1 比赛策略

更细致的规则解读与设计思路参考附件"比赛策略"

1.1 比赛规则解读

1.1.1 场地，时间与计分

场地如图(长15m宽8m)。分为防守区和进攻区，起始都在防守区，控球的机器人要在对应的底线处装球。篮筐高度为2430mm。

小组循环赛每场160s，淘汰赛每场240s，每次控球有20s的投篮时间，投篮时间按队伍计算，失去控球5s则控球变更。控球变更时比赛时间暂停，投篮时间重置，罚球时比赛时钟与投篮时钟都暂停。

投篮前中后都在三分线外得3分，其余投篮均2分；扣篮7分；运球成功一次1分，传球成功一次1分，在每个投篮时间内运球与传球得到的总分不超过2分，投篮时机器人在地面上的投影与防守区有交集，不得分。

1.1.2 运球、传球、投篮与扣篮

运球严格定义为机器人与持有的球脱离后，球落到比赛区地面并反弹被该机器人回收的过程。其中**机器人在与球分离时与再接住球时，球必须至少距离地面700mm**，球距离地面700mm区间内与机器人没有接触，**收球意味着球处于被机器控制的状态，仅仅与球接触一下是不行的。**

机器人在从防守区进入进攻区，必须现在防守区内运球一次；而且在投篮前必须至少运球一次，或者说传球成功后该机器人成功控球后不改变位置直接投篮。通过搜索robocon的论坛我们得知：**机器人是没有"走步"的，因此在满足需要运球的条件后，机器人是可以控着球走的。**但是也不能完全抱着球跑，**不能完全覆盖住球，起码在700mm上下350mm是可见的**，用一个类似网兜的设计托着球是跑允许的。

传球严格定义为**两机器人相距至少1000mm，且球在脱离机器人的运动过程中最多与地面接触一次，然后接球的机器人要稳稳接住。**接球机器人必须在进攻区内。

投篮前至少运球一次或者控球后不改变位置便可直接投篮，**投篮只能在进攻区进行。**

扣篮需要满足的条件为：在限制区，**独立跳起，在空中放开球，放开球时球没有向上的速度**，放开球前机器人不能钩住篮筐，放开球后可以钩住篮筐。

1.1.3 防守、控球变更与犯规

守方只能在防守区进行防守，防守的手段有：抢断、盖帽，阻挡和拦截，目的是**使攻方失去控球权，然后守方去捡球。或者攻方投篮失败，守方去抢篮板球。**

抢断：只能在攻方持球机器人运球或传球时球与该机器人完全脱离的情况下实施，允许护身接触，禁止其他部件接触

拦截：主要通过精准判断球路来阻止传球且不与攻方机器人接触的一 种防守方式，拦截时，如果守方机器人的任何部件（含护身）与攻方机器人接触，会被判定为拦截犯规。

阻挡：阻挡是指守方/攻方机器人以合法的防守位置和姿态防止攻方/ 守方机器人从自己身边通过的一种合法手段。允许护身接触，禁止其他部位接触或者守方将部件伸入攻方护身及其上方。

盖帽：攻方机器人投篮出“手”时，守方机器人设法在空中将球打掉的动作，如果守方机器人在空中击打攻方机器人投出的球时接触其除护身外的任何部件，守方机器人犯规。

控球变更的触发条件有：①控球的攻方的两台机器人在规定的设置时间内均未完成设置，②攻方成功投篮，③**攻方投篮失败且没有抢到篮板球或者没捡到球**，④球出界，⑥投篮时间结束，⑦攻方犯规，⑧**攻方丢球**，⑨守方持球申请变更，⑩攻方进入进攻区或者投篮前没有进行至少一次运球。

控球变更可以暂停比赛时钟，重置投篮时钟并且将所有机器人重置回防守区，只要攻方脱离球的控制5秒，便可控球变更。

对于犯规，主要注意防守时不要触碰攻方和球出界。**投篮时守方推动攻方，攻方将获得一次罚球机会，罚球相当于获得一次不消耗时间的投篮机会。**

1.2 机器人设计思路

整体上，以实现基础功能为主，扣篮成功一次7分固然诱人，但是成功率低、对机器损害大、而且在机器人上占空间以及重量，如果不能确保可以靠扣篮稳稳得分的话，花费大量空间设计单独的扣篮模块只会显得鸡肋。**扣篮模块不作为重点得分项，其设计是基于基础功能的附加产品，重点在于基础得分的实现。**

运球部分：在规则约束内的运球，可以有静止运球和边运动边运球两种，对于边运动边运球，需要在球脱手时施加一个向前的速度，然后机器人要去追球并稳稳收住，因为场地不是理想的，实际情况球可能会乱弹不容易收住球，无论是自动还是手动都较难实现，因此选用静止运球。即**由一个可开闭的容器从地上捡起球抬升到一定高度后打开，球自由落体再弹起，这期间容器移动到700mm处待命，待球弹回来时容器再闭合，稳稳接住球。**

传球部分：如果真的要完美实现传球，有些困难，因为要求接球机器人稳稳接住球，无论是用视觉、预设轨迹、手动接球都难以实现，但是传球这部分的分数又可被运球替代，因此决定不做精确的传球，只需要**将球向接球机器人丢出然后接球机器人在5s内重新控球即可。**

投篮部分：投篮本质上是一个斜抛运动，得到投球点与篮筐的坐标既可以建立抛物线。这其中有两个问题，一个是如何让投球的朝向面对篮筐，一个是过两点有无数条抛物线，哪一条抛物线是最佳的。对于前者，我们评估了视觉锁定篮筐、坐标锁定篮筐、投篮器与机器人移动双轴操控等等方案，发现还是手动调整朝向比较好，只要机器人运动设计的旋转比较方便即可。对于后者，最佳抛物线的选择必然需要大量的实验来支持，对于以篮筐为圆心的同心圆上，最佳抛物线是相同的，但即便这样，所需数据也太多，因此我们只预设三组参数，让操作者来自行把控距离，但是同时也能自行设置投球角度和速度。**因此我们的投篮器设置为一个扇形机构，有摩擦轮控制投球速度，引导器控制投球角度，引导器位于扇形最外侧圆弧处，通过其伸出扇形的长度来控制投球角度，最大为90°，最小为0°，这也可以满足传球的平抛要求。**

运动部分：在投篮部分中提到，因为要手动调整朝向，那么机器人应该能灵活旋转(原地旋转)，**三轮全向轮底盘**再合适不过了。

防守部分：为了避免无形中的犯规，我们重点在传球过程中进行抢断，因为运球时可能会接触，盖帽需要跳起来，不好实现。**为了更好的拦住球，我们在机器人的后侧配备网，可以在对方传球时更好拦下球，此时只需让对方丢球5s，即可控球变更。**

扣篮部分：虽然不把扣篮作为核心，但是也加入了扣篮模块。**表现为六个弹簧腿和投球器的可伸缩的升降杆。**扣篮时首先将升降杆伸出，投球器放到顶端，解锁弹簧腿使机器人跳起，待达最高点时，投球器水平出球，平抛运动进入篮筐，跳起不用太高，能够时投球器出球时球的下沿至少到2430mm即可，但是出球时机一定要把握准，至于出球速度，则需要大量现场实验来得到最佳速度。

1.3 实际比赛

如果为攻方，控球机器人首先运球一次，否则不能进入进攻区，机器人进入进攻区后，可以选择是传球给队友然后投篮还是自己去投篮。**如果自己去投篮**，要先在进攻区内运球一次刷满2分上限，然后移动到适合位置投篮，这里一定要移动到适合位置(预先设定的位置)，因为如果被撞罚球，是要在原位置投篮的，这样才能保证罚球能投进，因为设置的投篮速度与角度只有三组；**如果控球的受到了拦截，脱不开身，传球给队友**，如果队友能接住，传球成功得到1分并且此时队友属于控球后没改变位置，可以直接投篮，但是这样第一难以实现，第二可能接球点不是预设的位置；如果队友接不住，队友要在5s内控球，不让球出界，然后运球一次刷满2分，移动到预设位置投篮。**如果时机特别适合扣篮**，可以尝试提前走到限制区的某一位置扣篮，预设好出球速度。

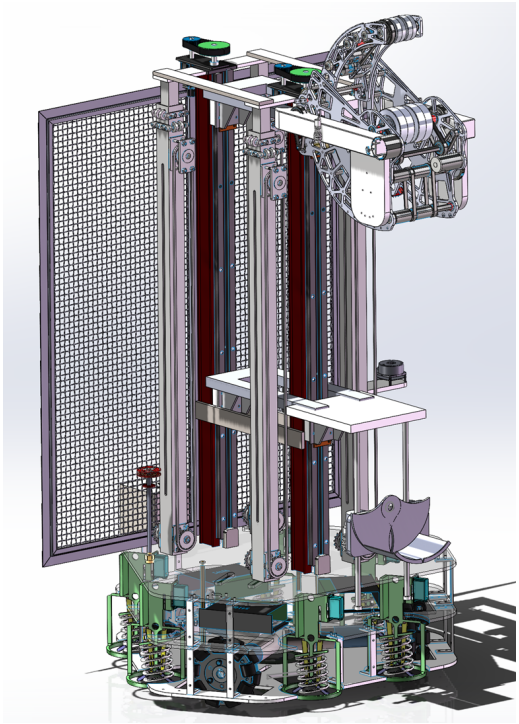
如果为守方，则要**在对面运球或者传球的时候找准时机挡下球**，但是不要让球出界，因为球出界是按照球最后一次碰到的队伍算的；此外还要抢篮板球。这之中有两个策略，一个是在没出界的前提下，**两个球分别拦截一辆对面的机器人**，让其在5s内不能重新控球，从而控球变更；另一个是**一个拦截一辆，另一个去抢控球**，如果能够成功控球，那么会判定为攻方丢球，直接控球变更。

由于一次控球的投篮时间只有20s，那么运球的速度应该尽量快，由上至少要允许两次完整运球、一次投篮和运动的时间，因此一次运球时间控制在2.5s以内，运动速度要快，因为控球用的容器比较深，可以防止球因为惯性飞出，三轮全向轮也最小化了转向的时间。投篮时间也尽量快，因为投篮是根据球脱手的时间判断有效性的，可以留出大多时间给调整位置和朝向。

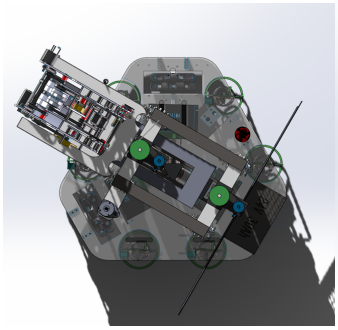
2 模型展示

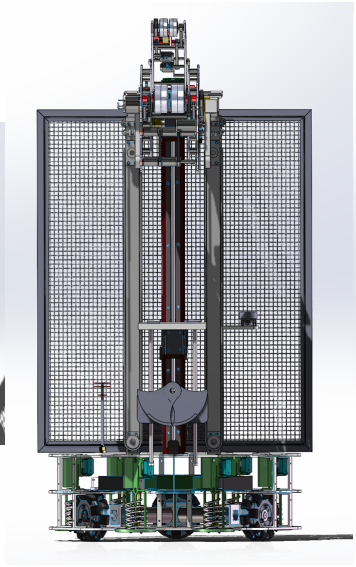
2.1 整体展示

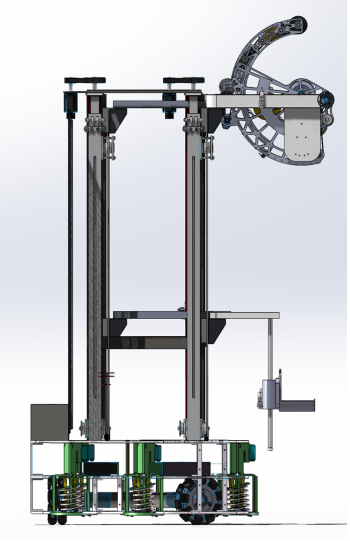
整体视图



俯视图　　主视图　　左视图



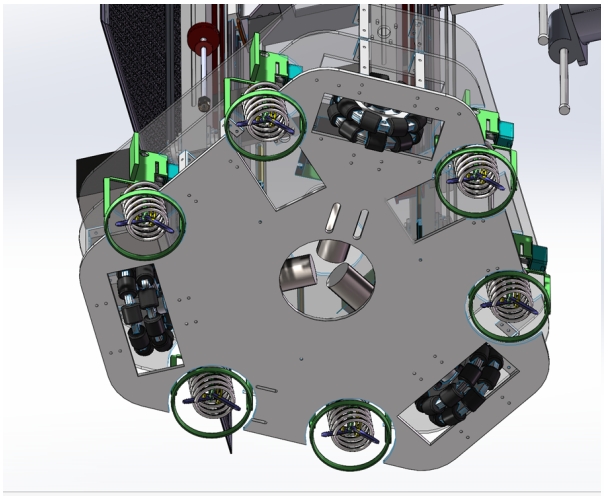


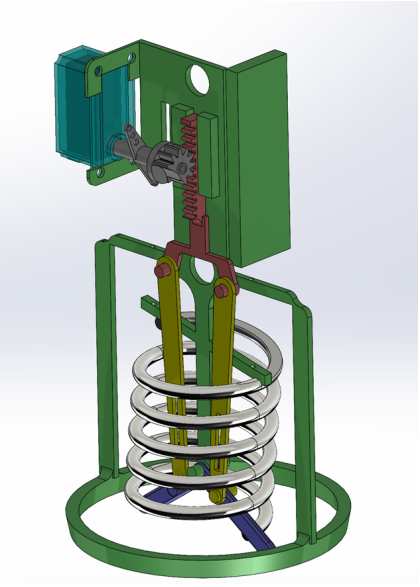


2.2 各功能模块

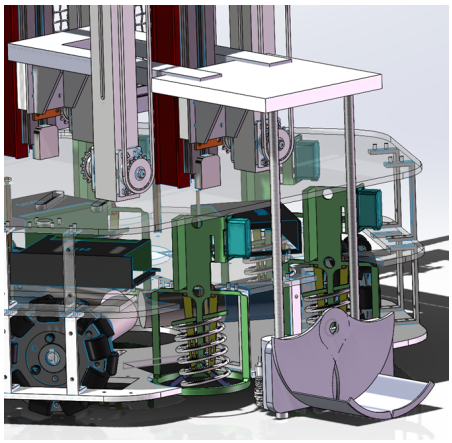
2.2.1 运动模块

全向移动轮　　弹跳机构

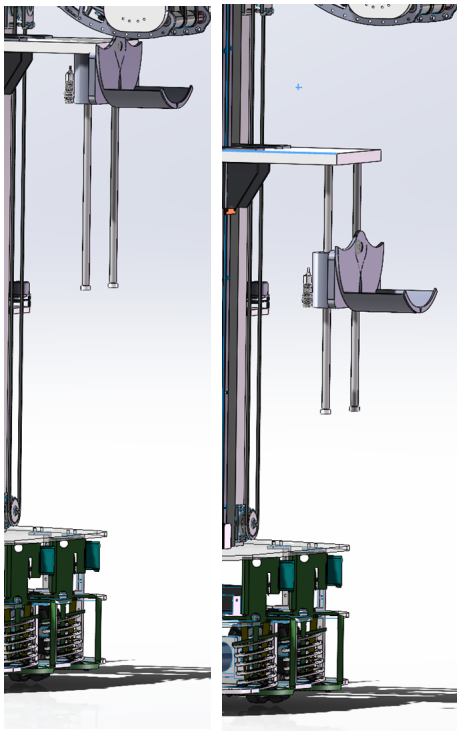




2.2.2 捡球，运球模块

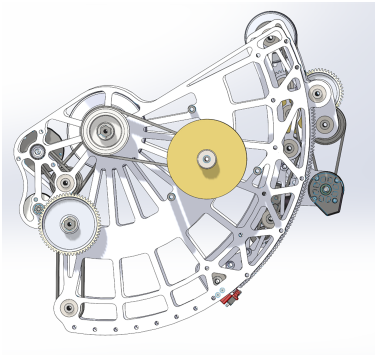


捡球：如图所示为捡球容器，容器打开，下降到地面高度，然后关闭容器，实现捡球。

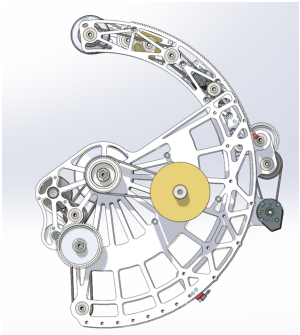


如图依次为将球抬升到一定高度释放，和移动到700mm处接球的过程

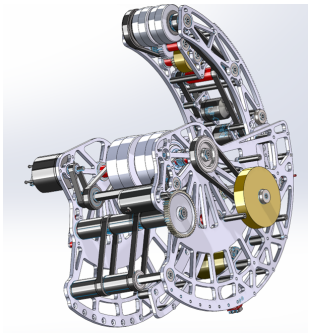
2.2.3 投篮器



如图为投篮器结构，图中为将球送入投球器时的准备状态，将球从正下方送入，然后通过引导传送带进入传球器。



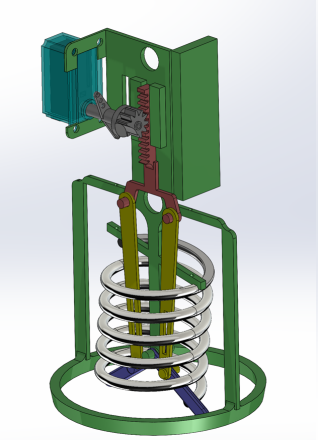
上方贴近圆弧侧伸出的结构为引导器，其伸出的长度对应了发球角度。

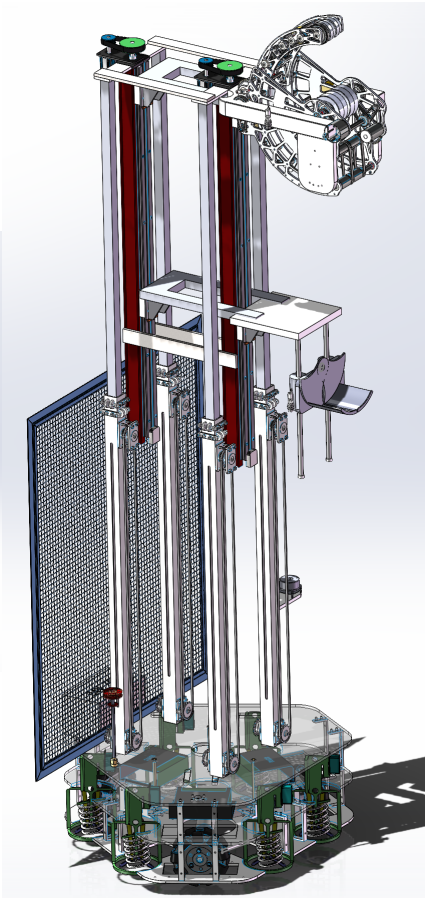


如图在引导器内部已经中轴附近分布着摩擦轮，对应了发球速度。

整体运动过程为：球从下方进来左边传送带摩擦进入机构→中央转轮将球运进准备位置→机构伸展，球一起移动→发射机构加速→中央转轮将球运到发射位置→出球

2.3.4 扣篮器





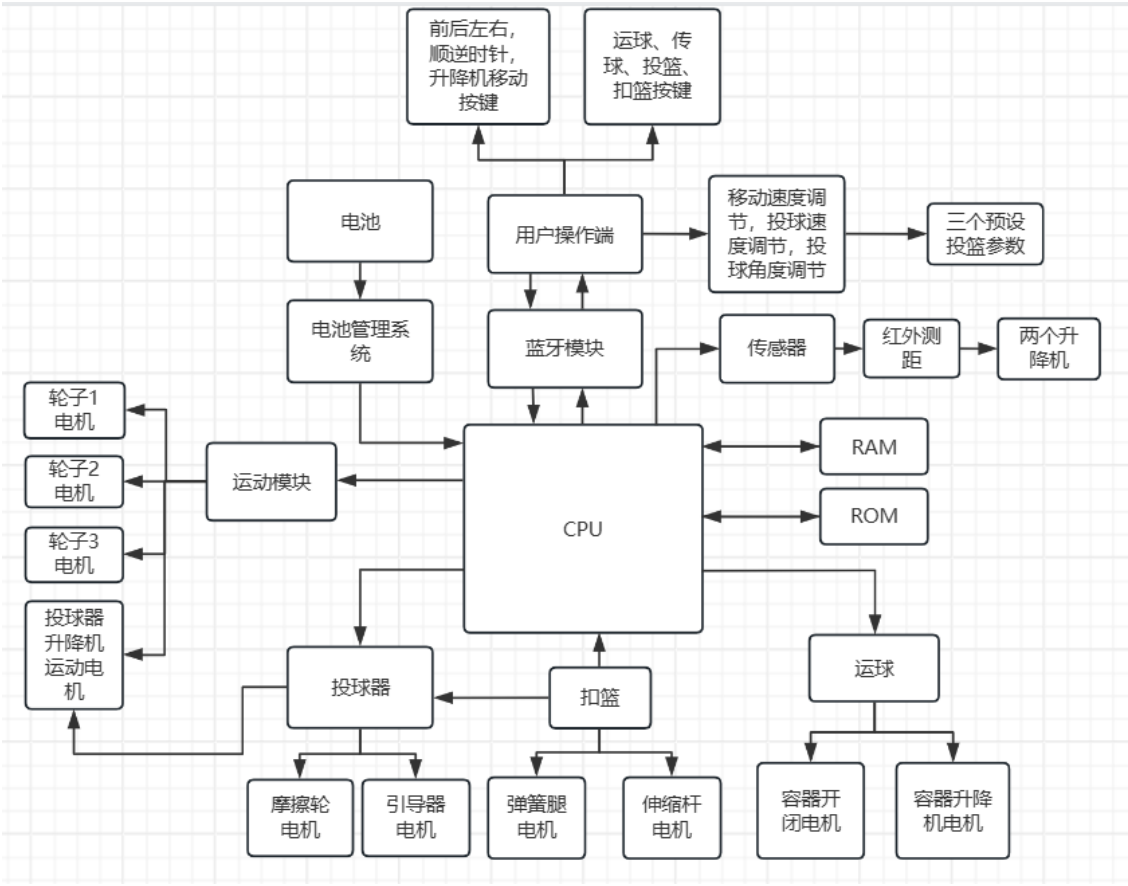
弹簧腿　　可升降的升降机杆

扣篮前杆先伸长到最高，投球器带球上升到顶端，然后解锁弹簧腿，待弹到最高点时投球器平抛出球。

3 电气硬件

电器硬件以及算法框架的细致解读见附件“算法框架”

3.1 系统框图



通过蓝牙模块从操作的前端将实时数据传输给cpu进行处理。cpu根据这些值对各种电机进行调整。负责**运动模块的电机**有：三个轮子电机，投球器升降的电机。**投球器的电机**有：摩擦轮电机和引导器电机，配合投球器升降机电机可以完成投篮、传球功能。**有关扣篮的电机****有：弹簧腿电机和伸缩杆电机，配合投篮器和投篮器升降机可以完成扣篮功能。**运球涉及的电机**有容器开闭电机和容器升降机电机，只要移动到球面前，运球是自动完成的。这其中**涉及的传感器**为红外测距传感器，用于测量两个升降机的高度。不需要速度传感器是因为各涉及速度的板块(轮子运动，升降机运动)都是开环控制，轮子的移动有移动速度调整按键控制缩放比例(最小0最大1)，升降机运动的速度是固定的，上下移动只需对应乘1或-1即可。ROM存储机器的固有程序，RAM存储运行时产生的临时数据。**在程序执行时会不断循环获取由操作端传来的各种数据，再由机器自动做出相应的反应。**

3.2 电机信息

由图可以得出需要的电机如下。共有25个电机。各种功能的实现就是依托这些电机之间的相互配合而实现的。

组成部分	电机驱动机构	电机数量
运动模块	全向轮	3
投球器	投球器升降	2
	捡球/运球器升降	2
	引导传送带	1
	中央转轮	1
	发射轮	2
	伸缩投球器	1
扣篮模块	弹簧腿	6
	长颈鹿伸缩杆	4
捡球机构	铲斗开闭	1
	铲斗升降	2

部分电机的相关参数如下：

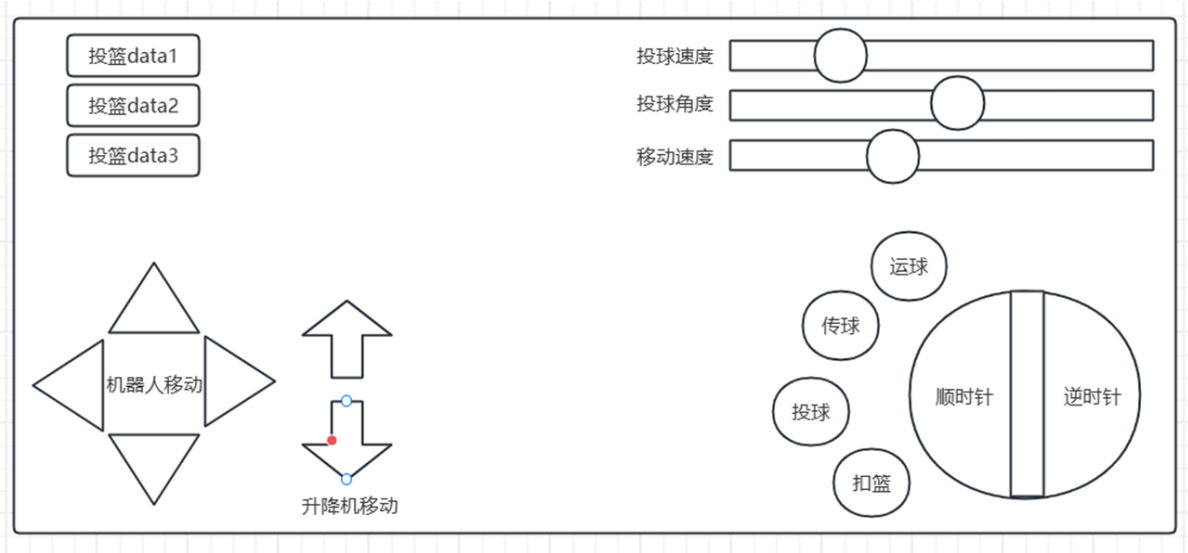
电机位置	电机种类	电机参数	电机优点
运动轮电机	无刷直流电机(Maxon EC 90flat)	额定功率:约 40W 额定转速:约 477RPM 额定力矩:约 0.0777Nm(总力矩) 电压:建议 12V或 24V	高稳定性，高动态响应，无刷更耐用
升降机电机	高性能伺服电机(如安川Σ-7系列)	额定功率 200-400W 额定扭矩 1Nm 最大转速 3000 RPM	高扭矩，高钢性，反应灵敏
捡球容器开闭电机	无刷直流电机(T-Motor MIN5208)	额定功率 50W 额定扭矩 0.1-0.3Nm 最大转速 5000RPM	适应高动态响应的轻负载齿轮驱动
摩擦轮电机	无刷直流电机(BLDC)(MaxonEC-160 系列)	额定功率:约180W 额定转速:约 1245RPM 额定扭矩:约 1.39Nm 电压:建议 24V	高扭矩，适合中高速
投球器引导传送带电机	高性能伺服电机(台达ECMA-C10207RS)	额定功率:约 25W 额定转速:约 382RPM 额定扭矩:约 6.13Nm 电压:建议 12V	适应高动态响应的轻负载齿轮驱动

其他补充：六个弹簧使机器人能弹跳，经计算，采用k=1200N/m的弹簧。红外测距使两个升降机的使用更加精准。

4 算法框架

4.1 整体控制策略

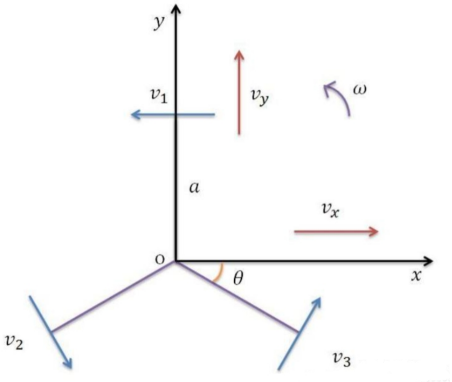
机器人主要由人工控制，其中一些功能的实现是自动的。用户可以控制的变量有：①前后左右移动，②顺逆时针旋转，③投球器升降机移动，④机器人运动速度调整，⑤投球速度调整，⑥投球角度调整，⑦三个预设的投球参数，⑧运球键，传球键，投篮键，扣篮键



自动部分：①按下运球键后，能够自动捡起球上升到一定高度后松开，然后接球的容器下降到700mm处接住球；②按下传球键后，接球的容器会将球抬升送去投球器中，投球器的引导器完全伸出，使球能以水平方向发射，可以通过投球速度旋钮调整出球速度。③设置好投篮角度和速度，按下投篮键，会将球投出，有三个预设好的投篮参数，按下后可以自动设置对应的角度和速度。④按下扣篮键后，伸缩杆会伸出，运球器会升高到最高点，弹簧解锁机器人跳起，在最高点投球器以一定速度平抛出球。（如图一）

手动部分	自动部分
前后左右移动	运球
顺/逆时针旋转	传球
升降机移动	扣篮
移动速度调整	三种投篮方式
投球速度调整	
投球角度调整	
三个预设的投球参数	

图一



图二

4.2 各种算法

功能的实现基于电机的配合，各种算法实际上就是将各种功能的运动过程拆解，再将每一步运动的速度、时间、位移等等转化为对应的运动模块的电机参数。因此**把运动参数转化为电机参数和把电机参数发送给对应的电机是核心**，但是这些都涉及电机的原理，不太好写，在此处省略，只写框架。**假设sendmotor(motor,motor_para)是将电机参数传给电机的函数，v2motor(v)类似的为将运动参数转化为电机参数的函数。**

因为本机器人需要大量的手动操作，因此**将操作端的参数变化转换为对应运动参数也是非常重要的。**在实际程序运行时，主函数是要从上电开始无限循环的，**假设用get函数可以得到操作端的某些参数，然后利用这些数据来使程序做出不同响应。**(具体有哪些参数见4.2.6 main函数)

以下代码均用python，因为简洁、可读性强。

4.2.1 移动算法

从俯视角度，以机器人投篮器正朝向作为y轴，建立右手坐标系。直接决定运动的有三个速度v_y, y_x, ω，其中a是速度点与转轴的距离。（如图二）

因为三个轮子的速度方向是确定的，可以以此得出速度的转换矩阵$(\theta = \frac{\pi}{6})$

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -1 & 0 & a \\ sin\frac{\pi}{6} & -cos\frac{\pi}{6} & a \\ sin\frac{\pi}{6} & cos\frac{\pi}{6} & a \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

其中三个速度互相不独立，为了简化操作，调速度使三个轮子速度同比例调整。运动的速度控制为开环控制，操作端的移动速度参数，对应了速度的以最大值为基础的放缩比例，可从0到1调，以此实现速度调控。

```
import numpy as np
v_max=#直线移动满速
w_max=#旋转角速度满速
status_x=0
status_y=0
status_w=0#三个运动方向的状态，当按正方向时为1，负方向时为-1，不按时为0。其中对w定义逆时针为正方向
```

```
a=d#机器人的最大半径
A=np.array([[-1,0,a],[0.5,-0.866,a],[0.5,0.866,a]])#上述的转换矩阵
speed_percentage=0.5 #调速用的缩放比例，满速时为1
def move(v_in):#运动函数，输入为实际运动的数值
    v_out=A@v_in    #计算得出对应v1，v2，v3的数值
    motor_para_1=v2motor(v[0][0])#转换为电机参数
    sendmotor(motor_move1,motor_para_1)#传给对应电机
    motor_para_2=v2motor(v[0][1])
    sendmotor(motor_move2,motor_para_2)
    motor_para_3=v2motor(v[0][2])
    sendmotor(motor_move3,motor_para_3)
if __name__ == "__main__":
    while(1):
        get(status_x)
        get(status_y)
        get(status_w)
        get(speed_percentage)
        v_in=np.array([[status_x*v_max*speed_percentage],
[status_y*v_max*speed_percentage],[status_w*v_max*speed_percentage]])
        move(v_in)
```

4.2.2 运球算法

运球算法整体过程为：**操作员手动控制机器人到球面前，按下运球键。容器打开并移动到地面处，关闭容器上升到高度h1，松开容器，移动到h2=700mm处，延迟一定时间后恰好为球第一次弹到700mm以上，此时闭合容器。**

对于反弹高度，假设衰减比为0.8，在1.25m落下时，自由落体到地面时间为0.5051s，速度为4.95m/s，反弹速度理论为3.96m/s。最大上升高度0.8m，到达0.7m时速度为1.4m/s，消耗时间为0.2612s，上升到最高点再落到0.7m处的时间为0.5469s，那么从1.25m放手到0.7m接住的过程中，升降机从1.25m运动到0.7m处应该在[0s,1.052s]，容器关闭的时机应该在[0.7663s,1.052s]。以上是在理论假设的基础上解得的值，实际还需要大量的实验来得到符合物理情况的值。

另外，因为对机器人有限高1.5m要求，而且投球器也占一定高度，但是允许折叠结构。因此将投球器也改为可以升降，而且因此可以控制传球的发射高度，也为传球提供了更多的灵活性。

至于对高度数据的测量，选择红外测距传感器，将直接测得的数据再根据传感器的原理再处理，得到以地面为0的高度值。

```
motor_open#负责容器开闭
motor_up#负责容器升降
h1=1.25#丢球时高度(理论值，具体高度需要实验测量)
h2=0.7#接球时高度
h3=0#捡球时高度
def close():#关容器
    ...
    sendmotor(motor_open,motor_open_close_para)
    return

def open():#开容器
    ...
    sendmotor(motor_open,motor_open_open_para)
    return

def h2motor(h):#把高度转化为电机参数的函数
    ...
    return motor_para
```

```
def dribble():
    motor_up_para1=h2motor(h3)
    sendmotor(motor_up,motor_up_para1)
    close()
    motor_up_para2=h2motor(h1)
    sendmotor(motor_up,motor_up_para2)
    open()
    motor_up_para3=h2motor(h2)
    sendmotor(motor_up,motor_up_para3)
    delay(770)#由之前理论中推算延迟0.77s关闭容器来接球，实际上还应该考虑升降台移动到0.7m处的时间以及程序数据处理的时间
    close()
```

4.2.3 传球算法

传球算法整体过程为：**首先在运球完控住球基础上，手动控制投球器升降机移动到能够卡住球的位置，然后按下传球键。引导器会首先回收，将球夹进投球器中，然后通过设定的投球速度调整摩擦轮的参数，传球为平抛，引导器完全伸出。**

```
status_up=0#升降机的状态，1为向上升，-1为向下，0为静止
v_up=#升降机运行速度，固定
motor_up_plus#控制投篮器升降的电机
motor_speedup#控制摩擦轮的电机
motor_leader#控制引导器的电机
def vball2motor(v):#把投篮速度转化为电机参数的函数
    ...
    return motor_para
def vangle2motor(angle):#把投篮角度转化为电机参数的函数
    ...
    return motor_para
def vup2motor(v):#把升降速度转化为电机参数的函数
    ...
    return motor_para
def pass(vball):
    motor_para1=vangle2motor(90)
    sendmotor(motor_leader,motor_para1)
    delay(700)#确保球被夹住
    motor_para_speed=vball2motor(vball)
    motor_para_angle=vangle2motor(0)
    sendmotor(motor_speedup,motor_para_speed)
    sendmotor(motor_leader,motor_para_angle)

if __name__ == "__main__":
    while(1):
        get(status_up)
        get(vball)
        motor_para_up=vup2motor(v_up*status_up)
        sendmotor(motor_up_plus,motor_para_up)#升降机运动
        pass(vball)
```

4.2.4 投篮算法

基本过程与传球一直，只是需要预设投篮速度和角度，按下预设的三个参数键时，会自动设置角度与速度，同时在操作页面的后端数据也要修改，防止数据被刷新掉。

```
shot_data1=[]
shot_data2=[]
shot_data3=[]#三组预设的投篮数据
status_shot1=0
status_shot2=0
status_shot3=0#三种数据是否选择
status_up=0#升降机的状态，1为向上升，-1为向下，0为静止
h_up=#投篮器的高度
def shot(vball,angleball):
    motor_para1=vangle2motor(90)
    sendmotor(motor_leader,motor_para1)
    delay(700)#确保球被夹住
    motor_para1=h2motor(h_up_max)#上升最高
    sendmotor(motor_up_plus,motor_para1)

    motor_para_speed=vball2motor(vball)
    motor_para_angle=vangle2motor(angleball)
    sendmotor(motor_speedup,motor_para_speed)
    sendmotor(motor_leader,motor_para_angle)

if __name__ == "__main__":
    while(1):
        get(status_shot1)
        get(status_shot2)
        get(status_shot3)
        get(vball)
        get(vangle)
        get(status_up)
        if status_shot1:
            vball=shot_data1=[0]
            vangle=shot_data1=[1]
        if status_shot2:
            vball=shot_data2=[0]
            vangle=shot_data2=[1]
        if status_shot3:
            vball=shot_data3=[0]
            vangle=shot_data3=[1]
        motor_para_up=vup2motor(v_up*status_up)
        sendmotor(motor_up_plus,motor_para_up)#升降机运动
        shot(vball,vangle)
```

4.2.5 扣篮算法

扣篮算法的基本过程：**首先将投球器升降机移动到可以夹住球的位置，然后按下扣篮键，此时先投球器夹住球升降杆先延长，最高可达2.4m，投球器带球把篮球送到最高点，然后跳起，弹跳到最高点时平抛投球，因为球脱手时不能有向上的速度。**

弹簧存储的能量是一定的，由于具体重量未知，具体可以跳多高h_jump和跳到最高点发球的时间t_jump都是需要实验来测得的。

```
h_jump=#最大跳起高度
t_jump=#到最高点高度
status_dunk=0#扣篮的状态
h_up=#投篮器的高度
def push():#伸缩杆伸出的函数，涉及伸缩杆的电机
    ...
    return
def jump():#跳起的函数，涉及弹簧腿的电机，较为复杂
    ...
    return
if __name__ == "__main__":
    while(1):
        get(status_dunk)
        get(status_up)
        get(vball)
        motor_para_up=vup2motor(v_up*status_up)#升降机移动
        sendmotor(motor_up_plus,motor_para_up)#先要移动到可以夹住球的位置
        if status_dunk:
            push()
            delay(1000)#确保完全伸出
            motor_para1=vangle2motor(90)
            sendmotor(motor_leader,motor_para1)
            delay(700)#确保球被夹住
            motor_para1=h2motor(h_up_max_plus)#伸缩杆伸出后上升最高h_up_max_plus,未伸出最高h_up_max
            sendmotor(motor_up_plus,motor_para1)
            jump()#跳起
            delay(t_jump)#根据跳起到最高点所用的时间进行延迟，并且同时需要考虑程序运行机器反应的时间
            shot(vball,0)#水平抛出,vball要根据其距离篮筐的距离，跳起的最高距离等等而定
```

4.2.5 main函数

main函数是要无限循环的，要在每次循环中更新从操作端传来的数据

```
if __name__ == "__main__":
    while(1):
        get(status_x)#对应左右键
        get(status_y)#对应前后键
        get(status_w)#对应顺逆时针键
        get(speed_percentage)#对应移动速度
        get(status_dribble)#对应运球键
        get(status_pass)#对应传球键
        get(status_up)#对应升降键
        get(status_shot)#对应投篮键
        get(status_shot1)
        get(status_shot2)
        get(status_shot3)#对应三个预设的投篮
        get(vball)#对应投球速度
        get(vangle)#对应投球角度
        get(status_dunk)#对应扣篮键

        v_in=np.array([[status_x*v_max*speed_percentage],
[status_y*v_max*speed_percentage],[status_w*v_max*speed_percentage]])
        move(v_in)#移动是要实时监测的，因为不按的时候为0
```

```
        motor_para_up=vup2motor(v_up*status_up)
        sendmotor(motor_up_plus,motor_para_up)#升降机运动
        if status_dribble:#运球键
            dribble()
        if status_pass:
            pass(vball)
        if status_shot1:
            vball=shot_data1=[0]
            vangle=shot_data1=[1]
        if status_shot2:
            vball=shot_data2=[0]
            vangle=shot_data2=[1]
        if status_shot3:
            vball=shot_data3=[0]
            vangle=shot_data3=[1]#获得预设投篮值
        if status_shot:
            shot(vball,vangle)
        if status_dunk:
            push()
            delay(1000)#确保完全伸出
            motor_para1=vangle2motor(90)
            sendmotor(motor_leader,motor_para1)
            delay(700)#确保球被夹住
            motor_para1=h2motor(h_up_max_plus)#伸缩杆伸出后上升最高h_up_max_plus,未伸出最高h_up_max
            sendmotor(motor_up_plus,motor_para1)
            jump()#跳起
            delay(t_jump)#根据跳起到最高点所用的时间进行延迟，并且同时需要考虑程序运行机器反应的时间
            shot(vball,0)#水平抛出,vball要根据其距离篮筐的距离，跳起的最高距离等等而定
```