

**关于我们**

我们是IRON MAPLE #19581，来自深圳南山（中加）学校。我们此次参加比赛的团队是由13名校内机器人社团的成员，外加一名来自深圳南山外籍人员子女学校的伙伴，以及十分靠谱且备受我们尊敬的一位导师组成。

我们曾参加过2023年重庆FTC®️ CENTERSTAGE正式赛，因此部分队员在赛事本身以及机器方面有一些经验。我们在过去的几个月内一直致力于为了此次比赛而致力于优化机器本身，让机器以及赛事伴随着我们的成长不断进步。同样，我们也致力于通过行动及努力，向更多的人推广FIRST®️，并以自己的方式为FTC做出贡献。

此前的比赛中，我们团队总有一些沟通不协调，角色不明确，工作任务冲突以及部件管理混乱的问题。在比赛结束后经过完善的总结，我们痛定思痛。所以在参加此次季后赛前，我们便制定了一个完整的框架，明确了团队内每个成员的角色和职责，并就灵活调配的方法指定了指引。

**ダイアグラム

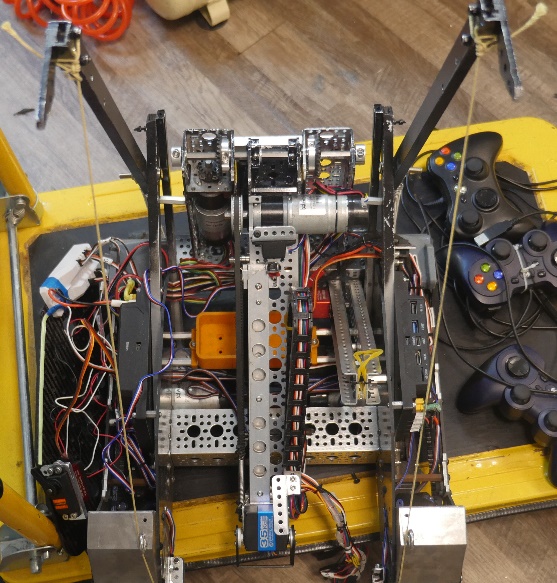
自動的に生成された説明グラフィカル ユーザー インターフェイス

自動的に生成された説明最终机器完整图片**

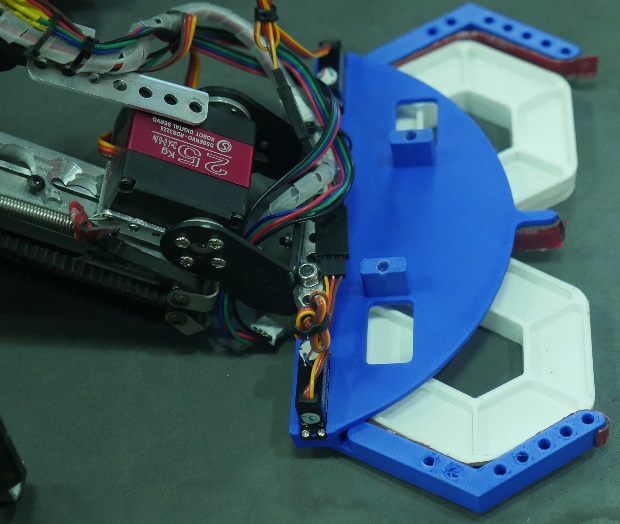
**硬件部分**

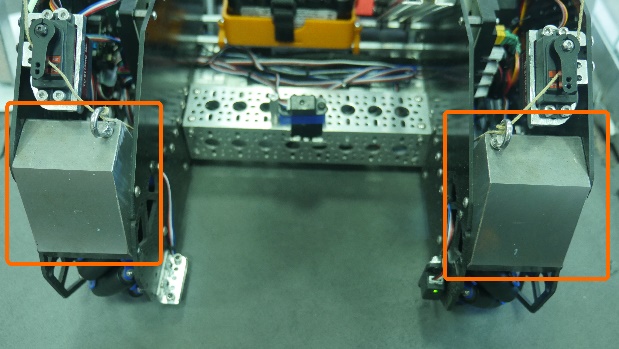
**底盘**

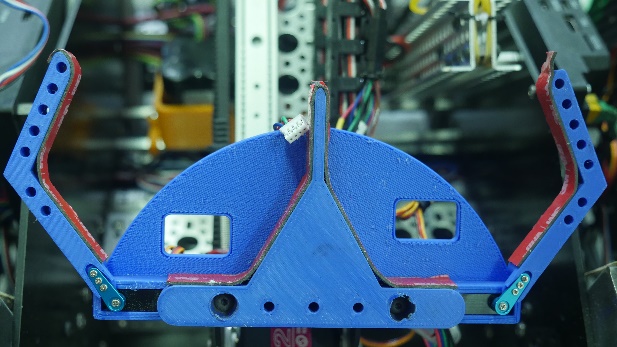
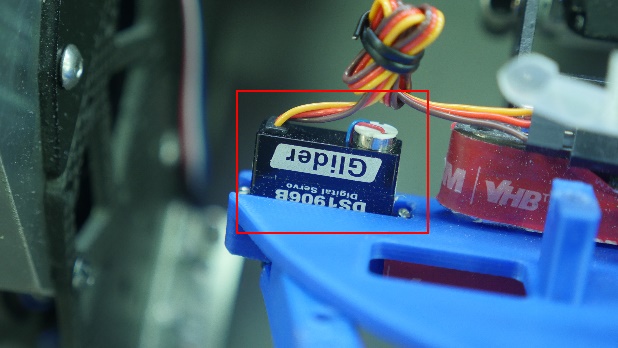
屋内, いっぱい, テーブル, 座る が含まれている画像

自動的に生成された説明我们的底盘沿用了此前正式赛的设计，采用麦克纳姆轮驱动，最大的优点就是灵活，可以向任何方向做平移运动。但此设计的一项痛点就是平移的时候会有不定的阻力导致平移不够精准，所以我们加入了独立编码器来为机器提供位置导航，三个从动轮都被放在了机器底部并加以弹簧提供压力使轮子始终着地。这些附件负责感应机器的移动并为控制程序提供精准定位。

**INTAKE夹取结构、爪子和配重**

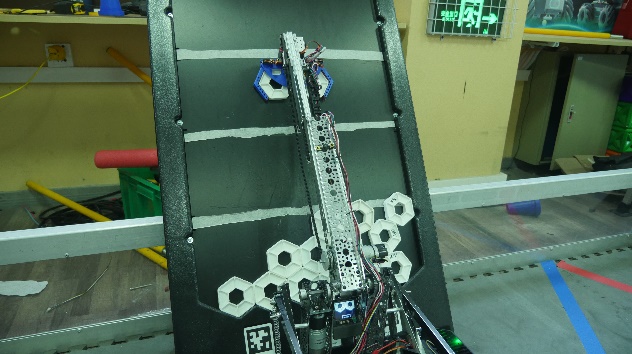
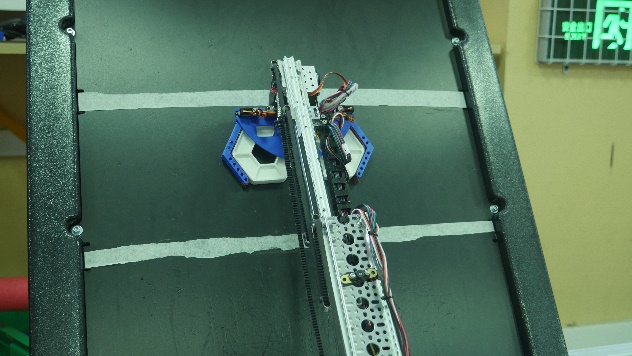
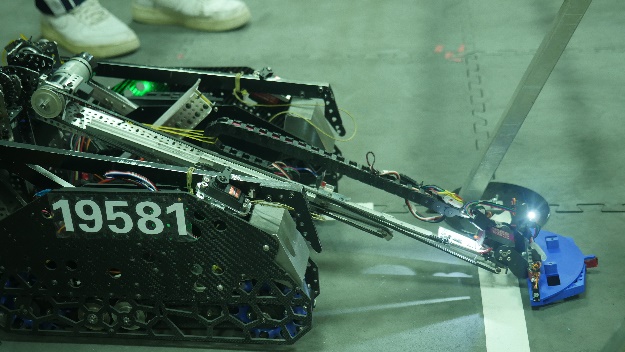
我们的机器在第一代中使用了皮带的intake，但我们在比赛过程中发现这种设计有很大的缺陷，所以在这几个月中我们对我们的机器进行了一次大改。

我们将intake的皮带改为机械臂，本次我们的结构在减轻成本的同时减轻了大部分重量。但在机器减重后，我们的机器在放置pixel时，由于机械臂位置发生改变，重心也随之改变，再放置完pixel后不可急加速，所以我们在车尾部增加了配重，这使我们机器的重心更低的同时抓地力更足，搭配我们的外置encoder使得机器在移动中更高速更精准。

由于我们的机械臂需要快速反转，所以我们的爪子进行了轻量化设计，我们从第一版使用的标准舵机改为小型类9g高压舵机，这个舵机能在我们抓取物较轻的情况下达到和标准舵机一样的抓取效果，并将重量从69g×2减轻至9g水 が含まれている画像

自動的に生成された説明×2。这让我们机器的机械臂在pid的控制下能够更快的达到目标位置并减少了机械臂在展开的情况下的头部的负重

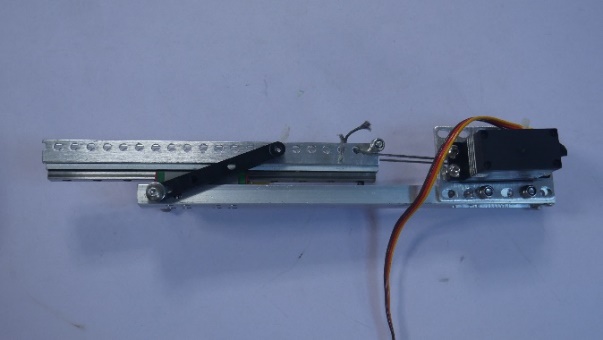
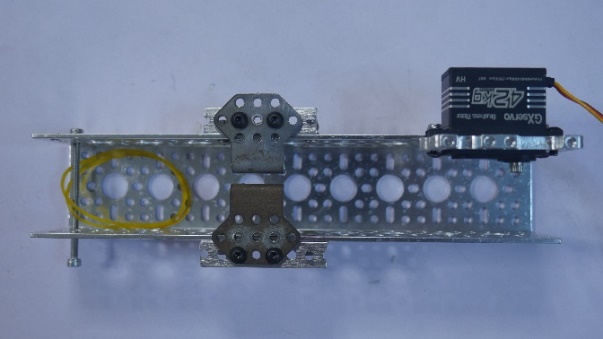
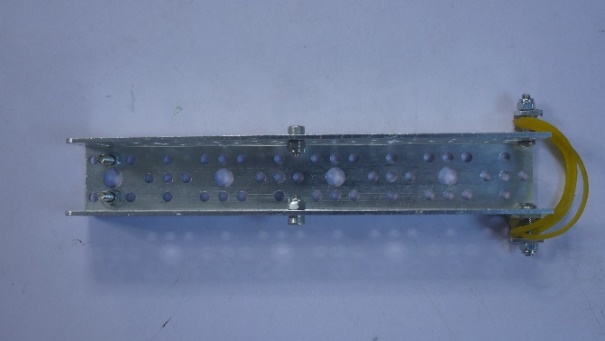
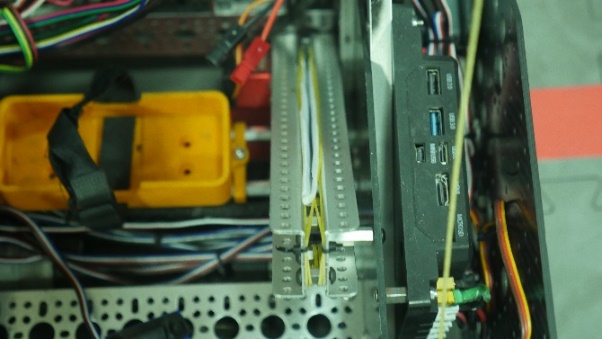
**机械臂**

在日常训练中，操作手发现了由于pixel在背景板上不断叠高而导致无法得分的问题。因此我们在机械臂上新增了伸缩结构，让我们的机器在放置低位置pixel时更加精准，不像第一版时由于要考虑到高位pixel所以手臂无法收缩调整高度至低位的问题。现在在pixel叠高后我们的机器也可以轻松得分。

**ENDGAME爬升结构**

Endgame时的悬挂我们也进行了全新的设计。机器此次的结构在节省机器内部空间的情况下同时也减轻了机器重量，并减少两个电机端口的占用。让我们得以在机械臂上使用两个舵机来翻转和伸缩。

**无人机**

我们第一代的无人机在去年的正式赛中使用皮筋与3d打印结合金属结构来推动无人机。但由于建模的问题，无人机经常固定不牢或者坠落。在后续的结构改造中，发现在空管结构下，皮筋传动效果实际上具有优势。因此，我们打算在原有的基础上进行改造。在过程中，我们发现如果皮筋尾部的力充分给到了无人机，弹射会变得快且稳定。后续在经历两代的改造后，我们最终在第三代的基础上做出了最终版本。最终版本在皮筋结构以及无人机本身厚度、轮廓上进行了最佳的改进。目前已能保持一致的弹道的同时，还能保证在遥远射程中的稳定性。

最终版本

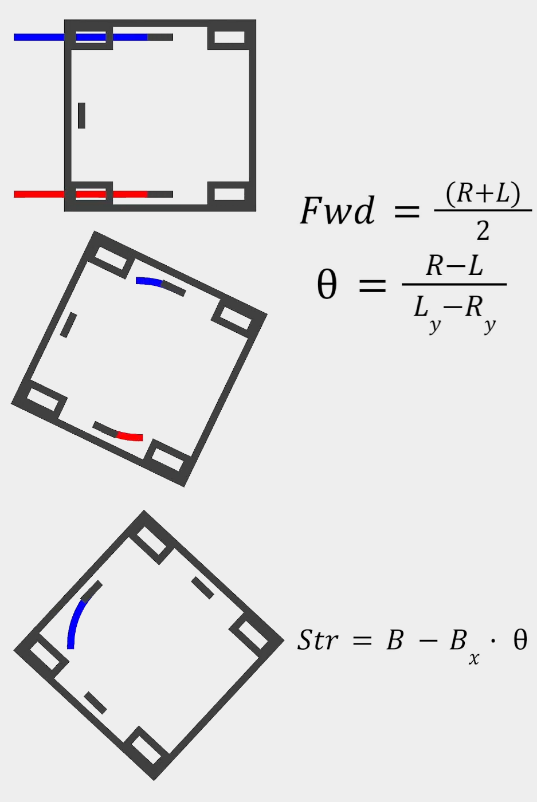
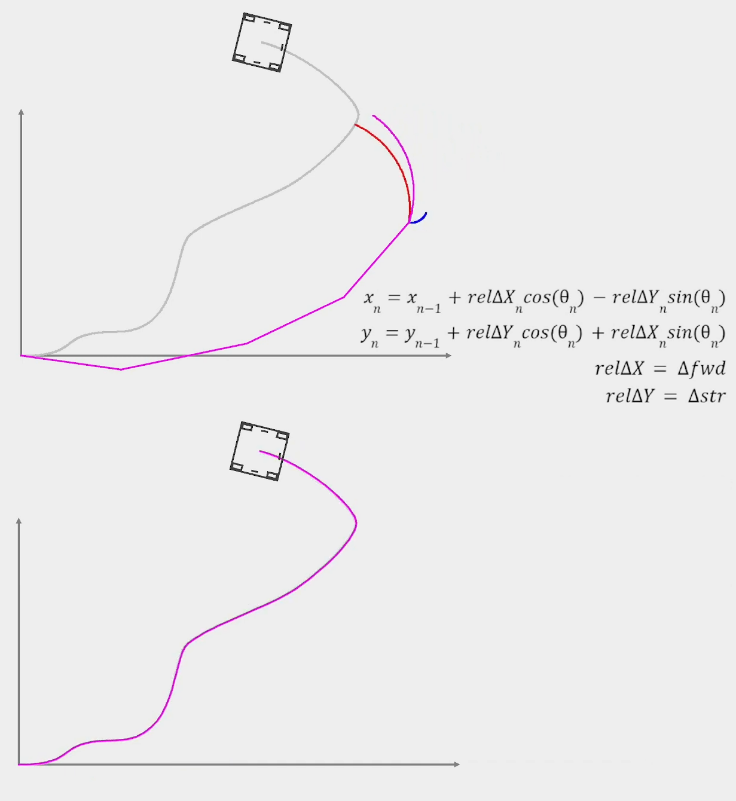
第三代

第一代

第二代

**软件部分**

**从动轮导航系统**

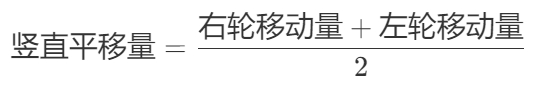
基本原理

我们使用了从动轮来感知机的运动，计算机器的位置、速度、加速度等

每个从动轮上都配有编码器，可以读取从动轮转了多少圈，以此推算出从动轮的位移

但是，从动轮的位移并不一定就是机器的位移，机器的旋转也会让从动轮转动，这就会造成误差，因此我们设计了一下的解决方案：

1. 计算竖直方向位移：使用对称、平行排列的竖直从动轮求平均值消除机器旋转带来的误差

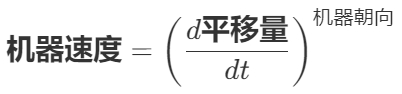


1. 计算旋转：左右从动轮相减得出旋转量

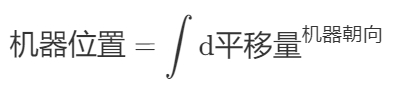
旋转量=(左轮移动量−右轮移动量)×比例k

接下来，进行微分/积分就可以得到机器的位置或速度

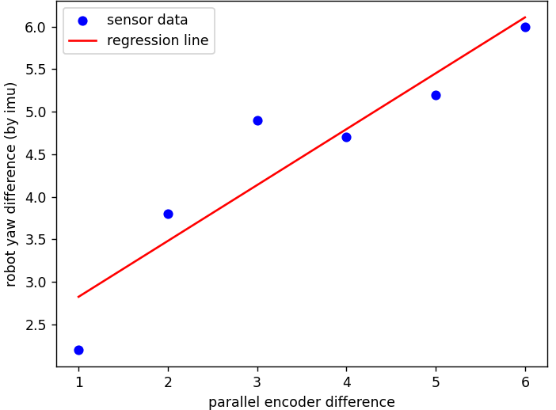
1. 计算速度：对机器平移进行微分



1. 计算平移量：对机器位移进行路径积分



校准方式

绝大多数队伍在使用从动轮时，需要知道三个轮子之间的确切相对位置，然而，机器在制作时存在公差，并不能保证精确。

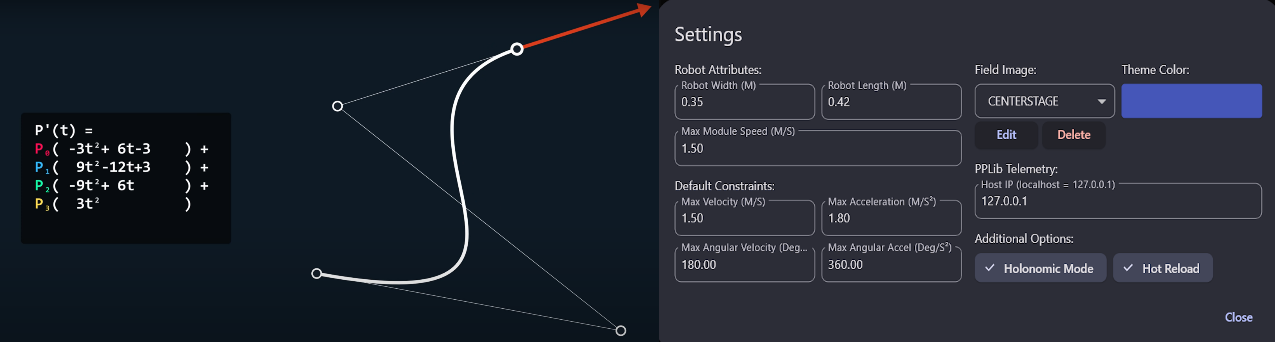
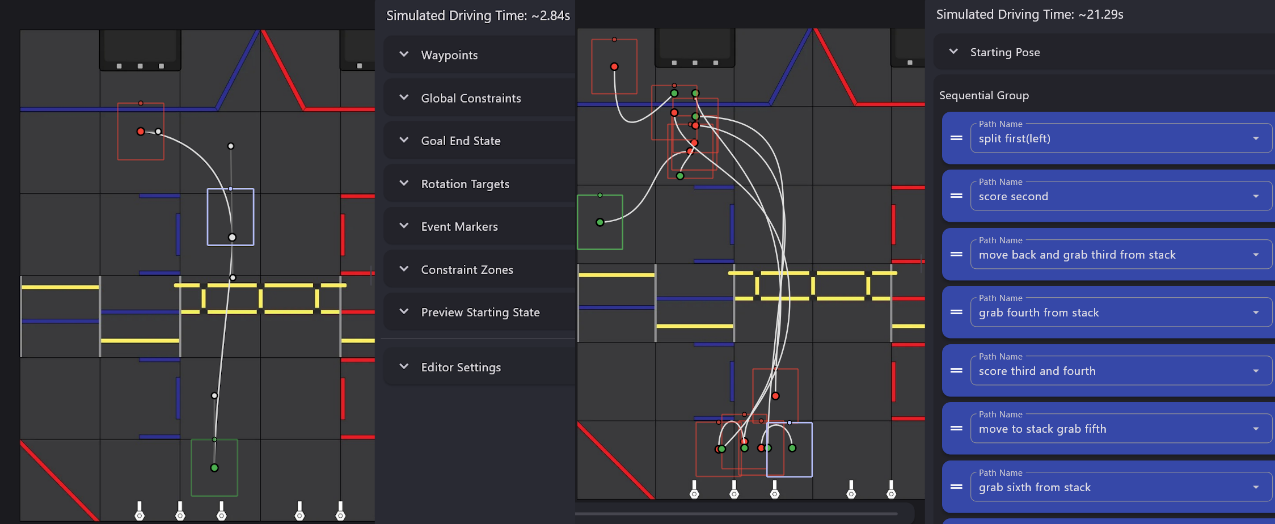
所以，我们写了一套校准程序，直接测量机器的读数，并画出回归线，以此测算出正确的参数

**自动阶段路线行走**

为了使得机器可以在自动阶段精准地完成指令，我们魔改了FRC队伍3015的开源工具Path Planner，对比传统的自动阶段编程，使用pathplanner进行路线设定有以下好处：

路径由贝塞尔曲线生成，保证机器移动丝滑流畅，且可以通过微调控制点精准控制走线轨迹

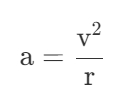
内置物理引擎，即使没有训练场也可以模拟出机器走线过程，甚至可以计算出确切的完成时间。

用户友好gsui界面，初学者也可以快速编辑自动

制定出精确的路径后，我们使用了一套创新的pid算法来控制机器，使其可以精准的跟随路径，具体原理见下图:

ダイアグラム

自動的に生成された説明



另外，我们的机器在跟随路径时，加入了速度曲线来使过程更加流畅。

**自动阶段自主决策系统**

绝大多数的机器在自动阶段只会执行预设指令，除了一开始的识别导航之外，其余一切动作都是按照设定的路线执行。

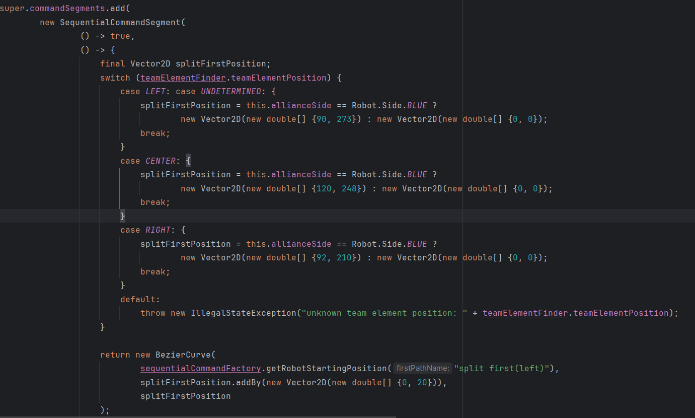
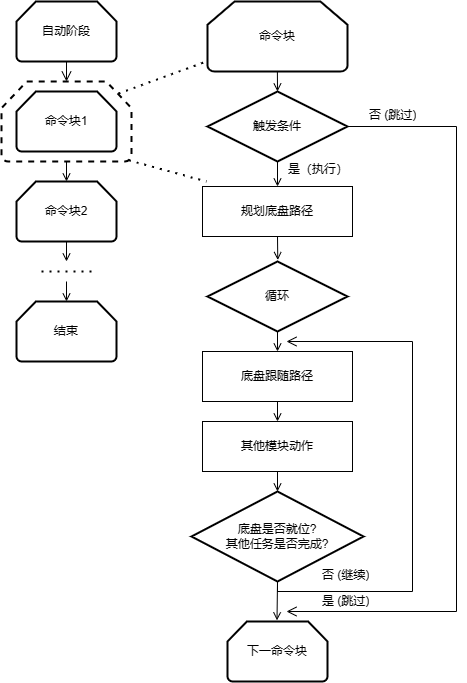
而我们在自动程序架构上作了创新，我们的自动程序由若干个命令快组成，每个命令快包含**触发条件、 路径生成函数、终止条件与上层结构指令**构成，详细如图1:



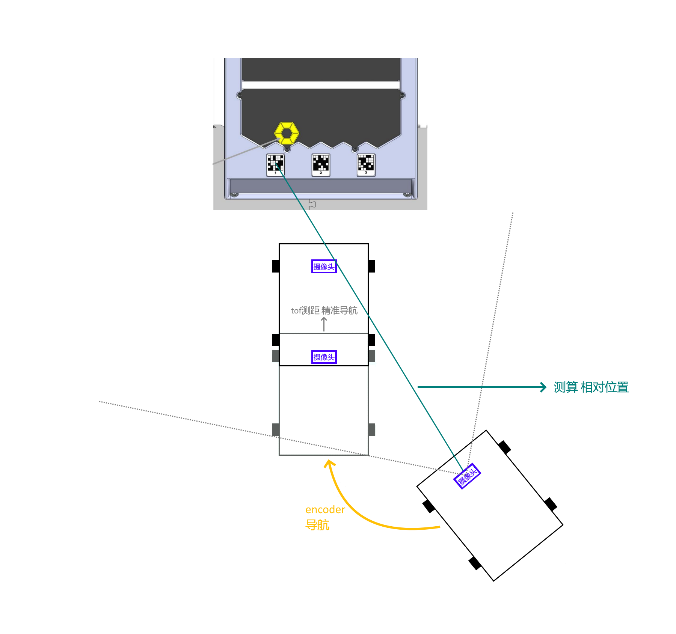
图3(下)（xia1）

图2(上)

图1

例如，自动阶段导航（放置紫色像素）我们是通过图2及图3中的程序实现的。除了基础的导航任务，用这套程序架构我们还可以做出很多功能，例如，当我们夹取一个像素堆失败后，机器不会浪费时间去背景板放置，而是会直接夹取下一个像素堆。

**自动瞄准系统**

在手动阶段，驾驶员自行控制机器接近目标并放置pixel需要很多时间反复微调，还可能出现失误，操作不慎撞上目标。于是，我们设计了一套继集编码器、智能摄像头和TOF距离传感器三种感知方式为一体的自动瞄准系统，实现自主瞄准功能。

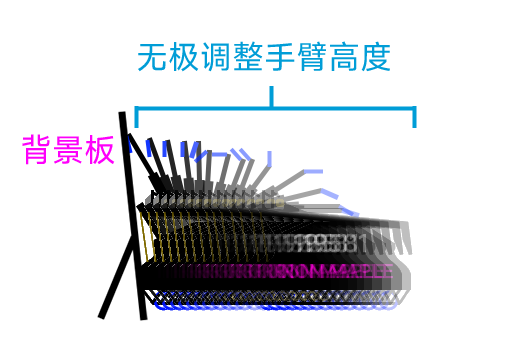
**第一阶段: 自动接近** 当驾驶员按下手柄上的自动瞄准按键，机器会首先通过智能摄像头计算出目标的大概位置。但是，智能摄像头存在延迟、帧率较低、容易丢失目标等问题。所以，这期间底盘会用从动轮编码器感应自身位置，用PID算法移动到目标前面，同时正对目标。

**第二阶段：自动贴合** 当机器到达目标面前，tof距离传感器启动，感知目标距离；而智能摄像头则负责感应目标的水平偏差。机器会在0.7秒内完成自动贴近目标

**第三阶段: 微调位置** 当机器完全贴近目标，驾驶员通过手柄选择具体瞄准位置。程序会自动保持与板子的距离（精度 <= 1cm）并调整机器水平位置直到完成瞄准。

目前为止我们能够实现三个阶段总共用时**不到4秒**

**新功能：手臂高度无级调整** 当机器已经完全贴合背板后，手臂可以根据操作手的指令无级调整像素放置高度。操作手只需决定防止高度，程序会自动计算出对应的底盘距离、手臂角度、伸展长度与舵机位置，并实时进行调整.

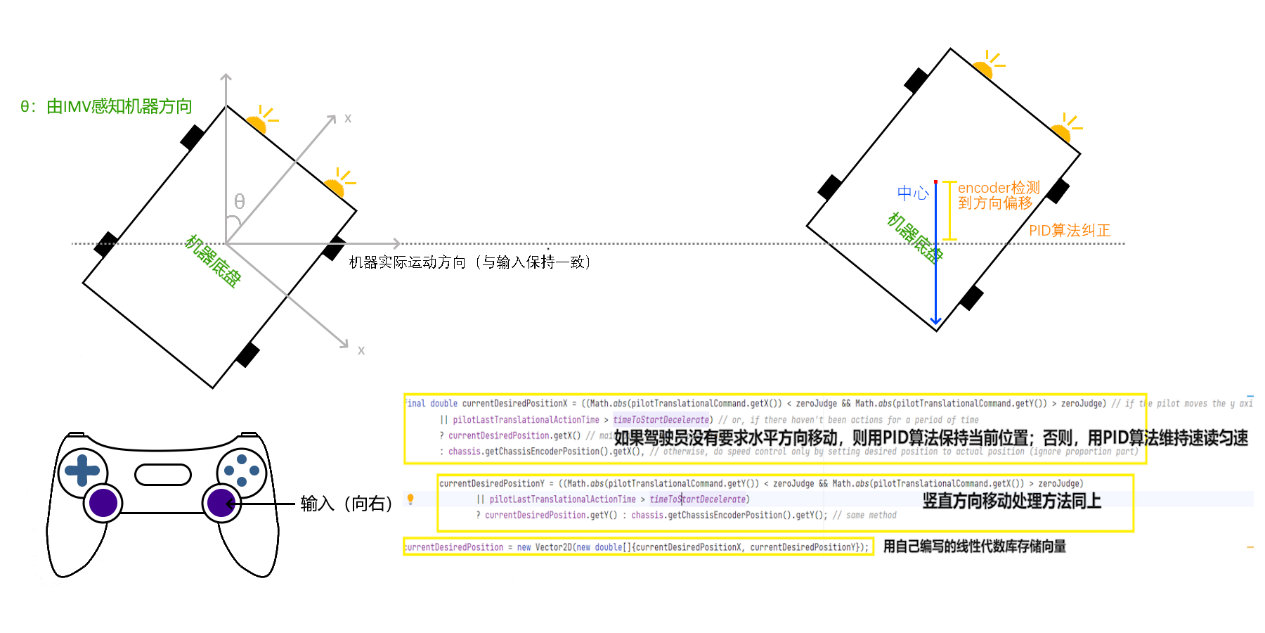


**手动阶段精准控制**

有了从动轮提供的导航信息，我们建立了一套兼顾灵活、精准、易用的控制系统。这套系统的核心是动态轨迹纠正系统

当驾驶员输入移动命令后，系统首先根据自身IMU获取的方向对驾驶员输入进行变换，无论机器朝向，机器的移动方向永远和驾驶员参照系的方向一致。这个功能简称“无头模式”

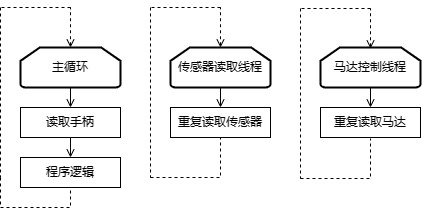
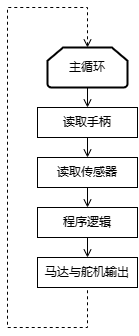
而且，机器在运动途中，从动轮会实时监测机器的实际运动，如果与驾驶员的输入存在误差，会用PID算法进行校正，使机器走一条直线



**程序优化问题**

一般来说，FTC机器的程序以单线程运行，机器依次读取传感器、处理程序逻辑、对电机进行控制。

但是当机器有太多功能时，这样做会导致程序延迟过高，于是我们将程序分成三个线程执行，如下所示：

优化前： 优化后：

主线程以100次每秒读取手柄并处理程序逻辑；马达控制线程以100次每秒控制电机；传感器通讯线程受到rev控制板自身硬件性能限制，只能以40次每秒读取传感器

在这种情况下，我们成功将操作延迟从50ms降低至10ms，并将PID的更新速率从20次每秒提升至40次每秒，大大提升了机器的流畅度