过高，静摩	分模块打印，整体依然使用 PLA，但在分层区前侧，内嵌小铝块；改缓分弹坡度，增加退弹坡道面积，使其静摩擦力远小于推力的分力；内壁增加小轴承，提高通过性	V1.2	机械工程师:胡佩泽 吴嘉琪

序号	问题描述	问题产生原因	问题解决方案&实际解决效果	机器人版本号或阶段	解决人员
		擦力大于轴承的推力分力，硬生生卡住并把坡道卡碎；拨弹盘中实际弹丸路径曲线和途中分析有一定偏差，外壁轴承安放位置，实际起到的作用有限			
6	整车装配完毕，距地高度与图中不符	全向轮减震计算与麦克纳姆轮计算并不相同，减震计算比例有误	根据相关资料，利用平行四连杆，重新计算减震位置，对减震器重新调节并安装	V1.1	机械工程师:胡佩泽 吴嘉琪
7	左右云台击打时弹丸无法命中装甲板中心	由于发射弹丸由于重力原因下落，在中远距离无法有效命中目标装甲板		V2.0	视觉软件工程师：景佳柱
8	左右云台识别到目标不稳定，丢失目标情况严重	研究通信协议，发现上位机与单片机通信过程中很多情况数据错位导致丢包情况，导致	修改通信协议，仿照裁判系统通信内容 弄个，将 V1.0 版本中简单的头尾帧通信改为 CRC 通信，极大的提	V2.1	嵌入式软件工程师：许佳奔 视觉软件工程师：景佳柱

序号	问题描述	问题产生原因	问题解决方案&实际解决效果	机器人版本号或阶段	解决人员
		云台识别有效识别目标概率大大降低	高了上位机，单片机通信过程中数据的有效性，大大提高了自瞄稳定性		
9	识别到目标时非常容易丢失目标	增加了预测算法后，刚刚识别到目标，云台距离目标装甲板距离过大，跟随过程中速度很快，预测时因为速度太大，在目标角度误差的基础上增加了期望角度导致云台 PID 积累， 直接将装甲板甩出范围。自瞄过程中极容易丢失目标	在刚识别目标时关闭预测，给与一个 0.2s 的时间误差， 云台与目标装 甲板基本重合后在开启运动，此时开启预测，有效避免了云台运动过快对预测数据产生干扰，极大的增强了预测情况下哨兵云台响应的稳定性	V2.2	嵌入式软件工程师:许佳奔 视觉软件工程师：景佳柱
10	Fastlio2 与 move_base 无法耦合	TFtree 出现问题，map 名称和 base_footprint 名称不对应	修改 fastlio2 的源码，使 TFtree 正确	V1.3	导航软件工程师：郝威程
11	避障功能不完善，有时候规划的全局路径会贴	膨胀半径设置不合适，参数修	增加膨胀半径，将局部避障算法	V2.3	导航软件工程师：郝威程

序号	问题描述	问题产生原因	问题解决方案&实际解决效果	机器人版本号或阶段	解决人员
	合障碍物，导致在真实场景下导航会导致剐蹭场地道具等	改粗糙，算法不够完善	TEB 改为 Astar 算法，规划路径有明显改善		
12	弹道散布较大	弹丸在在射出的过程中与枪管壁和测速壁发生碰撞，从而产生自旋，最终导致弹道散布大	通过车床对内径 17.5 外径 23 的枪管毛胚进行加工，使枪管变为前端内径 19 外径 21 后端内径 17.5 外径 23 的铝制阶梯圆管	V2.0	机械工程师：吴嘉琪
13	P 轴电机温度过高，难以维持额定扭矩	p 轴重心过于靠前	采用连杆机构进行传动，同时由于可利用空间不足，不能完全平衡重心，因此通过拉簧加以辅助	V2.0	机械工程师：吴嘉琪
14	云台系统卡弹严重	拨盘链路与云台链路转接口同轴度不高；拨弹盘机构表面平滑度不够，裸露在外的螺栓头部及螺母与弹丸间会发生卡死情况；拨盘链路中起润滑作用的轴承未起效果；更改云台俯仰角后，滑	对转接口打印件进行再设计，在原先定位结构基础上再设计两处榫卯定位结构；在原先拨盘的基础上进行再设计，通过合理使用沉头螺栓解决平整度问题；在链路打印件上切除出垫片结构，保证轴承有效运	V2.0	机械工程师：吴嘉琪

序号	问题描述	问题产生原因	问题解决方案&实际解决效果	机器人版本号或阶段	解决人员
		槽内曲线再设计不合理，试弹丸无法顺滑通过；yaw 轴链路出弹口内壁玻纤发生塑性形变，弹丸无法通过，发生卡死	作；通过建模仿真及实物测试，最终确定合适的滑槽内曲线；有限条件下，临时讲玻纤板磨出斜坡，卡死问题得以解决		
15	拨弹盘系统难以维修	拨盘链路直接与弹仓底板固定，使得每次维修都需要将底板与底盘分离，大大增加了维修周期与工作量	将螺栓从底板底面倒插入弹仓内，用螺母进行固定，继而使用两端攻丝铝柱，使拨弹盘系统形成快拆结构	V2.0	机械工程师：吴嘉琪
16	小陀螺时，底盘云台一起转	由于云台加上 750+弹丸后重量超过 6020 电机直连所能承受的负载，电机在超过 120°后，电机对角度的控制逐渐失灵	更换 yaw 轴的电机和传动方式	V1.1	机械工程师:胡佩泽
17	底盘框架及防护重量过高	防护强度过于冗余	除去上层防护铝管，更改底盘框架结构	V1.1	机械工程师:胡佩泽

序号	问题描述	问题产生原因	问题解决方案&实际解决效果	机器人版本号或阶段	解决人员
18	小陀螺时云台晃动大	餐盘轴承间的游隙大，会在小陀螺旋转时，被云台放大化，导致云台起伏大，影响发射和视觉识别，同时餐盘轴承材料并非轴承钢，后期磨损十分严重	更换交叉滚子轴承，重新设计轴系	V2.1	机械工程师:胡佩泽
19	防护不全面	前几代选用塑料围栏网，防护不全面	选用 2mm 玻纤板	V2.1	机械工程师:胡佩泽

1.7 团队成员贡献

姓名	基本信息 （专业、年级、队内角色）	主要负责工作内容描述	贡献度 （所有成员贡献度合计为 100%）
胡佩泽	机械设计制造及其自动化、大二、哨兵机器人机械	负责哨兵机器人底盘的设计、加工、装配和维修	15%
吴嘉琪	机械设计制造及其自动化、大二、哨兵机器人机械	负责哨兵机器人云台的设计、加工、装配和维修	15%
郝威程	机械电子工程、大三、哨兵组负责人、导航模块负责人	负责整车的进度以及团队的配合，负责哨兵自动导航部分的研发	15%
邢磊	机器人工程、大二、硬件成员	负责超级电容、主控板等研发	15%
许佳奔	工业设计、大二、哨兵控制、哨兵组负责人	负责哨兵控制部分	20%
喻文平	机械电子工程、大二、雷达站	负责雷达站的研发	5%
景佳柱	物联网工程、大二、哨兵算法	负责哨兵自瞄部分	15%

1.8 参考文献

[1] RM2021-北理工-中心供弹英雄机械开源

https://bbs.robomaster.com/forum.php?mod=viewthread&tid=11015&fromuid=72366

(出处: RoboMaster)

[2] RM2020 华南理工大学 华南虎 机械开源 下供弹步兵开源

https://bbs.robomaster.com/forum.php?mod=viewthread&tid=11030&fromuid=72366

(出处: RoboMaster)

[3] GitHub - RoboMaster/RoboRTS: An open source software stack for Real-Time Strategy research on mobile robots

[4] https://github.com/DoveJH/readme_source/blob/main/2020radar_station

[5] https://zhuanlan.zhihu.com/p/478630138

[6] https://github.com/ppogg/YOLOv5-Lite

[7] https://www.zhihu.com/column/c_1283402996664344576

1.9 技术方案复盘

回顾本赛季的技术方案，我想用我们的队训来概括一下，那就是：但行努力，莫问前程！从赛季初的方案制定到省赛一代双头哨兵的出车再到临近分区赛二代哨兵的实现以及最后国赛的三代哨兵，可以说是每一场“战役”都给了我们整个哨兵组努力的方向，也从不断的学习当中取长补短，不断优化。

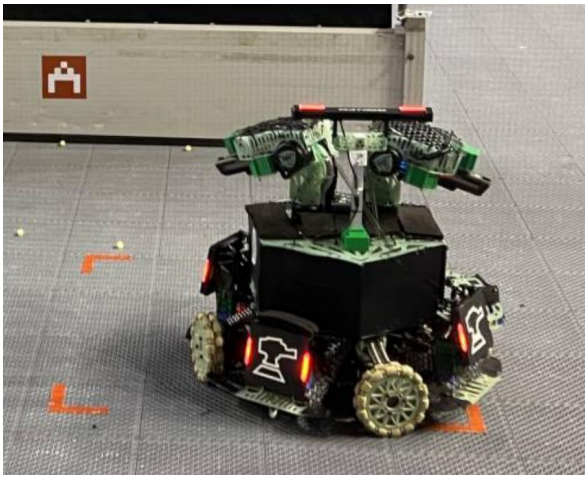
省赛前夕，一代双头哨兵才刚刚出车，时间的紧迫性导致无法让其达到既定的上场表现，以至于几局由于哨兵的问题而葬送，在此期间，为了保证哨兵不要主动死亡，我们抓紧赶制一个只能旋转的底盘来当做哨兵上场，正因如此，说明一代哨兵还是有着很大的问题，但在最后的表演赛上我们决定还是把它放在场上检验，也算是发挥了一下下了。

省赛回到实验室后我们对分区赛哨兵进行了无数次的讨论，甚至争吵，给我印象最深刻的就是底盘大 yaw 和云台弹仓部分的驱动方式。一代哨兵采用的是 GM6020 直驱，由于扭矩无法满足要求，导致在小陀螺的时候，会触发电机的过温保护，导致电机损坏和功能无法实现，因此在省赛后我们分别提出了三个拆了减速箱的 M3508 电机同时驱动大 yaw（当时已经能够实现）和带减速箱的 M3508 电机驱动小齿轮从而驱动大 yaw 这两种方案，在最后拍定方案的时候开了一个将近 4 个小时的会，22 赛季老队员也回来参与讨论，通过不断的比较以及理论分析，又与老师那边交流最终确定为两个方案的后者。通过不断的优化，完善，分区赛前哨兵算是勉强达到了我们既定的要求，由于时间安排的不合理还是有好多待完善的地方没有来得及进行完善。

分区赛依靠大家的通力配合，各兵种的不断发挥，成功弥补了 22 年败在 16 进 8 的遗憾，直接拿到了国赛的晋级资格。在分区赛后，深圳大学的哨兵在分区赛上的表现令我们无比佩服，而且由于结构相近，于是我们与深大进行了深入交流，通过与强队的交流中发现我们对细节的把控还是不足，于是在分区赛后，我们又开了一个个会议来对国赛哨兵进行讨论和完成方案的制定，通过对细节的不断优化以及经验总结，从而使得在国赛上哨兵是完成了我们既定的计划。

但行努力，莫问前程！制定目标，向着目标不断进发，无论前途是光明还是黑暗，是荆棘密布还是一马平川。极致者可敬，创新者无畏，回顾一年以来哨兵组的备赛过程，正是一个不断追求极致，不断追求创新的过程。没有反思和进步的人生不值一过，通过不断的交流讨论学习，不仅把比赛打好，更要把从比赛当中学习到的东西运用到生活当中，保持谦逊的态度，保持思考的能力，真正的做到：初心高于胜负！

1.9.1 赛场性能表现情况分析



上图是我们参加省赛山西站的一代哨兵，不管是防护还是两个小 yaw 云台的设计都比较粗糙，最让人难受的还是更换电池，当时设计了换电池方案是要把底盘抬起来，而且得几人配合才能够更换。一代车是按照最大尺寸设计，也导致哨兵的运输有一定困难。赛场的表现不尽人意，会出现超功率等现象，后来我们只能用一个自旋底盘来勉强上场。

上图是我们参加北部分区赛的二代哨兵赛场图。机械方面，重量的把控不到位，在检录时超重，我们临时把两个子云台的“头盖骨”拆下来，还拆了一些件才满足上场要求，防撞方面也没有对轮组进行防护，对大 yaw 的防护也仅仅是用扎带绑了四面网而已，控制和算法方面也有一些瑕疵，两个子云台的配合亦或者是导航的实现还有视觉的自瞄都出过或多或少的问题。可能大家的战术都不针对哨兵，所以发挥只能说是没有大问题。

上图是我们参加深圳国赛的三代哨兵赛场图，不管是从设计还是装配相较前两代都是精致了许多，而且针对前两代的问题也都进行了不断的完善，电子元器件的放置以及走线的合理性都在提高，但由于算法的鲁棒性不够高，导航功能没有在全国赛上完美呈现。在与深圳大学、哈尔滨工业大学（深圳）、上海交通大学的交手当中，明显的看出哨兵的反击能力还是不足，以及一些自主决策还是不够完善，面对深大的哨兵、上交的步兵，我们的哨兵还是有很大的提升空间。

1.9.2 赛场性能表现与规划对比分析

总体来说，从技术方案的制定到最后的落地大体还是比较合理的，但也有一些原因导致与规划出现了一定的差距。由于上半个赛季的疫情原因，导致我们备赛的时间大大缩短，许多方案没有经过严谨的推敲，许多机构没有经过足够的测试与完善，导致在出车后发现问题花费较多时间去更改，或者是在原有的基础上进行改善；控制与算法也是一大难题，哨兵今年是第一年下地，而且还是全自动运行，对于我们来说一些决策以及模拟赛场的状态和现象不足，再有就是理论方面的缺失，导致我们只能从调试现象出发来进行改进；导航今年是从零开始，还是有一些暗病的存在，在学校的场地中试验环境比较“安逸”，而在赛场比较“恶劣”。规划与落地产生差距还有一点就是对时间的把控不到位，一些重要的时间节点没有很好的利用起来，使一些本可以利用起来的时间浪费掉。

1.9.3 经验总结

偶然间在哔哩哔哩上看到了一个评论，就拿其作为最后的经验总结吧，不仅是对哨兵这个兵种，更是对整个队伍思想方面的统一，感谢整个哨兵组一个赛季的努力，初心高于胜负！

竞技体育

成绩是练出来的

冠军是打出来的

作为一个 RM 队员

比赛胜负的判定没有任何缓冲区域

决定成败可能是最后几秒钟的一念之差

就看谁能在逼近极限的同时

犯更少的错误

做出更加正确的决定

你发射的每一颗子弹

都没有机会再来一次

你场上或场下犯的每一个错

都有可能会断送整局比赛

甚至所有人整个赛季的努力

我们所爱的和别的运动不一样

你需要一个精诚合作的团队

需要一台好车

需要你的专业和冷静

需要你在比赛进行到千钧一发之际

钢铁洪流一般地向敌人冲锋

怎么战胜对手

那就是做好每一个技术点

焊好每一块板子

写对每一句代码

找到装备性能和战术执行能力的边界

抓住无数个　一克　一毫米　一焦耳

把你眼前的每一个小问题都解决好

这不是简单的对战比拼

这是全方位全流程的生死较量

六大兵种

上百个技术细节

三百六十多个日夜

耍小聪明

你能侥幸赢得了一秒

但你无法赢下七分钟

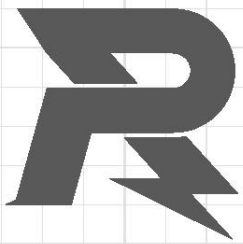
你问我绝招

绝招只有两个字就是

奉献

就是把你的全部

奉献给你热爱的一切



邮箱：robomaster@dji.com
论坛：http://bbs.robomaster.com
官网：http://www.robomaster.com
电话：0755-36383255（周一至周五10:30-19:30）
地址：广东省深圳市南山区西丽街道仙茶路与兴科路交叉口大疆天空之城T2 22F