INFINITE RECHARGE℠ at Home Guides
本地无限充能指导手册

FIRST® GAME CHANGERS℠ powered by *Star Wars: Force for Change*

2021 *FIRST*® Robotics Competition

FIRST ROBOTICS COMPETITION

FIRSTINSPIRES.ORG/ROBOTICS/FRC

CONTENTS 目录

INTRODUCTION 介绍

作为场地上唯一的机器人完成技能挑战对 FRC 队伍来说是一种全新的挑战。为了帮助队伍考虑他们如何以现有的机器人来练习和提高这些技能，FIRST 已经把各种指导整合在了一起，队伍可以从中寻求指导。这些指导完全是可选的，不需要逐步完成。这些指导的运用不是评审过程的一部分。虽然这些指导是围绕 2021 年技能竞赛挑战的应对而设计的，但队伍可以考虑如何开发和完善类似的活动并纳入未来的赛季中.

BREAKING DOWN THE AUTONAV CHALLENGE
细分自动导航挑战

自动导航挑战鼓励队伍探索复杂的自动导航程序。为了协助第一次接触此类挑战的队伍，本指导将对如何尝试自动导航做一个高度概括。

Tracking Robot Position 跟踪机器人定位

在确定采取哪种方法和如何进行之前，我们需要回顾对这两种方法至关重要的基本概念：确定和跟踪机器人的位置.

跟踪定位机器人的方式列举如下:

1. **Time 时间** - 测量机器人位置最简单的方法是在固定的时间内执行动作。传感器是不需要的，所以它适用于任何机器人设计；然而，时间实际上并不能测量与运动中发生的所相关的任何东西。这意味着实际的运动量可能会根据电池电压、车轮滑动或其他机器人碰撞导致的摩擦变化而变化（这些变花将导致机器人移动得比预期的更快或更慢，无法匀速运行）
2. **Sensors 传感器 -** 为了获得比定时运动更精确的性能，队伍可以使用传感器进行测量. 这些传感器 sensors 分为两个大类:
    - **Internal sensors 内部传感器** - 这些传感器测量机器人或机器人部件的运动，但不使用来自机器人外部的参考资料。 与时间测量相比，这些传感器消除了许多误差来源，但它们还是容易受到诸如车轮滑移或传感器漂移之类的影响。两种主要类型的内部传感器，队伍用于跟踪机器人的位置是:
        - **Encoders 编码器** - 测量轴的旋转（比如，通过扩展轴，车轮）。编码器可以跟踪直线位置，有一些可跟踪转弯 (许多机器人的驱动方式在转弯时会经历车轮打滑而降低编码器的可靠性).
        - **Gyroscopes 陀螺仪** - 跟踪角度位置（更具体地说，它们跟踪角速度，集成了返回角度位置的功能）。许多陀螺仪也有内置了加速度计 accelerometers（把一个或多个陀螺仪和加速度计集成在一个单元中的时候，我们通常称其为惯性测量单元或 IMU），它可以通过加速度计的双重积分的测量方式来跟踪位置，但双重积分会放大信号噪声，通常来说这种测量因有太多噪声而不适用于 FRC。

      许多队伍发现内部传感器应对一般导航足够可靠，但当需要精度时会转用外部传感器，通常用于得分项目.
    - **External sensors 外部传感器** - 外部传感器直接测量机器人周围的世界。结合预存的有关领域的知识，它们可以用来确定机器人的位置。外部传感器包含摄像头 cameras 和距离测量的传感器比如激光或红外线测距仪 Laser rangefinders (LIDAR), IR Rangefinders, 声波传感器 Ultrasonic sensors. 当队伍经常对于场地上的定位需要在一个精确的位置和/或方向的时候，会使用外部传感器，比如在得分项目上，尽管他们不必局限于这种应用上.

Choose an Approach 选择方法

两种常见的自动阶段机器人导航的方法:

1. **Individual Movement Approach 个体移动方法** - 把整个移动过程细分至更小的部分（比如直行，原地旋转，偶尔单弧度转弯）开发代码来完成每个单独的部分（理想情况下，在代码中重复利用的一些距离或角度的参数），然后将这些部分串在一起成为一个完整的例程.
2. **Path Planning Approach 路径规划方法** - 使用路径规划来产生一条丝滑的路径，分成一部或多部分，每个部分包含多弧度转弯或直线前进的动作.

Individual Movement Approach 个体移动方法

个人移动方法通常从开发个体代码例程开始，以驱动所需的距离（向前和向后），并转向所需的角度(同时左转和右转)。 然后，对于每个所需的路径，可以将其分解为这三个基本构建块。在编写这些构建块时，通常使用两种控制方法:

1. **Bang-Bang Control 起停式控制** – 起停式控制有两种状态，通常是开（虽然这不需要速度达到“全速”）和关。 只需将电机按指定值打开，并定期检查是否有满足结束条件。 一旦达到结束条件，停止电机。这种方法通常会导致目标在用于位置/角度时的显著过冲

***技术提示**: 请记住，时间和命令模板，包含 LabVIEW TeleopVI，已经包含了围绕 XXPeriodic 函数的循环。使用者不需要再加入长时间的运行循环在这些函数中，可以认为 XXPeriodic 是一种循环的迭代.*

---

2. **PID Control PID 控制** – PID 控制根据误差（以及潜在的误差积累和误差变化率）动态设置输出。在一个非常基本的层次上，你可以把它想象成一辆汽车接近停车场。一般情况下，它会逐渐减速，所以当它越靠近目标时就会移动得越慢。

Path Planning Approach 路径规划方法

这种方法通常是通过调整机器人上的控制回路（通常是速度控制）来使其能够遵循任意路径。 然后，对于你想行驶的每一条路，你都细分成一个个“路径点”，你希望机器人通过路径点并串联起来生成一条完整的路径

***技术提示**: 详细的 WPILib 工具帮助可以在 WPILib 文档中的轨迹生成和后续部分 Trajectory Generation and Following section 中找到。一步步的教程可以在轨迹教程 Trajectory Tutorial 里找到. LabVIEW 使用者可以使用这个库 this library 实现类似功能.*

Putting it Together 整合起来

一旦你决定了你采用的方法，并查看了你尝试行驶的道路，必须把这些构件都放在一起。这样的过程怎么做取决于你的机器人代码采用了什么语言和框架:

- **LabVIEW or Timed Robot Auto Init** – 组装一个自动例程的最基本方法是将每个构件作为一个方法或 VI，其中包含一个循环，并依次组装/调用它们。在 LavVIEW 中，这可以通过使用序列结构 Sequence Structure 来完成。在 C++或 Java 中，它可以通过按顺序从 AutoInit()方法调用构件来完成。虽然这种方法

是最简单的，但它可能有显著的缺点(至少在 C++\Java 中)。在 C++\Java 中构造这样的代码会使机器人在行驶时向代码中添加其他函数变得更加困难；因为你的代码正试图在一个循环中运行尝试完成当前的“构件”例程，它不能同时做其他事情。LabVIEW 有点不同，因为 LabVIEW 的代码本质上是并行的

- **LabVIEW or Timed Robot State Machine –** 如果你希望你的代码在运行其他行为的时候更灵活，请考虑状态机“state machine”。可以在因特网上的许多地方了解到状态机是什么。出于编程一个简单的 FRC 自动阶段程序的目的，它们通常包括以下内容:
    - **A state variable 1 个状态变量 –** 这个变量持续追踪当前状态.
    - **Conditional/Branched code 条件/分支代码 –** 基于状态变量的 AutoPeriodic()方法中的“switch”语句（或 LabVIEW 中的案例结构 case structure）或一系列“if”语句，该状态变量描述了在每个状态下要做什么.
        - 每个分支内部的代码通常应该执行一个操作（例如将电机设置为某种速度），然后测试是否达到了移动到新状态的标准。在简单的自动阶段的状态机里，流通常只是向前移动（当达到目标距离或角度时前进）。在更复杂的状态机下，流可以跳来跳去，而不一定是线性处理.
        - 使每个分支成为一个“构件”，设置电机速度，并检查目标是否已经达到，然后再进入下一个状态，从框架使用整体循环围绕 AutoPeriodic()方法来编写，而不是编写自己的循环
    - **Command-based framework 基于命令的框架** – 在基于命令的框架中，框架为你构造了一种状态机。每个构件都是一个命令，其中 Init()和/或 Execute()方法命令电机和 IsFinished()方法评估目标是否已达到。当使用更高级的控制时，你可能最终会使用诸如 PIDCommand 或 RamseteCommand 之类的来为你处理一些逻辑上的东西。 为了把构件组装成一个例程，可以使用 CommandGroups.把简单自动程序转换成基于命令的程序 Converting a Simple Autonomous Program to Command-Based 中使用旧的基于命令的库，但是，所描述的原则应该适用于“新”或“旧”基于命令的框架（尽管某些语法可能有所不同）.

TARGETING WITH VISION 视觉目标

关于使用 FRC 视觉系统寻找目标，有大量的文件可供查阅，但对如何使用该视觉返回的信息的描述不多。本指南旨在提供一个概述，让你知道应该如何运用视觉解决方案中获取的信息，并使用它来控制你的机器人.

Step 1: Decide Goals & Plan Approach 制定目的和计划方法

回顾问题，并确定你试图让软件做什么。这些是你可能会问的一些问题，希望对你有所帮助:

- 目标是否如预期般的在视觉系统的框架内吗？比如操控手和基于路径的自动导航让你靠近吗？或者如果没有找到目标，代码需要知道该怎么办？如果代码必须找到目标，这是用传动方式完成的？或者是否有一个炮塔或摄像头伺服机会被用到？
- 机器人需要知道到目标的距离才能执行所需的功能吗？ 如果是的话，那是来自视觉系统还是其他地方？ 在某些情况下，你可能希望知道举例，例如计算发射速度/角度或将机器人靠近目标。在其他情况下，距离可以通过其他方式处理（例如。 没关系，操控手会解决的。或者是由其他距离传感处理的）
- 机器人是否需要定向以特定的方式来达到目的（例如，它可能无法以锐角射击，或者它可能需要接近垂直来精确放置一个比赛道具）？如果是这样，捕捉达到极限的细节。代码应该如何处理机器人方向上超出这些限制的情况？

Step 2: Find the Target 找到目标

下面列出了三种常见的视觉系统方法:

1. **“Traditional computer vision” 传统计算机视觉** – 这包括使用 NI Vision 或 OpenCV 等库的非机器学习方法。 关于这些方法的信息可在 Vision Processing section of the WPILib documentation 中找到.
2. **Machine Learning 机器学习** – 这种方法通过使用标记过的目标的示例图像教会软件所寻找的目标是个什么样子。 关于这一方法的更多信息可在 Machine Learning tutorial in the WPILib documentation 该文件中找到.
3. **Off-the-shelf solutions 现成的解决方案** – 这种方法使用打包的现成解决方案来找到目标。这些解决方案通常可以允许一些用户做出一些微调，但这和尝试从零开始编写代码有很不同的感受。例子包含有 Chameleon Vision, Limelight, Opensight, 和 PhotonVision/Gloworm 等解决方案。请注意，其中一些解决方案需要特定的硬件，而有些是队伍为组装自己的硬件解决方案而设计的。WPI 开发的程序 GRIP 是一种介于传统计算机视觉和现成解决方案之间的一种混合。GRIP 提供类似于其中一些解决方案的接口，但会生成 OpenCV 代码，让用户可以进一步调试

Step 3: Vision System Output 视觉系统输出

对步骤 1 中的问题的回答以及正在使用的视觉系统的具体细节决定了视觉系统的输出以及如何进行通信。最常见的沟通方法是网络地址 Ne 两张桌子。 NetworkTables. 典型的视觉系统输出包括:

- **A measurement of horizontal offset 水平偏移的测量** - 偶尔这是一个角度，但更多的情况下，它是对帧中检测到的目标的偏移量的像素测量。 这可以直接使用或用于计算一个角度来通知机器人移动。

- **A measurement of vertical offset 垂直偏移的测量** - 偶尔这是一个角度，但更多的情况下，它是对帧中检测到的目标的偏移量的像素测量。 它通常用于使用方法计算到目标的距离类似于这里 here 所描述的（注意：这个特定的描述使用目标的宽度来描述，但是可以使用类似的数学和原理）
- **A measurement of the target width and/or height 目标宽度和/或高度的测量** - 这些几乎总是像素测量，如果使用，通常用于使用这里 here 描述的方法来评估与目标的距离（此描述使用目标宽度；可用数学很容易地转换为使用目标高度代替)。
- **Robot pose 机器人姿态** - 一些视觉系统使用姿态估计技术，比如 SolvePNP 提供相对于目标的机器人姿态的完整估计（姿态由位置和角度测量组成）

Step 4: Use the Output to Command the Robot 利用输出对机器人发出指令

一旦从视觉系统中获得关于目标的信息，它就可以用来对机器人发出指令。除了下面的信息，你可以找到各种“案例研究”，这些案例来自于 Limelight 文档 Limelight documentation 即便你没有 Limelight 这个硬件，文档也是有帮助的。

Convert Vision Information to Rotational Commands 转换视觉信息为旋转指令

如步骤 3 所述， 你的视觉系统应该提供一个输出，该输出为你提供有关目标在图像中的水平位置的信息。为了让你的机器人对准目标，使用这些信息为你的机器人生成一个旋转指令。典型来讲一个 control loop 需要被用到. 可行的两种方法如下（第三种替代方法在下面的“路径规划”中讨论）：

1. **Close the loop with vision 用视觉结束循环** - 重复使用此视觉测量直接生成转向命令。这种方法受摄像头帧率图像稳定性的限制。如果转得太快，你可能会超过目标后才得到一个新的图像，或者你可能会因为摄像头的摇动而得到不良数据。 这种方法是最简单的可行的，只要限制转弯率，就可以提高相机的适应力。
2. **Close the loop with a gyroscope 用陀螺仪结束循环** - 使用视觉信息生成一个角度转向，然后使用陀螺仪作为反馈转向该角度。 该方法利用陀螺的快速响应将机器人旋转到让视觉系统再次检查之前对目标进行估计，以验证目标现在是处于中心位置的（在一定的误差范围内）。 这种方法稍微比在方法 1 中描述的方法复杂一些， 可能不适用于更迅速的视觉系统。

***技术提示**: 虽然本节是按照控制机器人驱动系统方面编写的，但同样的原则也适用于机器人的旋转部分，如炮台结构。 想要在这种结构中适用方法 2，陀螺仪必须安装在机器人的移动部件上*

Convert Vision Information to Driving Commands 转换视觉信号为行驶指令

如果目的是将机器人导航到与目标相距特定距离，则可以使用距离测量来为机器人生成行驶命令。这些方法与上面描述的内容相同，只需要用“距离”代替“角度”，用“编码器”代替“陀螺仪”

Combine Rotation and Driving Commands 结合旋转和行驶指令

为了向目标行驶，你可能需要同时持续行驶并保证正确的行驶角度。要做到这一点，你可以将角度和传动输出的结果相结合。

技术提示: *如果来自两个循环的最大输出的组合可以超过系统的最大输出（例如，如果使用 PWM 控制器或%VBUS 模式的值超过 1.0），请检查此条件并划分输出，从而让两个指令中的较大值不超过系统最大值。否则，所需的转弯输出量将无法保证（例如，这两个输出的组合在左边产生 1.0 的最终输出，在右边产生 2.0 的输出；由于系统只能输出到 1.0，所以双方都得到相同的命令，并以相同的速度前进，而不是右边前进速度两倍于左边）*

Convert Vision Information to Shooting Commands 转换视觉信息为发射指令

如果比赛道具被发射或扔进一个目标，距离测量可以转换为与发射比赛道具相关的指令，而不是驾驶机器人。 而这取决于机器人的设计，这些指令通常是速度、角度或两者兼而有之。 虽然有物理方法可以通过这些测量获得一个很好的起点，捕捉 100%的可适用的物理法则在方程中是困难的。相反，经验数据通常被收集后，或者适合一个方程，或者输入一个查找表（有或没有插值），以转换距离测量信息（或视觉系统的其他输出）为发射指令

Alternative approach – Path-planning 可选方法 - 路径规划

另一种方法是使用 path-planning 路径规划工具在当前位置和期望目标之间生成平滑的机器人行驶路径。 这种方法需要确定机器人相对于目标的姿态。 代码使用现有的机器人位置作为起点，理想位置（通常是与实际目标/目标的小偏移）和角度作为最终的路径生成到达目的地的平滑路径。机器人可以完全遵循这条路径，也可以开始遵循这条路径，然后在路上通过读取视觉系统的更新姿态来计算新的路径

SELECTING DRIVERS 选择操控手

FRC 的一个重要技能是驾驶和操控机器人。 就像在体育、艺术、音乐和教育（仅举几个例子）中，技能是天赋和实践的应用产物。由于过度关注机器人的搭建，队伍往往会忽视考虑对于操控手的选择是具有竞争力的队伍和普通队伍之间的一个关键区别。本指南列出了一些有用的考量，帮助队伍选择操控手

操控手必须有良好的沟通技巧，展示操作机器人的天赋，并承诺在整个比赛季练习和获得经验。一些选择操控手的技巧和考量包括:

- **Talent vs Experience 天赋与经验** – 在评估操控手的时候，考虑候选人的原始天赋和他们积累的经验。 天赋，或候选人天生操作机器人的能力，是一个强大的差异设定可以让候选人出现差异化。一个有很高天赋但经验很少的候选人通常可以在表现上等于或大于有经验但天赋很少的候选人。与体育、艺术，音乐和教育类似，一个有高天赋的人不能仅仅依靠天赋，必须实践和反思才能提高。操控手的选择过程应该寻找有天赋的，敬业的候选人愿意投入时间和精力来培养他们的技能。考虑一下你将有多少时间来练习机器人。如果你没有时间练习，一个有天赋的候选人可能是正确的选择。如果你有更多的时间来练习，一个初始天赋较低的候选人，但在沟通和沉着方面表现出色，可能会随着时间的推移成为更好的选择。你不必等待你目前的机器人完成后，再评估操控手的天赋。往年的机器人，测试用的驱动系统，或当前机器人的底盘都可以用来评估操控手的天赋
- **Communication 交流** – 有效和尊重地交流新想法、策略和改进机会的能力对操控手来说至关重要。有效的操控手也能够和团队其他人沟通机器人改进和问题。操控手经常发现如果他们的驾驶团队内部有“化学反应”，那么交流和分享想法就更容易了，这往往反映了良好的融洽关系，通过分享经验比如队伍团建练习。操控组团队成员应该感到能安全的相互交流挫折和成功，并作为一个团队一起成长和学习。 赛后的比赛“汇报”是一个突出改进领域和庆祝成功的好方法.
- **Composure 冷静** – FRC 淘汰赛是一个复杂的环境，具有巨大的竞争和时间压力。有效的操控手将能够在压力下保持冷静，并在采取最佳行动之前冷静地评估情况。在评估操控手时，寻找能够在状况发生时适应良好的个人，这样的人能迅速采用替代方案。 而详细的计划可以减少意想不到的场景及恐慌，因为在淘汰赛的过程中，意想不到的事情很可能仍然会发生，操控手将被迫立即作出反应 .
- **Rules 规则** – 虽然阅读和理解规则不是操控组的唯一责任，但操控组的每个成员都必须彻底了解规则。尤其是和每个操控组成员在操控组中的角色相关的规则。成功的操控组通过持续关注所有的团队更新 Team Updates 和问答系统 Q&A 来保持对规则的持续警惕性 在所有赛事活动中都要备有更新过的比赛手册 Game Manual 的参考副本。有些比赛胜负是由一个罚分而决定的，所以理解规则是关键!

IMPROVING DRIVING PERFORMANCE 改善操控表现

几乎每年 FRC 比赛的一个标准部分是获取和利用比赛道具得分的比赛循环。这个循环，从刚刚得分，到获得比赛道具，到再次得分，通常被称为“循环”。对于大多数比赛，一个队伍的大部分比赛时间都是在重复这个循环，所以优化循环是至关重要的。本指南提供了一些关于如何练习，练习什么，以及如何进一步分析和改进循环时间的想法

除了本知道，你或许还想更多的社区资源中的类似话题，有两个资源对本指南有很大的启发，一个是 610 队的循环优化 *Cycling Optimization* 一个是 2168 队的操控组手册 *Drive Team Manual*.

How to Practice 如何练习

识别和实现提高操控手技能的过程和技术对于优化机器人的性能至关重要。提高驾驶技能最有效的方法之一是练习。练习不用等待机器人完成，以前的机器人、测试用的驱动系统或当前机器人的底盘都可以在机器人参赛之前开始练习。 虽然简单的练习（无论是物理上还是实际上通过模拟器或合作游戏）是有效的，但你如何练习也是非常重要的。 有几种技巧 这可以帮助操控练习更有效 - 其中一些包括:

- **Start Slowly and Work Up 慢慢起步稳扎稳打** – 当学习一项新技能时，最好慢慢开始。 成功的操控手往往从开车开始，慢慢完成任务。 一旦熟悉确认了控制、任务和事件的顺序，则加速每项任务的完成度。 如果在特定的速度或水平上练习需要扩展以建立信心或熟练程度，那没关系。最终操控手会发展出肌肉记忆和让下意识行动成为“第二天性”。在操控手变得“人机一体”后，并进一步完善他们的能力。.
- **Don’t Be Afraid of Crutches 不用害怕拄拐前行** – 学习新技能的一个有用方法是使用辅助元素。 一些例子包括以下内容:
    - 箭头贴在地板上，以帮助学习如何导航一个转弯序列
    - 线贴在地板上或现场元素上，以帮助定位或准确性, 或
    - 当学习的时候使现场元素呈现不同的颜色，以提高对比度
- 随着时间的推移，操控手从辅助元素以外的地方获取线索，例如机器人与操控手的特定距离时的外观，或者机器人保险杠依靠场地元素来对齐。此时，辅助学习插件可以慢慢去除
- **Drive with Finesse 灵活操控** – 团队有不同的表达方式，包括“慢是光滑的，光滑是快速的”和“时间是无价的”，但他们都有相同的基本目标：用精细的方法驾驶，而不是暴力和纯粹靠速度。过度地将比赛道具捣进场地元素里可能会有必要，但它并不总是有效的，相反会导致一些本可避免的机器人损坏、时间的浪费或犯规。有竞争力的操控手知道，五秒钟的精彩操控可以改变比赛的结果。
- **Practice with Purpose 有目的的练习** – 操控手往往会对队伍感到一种所有权和责任感，“不破坏机器人”，导致他们不能将机器人推向发挥其全部潜力的地步，并错过发现机器人早期在设计和建造阶段的可能的弱点。相反，“像你偷的一样操控它”技术有助于在事件发生前识别和解决问题。依靠团队的技术技能来保持机器人在顶端的形状；不要害怕推动机器人到它的极限，但仍然需要灵活精细操控

What to Practice 练习什么

在你能够的范围内，建立一个尽可能代表真实领域的实践环境。 布局和标记图和队伍版场地元素构建指南在比赛场地页面 Playing Field 查找，并作为这方面的有用参考:

- **Start Simple 简单起步** – 当用一台新打造的机器人开始练习的时候，简单起步即可。专注于机器人的基本功能，测试机器人在比赛中需要有的功能。操控手也需要熟悉机器人及控制。查找别扭和困难的东西能让队伍有机会迭代

- **Practice Drills 练习训练** – 音乐家知道学习演奏乐器的一种有效的方法是学习如何演奏音阶或特定的有序音符序列。音阶帮助音乐家发展速度、灵巧和音乐记忆。对于操控手来说，钻头对机器人来说就像天平对仪器一样。 练习可以包括驾驶回转或其他障碍/图案课程，拾取和释放比赛道具，并重复 将比赛道具精确地打分到/放在一个练习装置上。其中一些训练甚至可以在操控手选择期间使用，因为有才华的操控手可以通过训练来识别，然后通过练习增强他们的技能.

- **Cycle Tests 循环测试**– 练习比赛道具获取、行驶和得分的完整循环。会感到重复枯燥，但你对这个过程越熟悉，在压力下复制它就越容易。考虑计时循环或记录练习，以便你有客观的数据来回顾和理解不同的技术如何影响结果。确保练习不同的路径，行动和位置确保可能在比赛中遇到的各种情况.

- **Add Precision 增加精准度**– 试着通过训练来测试和提高你的驾驶精度。用于使用障碍物制造狭窄的最佳的驾驶路径。或重复对象拾取或交付的训练，直到始终能达到 100%的准确度。

- **Add Variables 增加变化量** – 操控手需要为比赛中可能发生的各种事情做好准备。想像在比赛中可能发生的所有“如果”，并开始为最可能发生的状况做好规划和练习:

    - 如果有一个最佳的路径或得分位置，但若它被阻碍了该怎么办?
    - 如果你的机器人的某些功能在比赛中崩溃了该怎么办?
    - 你如何响应防守?
    - 你如何分享和你的联盟队友共用的空间?
    - 你能在一些路线上倒着开车么?

Analyze and Upgrade Cycles 分析和改进的循环

虽然练习本身确实有助于改进循环，但分析这些循环中正在发生和没有发生的事情可以为迭代提供信息，并导致进一步的改进。 做到这些分析的具体思路有:

- **Think about the controls 对控制方式进行思考** – 思考机器人的控制方式。WPILib 提供了街机，曲率和坦克驱动模式的默认实现。你可以在 WPILib 文档 WPILib documentation 中了解更多关于这些差异的信息。其他排列已经在社区内发布，可能会或不会让你的操控手感觉更直观。另一个需要考虑的方面是机器人导向和场地导向的操控。 在场地导向的操控方式中，驾驶员在移动操纵杆时不再考虑机器人的方向，将操纵杆远离驾驶员，将机器人驶离驾驶员，而不必在意机器人的朝向。场地导向操控最常用于具有全方位运动的机器人，如麦克纳姆轮 mecanum 或 四驱全向轮 swerve drive 底盘的机器人，但可以使用差分驱动实现。 驱动控制应该尽早弄清楚，因为操控手需要时间来适应任何变化。确保控制是直观的。当第一次操控的时候如果操控手按到不正确的按钮是允许的，但需要密切观测这一点，因为这可能是一个迹象，那就是有可能按键的布局让操控手不适应.

- **Think about the controller 对控制器进行思考** – 另一件要尽早查看和测试的事情是确保操控手正在使用的控制器。不同的队伍在操纵杆、游戏控制器，方向盘和油门的操控组合的选择上取得了成功。找到一个直观的选项，让操控手有正确的运动范围.

- **Analyze the cycles 分析循环 –** 分析它们执行的循环。 将计时分解为三个环节：比赛道具获取，场地游走，比赛道具交付或计分。这三个环节中哪一个占用了循环中最大的比例？ 如何改进每个环节？ 是否有设计或软件上的调整可以提高你的速度？有没有无谓的时间和运动？精度如何影响循环的有效性?

- **Add automation 加入自动化** – 减少循环时间的一种方法是加入自动化。例子有:
    - 使用计算机视觉帮助列举某些目标和目的地，如在视觉目标 Targeting with Vision 部分讨论过的那样
    - 自动定位机器人机制, 或 机器人内部比赛道具的自动移动.