RoboMaster 2018 夏令营
技术报告

18 组

目录

1 组员及分工

姓名	RM 编号	技术方向	组内工作
金子旭	RM000222	机械	机械设计与装配
黄毅	RM000231	机械	装配、视频记录
ROWENA LU	RM000290	机械	建模、装配
丁致远	RM000059	嵌入式	嵌入式、算法 UI
何清华	RM000264	算法	第一阶段算法、视觉
周梓清	RM000256	算法	第一阶段算法、视觉

2 需求分析

2.1 赛题分析

本次赛题分为两个部分.第一部分为个人战，要求机器人根据准备时间给出的条件，从资源区运送正确颜色的积木块到比赛区.第二部分为城堡攻防战.城堡由 4 个积木块十字交叉拼出，城堡周围路基为 1 个积木块，机器人需要在规定时间内尽多搭建（守城方）或拆除（攻城方）场内城堡。

2.2 策略设计

本次赛题的主体为运送积木块。由于硅胶轮的摩擦力大，容错率高于夹子，机械设计选其作为收取积木块的装置。又因为资源区中积木块摆放紧密，从积木块短边收取比从长边有更多的空间，所以机械设计选择了从短边进行积木块收取。此外，为了减少机器人在场上运动搬运的时间，一个存储积木块的机构是需要的。

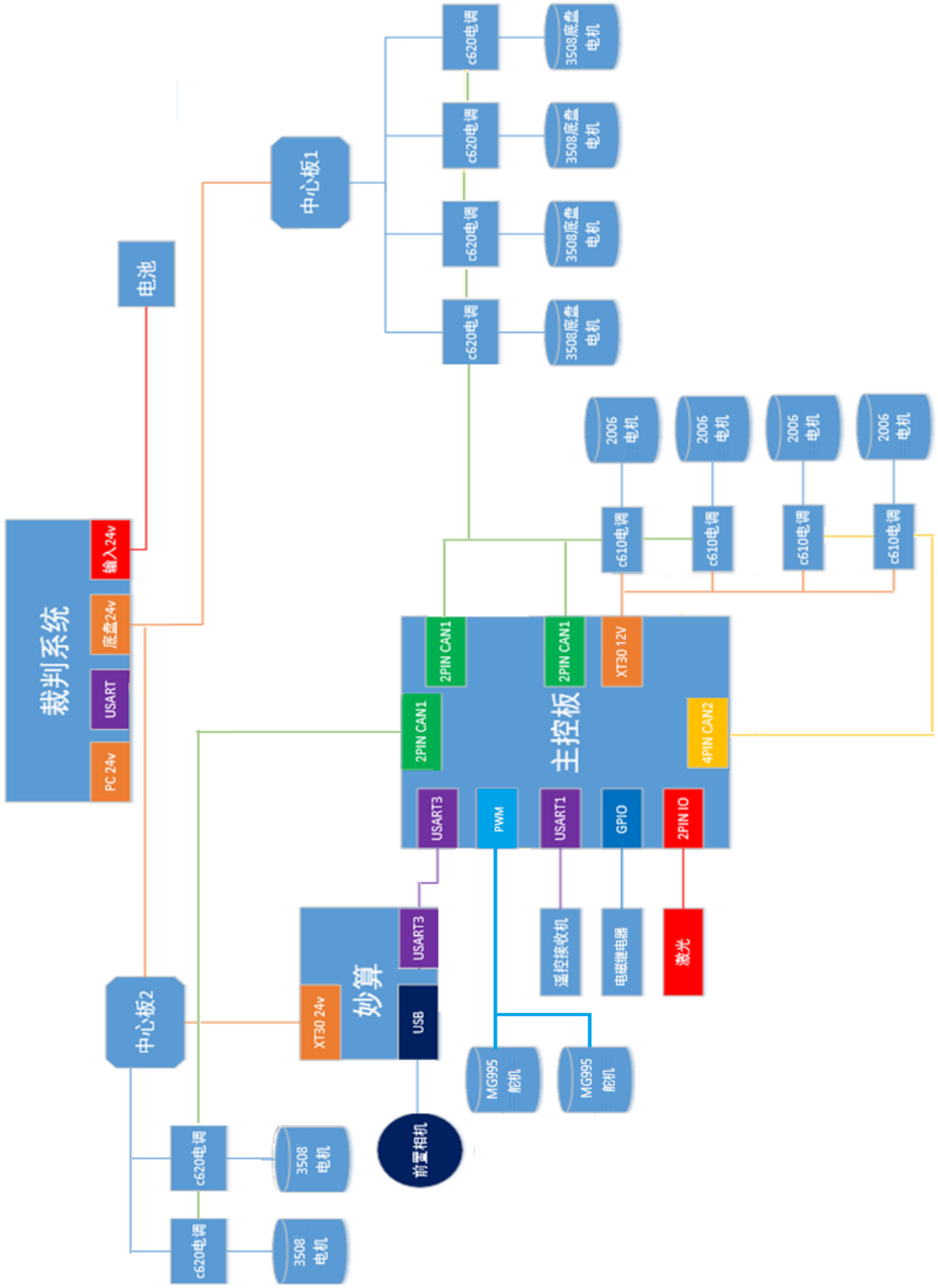
为了不放弃第二阶段 15 秒图传交替中断的时间，嵌入式会在代码中增加一些自动控制，如旋转一定角度、前进一段距离、自动抓取/放置积木块等。

算法的主要任务为计算出第一阶段积木块排放的位置，并将其展现给操作手。UI 界面的设计能让减少操作手在赛场上的失误率，同时也能让操作手更好地定位机器人在场上的位置。同时为了配合硅胶轮收取方块的位置，我们还决定通过视觉辅助来帮助机器人定位。

3 关键词

硅胶轮；单摇臂；通信；控制；深度搜索；视觉识别

4 总体方案(横向)


裁判系统
输入24v
底盘24v
USART
PC 24v
电池
中心板1
c620电调
3508底盘电机
2006电机
c610电调
主控板
2PIN CAN1
XT30 12V
4PIN CAN2
USART3
PWM
USART1
GPIO
2PIN IO
中心板2
妙算
XT30 24v
USB
遥控接收机
电磁继电器
激光
MG995舵机
前置相机
3508电机

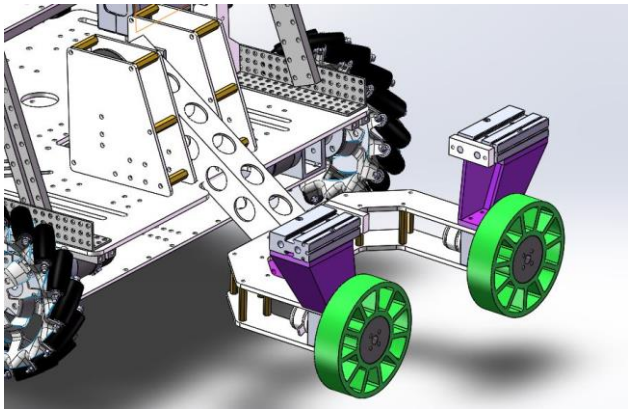
5 模块方案

5.1 机械

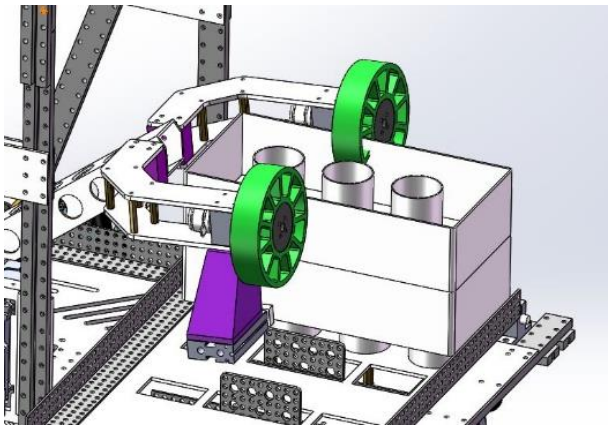
机器可主要分为三个部分，分别是收集机构、抬升机构和储存机构，该三部分相互配合即可完成对积木的收集与储存，下面将对这三个模块分别进行介绍。

5.1.1 收集机构

收集机构是一个以前端为双硅橡胶摩擦轮的单摇臂机构，前端硅胶轮的邵氏硬度为 35 度，在吸取积木后的形变和为 40mm，可有效防止积木在两摩擦轮轴线上的周向发生转动，从而实现积木的稳定收集。



同时该机构也可实现积木的放置与城堡的积累，可直接通过硅胶轮的形变将积木块从城堡中拆出或将积木块按入城堡。另外为防止从仓库取出积木块时积木块的连接，在摩擦轮的上方设计有 2 个缸径 10、行程 25 的微型气缸，在摩擦轮将累在一起的上层积木取出时，气缸将下层积木加紧固定，从而使摩擦轮可通过摩擦力将压下一起的积木分开。

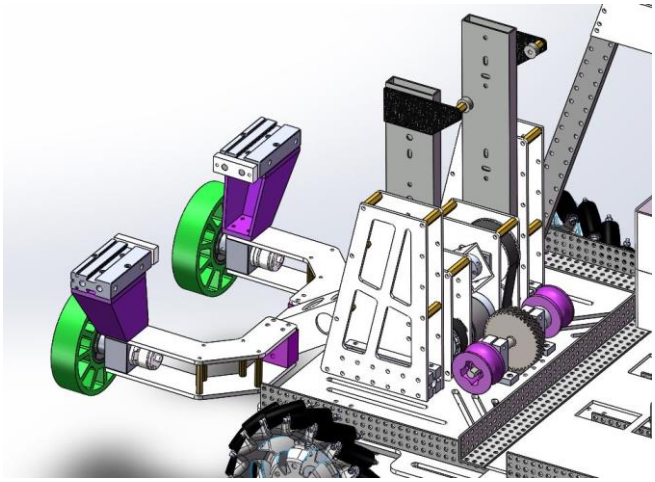


这部分最初的想法是 Rowena Lu 提出的，在提出后大家进行了讨论并对方案进行了改进，并由金子旭进行了具体的设计建模与方案论证，最后有负责机械的几位将其装配出来。

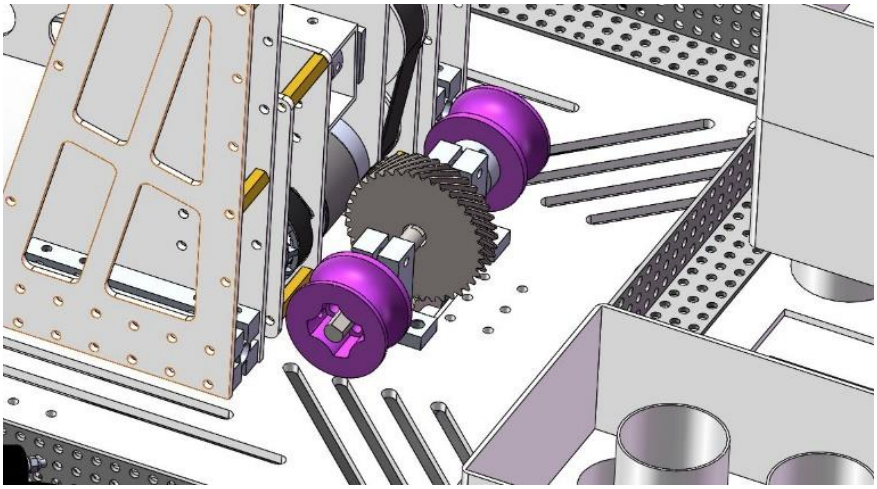
5.1.2 抬升机构

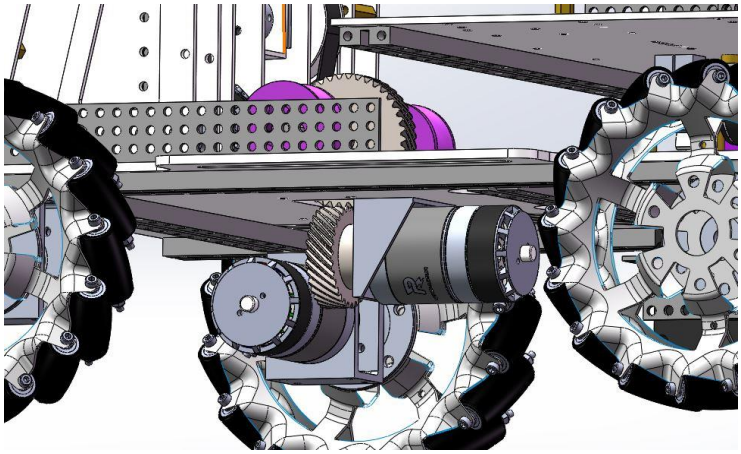
在积木累积的过程中积木的高度会随积木的层数而不断增加，而摩擦轮在对积木的收集与放置使需要尽量与积木保持平行，这就需要一个抬升装置能实时调整机构的高度，以保证其与积木的平行。

该机构为一通过卡夫拉线控制的双滑轨抬升装置，通过线轴的转动来改变放出线的长度，并通过滑轨与收集机构自身的重力实现线的涨紧，从而实现通过线轴的转动控制收集机构高度的目的。



线轴部分的动力我们使用了 45°的斜齿轮进行传动，这样设计可将电机的位置移到底板下方，从而将底板上方的空间节约了出来，为气动部分的控制元件及抬升机构的线路留出空间。斜齿轮产生的轴向力我们通过两侧的光轴支架进行支撑，使结构保持稳定。



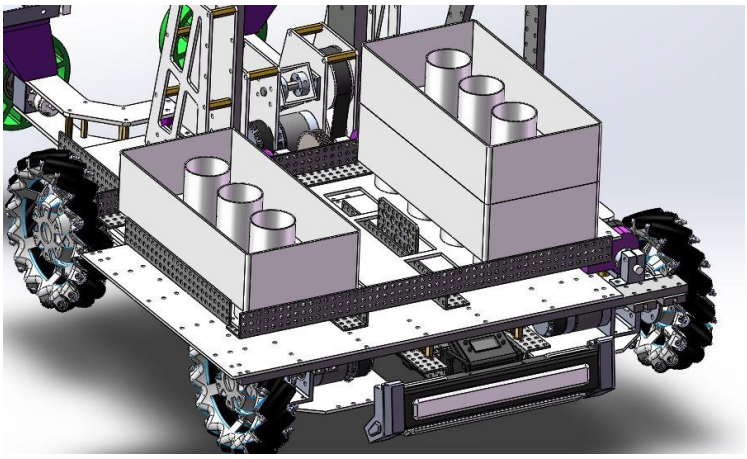


这部分主要是由金子旭提出设计并完成方案论证，由黄毅与 Rowena Lu 完成具体装配。

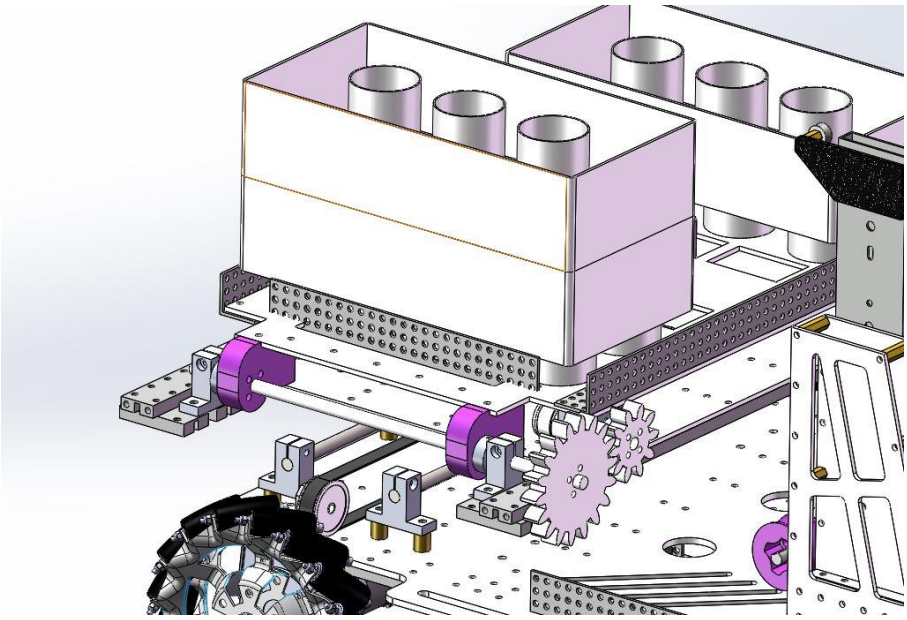
5.1.3 储存机构

在本次赛题中，机器所需要完成的主要任务为对积木的收集与放置，而资源区中的积木的摆放方向均为短边面向场地，若设计为从积木的长边收集，则机器需整体进入资源区内部，这在比赛过程中会对机器收集积木的效率造成很大的影响，所以我们将机器设计为从积木的短边收集，这样的设计能使机器在收集时不需要进入资源区内部，降低了操作难度，并提高了效率。在热身赛中，我们发现若机器需将积木摆放在距其较近的基座时，“夹一块、放一块”策略的效率要高于一次性储存大量积木，并同时放置很多积木的方案。但在需要机器长距离移动时，大容量储存机构的效率是要明显高于单取单放的策略的，这就需要机器既能方便的单个夹取，又要能有一个一次性储存较多积木的能力。针对这一发现，我们经过实验并最终认定最佳的储存数量为 4 块。

该储存机构的结构可分为底层滑台与上层翻板，底层滑台通过以压齿板与同步带固定，从而实现了通过同步带控制滑台的左右移动，进而实现了积木的双列储存，实际实验最大的容量为 12 块。当然，我们通过实验发现在存储数量为 4 块时可同时兼顾各种距离不同的积木，故最佳的储存数量为 4 块。



同时该储存机构还可直接将储存的积木翻出，快速实现将收集到的对方积木运回资源区的目的。该部分机构通过一个通过雕刻机切割出的 pom 板 4 模齿轮实现翻板的转动，RM2006 的最大持续扭矩为 1N.m，该机构的减速比为 5：9，由此可由计算得知将大量积木翻开的难度并不大，从而验证机构的可行性。



这部分机械部分由金子旭提出设计，由金子旭、黄毅和 Rowena Lu 共同完成具体的装配。有丁致远提出利用红外传感器实时监测仓库内积木的数量，并由其完成了该部分机构与单摇臂部分的半自动操作。

5.2 嵌入式

本队的嵌入式部分主要有三大任务：

i. 编写控制程序，使各机械臂能按照指定角度/速度运动；

ii. 与算法/视觉组配合，与 *Manifold*（妙算）通信，使机器能根据图像识别反馈的数据进行校准；

iii. 完成半自动程序的编写任务.

由于本次的机械结构较为复杂，自由度较多，控制也较为复杂，很多参数依赖于机器人的当前状态，因此，机器人所需较高的自动化程度。对于嵌入式系统的自动和半自动而言，该两种控制模式所分配的任务如下图所示。

嵌入式部分使用的电气设备如下所示：

摩擦轮[0]：	M2006 + C610	**仓库挡板：**	MG995
摩擦轮[1]：	M2006 + C610	**图传转台：**	MG995
机械臂升降[2]：	RM3508 + C620	**继电器：**	德力西 CDG1-1DD
机械臂[3]：	RM3508 + C620	**电磁阀：**	德力西 4V210-DB
仓库横向[0]：	M2006 + C610	**光电模块：**	基于 LM393
仓库旋转[1]：	M2006 + C610	**指示灯：**	若干

5.2.1 自动拾取积木块

初始条件：在拾取或放置积木块程序启动前，程序会将*机械臂[3]*抬升到向上抬升到与水平方向呈 60°夹角的位置，*机械臂[2]*回到最下面，*仓库电机[0]*回到中间位置。

i. 机器人根据 *CV* 反馈自动（或根据图像回传手动）调整到积木块中心位置；

ii. 仓库电机自动将爪子所对应的位置切换到指定的仓库（共两个），仓库后安装的若干红外传感器识别，并由程序计算当前仓库中所剩余的积木块的数量；

iii. 如果当前仓库已满，则取消本次取块操作，并将目标仓库自动切换到另一个；

iv. 机械臂由初始位置下落，摩擦轮转动，将积木块吸取到爪子内；

v. 程序根据目前仓库中的积木块数量，计算爪子需向上抬升的高度，并抬升到此高度；

vi. 爪子向后翻转，将积木块扣在仓库中；

vii. 摩擦轮继续转动，积木块脱离爪子，进入仓库；

viii. 爪子向前反转，回到初始位置；

ix. 爪子下降，回到初始位置；

x. 摩擦轮回零，仓库回到中部初始位置（晃动时，未到位卡合的积木块会自动卡合），取块操作完成。

为了使电机稳定到某个位置，有大量电机需实现较精准的角度控制，因此控制过程中使用了大量的速度环嵌套位置环 *PID* 控制算法，以实现系统的稳定性。由于机械臂本身较重，在不同角度时所受外力不同，电机出现无法控制角度、难以驱动的问题，因此需要在不同阶段使用不同的 *PID* 参数常量。此为所有控制过程中的较难点，经机械组优化后该情况好转。

5.2.2 自动放置积木块

初始条件：同自动拾取

i. 机器人调整到需放块的位置；

ii. 仓库电机自动将爪子所对应的位置切换到指定的仓库（共两个），仓库后安装的若干红外传感器识别，并由程序计算当前仓库中所剩余的积木块的数量；

iii. 如果当前仓库没有积木块，则取消本次放块操作，并将目标仓库自动切换到另一个；

iv. 程序根据目前仓库中的积木块数量，计算爪子需向上抬升的高度，并抬升到此高度；

v. 摩擦轮转动，将积木块吸取到爪子内；

vi. 爪子向前翻转到最低点，机械臂回落到最低位置；

vii. 摩擦轮继续转动，积木块脱离爪子，落到地面；

viii. 爪子抬起，回到初始位置；

ix. 摩擦轮回零，仓库回到中部初始位置，放块操作完成

5.2.3 视觉组对接

为了更方便进行机器人和积木块的位置校准，视觉组使用 *Manifold* 在 *OpenCV* 平台编写了自动识别程序。

为接受到视觉计算结果（机器人摄像头与目标积木块在 *x-axis* 的偏移量，单位为 *px*），*STM32F427IIH6* 开发板与 *Manifold* 使用 *UART* 协议进行通信，波特率为 9600，数据位为 8 为位，停止位为 1 位，无校验位。实际接收时，开发板需获取前四位，格式如下：

符号位 [0]	数据位 [1]	数据位 [2]	数据位 [3]
0 为正，-为负	百位	十位	个位

接收到数据后，触发 *Usart3_callback()*函数，解析接收到的数据，并通过 *PID* 位置环计算出底盘在 *x-axis* 的移动速度，并累加到 *chassis.vx*，实现底盘的自动对位。

据实际测试，视觉组的程序在近距离靠近积木块时才比较准确，距离积木块一定距离时，并不能完全达到理想效果。为了避免视觉识别出现其他误差，自动对位仅在按下键盘某个按键时触发，松开后即停止。

5.3 算法

5.3.1 第一阶段算法

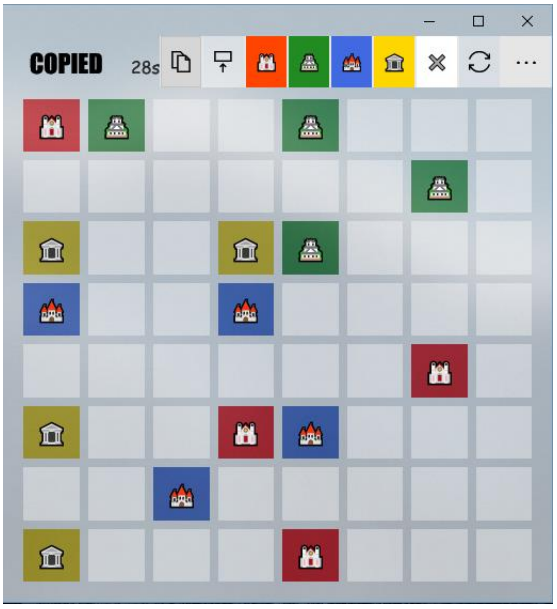
考虑到第一阶段算法题复杂度不高，相比于动态规划，我们选用了代价较大但是代码逻辑更简单的深度搜索。以下为搜索顺序:

i. 输入地图（用 R, Y, B, G, W 表示各个颜色的城堡）
ii. 在地图上从左上角开始寻找第一个城堡，标为已访问并记录颜色
iii. 用深度搜索依次搜索每一个格子上、左、下、右方向的格子，直到找到与其颜色相同的一个城堡并标记，完成一条路径
iv. 在深度搜索过程中标记已经走过的格子
v. 寻找一个未被标记的城堡，作为下一次搜索的起点，重复上面步骤
vi. 重复直到 8 条路径都找到，中间如果遇到无解的情况则退回上一步尝试另一种走法
vii. 检测是否所有格子都被标记过
viii. 如是，则结束搜索，打印出最后生成的地图；如不是，则退回上一步继续下一种可能性

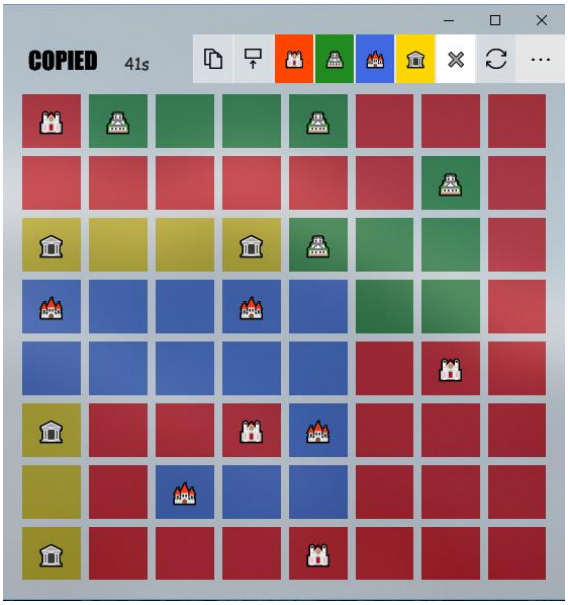
5.3.2 算法 UI

为方便操作手迅速定位应放置积木块的位置，本队使用基于 XAML 的 UWP 框架编写了算法配套 UI（如下图）.

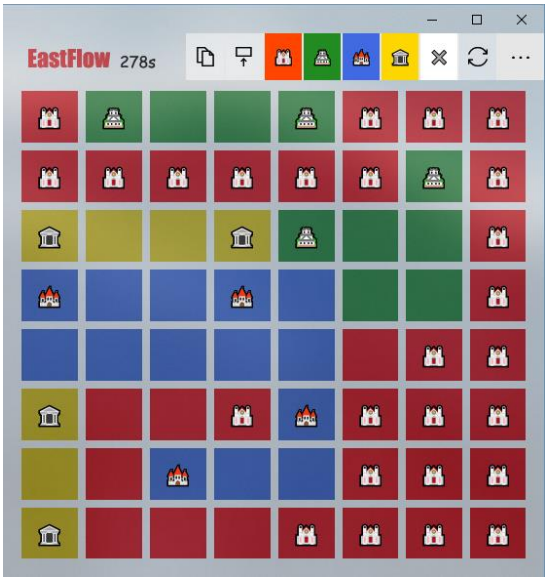
在算法运行前，在 UI 界面上点击空白位置，即可将此位置标定为城堡；



在算法运行结束后，UI 界面上可以自动显示出路基点的位置；

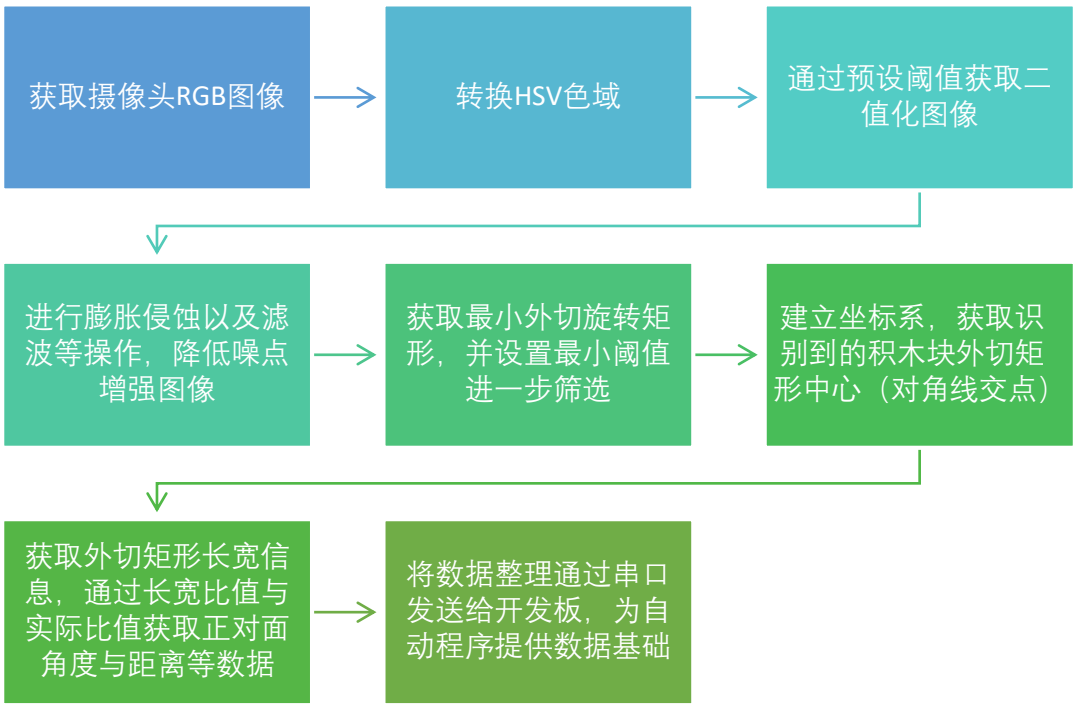


在算法运行后，第二操作手可以继续手动标定机器人已经放置完成的积木



5.3.3 视觉

视觉识别程序工作原理及流程图



技术细节

图像预处理：从摄像头获取图像，并将图像分辨率调低、进行模糊处理。模糊的图像可以大大减少单张图片的计算量，提高计算速度。同时 RGB 色域也被转换为了 HSV 色域，更符合人眼对颜色的认知。

各色积木块的二值化图像获取：调用 inRange 函数，使用设置好的阈值对图像进行二值化操作，使画面上除积木块以外的全为黑色，积木块则为白色。输出二值化图像以方便调参与确认各色积木块位置。

> **阈值的设置：**因为比赛场地（浅灰色）与积木块（亮色）的色差较为明显，为了能在识别积木块的基础上减少环境干扰项，程序中将阈值调整得较为宽松。在背景颜色影响不大的情况下，大范围阈值允许了其他噪声（如光线因素）对积木块识别的影响，使得识别更加准确。

降低噪点提高识别度：进行二值化操作后，通过膨胀与侵蚀操作初步消除背景大部分噪点。但在消除噪点的同时，膨胀与侵蚀也会造成积木块边缘模糊，不好确认形状。为了得到更清晰的边缘数据，程序中还使用了 blur 函数进行锐化操作。

获取最小外切旋转矩形：二值化后积木块的边缘则为黑色与白色色素的交界处，通过识别图像上的这些像素点的坐标点集，我们可以计算出白色区域的最小外切矩形。程序将外切矩形的四个顶点进行记录，并将外切矩形输出在图像上以供测

试使用。同时为了进一步降低干扰，占整个图像面积百分比过小的外切矩形将被过滤，防止背景上没有被过滤的噪点被误识。

获取积木块偏移量：通过刚才获取的最小外切矩形的顶点坐标，我们可以计算出矩形的中间坐标（对角线交点），并据此计算积木块相对于图像中心点（积木块方便 夹取的最佳位置）的 X 轴与 Y 轴偏移量，用正负分别表示外切矩形中心与图像中心的左右关系。这些数据将发送给开发板，给自动校准程序提供信息基础，以实现自动对位的效果。

平台

图像识别程序用 *C++*语言进行编写，调用了 *OpenCV* 图像识别库，并在 *Manifold* 上运行。通过开机登陆后脚本的运行，图像识别程序将在 *Manifold* 接入摄像头的情况下开机运行。

在图像识别程序运行的同时，通信程序也将运行。通信程序会通过 *Manifold* 上 *UART3* 串口，以波特率 *9600*，每秒 10 次的速度传送外切矩形与图像中心点偏移量传送到主控板，与嵌入式对接。

6 理论分析

6.1 机械选型

在底盘设计之初，我们考虑到每个路基间的中心距为 930mm，且考虑到积木的长边尺寸约为 250mm，故实际间距约为 680 左右，若想使机器在场地中存在很多城堡的情况下仍能灵活移动，就要求机器的最长边在 500mm 左右，故我们将底盘的尺寸设计为 500*450，以便其能在场地中方便的移动。

在该机器中，动力部分主要有 3508 电机、2006 电机及气缸。设计中对气缸的推力要求不高，能为积木提供一个适度的摩擦力即可，该 HN 10-25 气缸的缸径为 10、行程为 25、推力为 17Kg，足以满足现有需求。

单摇臂部分我们使用了 3508 电机通过减速比为 24：35 的同步轮驱动，但在实际使用中发现在控制时对电机的负担较大，短时电流以接近 10A，短时功率已接近 250W。虽能使用，但这会大大降低电机的寿命，为降低电机的负担我们对单摇臂部分加装了皮筋平衡器，一定程度上降低电机负担，延长了其寿命。

在测试中我们发现单摇臂的中心转轴经常会出现因顶丝对轴的径向力较大而造成中心轴产生形变，进而导致其极难拆卸维护。而且在这种情况下会因轴的形变使顶丝并不能紧固紧，而且在设计时为了使机构的体积较小，我们选用的同步轮是无台阶的 AF 型同步轮，这导致了在出现顶丝松动时的维护十分不便。

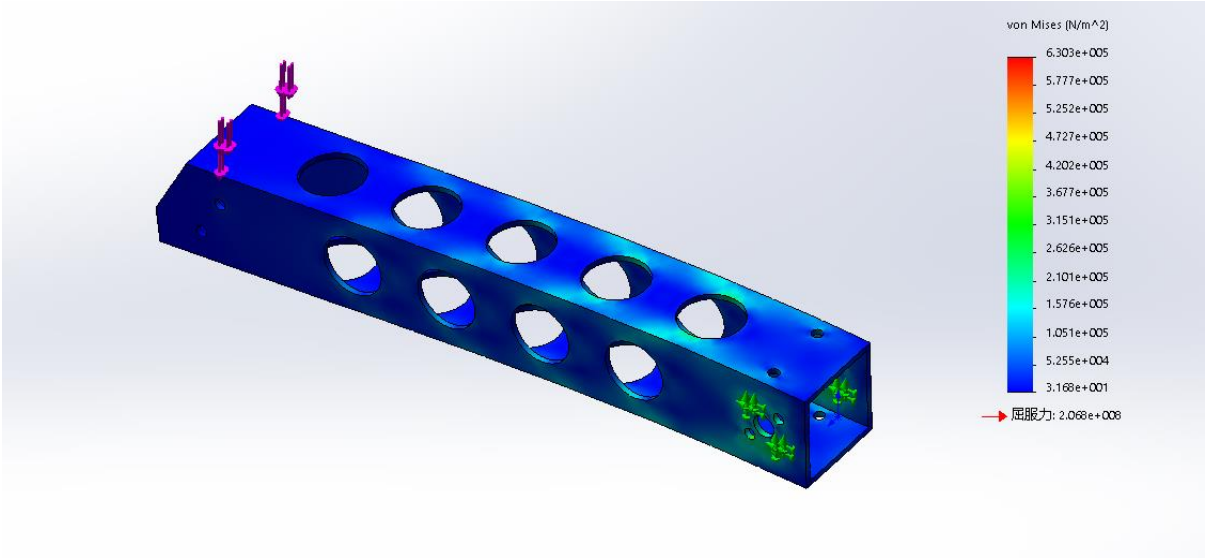
产生这种问题的原因主要是光轴选用的是软轴，使用的顶丝为 12.9 级的凹面顶丝，故在使用时光轴经常会出现光轴被顶丝压出较大的环形，且同步轮配套的顶丝为规格较大的 M5 顶丝，在承受力矩时会出现因顶丝过大而导致的轴的打滑现象的发生。

最后经过金子旭与何清华的思考，考虑到我们并没有板牙丝锥等工具，我们最终决定使用外接一个顶丝孔使用 M4 螺纹的法兰盘，并将中心的软轴改为同规格的经过调质的硬轴，从而解决了问题。

6.2 机械可行性

在该机器的设计中单摇臂是质量应该是尽可能的轻，而单摇臂的主体部分受力较大，这样要求这跟主体梁要在质量与强度中达到一个平衡。为实现这个目的，我们使用了 SW 中的 simulation 进行静力分析，模拟该主体梁在固定在单摇臂的轴上时收到一个向下的负载的情况下该梁的受力情况。

根据模拟出的结果可更改设计以进一步减轻主梁的质量，以减小单摇臂的整体质量，从而使控制难度更低。



6.3 算法可行性

第一阶段算法我们选用了深度搜索，时间复杂度为$O(n^2)$。在给出的题目足够随机的情况下，用其遍历 8*8 的棋盘可以在 2 分钟准备时间内完成。一下给出 5 组测试数据所使用的时间：

序号	所使用地图	算法使用时间
1	WWGWWYWW WWWWWWGW WWYWWWWY WWWBYWWG RBWWWWBW WWBWWWRW RWWRWWWW WWWGWWWW	0.012000s
2	WWWWBWWR WWWYWWWW WWWBWWYW BBWWGWGW WWWWRWWW WWWRWWWW WGRWWWWW GYYWWWWW	0.114000s
3	WWWWWWBR WRWWBWWW WWWRWWWW WWWWWWWW GWGWWWWW WGWWWWGR WWWWWWYB YWYWYBWW	0.075000s
4	BRWWWWWW	0.146000s

	WWBRWWWW RWWBWGWY WWWWWWWW WRGWWWYW WWWBWWWY WWWYGWWW GWWWWWWW	
5	WWWWGWWG WRWWWWWB WWRWWWWW WWYWWWBW WWWYWWWR BWWWWWWW WYGWGWWW BWWYWWWR	3.168000s
平均值		0.703000s

加上数据录入的时间，一般来说每场比赛只需要 30s 来形成地图，对于有 2 分钟准备时间的第一阶段来说时间很充裕。

7 制作与测试流程

热身赛是我们的机器人第一次进入场地进行测试，除了部分功能还未完成以外，已完成的功能与我们的设想基本相符。热身赛结束，进行最终比赛之前的几天中，我们一直在继续建造尚未完成的功能。

8 结果与评价

作为一个所有成员都有一定机器人经验的团队，这次机器人的建造过程基本上遵照了我们的设想。每一个机械设计都有许多优秀的机械结构的影子，每一个程序细节都是经验累积下方便赛场操作的设计，策略与联队也都不是临场的随意决定。但在同时我们也清楚地明白自己的眼界所在限制了机器的表现。希望以后能够接触到高出我们层级的技术，并且从中有所收获。

如果我们能有更好的时间管理，在接近项目尾声将有更多时间进行细节的完善。虽然操作练习与自动化程序调试也在我们的预想之中，但是突发的技术故障却让我们没有时间去完成。项目最初制定的一个可执行且有容错率的时间表应该能够解决这个问题。

9 感想与感悟

本次活动让每一位组员切身了解到时间管理的重要性，尤其是在像本次夏令营这样一个要求在短时间内完成大量既定任务的情况下，合理的时间管理就显得尤为重要，若能良好的对时间进行一个规划，并按照设定的计划稳步继续，相信能很大程度上减少熬夜这类情况的发生。

同时也正是因为本次活动要求在较短时间内完成较多的既定任务，这样一个合理的工作顺序就显得尤为重要，这点在机械上体现的尤为明显。 在设计方案确定之后应优先抓紧时间完成建模，并不断改进建模，以完成机械的改进，而不应该在一开始就利用已有零件搭建实体。 若没有遵守这个顺序，在大多数情况下都会浪费较多的时间在部件的安装与拆卸上，导致无法高效的完成任务。 所以这次活动也让我接触到了一个标准的产品设计流程，使队伍中的每个人都有所收获。

10 附录

10.1 嵌入式代码

startup.c *(已排除原有代码)*

```
int NO_AUTO_MODE = 1;
```

execute_task.c *(已排除原有代码)*

```
#include "execute_task.h"

#include "can_device.h"
#include "uart_device.h"
#include "cmsis_os.h"
#include "calibrate.h"
#include "pid.h"
#include "sys.h"

//uint8_t test_key_value;
extern uint8_t relay;
extern uint8_t if_pick;
extern uint8_t if_put;
extern uint8_t which_storage; // 0 is left, 1 is right
extern uint8_t usart3_recv[];
extern int CV_offest;

long pick_time_begin = 0;
//uint16_t servo_angle = 1000;

void Usart3_callback()
{
    if (usart3_recv[0] = '-')
        CV_offest = -(usart3_recv[1] * 100 + usart3_recv[2] * 10 +
usart3_recv[3]);
    else if (usart3_recv[0] = '0')
    {
        CV_offest = usart3_recv[1] * 100 + usart3_recv[2] * 10 + usart3_recv[3];
    }
}
```

```c
void execute_task(const void *argu)
{
    //测试电机初始化
    arm_moto_init();
    storage_moto_init();
    uart_init(USER_UART3, 9600, WORD_LEN_8B, STOP_BITS_1, PARITY_NONE);
    uart_recv_callback_register(USER_UART3, Usart3_callback);
    uart_receive_start(USER_UART3, usart3_recv, 4);

    write_led_io(LASER_IO, LED_ON);

    while (1)
    {
        if ((rc.kb.bit.X && rc.kb.bit.SHIFT == 0) || rc.sw1 == 1)
        {
            if_pick = 1;
            pick_time_begin = HAL_GetTick();
        }
        if ((rc.kb.bit.X && rc.kb.bit.SHIFT == 1) || rc.sw1 == 2)
        {
            if_put = 1;
            pick_time_begin = HAL_GetTick();
        }

        if (rc.kb.bit.Q)
        {
            which_storage = 1;
        }
        if (rc.kb.bit.E)
        {
            which_storage = 2;
        }

        arm_moto_control();
        storage_moto_control();
        io_pwm_control();

        osDelay(7);
```

test_custom.c

```c
#include "execute_task.h"
#include "can_device.h"
#include "uart_device.h"
#include "pid.h"
#include "sys.h"
#include "rm_hal_lib.h"
#include "cmsis_os.h"

int16_t test_moto_speed = 0;
int16_t test_moto_current[1];

// test values
uint8_t relay = 0;
uint8_t get_height = 0; // do not set it to 1 if auto mode is enabled

uint8_t if_pick = 0;
uint8_t if_put = 0;
uint8_t which_storage = 1; // 1 is left, 2 is right
uint8_t usart3_recv[4];
int lift_angle = 0;
extern long pick_time_begin;
int time_after_start;
extern int NO_AUTO_MODE; // set it @ startup.c
int CV_offest = 0;

uint8_t ir_sensor[6];
```

```
int16_t arms[3][4];
// CAN1: 0x1ff
//   [X] [0] angle_set [1] speed_set [2] current
//       [0] 5 - fric wheel 1 (2006)
//           init: 0, situ1: 60, situ2: N/A
//       [1] 6 - fric wheel 2 (2006)
//           any situ: - (fric wheel 1)
//       [2] 7 - up&down      (3508)
//           init: N/A, situ1: N/A, situ2: -500
//       [3] 8 - pitch        (3508)
//           init: 60, situ1: 0, situ2: 270
//           max: 1330

int16_t storage[3][2];
// CAN2: 0x200
//   [X] [0] angle_set [1] speed_set [2] current
//       [0] 1 - l/r (2006)
//           l: -100(init), med: 500, r: 1100
//       [1] 2 - somewhat (2006)

int16_t height;

void io_pwm_control(void)
{

    if (relay == 1)
        write_digital_io(7, 1);
    else

        write_digital_io(7, 0);

    if (get_height == 1)
    {
        height = 0;                          // reset no. of blocks
        read_digital_io(1, &ir_sensor[0]); // block 0
        read_digital_io(2, &ir_sensor[1]); // block 1
        read_digital_io(3, &ir_sensor[2]); // block 2
        read_digital_io(4, &ir_sensor[3]); // block 3
        //read_digital_io(5, &ir_sensor[4]); // block 4
        //read_digital_io(6, &ir_sensor[5]); // block 5
        for (int i = 0; i < 4; i++) // get no. of blocks
        {
            if (ir_sensor[i] == 0)
                height++;
        }
    }
}

void arm_moto_control(void)
{
    //舵机控制函数周期设定
    //set_pwm_group_param(PWM_GROUP1, 20000);

    //开启控制端口
    //start_pwm_output(PWM_IO1);
    //set_pwm_param(PWM_IO1,2200);

    time_after_start = (HAL_GetTick() - pick_time_begin) / 100;
    if (!NO_AUTO_MODE)
    { // auto pick
        if (if_pick == 1)
        {

            // pid_init(&pid_arms[0][3], 7000, 0, 50, 0.1, 0.1);

            if (time_after_start < 5)
            {
                if (which_storage == 1)
```

```
        storage[0][0] = -80;
    if (which_storage == 2)
        storage[0][0] = 980;
    pid_init(&pid_arms[0][3], 7000, 0, 15, 0.1, 0.1);
    arms[0][3] = 0; // pitch goes downward
}

if (time_after_start >= 5 && time_after_start < 10)
{
    arms[0][0] = 68; // fric goes, brick picked
}

if (time_after_start >= 10 && time_after_start < 20)
{
    height = 0; // reset no. of blocks

    read_digital_io(1, &ir_sensor[0]); //block 0
    read_digital_io(2, &ir_sensor[1]); //block 1
    read_digital_io(3, &ir_sensor[2]); //block 2
    read_digital_io(4, &ir_sensor[3]); //block 3
    //read_digital_io(5, &ir_sensor[4]); //block 4
    //read_digital_io(6, &ir_sensor[5]); //block 5

    for (int i = 0; i < 4; i++) // get no. of blocks
    {
        if (ir_sensor[i] == 0)
            height++;
    }

    switch (height)
    {
    case 0:
        lift_angle = 560;
        break;
    case 1:
        lift_angle = 620;
        break;
    case 2:
        lift_angle = 770;
        break;
    case 3:
        lift_angle = 920;
        break;
    default:
        if_pick = 0; // 4 blocks in this storage
        if (which_storage == 1)
            which_storage = 2;
        else
            which_storage = 1;
        break;
    }
    arms[0][2] = lift_angle; // up&down goes upward to value set
}

if (time_after_start >= 20 && time_after_start < 40)
{
    pid_init(&pid_arms[0][3], 7000, 0, 15, 0.1, 0.1);
    arms[0][3] = 200; // pitch goes upward to value set
}

if (time_after_start >= 45 && time_after_start < 50)
{
    arms[0][0] = 170; // fric continues going, brick stored
}

if (time_after_start >= 50 && time_after_start < 60)
{
    pid_init(&pid_arms[0][3], 7000, 0, 40, 0.1, 0.1);
    arms[0][3] = 70; // pitch goes back to critical vaule
```

```
    }

    if (time_after_start >= 65)
    {
        lift_angle = 0;
        storage[0][0] = 500;
        arms[0][2] = 0; // up&down resets
        arms[0][0] = 0; // fric goes back
        if_pick = 0;
    }
}

if (if_put == 1)
{
    if (time_after_start >= 0 && time_after_start < 10)
    {
        if (which_storage == 1)
            storage[0][0] = -80;
        if (which_storage == 2)
            storage[0][0] = 980;
    }

    if (time_after_start >= 10 && time_after_start < 20)
    {
        // here is number of blocks

        height = 0; // reset no. of blocks

        read_digital_io(1, &ir_sensor[0]); //block 0
        read_digital_io(2, &ir_sensor[1]); //block 1
        read_digital_io(3, &ir_sensor[2]); //block 2
        read_digital_io(4, &ir_sensor[3]); //block 3
        //read_digital_io(5, &ir_sensor[4]); //block 4
        //read_digital_io(6, &ir_sensor[5]); //block 5

        for (int i = 0; i < 4; i++) // get no. of blocks
        {
            if (ir_sensor[i] == 0)
                height++;
        }

        switch (height)
        {
        case 1:
            lift_angle = 560;
            break;
        case 2:
            lift_angle = 620;
            break;
        case 3:
            lift_angle = 770;
            break;
        case 4:
            lift_angle = 920;
            break;
        default:
            //if_put = 0; // 0 block(s) in this storage
            if (which_storage == 1)
                which_storage = 2;
            else
                which_storage = 1;
            break;
        }
        arms[0][2] = lift_angle; // up&down goes upward to value set
    }

    if (time_after_start >= 20 && time_after_start < 27)
    {
        pid_init(&pid_arms[0][3], 7000, 0, 50, 0, 0);
```

```
                arms[0][3] = 1 70; // pitch goes from critical to value set
            }

            if (time_after_start >= 27 && time_after_start < 33)
            {
                pid_init(&pid_arms[0][3], 7000, 0, 15, 0, 0);
                arms[0][3] = 210; // pitch goes from critical to value set
            }

            if (time_after_start >= 36 && time_after_start < 40)
            {
                arms[0][0] = -68; // fric
            }

            if (time_after_start >= 40 && time_after_start < 47)
            {
                pid_init(&pid_arms[0][3], 7000, 0, 50, 0.1, 0.1);
                arms[0][3] = 70; // pitch goes back to critical vaule
            }
            if (time_after_start >= 47 && time_after_start < 60)
            {
                pid_init(&pid_arms[0][3], 7000, 0, 15, 0.1, 0.1);
                arms[0][3] = 0; // pitch goes back to critical vaule
                pid_init(&pid_arms[0][3], 7000, 0, -1, 0, 0);
                arms[0][2] = 0;  // reset downward
            }

            if (time_after_start >= 60 && time_after_start < 65)
            {
                arms[0][0] = -140; // brick released
            }

            if (time_after_start >= 65 && time_after_start < 70)
            {
                pid_init(&pid_arms[0][3], 7000, 0, 50, 0.1, 0.1);
                storage[0][0] = 500;
                arms[0][3] = 70; // pitch resets
                arms[0][2] = 0;  // up&down resets
                arms[0][0] = 0;  // fric goes back
                if_put = 0;
            }
        }

        if (if_pick == 0 && if_put == 0)
        {

            pid_init(&pid_arms[0][3], 7000, 0, 50, 0.1, 0.1);
            storage[0][0] = 500;
            arms[0][3] = 70; // pitch resets
            arms[0][2] = 0;  // up&down resets
            arms[0][0] = 0;  // fric goes back
        }
    }
    // calculate and send currents to motors
    arms[2][0] = pid_calc(&pid_arms[1][0], moto_arms[0].speed_rpm,
                          pid_calc(&pid_arms[0][0], moto_arms[0].total_angle /
36.0, arms[0][0]));
    arms[2][1] = pid_calc(&pid_arms[1][1], moto_arms[1].speed_rpm,
                          pid_calc(&pid_arms[0][1], moto_arms[1].total_angle /
36.0, -arms[0][0]));
    arms[2][2] = pid_calc(&pid_arms[1][2], moto_arms[2].speed_rpm,
                          pid_calc(&pid_arms[0][2], moto_arms[2].total_angle /
19.2, arms[0][2]));
    arms[2][3] = pid_calc(&pid_arms[1][3], moto_arms[3].speed_rpm,
                          pid_calc(&pid_arms[0][3], moto_arms[3].total_angle /
19.2, arms[0][3]));

    send_arm_moto_current(arms[2]);
}
```

```
void arm_moto_init(void)
{
    // PID init:
    //   [0] angle_set
    //       [0] 5 - fric wheel 1 (2006)
    //       [1] 6 - fric wheel 2 (2006)
    //       [2] 7 - up&down      (3508)
    //       [3] 8 - pitch        (3508)
    pid_init(&pid_arms[0][0], 7000, 0, 35, 0.1, 0);
    pid_init(&pid_arms[0][1], 7000, 0, 35, 0.1, 0);
    pid_init(&pid_arms[0][2], 7000, 0, 26, 0.1, 0);
    pid_init(&pid_arms[0][3], 7000, 0, 15, 0, 0);

    // PID init:
    //   [1] speed_set
    //       [0] 5 - fric wheel 1 (2006)
    //       [1] 6 - fric wheel 2 (2006)
    //       [2] 7 - up&down      (3508)
    //       [3] 8 - pitch        (3508)
    pid_init(&pid_arms[1][0], 7000, 0, 10, 0, 0);
    pid_init(&pid_arms[1][1], 7000, 0, 10, 0, 0);
    pid_init(&pid_arms[1][2], 7000, 0, 10, 0, 0);
    pid_init(&pid_arms[1][3], 10000, 0, 10, 0, 0);

    // relay @ GPIO 7
    set_digital_io_dir(7, IO_OUTPUT);
    // LEDs on camera @ GPIO 8-9
    set_digital_io_dir(8, IO_OUTPUT);
    set_digital_io_dir(9, IO_OUTPUT);
}

void storage_moto_control(void)
{
    storage[2][0] = pid_calc(&pid_storage[1][0], moto_storage[0].speed_rpm,
                             pid_calc(&pid_storage[0][0],
moto_storage[0].total_angle / 36.0, storage[0][0]));
    //storage[2][1] = pid_calc(&pid_storage[1][1], moto_storage[1].speed_rpm,
    //                         pid_calc(&pid_storage[0][1],
moto_storage[1].total_angle / 36.0, storage[0][1]));
    send_storage_moto_current(storage[2]);
}

void storage_moto_init(void)
{
    // PID init:
    //   [0] angle_set
    //       [0] 1 - l/r (2006)
    //       [1] 2 - somewhat (2006)

    pid_init(&pid_storage[0][0], 10000, 0, 15, 0.1, 0.5);
    pid_init(&pid_storage[0][1], 7000, 0, 0.5, 0.1, 0);

    // PID init:
    //   [1] speed_set
    //       [0] 1 - l/r (2006)
    //       [1] 2 - somewhat (2006)

    pid_init(&pid_storage[1][0], 10000, 0, 3, 0.1, 0.5);
    pid_init(&pid_storage[1][1], 7000, 0, 1, 0.1, 0);

    // IRs @ GPIO 1-6
    set_digital_io_dir(1, IO_INPUT);
    set_digital_io_dir(2, IO_INPUT);
    set_digital_io_dir(3, IO_INPUT);
    set_digital_io_dir(4, IO_INPUT);
    //set_digital_io_dir(5, IO_INPUT);
    //set_digital_io_dir(6, IO_INPUT);
```

```
    if (!NO_AUTO_MODE)
        storage[0][0] = 500;
    else
        storage[0][0] = 0;
}
```

10.2 第一阶段算法代码

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <vector>
#include <map>
#include <utility>
#include <stdlib.h>


using namespace std;

const int MAX_N = 8;
const char* colorRefer = "RBGY";
bool flag = false; //mark whether the solution is found

char graph[MAX_N + 5][MAX_N + 5];
char graph_for_print[MAX_N + 5][MAX_N + 5]; //keep the original input unchanged

int gr[] = {-1, 0, 1, 0}, gc[] = {0, -1, 0, 1}; //moving direction arrays
int color[MAX_N][MAX_N] = {0};

//STL
map<char, int> toNum;
vector<pair<int, int> > paint[4 + 5];

int currentIndex = 1;
int prevIndex = 0;

bool vis(int row, int column) {
    //check if element graph[row][column] is visited
    return color[row][column]; //state compression
}

void printGraph() {
    //repaint the graph
    for (int i = 0; i < MAX_N; i++) {
        for (int j = 0; j < MAX_N; j++) {
            graph_for_print[i][j] = graph[i][j];
        }
    }
    for(int i = 0; i < 4; i++) {
        for(int j = 0; j < paint[i].size(); j++) {
            int x = paint[i][j].first, y = paint[i][j].second;
            graph_for_print[x][y] = colorRefer[i];
        }
    }
    printf("\n");
    //print the solution
    for(int i = 0; i < MAX_N; i++) {
        for(int j = 0; j < MAX_N; j++) {
```

```
            printf("%c", graph_for_print[i][j]);
        }
        printf("\n");
    }
}


void dfs(int row, int column, char ch) {

    if (flag) return;

    int newRow, newCol;
    for (int k = 0; k < 4; k++) {
        newRow = row + gr[k];
        newCol = column + gc[k];
        //find the path
        if (0 <= newRow && newRow < MAX_N && 0 <= newCol && newCol < MAX_N
&& !vis(newRow, newCol)) {
            if (graph[newRow][newCol] == 'W') {
                color[newRow][newCol] = 1;
                paint[toNum[ch]].push_back(make_pair(newRow, newCol));
                dfs(newRow, newCol, ch);

                //unmark if the path is invalid
                currentIndex++;
                paint[toNum[ch]].pop_back();
                color[newRow][newCol] = 0;
            } else if (graph[newRow][newCol] == ch && !vis(newRow, newCol)) {
                //reached the same color, path found
                color[newRow][newCol] = 1;
                bool finished = true;

                //find other colors' paths
                for (int i = 0; i < MAX_N && finished; i++) {
                    for (int j = 0; j < MAX_N && finished; j++) {
                        if (graph[i][j] != 'W' && !vis(i, j)) {
                            color[i][j] = 1;
                            dfs(i, j, graph[i][j]);

                            finished = false;
                            color[i][j] = 0;
                        }
                    }
                }

                if (finished) {
                    bool full = true;
                    for (int i = 0; i < MAX_N && full; i++) {
                        for (int j = 0; j < MAX_N && full; j++) {
                            if (!vis(i, j)) {
                                full = false;
                            }
                        }
                    }

                    if (full) {
                        if (currentIndex != prevIndex) {
                            printGraph();
                            flag = true;
```

```
                            return;
                        }
                    } else {
                        color[newRow][newCol] = 0;
                    }
                } else {
                    color[newRow][newCol] = 0;
                }
            }
        }
    }
}

int main() {
    //read input && init
    for(int i = 0; i < MAX_N; i++) {
        for(int j = 0; j < MAX_N; j++) {
            scanf("%c", &graph[i][j]);
        }
        getchar();
    }

    toNum['R'] = 0;
    toNum['B'] = 1;
    toNum['G'] = 2;
    toNum['Y'] = 3;

    //find solution
    for(int i = 0; i < MAX_N && !flag; i++) {
        for(int j = 0; j < MAX_N && !flag; j++) {
            if(graph[i][j] != 'W') {
                color[i][j] = 1;
            dfs(i, j, graph[i][j]);
            }
        }
    }

    if(!flag) printf("\nSolution Not Found.\n");
    return 0;
}
```

10.3 图像识别代码

```
#include <stdio.h>
#include <iostream>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/core/core.hpp>
#include <memory>
#include <string.h>
#include "trace.h"
#include “trace.cpp”

using namespace std;
```

```
using namespace cv;

extern int iLowH, iLowS, iLowV;
extern int iHighH, iHighS, iHighV;

int angle_x[4];
int angle_y[4];
void showpeacture(Mat peacture);

//初始化尺寸
const int y = 480;
const int x = 640;

//初始化变量
int RGB[3][y][x];//RGB == 012
int R = 0, G = 1, B = 2, Y = 3;
int HSV[3][y][x];//HSV == 012
int H = 0, S = 1, V = 2;
bool rgb_color[y][x][4] = { 0 };//RGBY = 0123
bool hsv_color[y][x][4] = { 0 };
Mat matSrc, matDst_RGB, matDst_HSV;
Mat matDst_GRAY[6];
Mat normImage;//归一化后的图
Mat scaledImage;//线性变换后的八位无符号整型的图

int thresh = 30; //当前阈值
int max_thresh = 175; //最大阈值

Scalar Red(0, 0, 255), Green(0, 0, 255), Blue(0, 0, 255), Yellow(0, 0, 255);

bool LR;
vector<int> centerX;
void drawRect(const RotatedRect &box, Mat &dst, vector<int> *centerX) {
    Point2f vertex[4];
    box.points(vertex);
    //绘制出最小面积的包围矩形
    line(dst, vertex[0], vertex[1], Scalar(100, 200, 211), 3, LINE_AA);
    line(dst, vertex[1], vertex[2], Scalar(100, 200, 211), 3, LINE_AA);
    line(dst, vertex[2], vertex[3], Scalar(100, 200, 211), 3, LINE_AA);
    line(dst, vertex[3], vertex[0], Scalar(100, 200, 211), 3, LINE_AA);
    //绘制中心的光标
    Point s, l, r, u, d;
    s.x = (vertex[0].x + vertex[2].x) / 2.0;
    s.y = (vertex[0].y + vertex[2].y) / 2.0;
    l.x = s.x - 10;
    l.y = s.y;

    r.x = s.x + 10;
```

```
    r.y = s.y;

    u.x = s.x;
    u.y = s.y - 10;

    d.x = s.x;
    d.y = s.y + 10;
    line(dst, l, r, Scalar(100, 200, 211), 2, LINE_AA);
    line(dst, u, d, Scalar(100, 200, 211), 2, LINE_AA);

    //if ((s.x) - 320)-0

    //各项参数显示 I see
    char move1[25];
    char move2[20];
    sprintf(move1, "x: %d y: %d d: %d", int(s.x) - 320, int(s.y) - 240,
int(box.angle + 90));
    sprintf(move2, "h: %d w: %d", int(box.size.height), int(box.size.width));

    putText(dst, move1, Point(s.x, s.y), FONT_HERSHEY_PLAIN, 1, Scalar(124, 252,
0), 1, 8, false);
    putText(dst, move2, Point(s.x, s.y + 15), FONT_HERSHEY_PLAIN, 1, Scalar(124,
252, 0), 1, 8, false);

    centerX->push_back(s.x - 320);
}

int main(int argc, char* argv[]) {
    /*图像分辨率定义初始化*/
    int size = y * x;
    //初始化摄像头并读取->matDst
    VideoCapture cap(0);
    if (!cap.isOpened()) {
        return -1;
    }
    /*设置摄像头参数
    capture.set(CV_CAP_PROP_FRAME_WIDTH, 1080);//宽度
    capture.set(CV_CAP_PROP_FRAME_HEIGHT, 960);//高度
    capture.set(CV_CAP_PROP_FPS, 30);//帧数
    capture.set(CV_CAP_PROP_BRIGHTNESS, 1);//亮度 1
    capture.set(CV_CAP_PROP_CONTRAST,40);//对比度 40
    capture.set(CV_CAP_PROP_SATURATION, 50);//饱和度 50
    capture.set(CV_CAP_PROP_HUE, 50);//色调 50
    capture.set(CV_CAP_PROP_EXPOSURE, 50);//曝光 50
    */
    //打印摄像头参数
    printf("width = %.2f\n", cap.get(CV_CAP_PROP_FRAME_WIDTH));
    printf("height = %.2f\n", cap.get(CV_CAP_PROP_FRAME_HEIGHT));
```

```
    printf("contrast = %.2f\n", cap.get(CV_CAP_PROP_CONTRAST));
    printf("saturation = %.2f\n", cap.get(CV_CAP_PROP_SATURATION));
    printf("exposure = %.2f\n", cap.get(CV_CAP_PROP_EXPOSURE));

    while (1) {
        cap >> matSrc;
        resize(matSrc, matDst_RGB, Size(x, y), 0, 0, INTER_NEAREST);//获取 RGB 图像
并变换尺寸
        cvtColor(matDst_RGB, matDst_HSV, COLOR_BGR2HSV);//RGB->HSV

        for (int i = 0; i < 4; i++) {
            //int i = 1;
         ////////////////////////////////////////////
         chose_color(i);//选择要识别的颜色/RED=0,GREEN=1,BLUE=2,YELLOW=3;
         ////////////////////////////////////////////
            inRange(matDst_HSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH,
iHighS, iHighV), matDst_GRAY[i]);
            if (iHighH == 180) {
                inRange(matDst_HSV, Scalar(0, iLowS, iLowV), Scalar(10, iHighS,
iHighV), matDst_GRAY[4]);
                for (int r = 0; r < y; r++) {
                    for (int j = 0; j < x; j++) {
                        if (matDst_GRAY[4].at<uchar>(r, j) == 255) {
                            matDst_GRAY[i].at<uchar>(r, j) = 255;
                        }
                    }
                }
            }

            Mat element = getStructuringElement(MORPH_RECT, Size(5, 5));
            Mat element2 = getStructuringElement(MORPH_RECT, Size(5, 5));
            //开操作
            morphologyEx(matDst_GRAY[i], matDst_GRAY[i], MORPH_OPEN, element);
            //闭操作 (连接一些连通域)
            morphologyEx(matDst_GRAY[i], matDst_GRAY[i], MORPH_CLOSE, element2);
            blur(matDst_GRAY[i], matDst_GRAY[i], Size(2, 2));

            Mat matDst_GRAY_1 = Mat::zeros(matDst_GRAY[i].rows,
matDst_GRAY[i].cols, CV_8UC3);
            vector<vector<Point>> contours;
            vector<Vec4i> hierarchy;
            findContours(matDst_GRAY[i], contours, hierarchy, RETR_CCOMP,
CHAIN_APPROX_SIMPLE);
            int index = 0;

            constexpr auto sizeLimit = 8000.0;
            if (contours.size() > 0) {
                //RotatedRect max = minAreaRect(Mat(contours[0]));
```

```
                for (auto contour : contours) {
                    auto rect = minAreaRect(Mat(contour));
                    if (rect.size.width * rect.size.height > sizeLimit) {
                        drawRect(rect, matDst_RGB, &centerX);
                    }
                }
            }
        }
        int minDist = 500;
        if (!centerX.empty()) {
            for (vector<int>::const_iterator iter = centerX.cbegin(); iter != 
centerX.cend(); iter++)
            {
                if ((abs(*iter)) < abs(minDist)) minDist = *iter;
            }

            if (minDist > 8) { cout << 2 << endl; }
            else if (minDist < -8) { cout << 0 << endl; }
            else { cout << 1 << endl; }

        }
        //lmin = 0;
        //rmin = 0;
        centerX.clear();

        //see_black_line(matDst_HSV, matDst_RGB, matDst_GRAY[5]);
        //imshow("GRAY", matDst_GRAY[3]);
        //imshow("HSV", matDst_HSV);
        imshow("RGB", matDst_RGB);
        if (waitKey(50) == 27)
            break;
    }
    return 0;
}

void showpeacture(Mat peacture) {
    namedWindow("MyWindow", CV_WINDOW_AUTOSIZE);
    //在 MyWindow 的窗中中显示存储在 img 中的图片
    imshow("MyWindow", peacture);
    //等待直到有键按下
    waitKey(0);
    //销毁 MyWindow 的窗口
    destroyWindow("MyWindow");
}
```

2.trace.cpp

```
#include "opencv2/opencv.hpp"
#include "iostream"
#include "trace.h"
```

```
using namespace cv;
using namespace std;

int iLowH, iLowS, iLowV;
int iHighH, iHighS, iHighV;

void chose_color(int color) {
    //RED
    if (color == 0) {
        iLowH = 156; iHighH = 180;
        iLowS = 140; iHighS = 255;
        iLowV = 0; iHighV = 255;
    }
    //GREEN
    if (color == 1) {
        iLowH = 63; iHighH = 80;
        iLowS = 130; iHighS = 255;
        iLowV = 0; iHighV = 255;
    }
    //BLUE
    if (color == 2) {
        iLowH = 100; iHighH = 124;
        iLowS = 125; iHighS = 255;
        iLowV = 0; iHighV = 255;
    }
    //YELLOW
    if (color == 3) {
        iLowH = 14; iHighH = 23;
        iLowS = 130; iHighS = 255;
        iLowV = 0; iHighV = 255;
    }
}

void see_black_line(Mat input, Mat output, Mat GRAY) {
    inRange(input, Scalar(20, 0, 0), Scalar(100, 255, 150), GRAY);
    Canny(GRAY, GRAY, 3, 9, 3);

    vector<Vec4i> Lines;
    HoughLinesP(GRAY, Lines, 1, CV_PI / 360, 170, 30, 15);
    for (size_t i = 0; i < Lines.size(); i++)
    {
        line(output, Point(Lines[i][0], Lines[i][1]), Point(Lines[i][2],
Lines[i][3]), Scalar(0, 0, 255), 2, 8);
    }
}
```

3.trace.h

```
#ifndef TRACE_H
#define TRACE_H
```

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/core/core.hpp>

void chose_color(int color);
void see_black_line(cv::Mat input, cv::Mat output, cv::Mat GRAY);

#endif // !TRACE_H
```

10.4 UI 代码

MainPage.xaml.cs

```
using Windows.UI;
using Windows.UI.Core;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Controls;
using Windows.ApplicationModel.DataTransfer;
//using static RoboMaster.CellModel;
using static Windows.UI.Colors;
//using static RoboMaster.CellModel.What;

using System;
using Windows.UI.ViewManagement;
//using Windows.ApplicationModel.Core;



namespace RoboMaster {
    public sealed partial class MainPage {
        private readonly Button[] _buttons = new Button[64];
        //private readonly CellModel _cells = new CellModel();
        private int[] colors = { 0, 0, 0, 0 };
        private bool readClip = false;

        private AcrylicBrush myWhite = new AcrylicBrush();
        private AcrylicBrush myRed = new AcrylicBrush();
        private AcrylicBrush myGreen = new AcrylicBrush();
        private AcrylicBrush myBlue = new AcrylicBrush();
        private AcrylicBrush myYellow = new AcrylicBrush();

        public MainPage() {
            InitializeComponent();

            ApplicationViewTitleBar titleBar = ApplicationView.
                                        GetForCurrentView().TitleBar;
            titleBar.ButtonBackgroundColor = Colors.Transparent;
            titleBar.ButtonInactiveBackgroundColor = Colors.Transparent;

            myWhite.BackgroundSource = AcrylicBackgroundSource.HostBackdrop;
            myWhite.TintColor = Color.FromArgb(240, 255, 255, 255);
            myWhite.FallbackColor = Color.FromArgb(200, 220, 220, 220);
            myWhite.TintOpacity = 0.8;

            myRed.BackgroundSource = AcrylicBackgroundSource.HostBackdrop;
            myRed.TintColor = Red;
            myRed.FallbackColor = OrangeRed;
            myRed.TintOpacity = 0.5;

            myGreen.BackgroundSource = AcrylicBackgroundSource.HostBackdrop;
            myGreen.TintColor = ForestGreen;
```

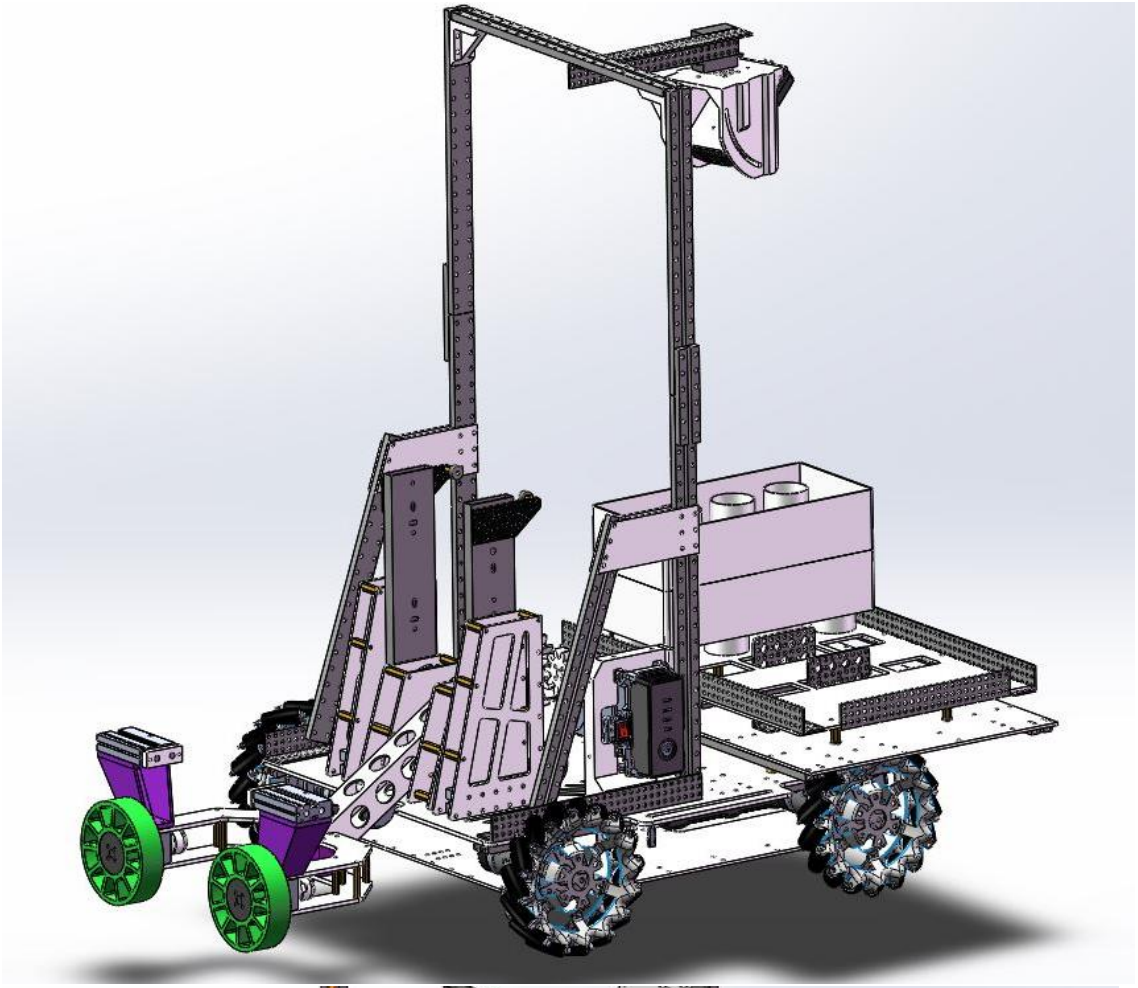
```
    myGreen.FallbackColor = ForestGreen;
    myGreen.TintOpacity = 0.5;

    myBlue.BackgroundSource = AcrylicBackgroundSource.HostBackdrop;
    myBlue.TintColor = RoyalBlue;
    myBlue.FallbackColor = RoyalBlue;
    myBlue.TintOpacity = 0.5;

    myYellow.BackgroundSource = AcrylicBackgroundSource.HostBackdrop;
    myYellow.TintColor = Gold;
    myYellow.FallbackColor = Gold;
    myYellow.TintOpacity = 0.5;


    for (var i = 0; i < 64; ++i) {
        var position = (i / 8, i % 8);
        var button = DefaultButton();
        button.Tag = position;
        button.Background = myWhite;
        button.Click += ObjectClick;

        MainGrid.Children.Add(button);
        Grid.SetRow(button, position.Item1 + 1);
        Grid.SetColumn(button, position.Item2);

        _buttons[i] = button;
    }
    Mode = myRed;
    Title.Foreground = new SolidColorBrush(Black);
}


private static Button DefaultButton()
    => new Button
    {
        FontFamily = new FontFamily("Segoe UI"),
        FontSize = 20,
        Margin = new Thickness(4),
        Background = new AcrylicBrush(),
        HorizontalAlignment = HorizontalAlignment.Stretch,
        VerticalAlignment = VerticalAlignment.Stretch,
    };

void Do(DispatchedHandler action) {
    var act = Dispatcher.RunAsync(CoreDispatcherPriority.Low, action);
}

private Brush _mode;

private Brush Mode {
    get => _mode;
    set {
        Do(() => Title.Foreground = value);
        _mode = value;
        if (_mode == myWhite)
            for (int i = 0; i < 4; i++)
                colors[i] = -10;

    }
}

private void ObjectClick(object sender, RoutedEventArgs e) {
    if (colors[0] == 0)
        TimeCalc();
    var button = (Button) sender;
    var position = ((int, int)) button.Tag;

    if (_mode != myWhite)
    {
```

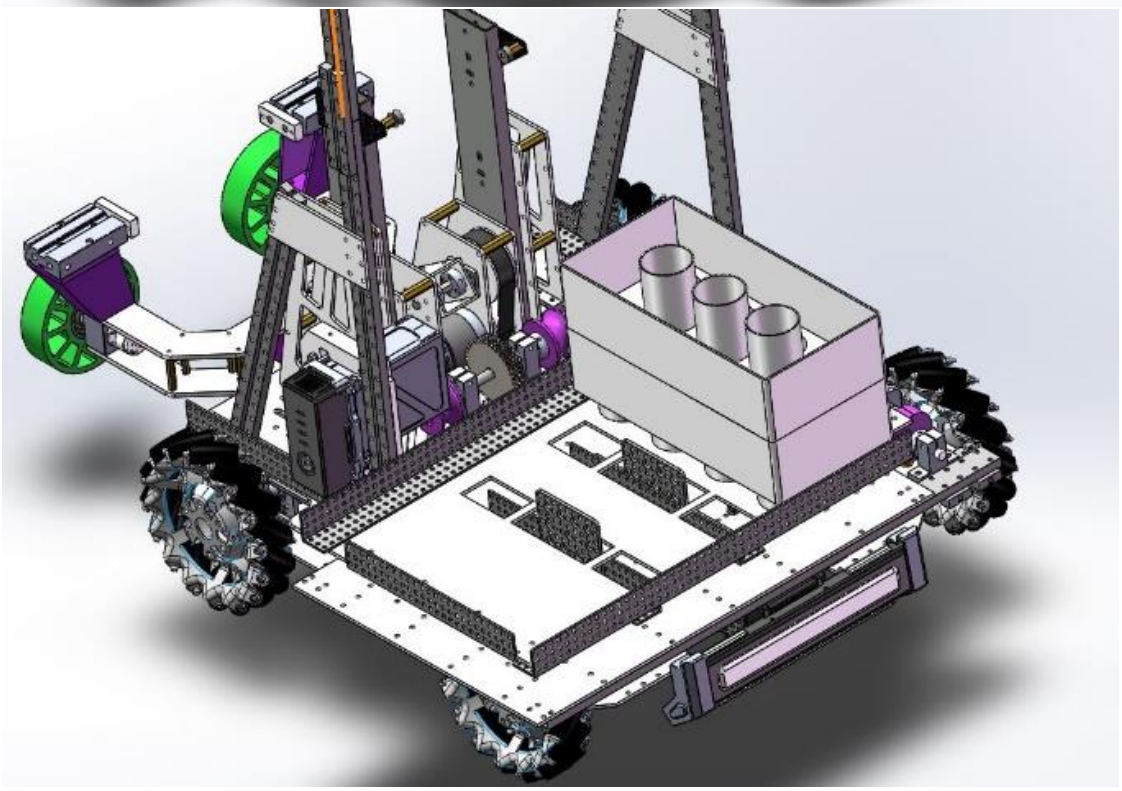
```
            Title.Foreground = _mode;
            if (_mode == myRed)
            {
                colors[0]++;
                Title.Text = "EastFlow";
                button.Content = "✡";
                if (colors[0] > 4)
                {
                    _mode = myGreen;
                }
            }
            if (_mode == myGreen)
            {
                colors[1]++;
                Title.Text = "EastFlow";
                button.Content = "🏯";
                if (colors[1] > 4)
                {
                    _mode = myBlue;
                }
            }
            if (_mode == myBlue)
            {
                colors[2]++;
                Title.Text = "EastFlow" ;
                button.Content = "🕌";
                if (colors[2] > 4)
                {
                    _mode = myYellow;
                }
            }
            if (_mode == myYellow)
            {
                colors[3]++;
                Title.Text = "EastFlow" ;
                button.Content = "🏛";
                if (colors[3] > 3)
                {
                    GenerateClip();
                }
            }

            button.Background = _mode;
        }
        else
        {
            button.Content = "";
            Title.Text = "EastFlow";
            Title.Foreground = new SolidColorBrush(Black);
            button.Background = myWhite;
        }
    }

    private void AddRed_Click(object sender, RoutedEventArgs e)
                                                    => Mode = myRed;
    private void AddYellow_Click(object sender, RoutedEventArgs e)
                                                     => Mode = myYellow;
    private void AddGreen_Click(object sender, RoutedEventArgs e)
                                                    => Mode = myGreen;
    private void AddBlue_Click(object sender, RoutedEventArgs e)
                                                    => Mode = myBlue;
    private void AddGray_Click(object sender, RoutedEventArgs e)
                                                     => Mode = myWhite;

    private void Reset_Click(object sender, RoutedEventArgs e) {
        //_cells.ClearAll();
        Mode = myRed;
        Title.Foreground = new SolidColorBrush(Black);
        Title.Text = "EastFlow";
```

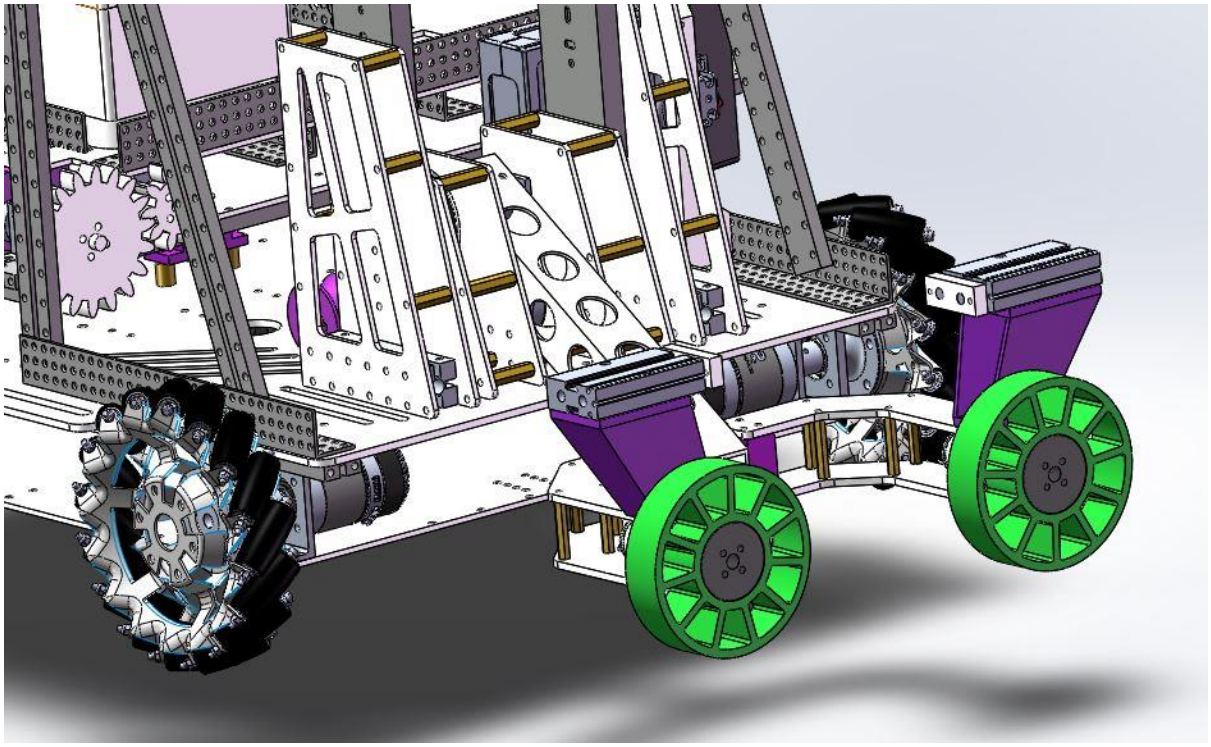
```
        colors[0] = 0;
        colors[1] = 0;
        colors[2] = 0;
        colors[3] = 0;
        foreach (var button in _buttons)
        {
            button.Background = myWhite;
            button.Content = "";
        }

    }

    private void Generate(object sender, RoutedEventArgs e)
    {
        GenerateClip();
    }


    private async void Load(object sender, RoutedEventArgs e)
    {
        char[] block = new char[64];
        string readstr = "", str = "";
        DataPackageView con = Clipboard.GetContent();
        if (con.Contains(StandardDataFormats.Text))
        {
            readstr = await con.GetTextAsync();
        }
        bool city = readstr.Contains("W");
        if (readstr.Length >= 8 * 8)
        {
            for (int i = 0; i < 8; i++)
                str += readstr.Substring(10 * i, 8);

            block = str.ToCharArray();

            for (int i = 0; i < 64; i++)
            {
                if (_buttons[i].Content == null ||
                        _buttons[i].Content == "")
                    switch (block[i])
                    {
                        case 'R':
                            _buttons[i].Background = myRed;
                            if (city)
                                _buttons[i].Content = "✡";
                            break;
                        case 'G':
                            _buttons[i].Background = myGreen;
                            if (city)
                                _buttons[i].Content = "🏯";
                            break;
                        case 'B':
                            _buttons[i].Background = myBlue;
                            if (city)
                                _buttons[i].Content = "🏰";
                            break;
                        case 'Y':
                            _buttons[i].Background = myYellow;
                            if (city)
                                _buttons[i].Content = "🏛";
                            break;
                        case 'W':
                            _buttons[i].Background = myWhite;
                            break;
                    }
            }
        }
    }
```

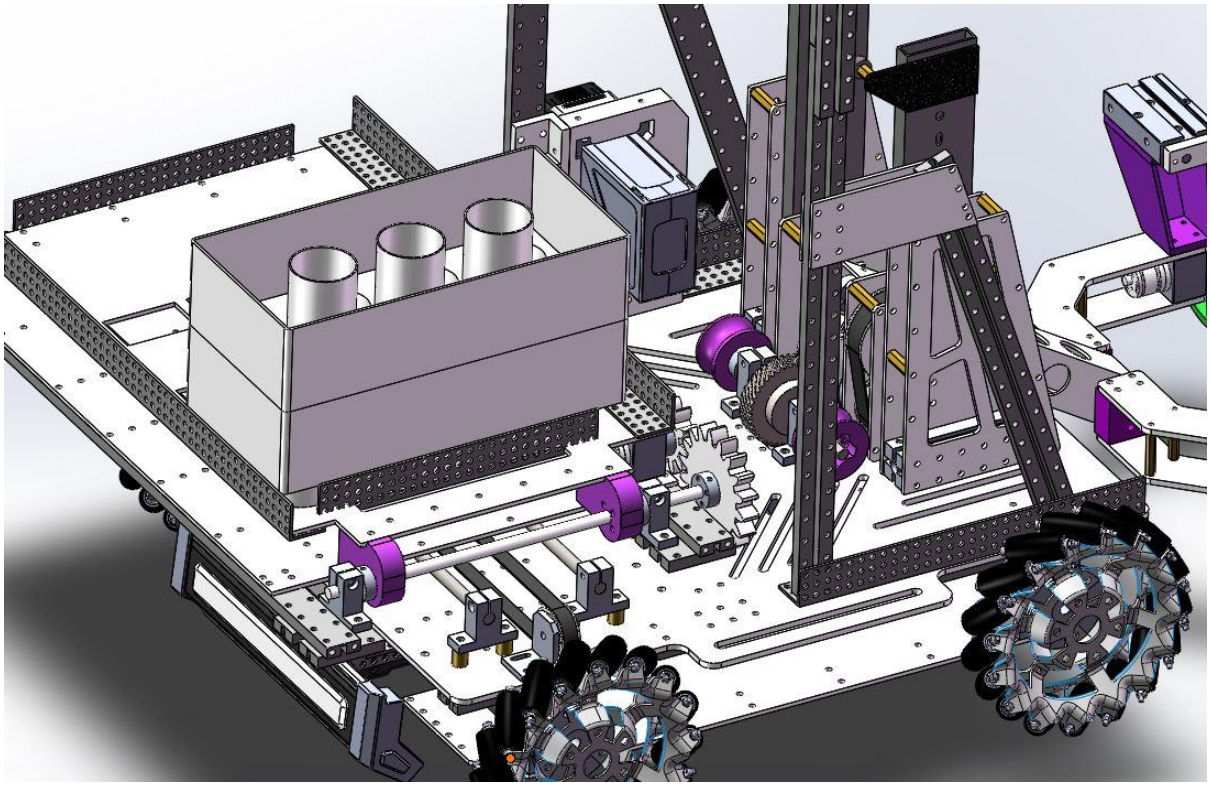
```
    private async void GenerateClip()
    {
        string text = "";
        for (int i = 0; i < 64; i++)
        {
            if (_buttons[i].Content == "")
                text += "B";
            else if (_buttons[i].Content == "")
                text += "G";
            else if (_buttons[i].Content == "")
                text += "Y";
            else if (_buttons[i].Content == "")
                text += "R";
            else
                text += "W";

            if ((i + 1) % 8 == 0)
                text += "\r\n";
        }

        DataPackage dataPackage = new DataPackage();
        dataPackage.SetText(text);
        Clipboard.SetContent(dataPackage);
        readClip = true;
        Title.Foreground = new SolidColorBrush(Black);
        Title.Text = "COPIED";
    }


    private async void TimeCalc()
    {
        int time = 0;
        char[] block = new char[64];
        string readstr = "", str = "";

        while (true)
        {
            await System.Threading.Tasks.Task.Run(() =>
            {
                System.Threading.Thread.Sleep(1000);
            });

            time++;
            Time.Text = time + "s";
        }
    }
}
```

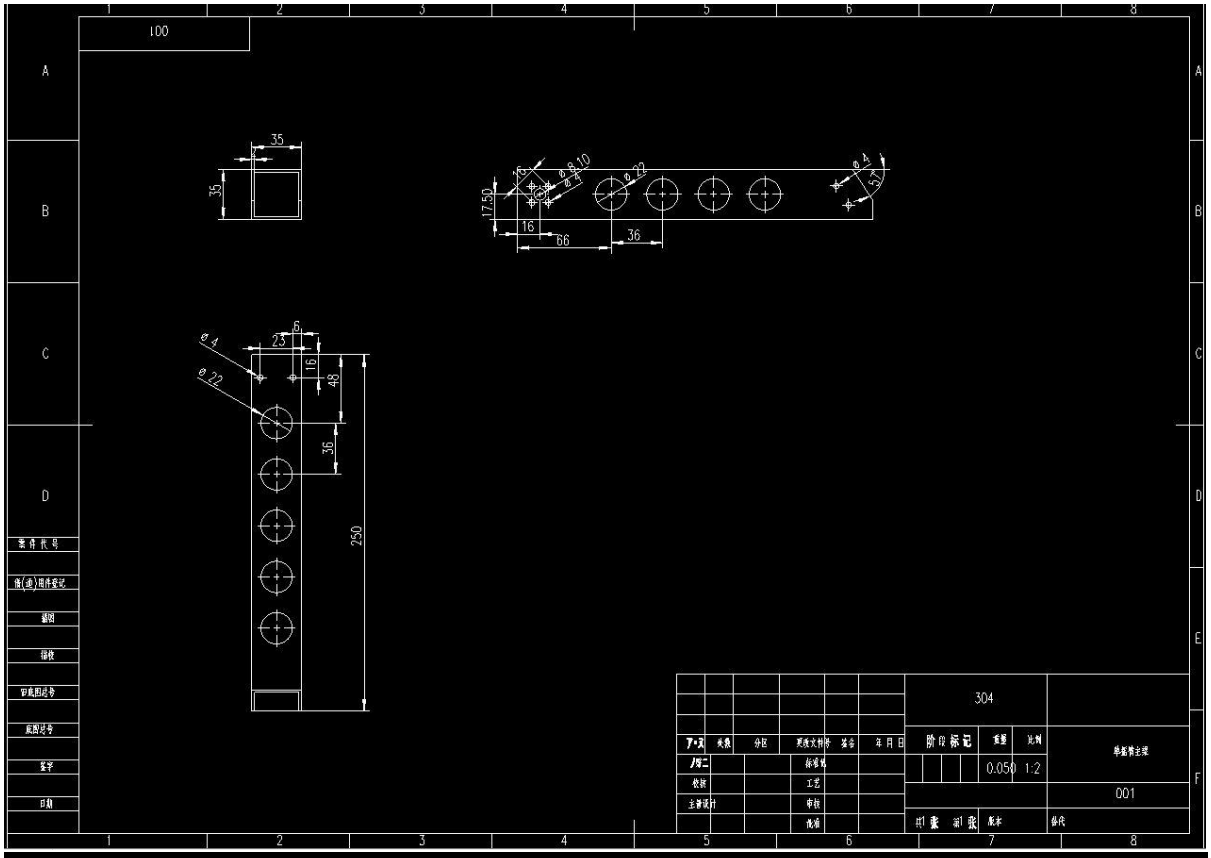
10.5 机械制图

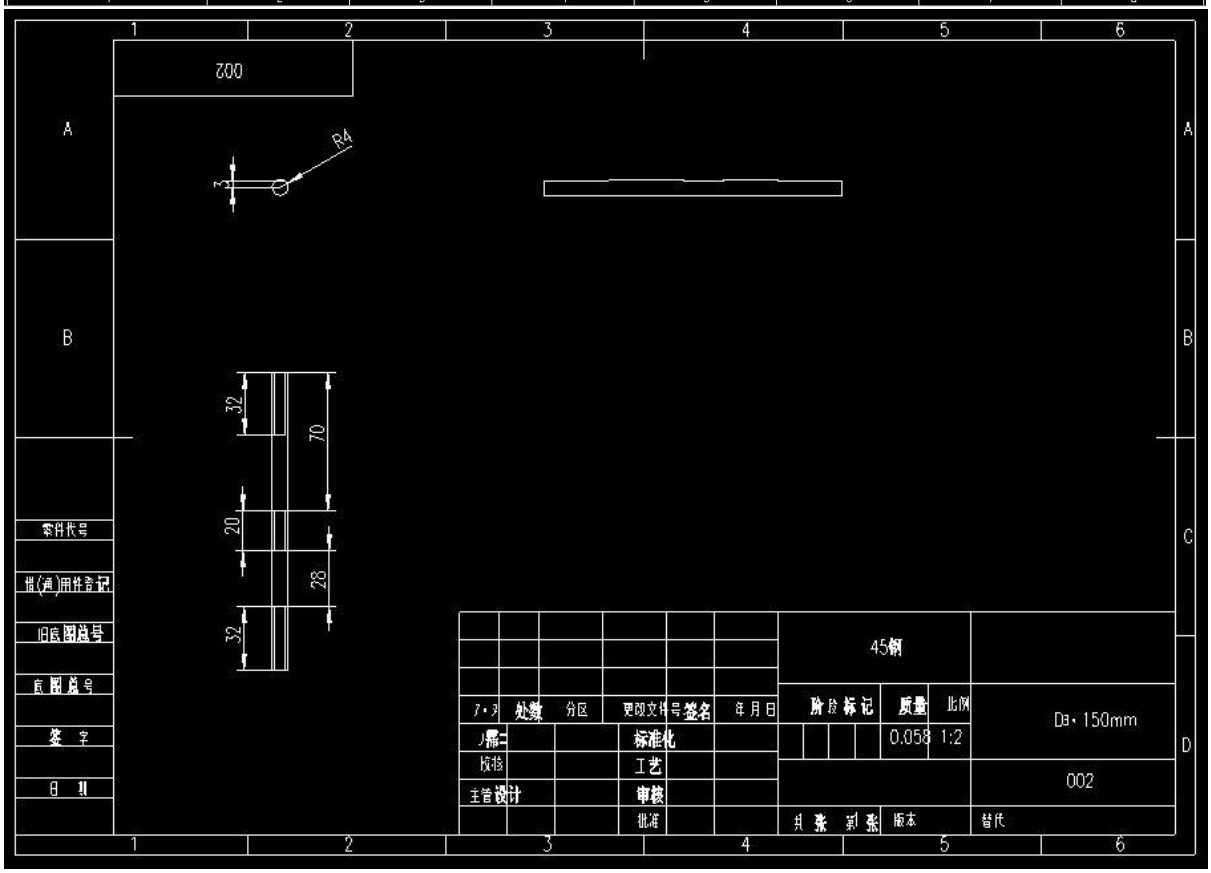


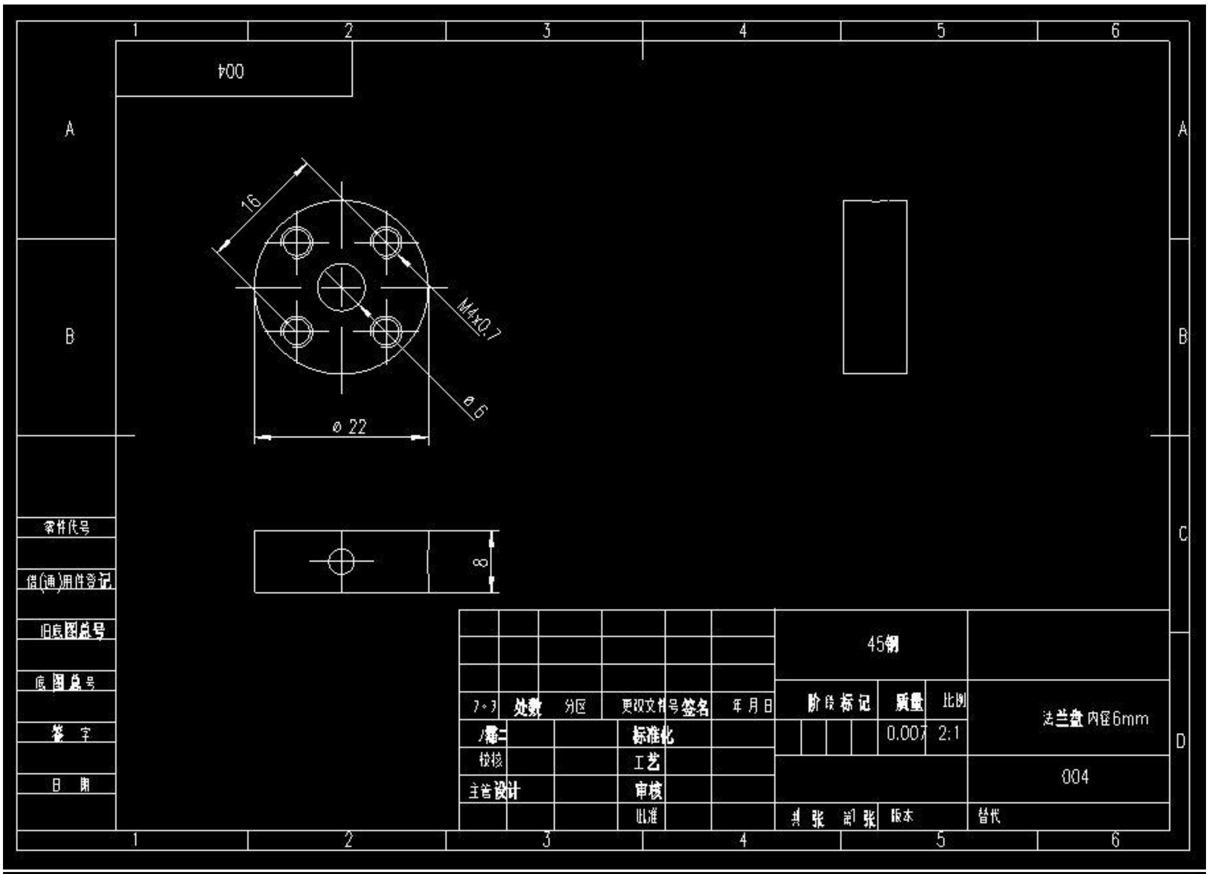
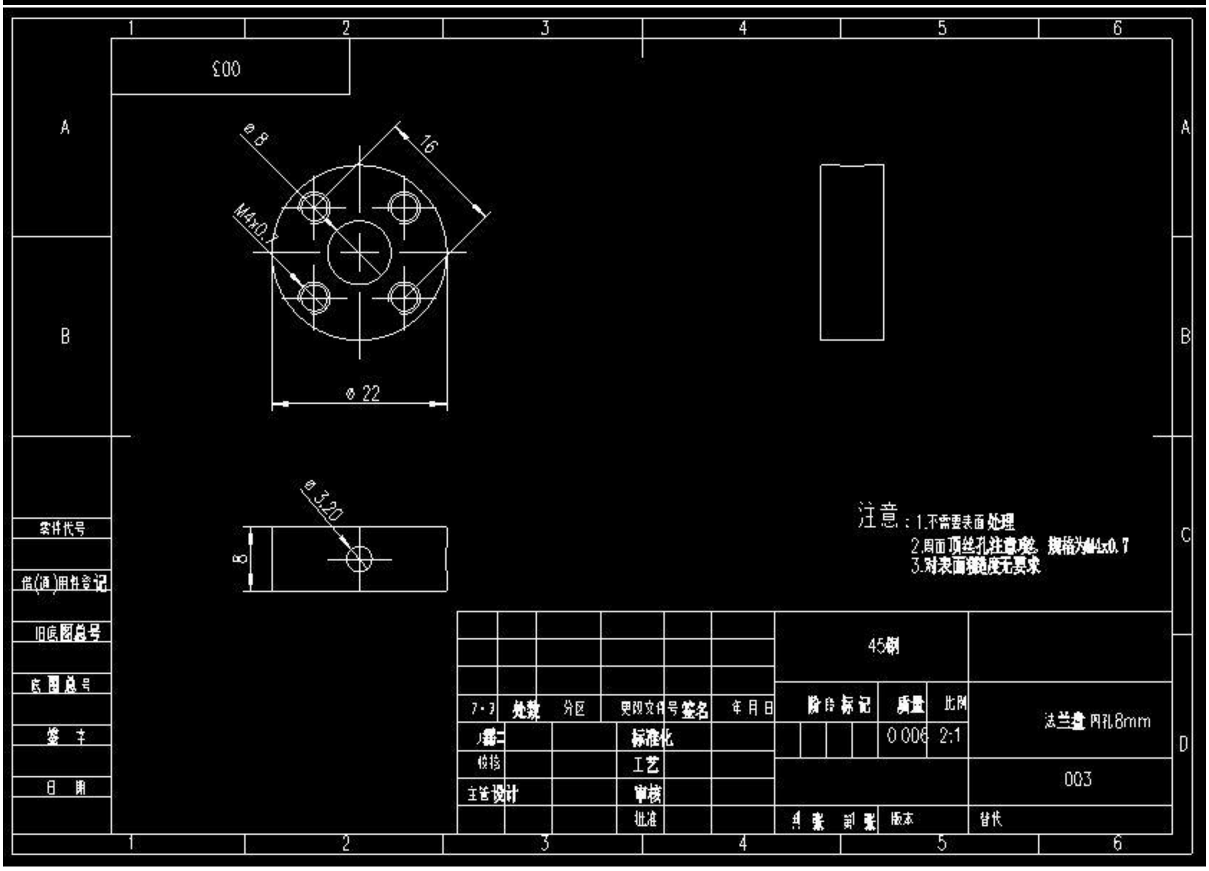


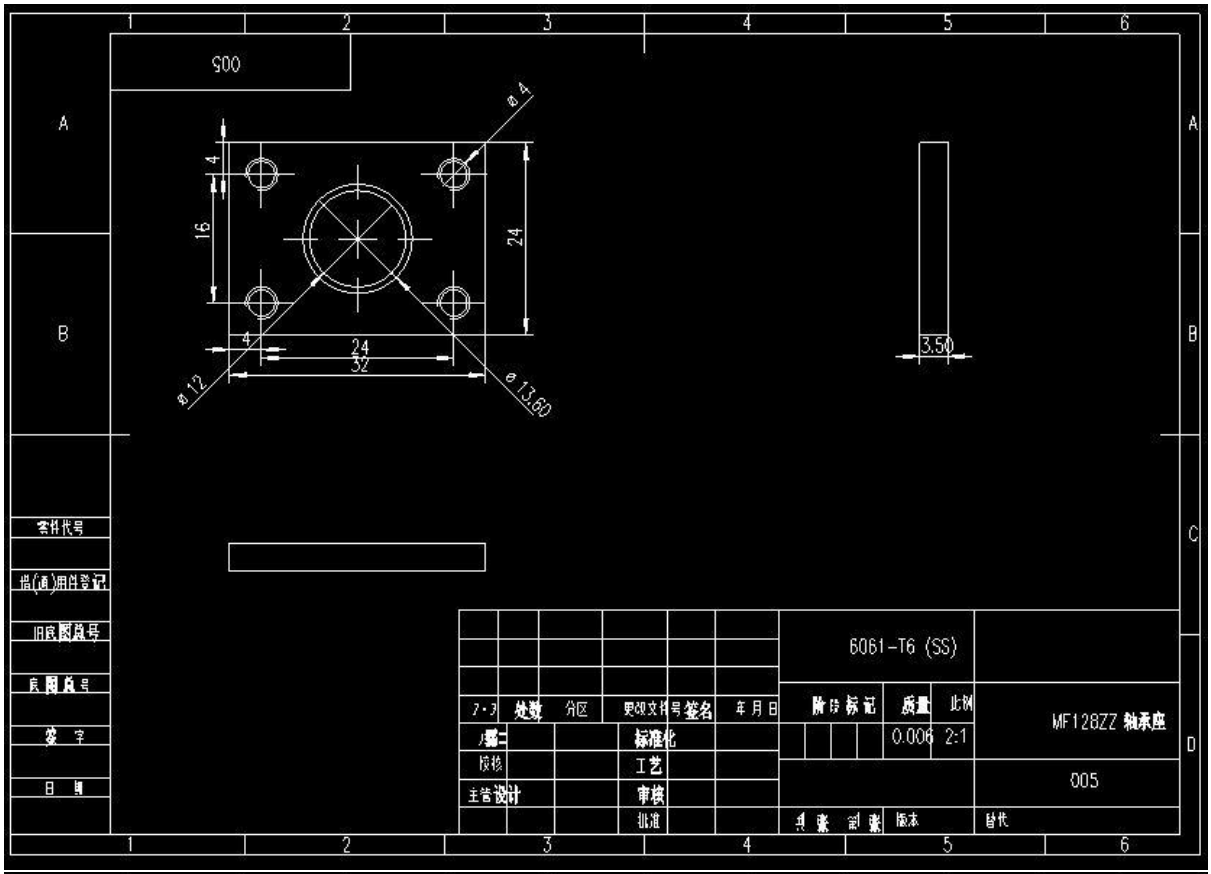


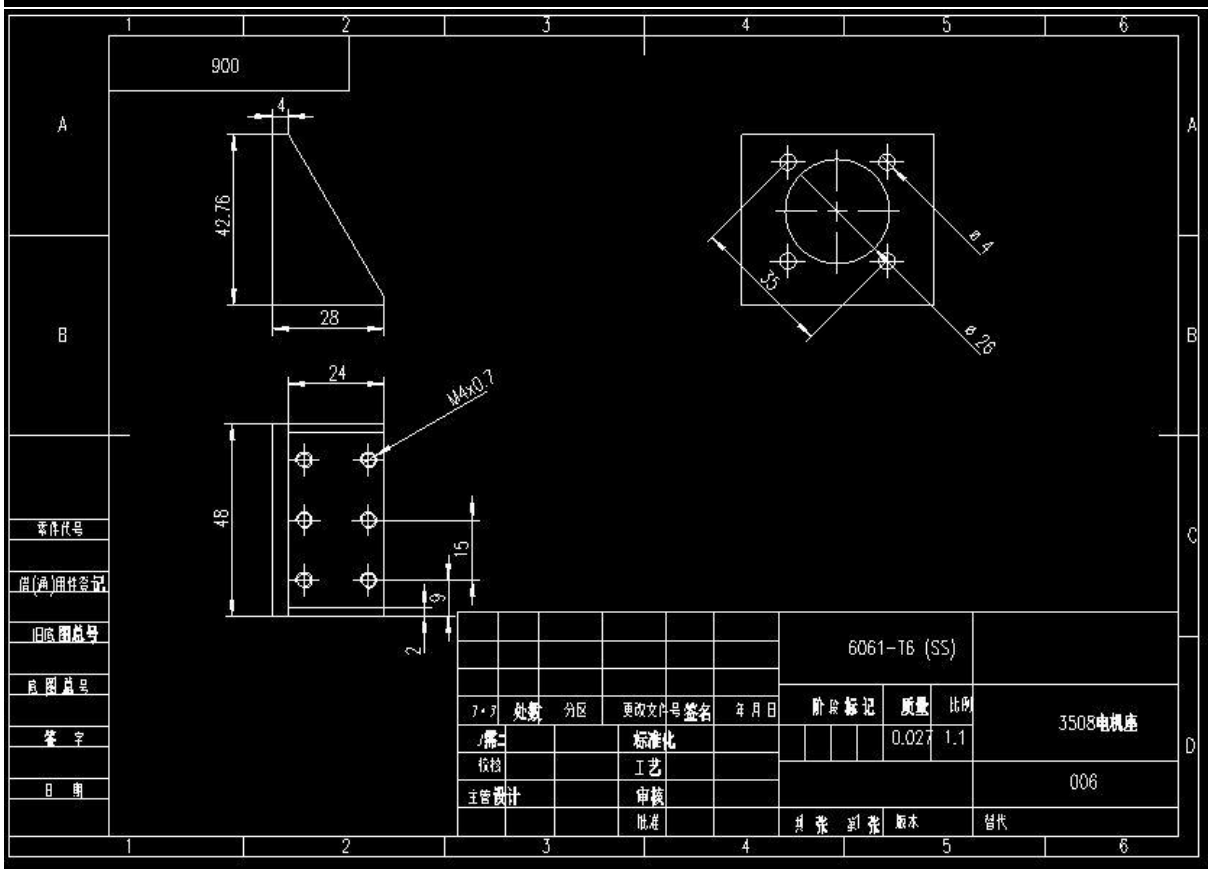



100
35
35
17.5
16
66
36
6
23
16
48
36
250
304
001


002
R4
3
32
70
20
28
32
45钢
Ø150mm
002


004
16
M4x0.7
ø6
ø22
8
零件代号
借(通)用件登记
旧底图总号
底图总号
签字
日期
45钢
阶段标记
质量
比例
0.007
2:1
法兰盘 内径6mm
004
处数
分区
更改文件号
签名
年月日
标准化
校核
工艺
主管设计
审核
批准
共张
第张
版本
替代

003
ø8
16
M4x0.7
ø22
ø3.20
8
注意：1.不需要表面处理
2.周面顶丝孔注意攻丝，规格为M4x0.7
3.对表面粗糙度无要求
零件代号
借(通)用件登记
旧底图总号
底图总号
签字
日期
45钢
阶段标记
质量
比例
0.006
2:1
法兰盘 内孔8mm
003
处数
分区
更改文件号
签名
年月日
标准化
校核
工艺
主管设计
审核
批准
共张
第张
版本
替代


S00
ø 4
4
16
24
4
24
32
ø 12
ø 13.60
3.50
6061-T6 (SS)
MF128ZZ 轴承座
0.006
2:1
005


900
4
42.76
28
24
M4x0.7
48
15
9
2
ø 4
33
ø 26
6061-T6 (SS)
3508电机座
0.027
1.1
006