电控技术报告

[图1 通信流程图（虚线部分为待实现/方案待更改） 1](#_Toc28595)

[1.1 步兵赛季总结 2](#_Toc30484)

[1.2 步兵规划 2](#_Toc5234)

[2.1 哨兵赛季总结 3](#_Toc5964)

[2.2 哨兵规划 4](#_Toc23953)

[3.1 平衡步兵技术报告（zjh） 4](#_Toc16184)

[4.1 硬件组规划 5](#_Toc23057)

[4.2分电板相关经验（pdx） 6](#_Toc29321)

[5.1 代码管理相关 8](#_Toc13515)

[6.1 一些小tips 9](#_Toc14070)

##### 以后每一届都在这个文档里面按照格式更新下赛季进度和下个赛季规划，不仅能起到记录进度的作用，也是一种经验的传承

##### 588e67945dbf70ba0ff425593e04463

图1 通信流程图（虚线部分为待实现/方案待更改）

底盘与底盘电机之间的通讯依靠can1

底盘与电管（裁判系统）之间的通讯依靠huart3

底盘与云台之间的通讯依靠can2

云台与拨轮、摩擦轮之间的通讯依靠can1

云台与小电脑通讯依靠huart1，与遥控器通讯依靠huart3

舵机的控制依靠htim1的PWM波控制

##### 步兵赛季总结

------------------2024------------------------

步兵现在存在的问题有（按重要性排列）：

1. **底盘限功率**的代码写得不太好，现在采用的方案是，根据裁判系统给的最大上限功率来对应一个底盘速度，这个方案有几个缺点：1.底盘的状态并不是恒定的，除了普通的移动外，还有小陀螺，移动小陀螺，旋转等，采用这个方案的话就得对每个状态进行速度对应，非常繁琐 2.采用该方案拟合出来的速度不一定适合，可能会存在浪费功率的情况
2. **步兵的自瞄**现在用的是三串级pid，实际测试发现虽然稳定（不抖动），但响应太慢，抗干扰能力也不行，所以还是得换成二串级pid，可参考哨兵代码

3.**超级电容功率控制板**如果要再产的话记得让硬件那边的人把can接口的HL与我们常用的can线线序对应一下。如果尝试一段时间不好弄的话，发送功率值的功能就用几个IO口来发送给功率控制板（让硬件的打板记得预留），UI显示就用几个led放置在图传前显示。

（lzf）

没能与功率控制板通信稳定通信，所以就不能给控制板发送当前等级的最大允许功率，目前就是干脆给功率控制板定死一个最低等级的功率，通信不了的原因可能是因为硬件问题导致的can通信不稳定，也可能是因为代码的什么原因，但据说哨兵上测试是稳定的，所以应该是代码问题.

4.**底盘裁判系统UI显示**的任务如果不执行或者执行一两遍就没掉了的话记得加大堆栈空间以及提高任务优先级。目前是一次性只能绘制两个及以下的图形才能推送得到服务器，可能是代码的推送函数的推送速度太快导致的。并且UI还存在在实验室本地显示，但在赛场上不显示的问题，可以对照平衡步兵那份的UI推送检查一下是什么问题。尝试过自动获取ID与写死ID，但在赛场上都没效果(lzf)

Ui界面，步兵的界面在赛场上出不来，但是平衡的可以，因为平衡这个工程是其他学校开源的，可以看看与步兵的存在哪些区别，或者看看论坛上别人有没有这样的情况，如果能解决是最好，毕竟操作手有很多信息是很难自己获取的，有了界面可以让操作手获取更多的信息，特别是超级电容的容量

5.步兵的代码**存在一个隐患**，那就是重新使用cubemx生成代码后，会导致代码在步兵上跑不了，这是因为往界把代码写在了begin end以外的地方，这就导致更新完这段重要的代码就没了，所以这个问题最好解决，不然会成为以后的绊脚石，解决方法：使用github的代码对比功能，对比生成后和生成前的代码，把这些代码写到begin end里面

###### 步兵规划

------------------2024------------------------

1. 限功率的方法要换一个，可采用

<https://www.robomaster.com/zh-CN/resource/pages/activities/1013> 里介绍的方法二，实时计算底盘功率然后给四个电机的速度设定值同时乘一个因数，同时放大/缩小速度倍数才不会导致运动混乱，之前想的思路是同时加减速度，但是会导致运动混乱，之前如果早看到这篇文章就不用搞这么麻烦了（侧面论证多看论坛的重要性）

1. 自瞄调pid
2. 功率控制板与a板的通讯要完善，功率控制板与a板的can通信如果还是不行，就换成串口通信，正好控制板上也留了一个串口，见图一
3. ui界面不推送的问题

。

4.新赛季要对步兵进行迭代升级，可能是换成全向轮，全向轮/舵轮开源代码，可作为参考 https://bbs.robomaster.com/forum.php?mod=viewthread&tid=17629

2.1 哨兵赛季总结

------------------2024------------------------

S2 s1(一档) s1(二档) s1(三档) /

一档 无力 导航模式 导航+小陀螺/

二档 普通跟随 运动小陀螺 云台扫描+小陀螺/

三档 同上+自动射击 同上+自动射击 同上+自动射击/

1. 哨兵现在最大的问题是，通讯流程太过于复杂，由于小电脑在云台，底盘的里程计数据要发给小电脑，是先发给云台c板，再发给小电脑，小电脑的导航数据要发给底盘进行运动也是要经过云台，这种通讯方法会导致1.延迟，会导致巡航的延迟性 2.can2总线上数据太多，影响云台电机与c板通讯或者影响底盘遥控器数据的更新；所以现在打开导航挡位，云台是控制不了的，因为云台电机数据接收乱掉了，但是底盘可以靠导航发过来的数据操作底盘运动

所以，必须简化通讯过程，个人建议：把雷达换到云台上面，然后底盘到云台就能空出两根滑环线，可以拿来当串口线，作为底盘和小电脑的双向通讯线（图一）；而且，把雷达从底盘换到云台，也有很大的好处，**导航发过来的数据格式是前进速度还有旋转速度，哨兵代码里是把这个旋转速度进行累加得到旋转角度，再把这个前进速度分解，就能得到x,y轴的速度了**，这样就可以边开小陀螺边导航了（强校都是这样的），当然，关于开小陀螺时里程计怎么计算，个人认为，可以拿小电脑算出来的x,y轴速度来计算里程计

1. 哨兵的射弹需要配合当前热量和最大热量限制去做相应的限制，2024赛季在这一块的处理仅仅是打一段时间，停一段时间，结果因为没实际热量测试，上场后直接超热量死亡了（没办法，在赛场上写出来的代码），一旦超热量了还在发弹，掉血是非常快的，基本宣判死亡

。之所以采用这种方法来进行热量限制，是因为哨兵底盘的裁判系统与a板之前通讯存在延迟（步兵不会），导致这个热量数据迟迟不更新，而这个要进行热量控制，数据更新必须是实时性的，所以不解决这个延迟的问题，这个热量限制估计进行不了，之所以会存在延迟，可能软件方面哪里写得不好，导致串口频率比较低，可以参考步兵的串口通讯

如果解决了这个通讯延迟的问题的话，这个实时热量数据就可以用来坐发弹限制了，最简单的方案：采用合适的射频（拨盘速度环控制），如果热量快到上限了，就使用舵机切枪管或者停止射弹，当然舵机切枪管是最好的不浪费热量的方式，因为两个枪管的热量是分开计算的，**哨兵云台代码里有舵机切枪管的逻辑**，但是当时由于舵机坏了+时间紧，没有过多测试，只知道给舵机一个信号，它能来回切到正确的位置；代码方面，可以使用switch语句做状态机来跳转状态（见个人代码风格）。

如果实在解决不了通信的问题，那还是用回打一段时间，停一段时间的方法吧，只不过用这个方法的话，射频要先确定好（可以根据自瞄距离来切射频，太远射频就小点，射频大，打的时间就短，射频调速度环的设定值），再来标定攻击时间和冷却时间

1. 哨兵的自瞄算是调好了，但是打不准，两个轴都存在差值，这个差值可能是定制，也可能随着距离而改变，又或者在某个距离内改变，某个距离外为定值

2.2 哨兵规划

------------------2024------------------------

2.2.1 控制组规划

1.把底盘和小电脑的can2通信换成串口通信

2.自瞄补偿

3.枪口热量限制

2.2.2 导航组规划

1.目前定位只靠电机里程计进行定位，如果存在打滑的情况，定位将产生偏差，所以最好是融合一下imu来提高定位准确性

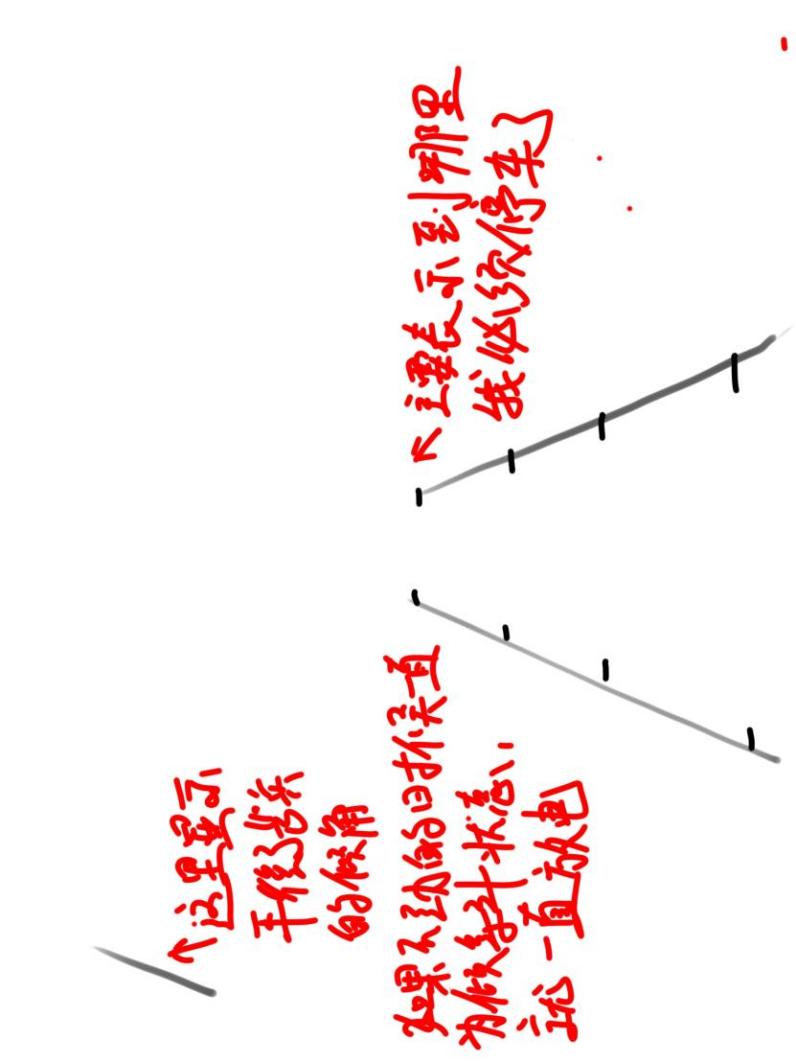
<https://mp.weixin.qq.com/s/CUjqrUIP1WGsGaPi5ezzrw>

<https://mp.weixin.qq.com/s/7w-6S2Udb-heXhSFWXF_PQ>

1. 如果使用双雷达来扫描，便能解决单雷达的死角问题，但是双雷达的话，要把两个雷达的点云数据融合在一起才能在一张地图上面进行规划，避障
2. 在视觉看不见的地方，如果能对以及确定是敌方的机甲的位置进行跟踪，那么便能提前做出相应的反应
   1. 平衡步兵技术报告（zjh）

------------------2024------------------------

平衡步兵现在最要紧的是换功率控制板，然后跟功率控制板通讯上，然后用灯珠和用ui就看你们了，应该挺简单的，ui界面要增加三条线



这是最重要的

第二重要的是底盘质心问题得解决，最好保护壳宽度加长，稍微把装甲板嵌进去点，能过检录就行

平衡代码要是有什么不懂的可以问我

还有就是**平衡底盘有全向轮步兵要用到的解算，像梯度控制，全向轮解算**

平衡步兵云台代码挺纯净的可以拿那个去改新步兵云台代码，看你们

新步兵控制框架你们看看用我们还是用南航的

* 1. 平衡步兵规划

------------------2024------------------------

等哪一届经费充足了可以把普通底盘换成轮腿，机动性更强，更加稳定，需要再加4个海泰04作为关节电机，一个3000块，可以先做出网上开源的小轮腿，学一学算法控制，有了经验才好上手

1. 目前的话平衡就是调一调参数，看看能不能让控制更加稳定一些，打滑检测的阈值这些要调一调，地上铺满子弹来模拟赛场情况
2. 功率控制板通讯
3. Ui界面完善
   1. 硬件组规划

------------------2024------------------------

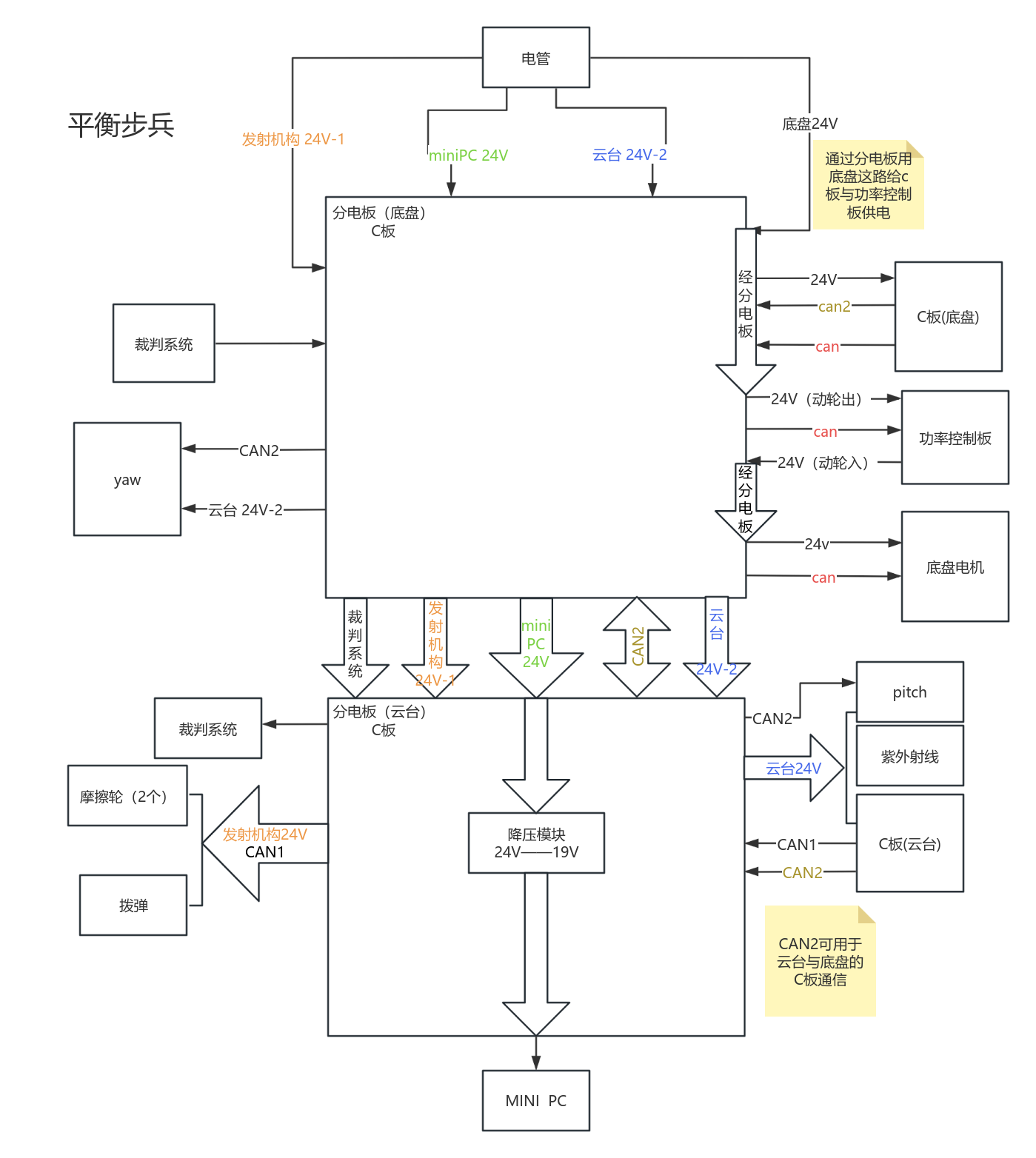
1. 无线烧录模块的生产工作，争取多生产一点给控制那边调车用

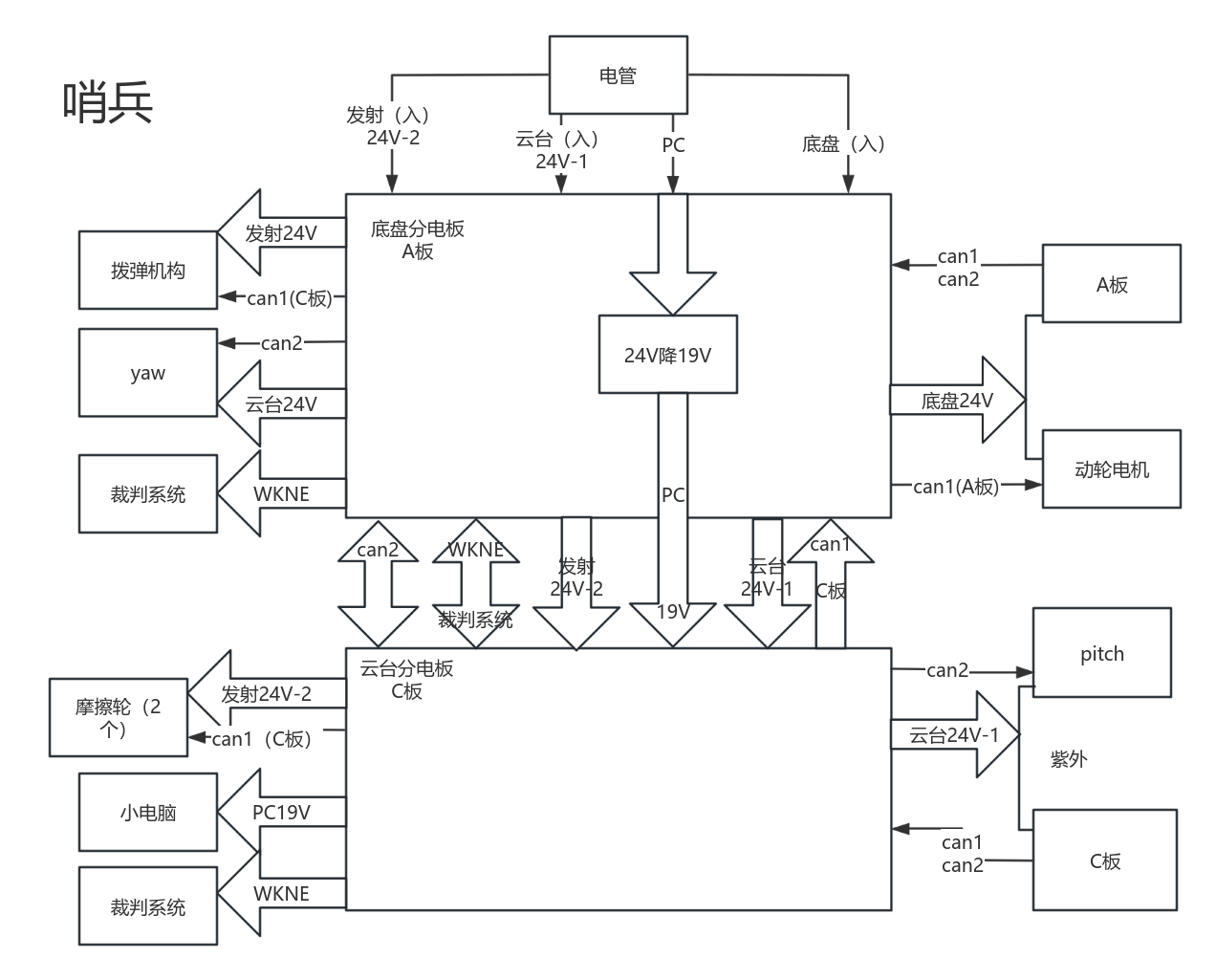
[高速无线DAP调试器Lite - 嘉立创EDA开源硬件平台 (oshwhub.com)](https://oshwhub.com/ylj2000/dap_hs_esp_open)



1. 到时候新英雄还有新步兵需要画分电板，争取面积小一点，丝印少错点
2. 多焊点功率控制板
3. 硬件组的活比较少，所以大家可以兼职一下控制那边的活，要主动去学一学代码，等待组长安排进控制组

4.2分电板相关经验（pdx）





Can1用于控制各系统

Can2用于云台与底盘的主控板的通讯

哨兵分电板有关pitch和yaw的丝印有误，但不影响使用，要注意区分。（别问为什么有误，问就是在初期时设计的，不熟练，哎嘿）

在画板前一定要与机械组的同学好好沟通，讨论好各接口的位置所在，不然就会像平衡和哨兵的接线一样冗余，不稳定。集电滑环的线长和粗细一定要适合，平衡的线过细又短，结果用了24根线还进行了延长，连接不稳定。哨兵的线7根一簇，加上线粗，冗余一团而且不好调整。

Can线接口占面积小，可以多预留几个，以防万一。

动轮（电机）指机甲的轮胎电机，初期设计的命名不规范。自己设计时可以重新命名。

降压模块因为是用于小电脑，用电表简单测压后发现电压达15V就没管了，到末期裁判系统到才启用，上电后发现不能用，但因为时间原因没法修改，所以就直接外接成品，造成不必要的冗余。自己设计降压模块时一定要及时上电启动测试，免生事端。

5.1 代码管理相关

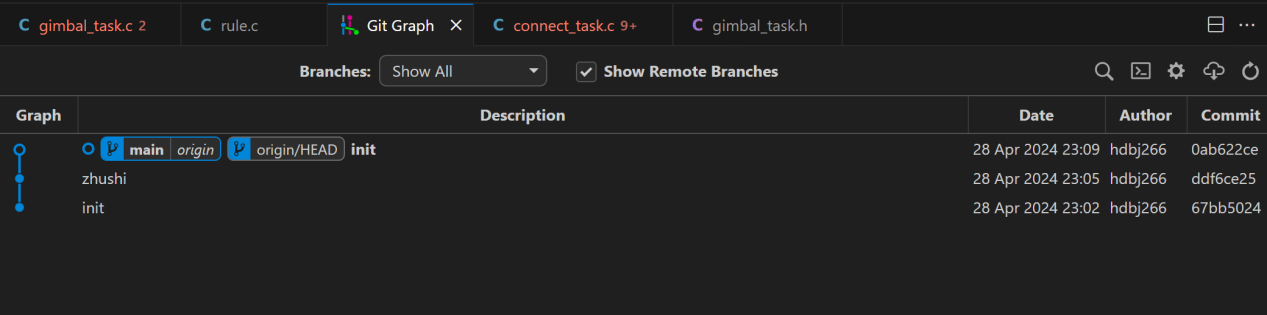
使用vscode安装git进行代码管理是一种很方便，很实用的代码管理方式，可以使用vscode进行代码管理（做标记，代码回溯，建立分支，换分支，上传代码，克隆工程），而写代码和调试就在keil里。（相恨见晚，导致电脑里有几十个工程）

关于安装，注册那些东西，可以看这篇文章

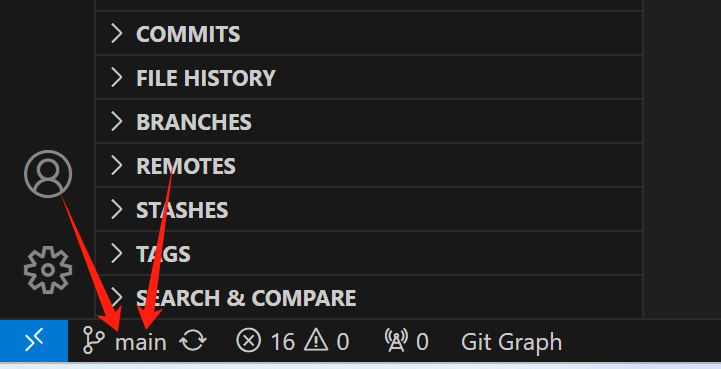
[vscode如何连接github/gitee远程仓库详细步骤（ssh+https）最全最详细\_vscode github-CSDN博客](https://blog.csdn.net/Bin_niB/article/details/135915738)

我认为最实用的地方如下：

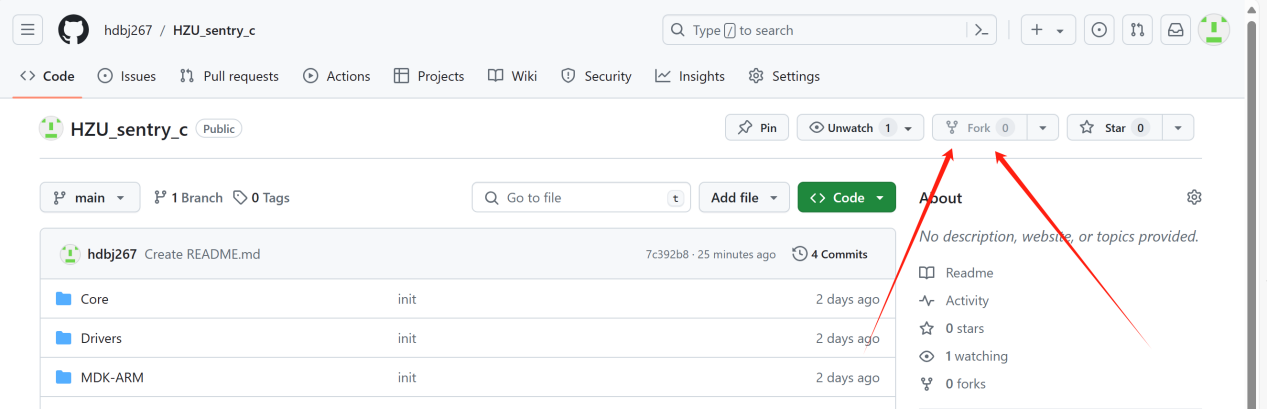
1. 改完你的工程后，可以使用commit功能，给你的修改打上备注，同时在vscode安装插件git graph,可以看每一次commit打的备注和修改的内容以及可以进行代码回溯，下图为git graph插件



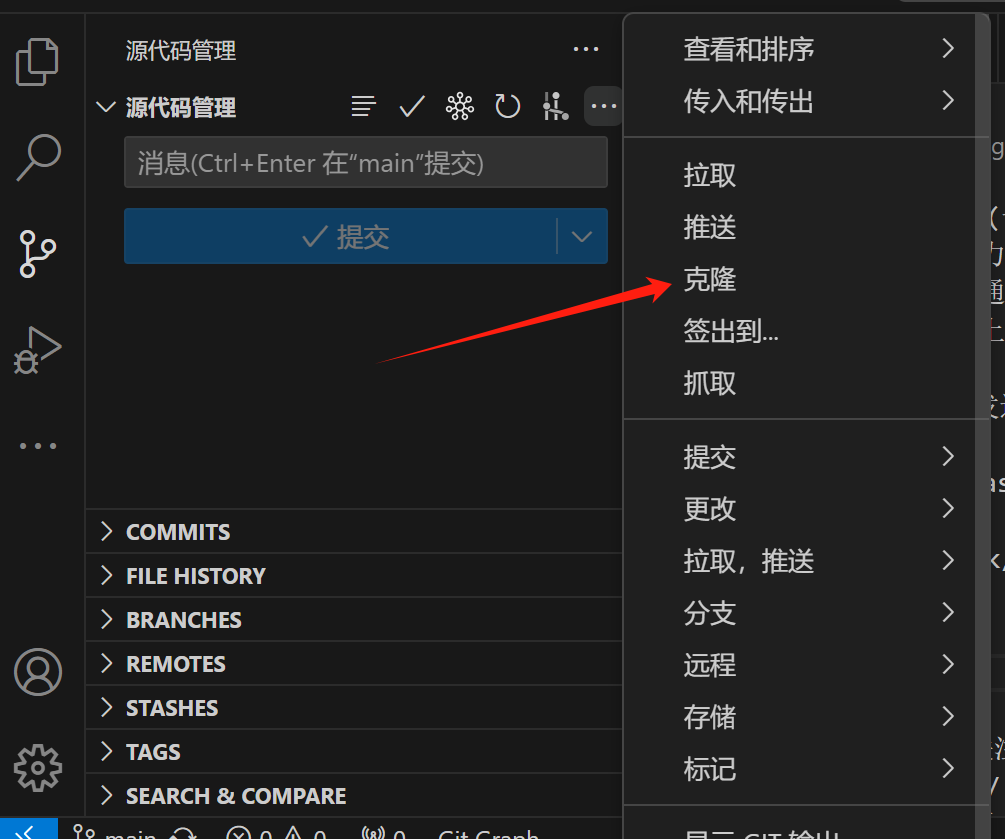
1. 使用工程分支（比如我上传的步兵底盘就用了分支），可以很轻松的复制你的工程，这个功能可以用于你想进行代码测试但又不想修改原工程的情况（超实用的好吧），而且新的分支还在同一个工程文件夹里面，但是两个分支不会起冲突，只需要点一下左下角就可以换分支，同时，换完分支后，你的本地代码就被修改了，就可以用keil进行另一个分支的调试



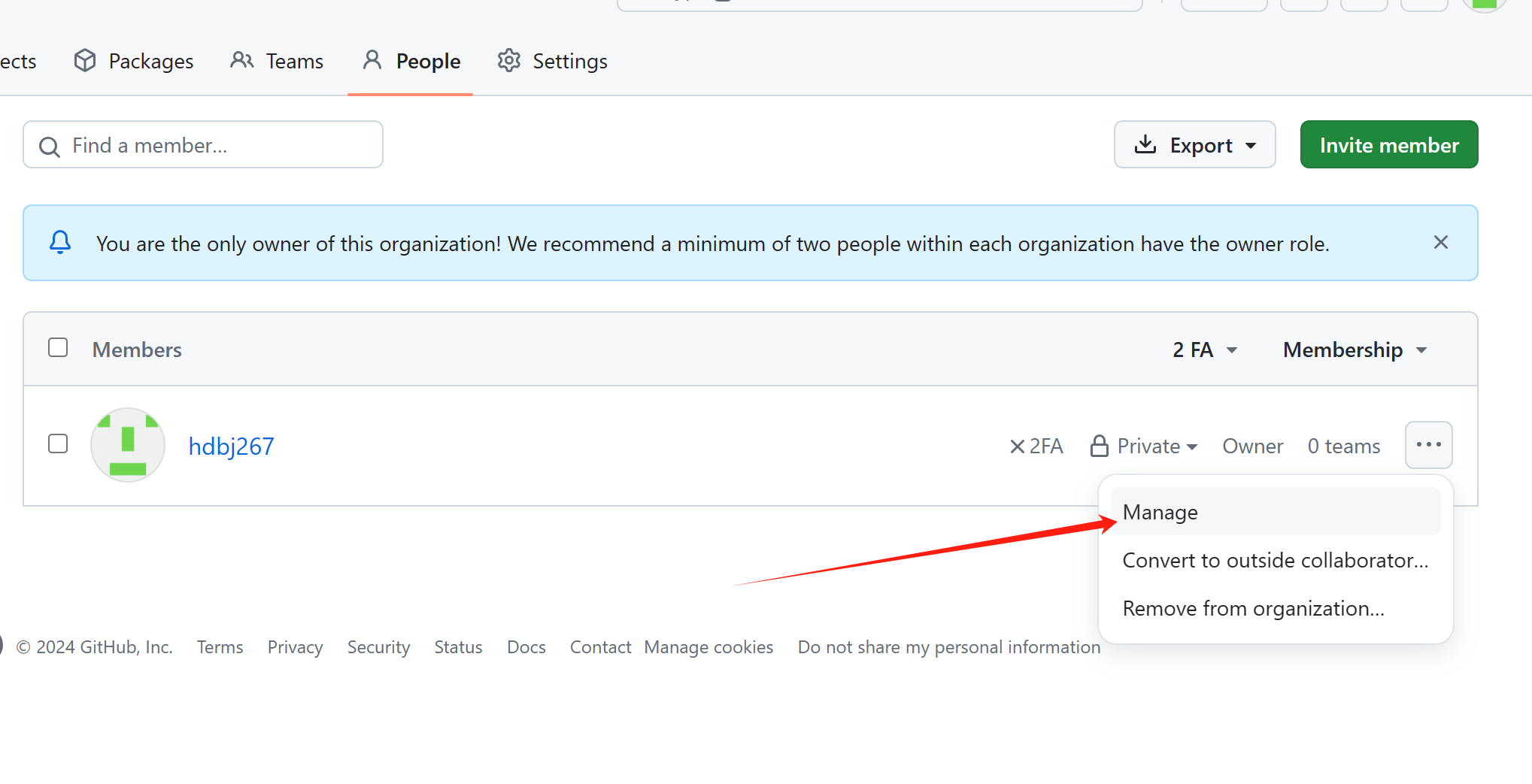
1. fork功能，将别人仓库的工程复制到你的仓库，然后就可以在vscode使用克隆功能将你的仓库工程下载下来，这样子，其他功能都可以基于你的仓库工程来进行，就不需要提交和推送代码到别人仓库了



1. 你永远不需要担心你找不到你的工程放在哪里了，找不到本地文件夹了，只需在你的仓库里面重新克隆一份就好了



5.使用团队模式创建仓库，给团队里的人代码上传的仓库权限，同时将团队里所有人都设置为推送审核人，这样一来，一旦有人想将代码合并到仓库，就需要其他人的审核，也能让大家都互相了解工程进度



6.1 一些小tips

------------------2024------------------------

1.keil的watch窗口观察变量，他的数据刷新频率是有上限的，如果需要看频率更新比较高的数据，需要使用波形观察软件，这里推荐es-dap-view，刚好可以搭配无线dap使用

下载地址：[PC软件-上海东软载波微电子有限公司 (essemi.com)](https://www.essemi.com/index/article/plist?cid=114)

1. keil的watch窗口数据如果不更新了，打开debug-view里最下面那个勾上就可以了；要删除watch窗口的变量，必须在代码未运行时才能删；watch窗口里面的变量可以赋值，但是一旦退出debug，赋值将失效
2. 调自瞄时，最好不要把底盘的电拔了，虽然说会有一定危险，但是加不加底盘跟随会对云台控制来说是有区别的，可以先把底盘电拔了，等调试稳定了，正式开始调pid了，再把电插回去（或者拔can1信号线也行）
3. c板的电源很脆弱，一定不要带电拔插swd下载口！！！，而且不是把24v电源线拔了就行的，因为串口那边还给c板供着电呢，所以那根下载线尽量不要去拔它，要拔也是关电拔
4. 测试自瞄命中时，可以把弹速拉到28左右（这一届是最高30，下一届不知道），因为弹速大，子弹飞行时间就可忽略不计，弹道下坠就小，还有预测也不用考虑子弹飞行时间
5. 实验室是有一套裁判系统可以用的（以前买的），电源管理-装甲板-枪管，有这三样其实以及够了，平时可以利用它们进行测试底盘功率，枪口热量等，枪口热量可以连上服务器，然后看操作手界面中心那个圈，很直观，如果看底盘裁判系统结构体变量可能会有延迟！！

7.1 一些错误的研究方向

------------------2024------------------------

1.过于关注性能指标，错误的，如果基本功能都没完善，就不需要关注相应的指标提升方面

2.采用不切实际的理论，错误的，在实际的工程应用中，效果才是王道，不要去分析太多的理论，就比如小陀螺走不直的问题，不应该去考虑什么科氏力导致走偏，可能这是答案，但不是解决工程问题的答案，直接采用暴力补偿法就能直接看到效果了，往哪边偏就给一点相应的反方向的速度用来抵消就好了

3.想的太多，做的太少，可能会导致变成，理论上的大师，实操上的矮子