# 一只旗兵技术文档

2018RoboGame
勇攀高峰组

2018/10/31

## 0.1 机械部分

### 0.1.1 整体设计

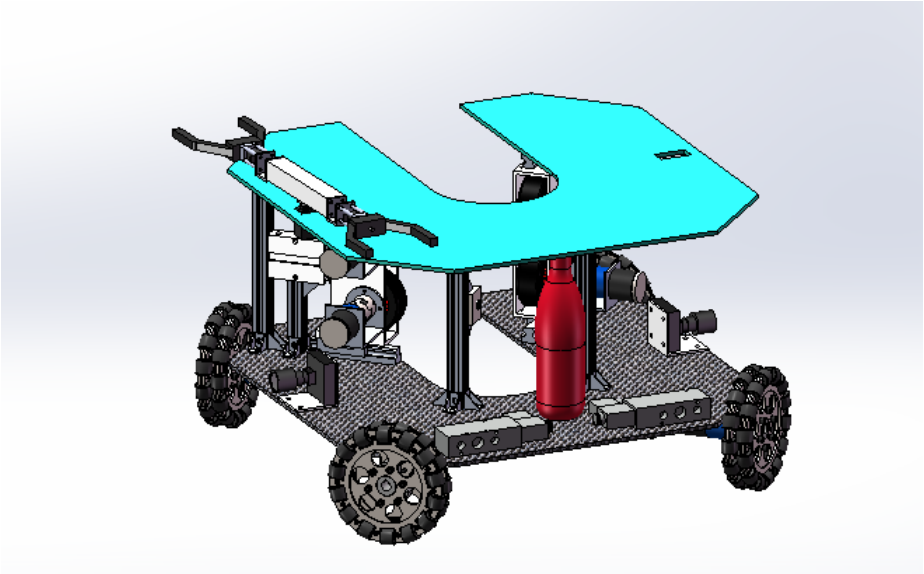


图 1: 机器人整体概览

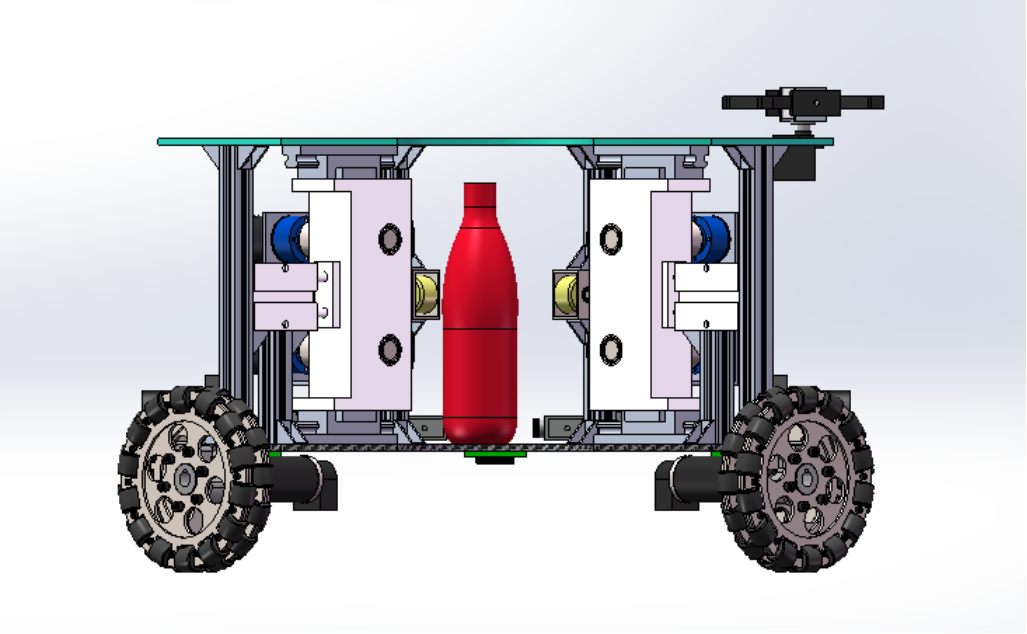


图 2: 机器人正视图

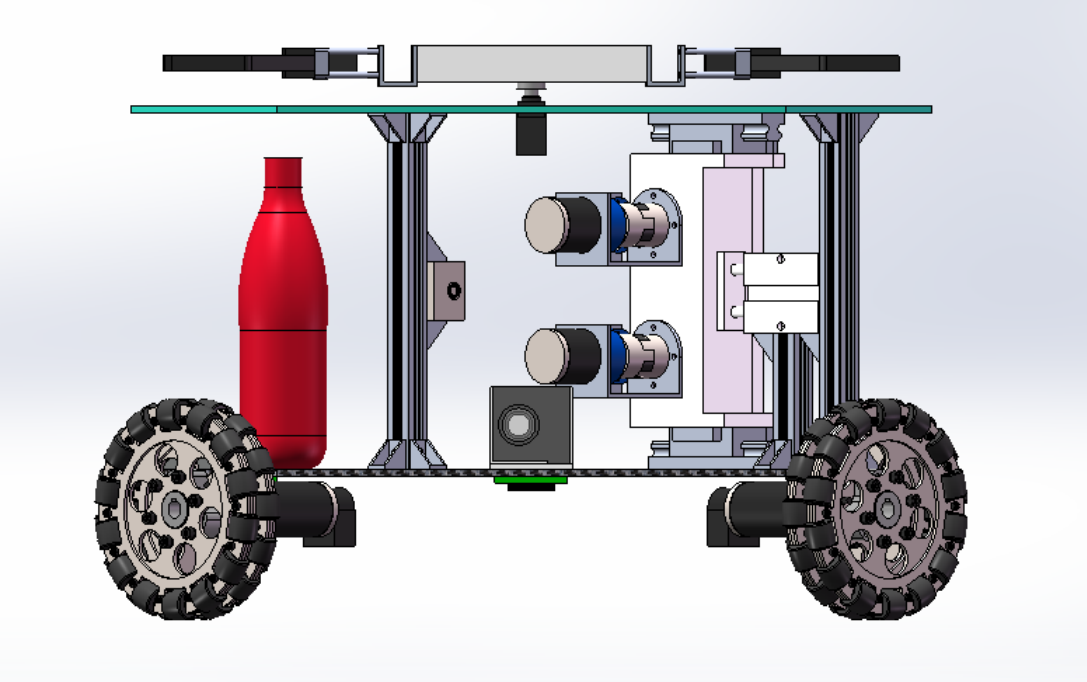


图 3: 机器人侧视图

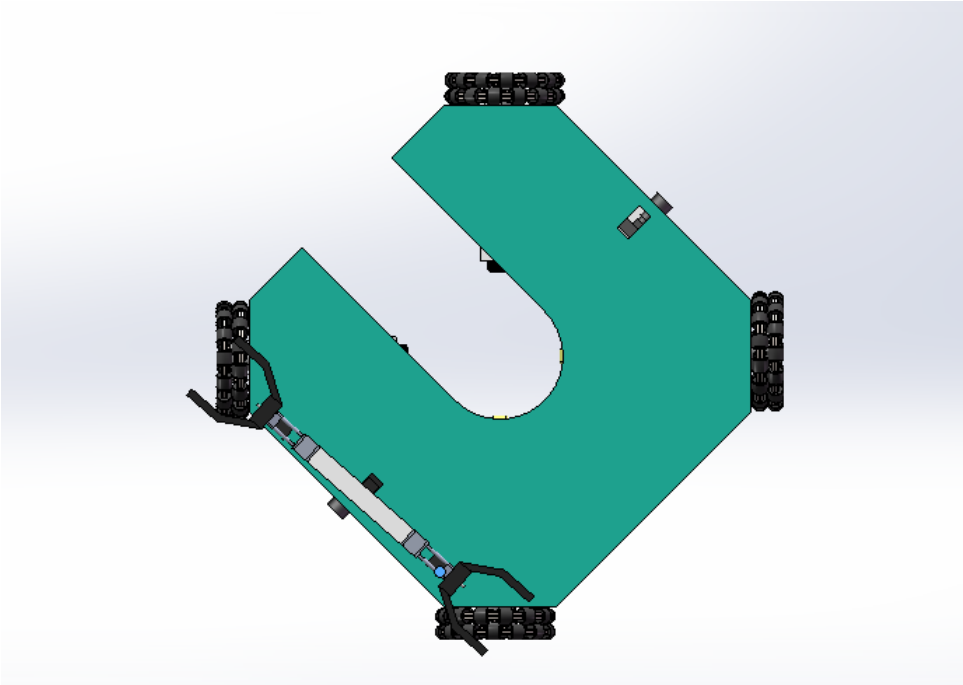


图 4: 机器人俯视图

### 0.1.2 关键模块

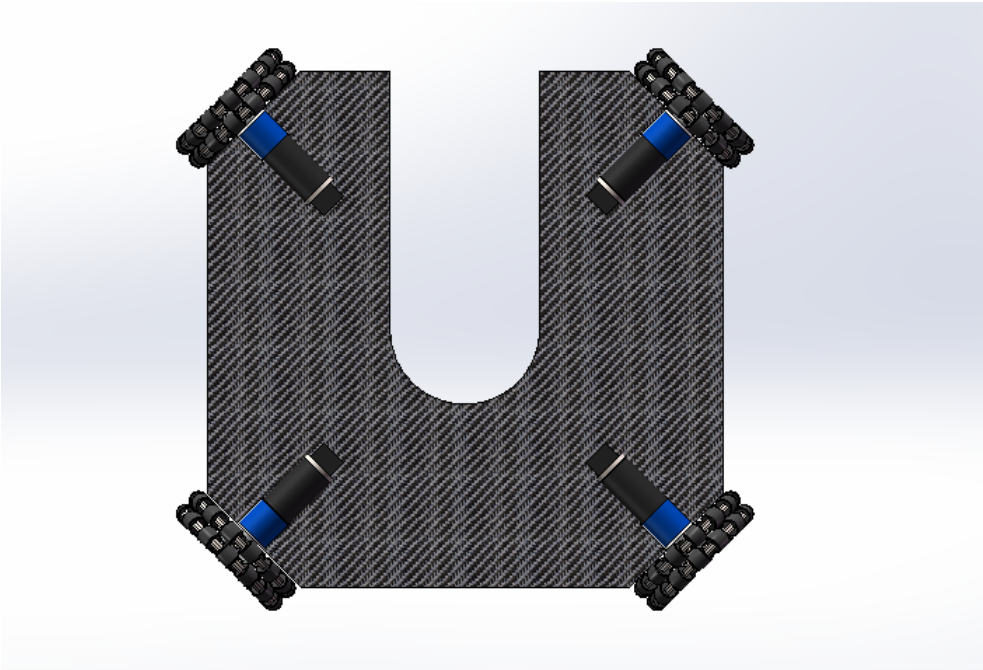


图 5: 底盘移动结构

移动结构的实现采用轮式。轮式结构由四个正交的全方位轮组成，四轮都配有驱动电机，通过四轮运动的矢量叠加，实现机器人前后左右四个方向移动。这样可以使底盘的稳定性好，运动更简单平稳，驱动力大。

使用全向轮使机器人的横向行走变方便，这样机器人移动到放置目标的台子时，可以轻松的通过横向移动去抓取小旗。使整个过程机器人只进行直走，不需要转向。

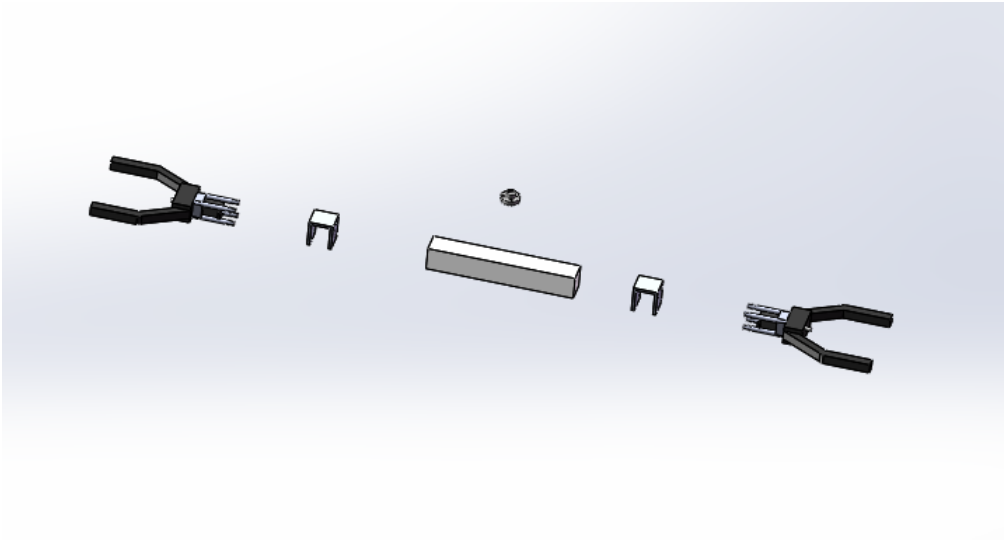


图 6: 手臂抓取结构

抓取系统由舵机、旋转杆和机械手共三部分组成，其中舵机控制整个机械臂的旋转，当位于侧面的机械手靠近要抓取的小旗时，舵机控制机械臂旋转 90 度，以便使机械手正对小旗。当抓起物块后，机械臂再旋转 180 度使另一边的机械手抓取下一个小旗。旋转杆是连接舵机和机械手的部分，它采用 T 形结构，使机械手的高度与平台上的小旗平齐，舵机放于顶盘上，使得整个旋转杆不至于过长而导致结构的不稳定，两个机械手抓取中心之间的长度为 50cm，以便一次性放置两个小旗。

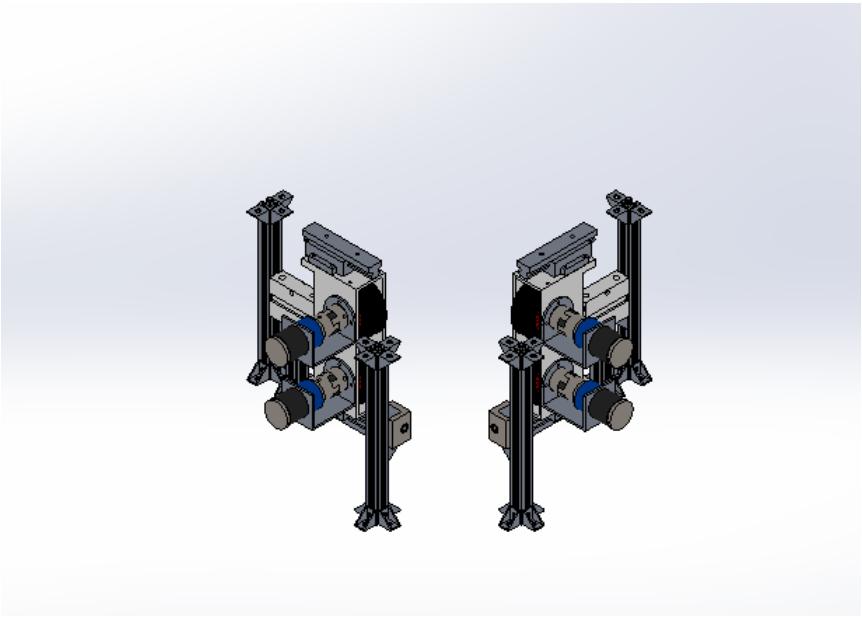
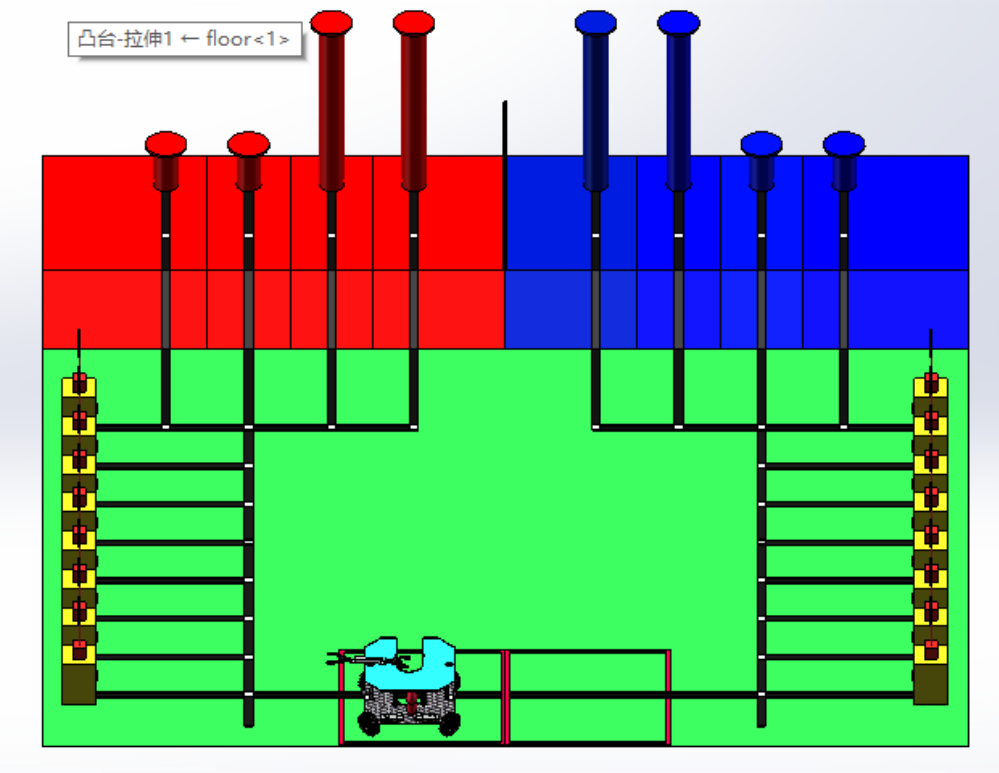


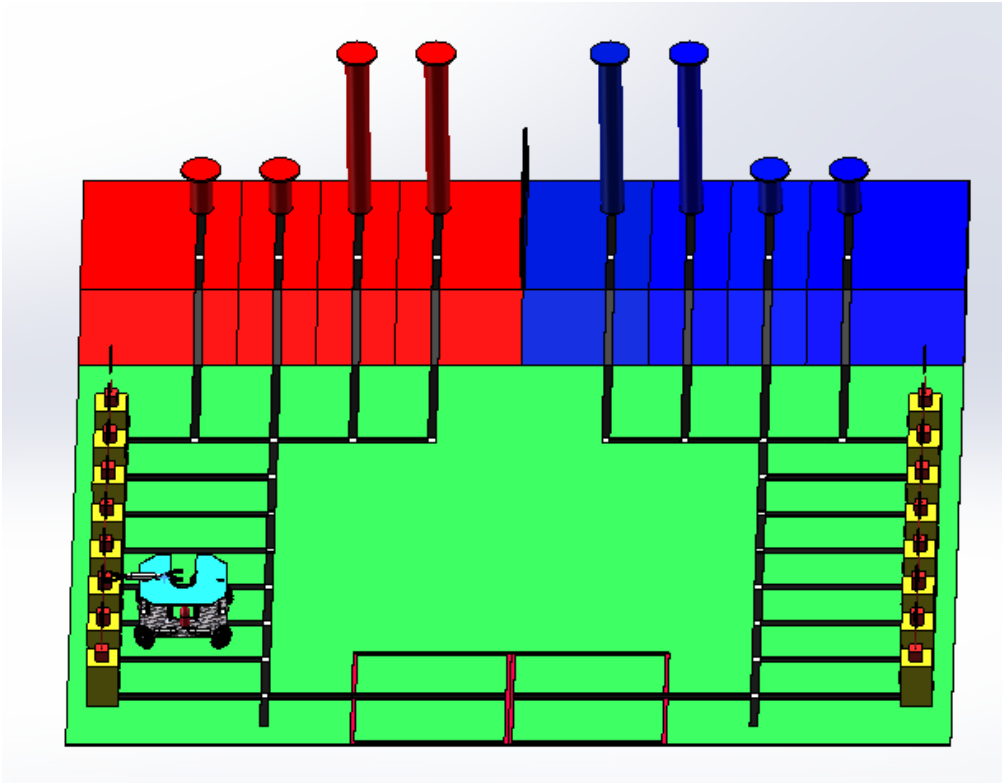
图 7: 气动爬杆结构

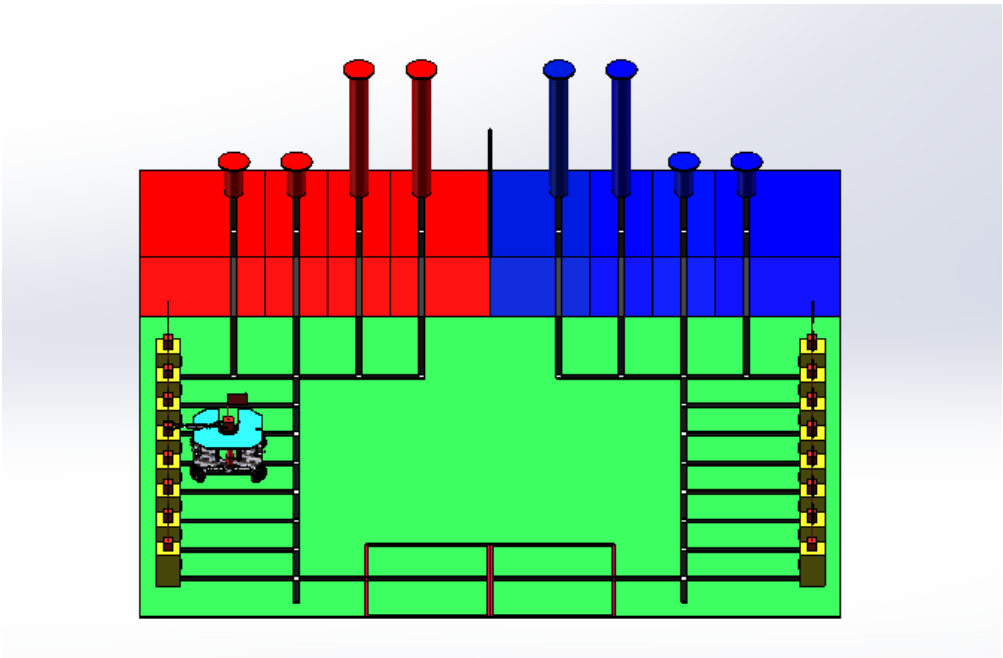
爬杆结构使用轮式爬杆的方法，设计时采用两组（共 4 个）驱动轮、两组（共两个) 固定轮、固定杆和气缸组合而成。两组驱动轮之间、两组固定轮之间都是垂直的。抱杆阶段，气缸伸长，带动推杆推动两个机架远离气缸，两对驱动轮相向运动，距离减小，驱动轮逐渐贴近立柱，直到被气缸顶至抱紧立柱。在立柱被抱紧后，开始上爬阶段，四个电动机同时工作，四个驱动轮在沿着杆上滚动，向着杆顶运动，达到爬杆的目的。固定轮用以加大轮组与爬杆的接触和摩擦从而使得轮组更易带动整体向上爬动。每个轮子都有一个单独的电动机驱动，四个驱动轮以相同的转速运动，带动机器人上爬到指定高度去放置物块。

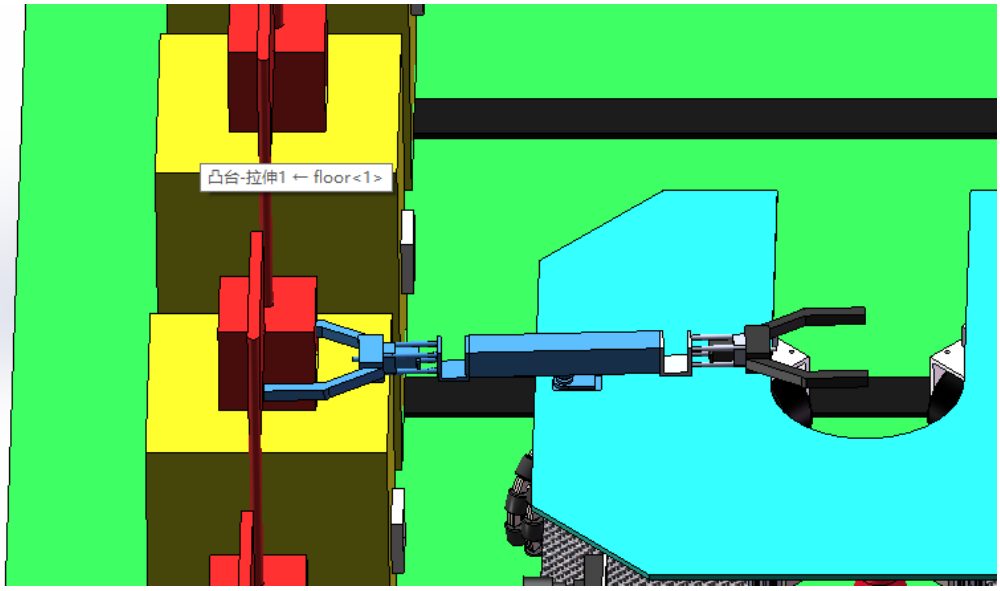
气缸的有杆腔和无杆腔各有 1 个接气口，用一个二位五通单电控电磁阀控制，阀上有 1 个来自压缩空气的进气口 P，2 个分别通向气缸的 A、B 口。A 和 B 始终是一个进气一个出气通给气缸，气缸也就只有伸出和缩回两种状态。给电磁阀通电只是切换一下 A、B 口的原始通气状态，即 A 和 B 反向进出气，因此气缸也跟着反向运动。这时给电磁阀断电，阀就会使 A、B 口自动回复到原位（阀里的复位弹簧作用），即 A、B 口进出气又换向了，气缸也就跟着回复。

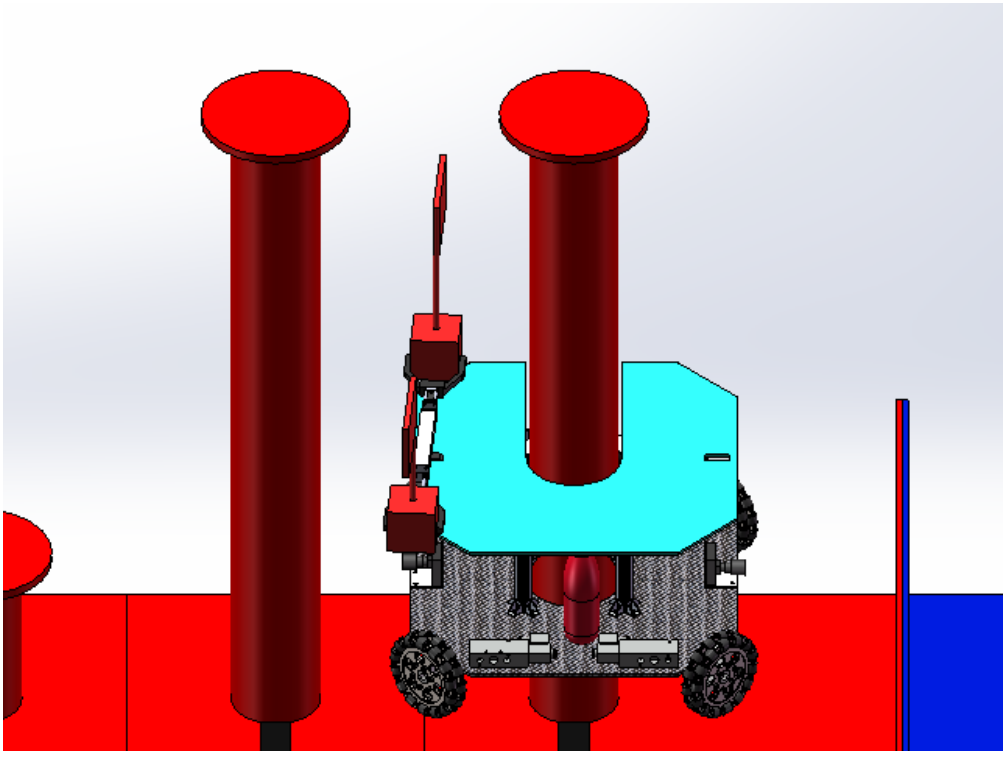
### 0.1.3 比赛模拟

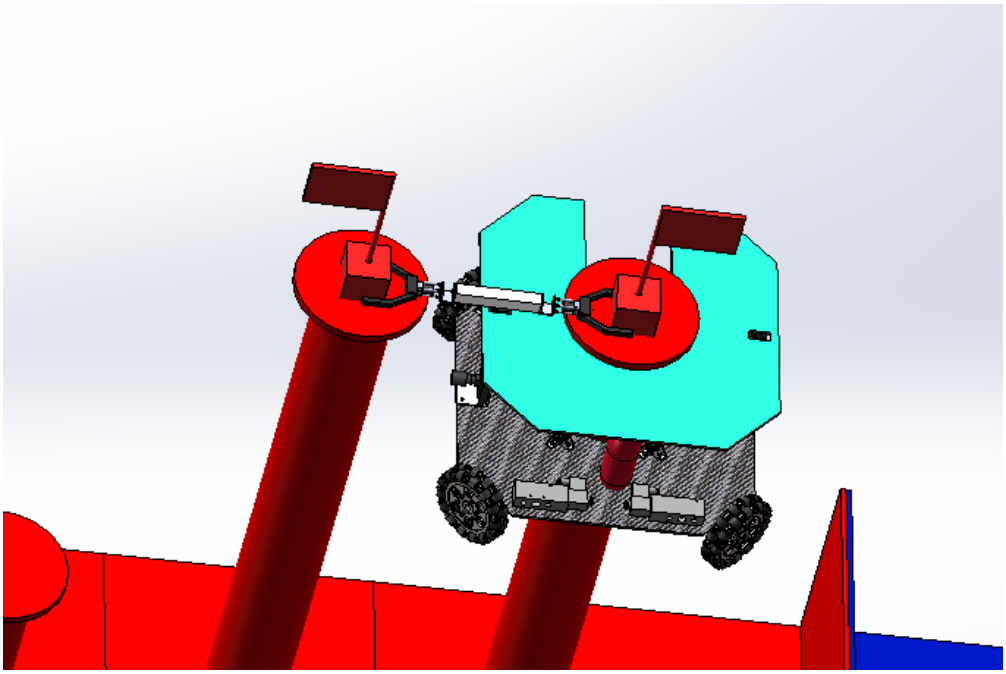







凸台-拉伸1 ← floor<1>





## 0.2 电路部分

### 0.2.1 整体设计

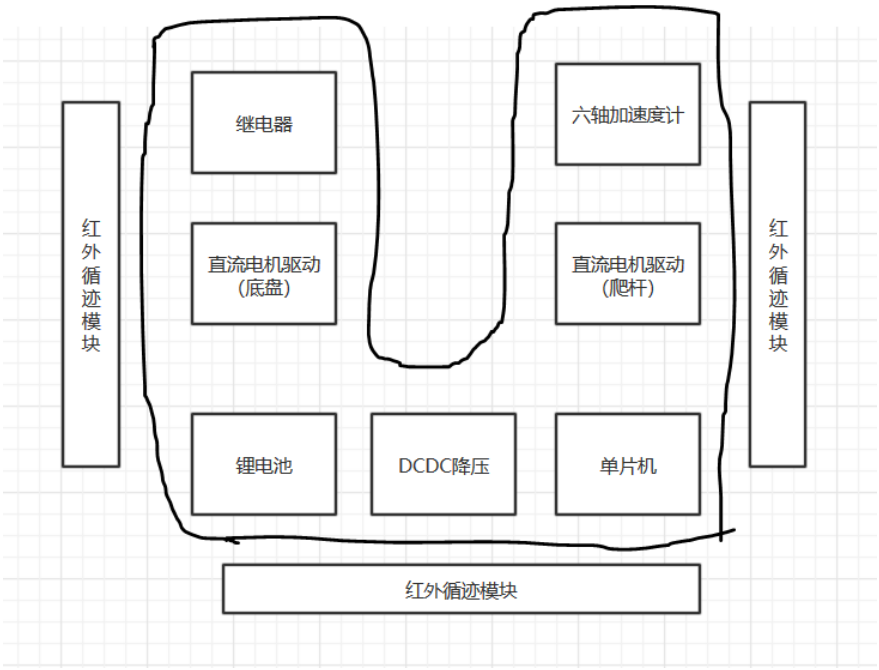



图 8: 下层底盘电路设计

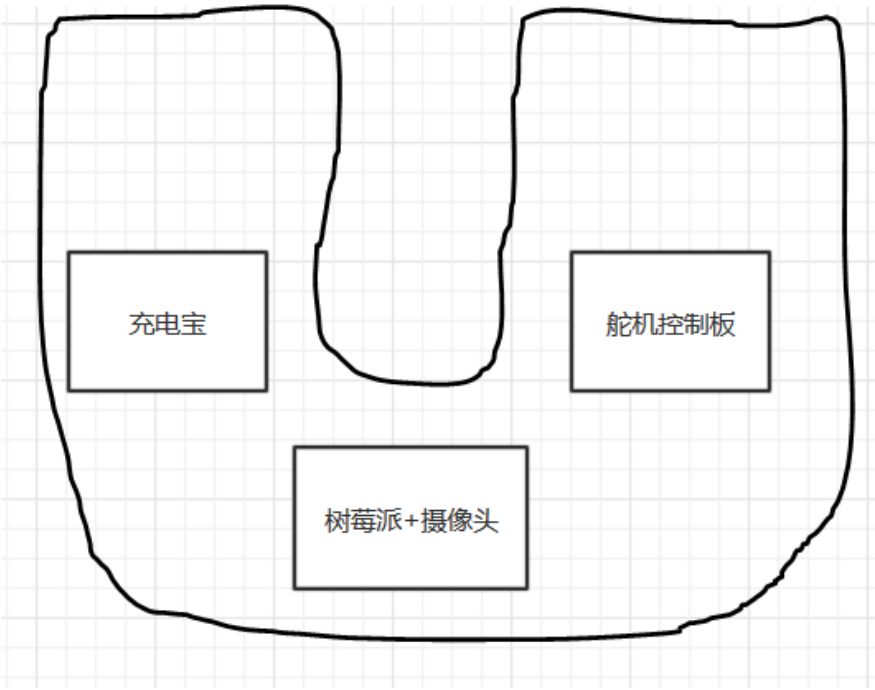



图 9: 上层顶盘电路设计

### 0.2.2 关键模块

- **锂电池**：选用 22.2V 6S 航模电池，目的是减轻重量，同时保证足够功率。
- **DCDC 降压**：三路降压，分别降至 3.3V(单片机供电),5V(传感器供电),7.4V(舵机供电)。
- **单片机**：使用电赛的一块自制系统板，和市面上的开发板区别不大，但不能接受 5V 供能。
- **电机驱动**：MOS+H 桥设计，保证较大功率时不会损坏。共计 6 块，分别驱动四个底盘直流电机和两个爬杆直流电机。
- **六轴加速度计**：由于机械和电机特性，机器人移动的时候可能会歪斜，需要用加速度计测量角速度和偏转角，形成闭环系统，用来确保移动方向的正确性。
- **五路红外循迹**：将五对对管集成在一个模块上，减小了电路体积，保证循迹的稳定可靠。注意的是，五路之间间距的把握和对地面测量距离的范围。
- **舵机控制板**：通过串口发送指令，规定某个舵机以某种行为运动，节省主控资源。
- **树莓派套件**：进行图像识别等上层操作。

## 0.3 控制部分

### 0.3.1 图像识别

图像识别算法使用 python + Opencv，主要方法是把图像转换成 HSV 格式，因为比赛使用颜色的 HSV 特点非常鲜明，首先，每种颜色的 H 值很不相同，而且 S 即饱和度相比环境色，显得非常之高。所以，可以事先调参，获取各种颜色的 HSV 范围，通过卷积，求出符合范围的面积最大的地方，那就是某种颜色所处的位置，从而可以将物块排序。

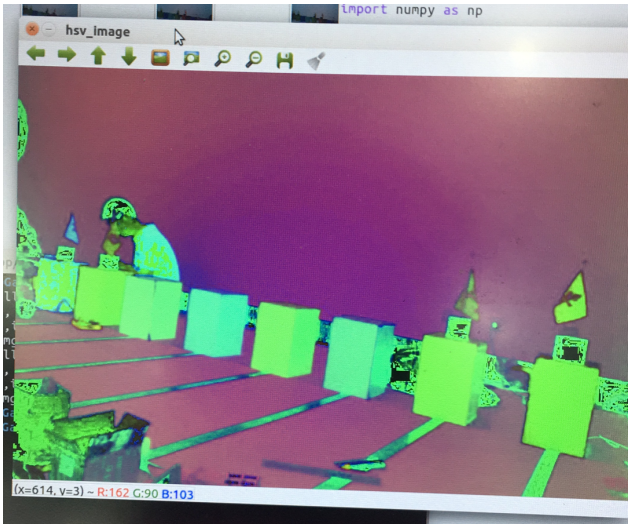



图 10: HSV


图 11: 识别结果

### 0.3.2 运动姿态调整

由于机械和电机特性，很难保证在开环的情况下，机器人能延直线运动，于是需要使用六轴加速度计来计算与初始状态的偏差角，并通过差速，修正姿态。

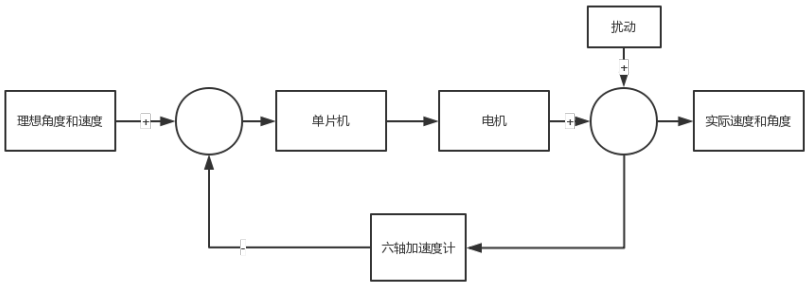



图 12: 姿态控制方框图

### 0.3.3 运动速度控制

PID 控制算法是一个在工业控制应用中常见的反馈算法。该算法把收集到的数据和一个参考值进行比较，然后把这个差别用于计算新的输入值，这个新的输入值的目的是可以让系统的数据达到或者保持在参考值。PID 控制器可以根据历史数据和差别的出现率来调整输入值，使系统更加准确而稳定。

具体来说，针对传感器获得的数据 c(t) 与参考值 r(t) 之间的差值 e(t)，PID 利用比例系数 Kp，积分系数 Ki 和微分系数 Kd 来计算新的输入值 u(t)。

本系统中采用四路红外探测模块传感器进行寻迹，对四路探测的输出进行加权，结果作为 c_ k，即 x_ i 为第 i 路输出， _ i 为权值。参考值 r_ k 则通过机器人处于巡线中央时的 c_ k 确定。系统产生的输出用于控制电机的速率，通过左右轮差速控制机器人偏转方向，从而稳定在巡线中央。

具体实现上，利用 stm32 的定时器模块产生固定频率的中断，每隔一定时间，查询 GPIO 管脚、获得四路输入值、计算 c_ k、e_ k 与 u_ k，并调节输出。假如频率过低，系统响应会过慢，而效率过高会浪费资源。

stm32 中，具体代码实现如下：

```
typedef struct
{       // PID控制结构体，用于存储PID需要的参数
        int setpoint;                                   // 设定目标
        float proportion ;                      // 比例常数
        float integral;                         // 积分常数
        float derivative;                               // 微分常数
```

```
7           int last_error;                         // e[-1]
8           int prev_error;                         // e[-2]
9   } PIDtypedef;
10
11  int incPIDcalc(PIDtypedef *PIDx, u16 point)
12  {       // 对某个给定PID工作单元，计算输出差值
13          int iError, iincpid;
14          iError = PIDx->setpoint - point;                // 当前误差
15          iincpid =
16                  PIDx->proportion * (iError - PIDx->last_error)
17                  + PIDx->integral * iError
18                  + PIDx->derivative * (iError - 2 * PIDx->last_error + PIDx->pr
19          PIDx->prev_error = PIDx->last_error;    // 存储误差，用于下次计算
20          PIDx->last_error = iError;
21          return (iincpid) ;
22  }
```