1 算法总览

机器人预实现的功能：移动，原地转向，运球(捡球抬升一定高度并且能够接住)，传球，投篮(三个参数选择)。
根据对机器人的设计，其中的手动与自动的功能如下：

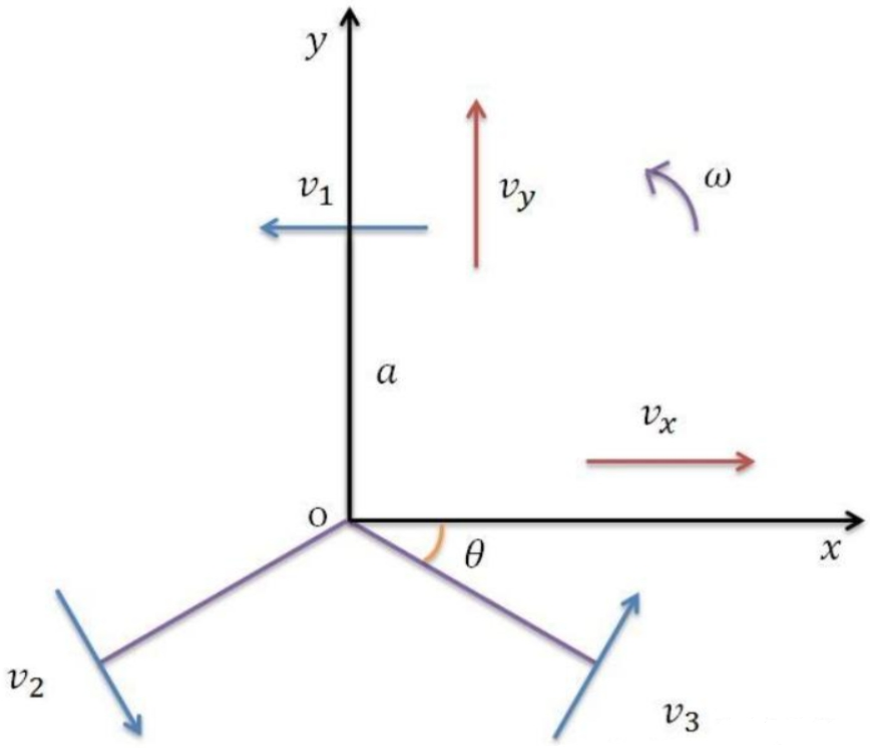
手动部分：在操作者的控制端，可控制的按键有：①前后左右移动，②顺逆时针选择，③运球按键，④三个预设的投篮参数按键，⑤传球按键，⑥调速，⑧投球角度旋钮，⑦投球速度旋钮，⑧投球器高度位移

(加一张图片展示操纵的界面)

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。

2 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。



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$(\theta = \frac{\pi}{6})$

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

求逆可得到另一种转换(但是用不到)

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \begin{bmatrix} -\frac{2}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & -\frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} \\ \frac{1}{3a} & \frac{1}{3a} & \frac{1}{3a} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

每个轮子都配有一个电机，电机参数为：

由于三个速度互相不独立，同时为了简化操作，设置调速度时三个速度同时按同比例调整。基于python的算法框架如下

```
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 v2motor(v):#由于三个轮子的电机是完全相同的，其速度与电机转速直接的转换函数也是相同的
    ...
    return motor_para
def sendmotor(motor,motor_para):#将电机参数传给电机的函数
    ...
    return
def get(*add,value):#得到操纵端相关值输出的函数
    ...
    return value

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)
```

3 运球算法

运球涉及两个电机，分别负责容器的开闭与容器的升降。

负责容器开闭的电机为
负责升降的电机为

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 传球算法

传球涉及到了投球器，所需要的电机有三个：控制投球器升降的电机，摩擦轮的电机，控制引导器伸出长度的电机

控制投球器升降的电机与控制容器的电机相同
摩擦轮的电机为
控制引导器伸出长度的电机为

过程具体为：投球器下降，引导器回收夹住球，摩擦轮加速然后引导器伸出
其中投球器下降是需要手动实现的，其他的只需按一下传球键即可

对于引导器，当其完全回收时传球角度为90°，完全伸出时角度为0°。

```
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)
```

5 投篮算法

投篮涉及的算法与传球大致上一样，只是参数不同，而且需要计算抛物线。这里略去计算抛物线的过程，因为过两点的抛物线有无数条，但是最合适的数据需要经过大量实验才能确定。投篮时固定投篮器上升到最高。

```
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]#同时讲操作界面的后端数据也修改，防止循环结束后数据被刷新
掉
        motor_para_up=vup2motor(v_up*status_up)
        sendmotor(motor_up_plus,motor_para_up)#升降机运动
        shot(vball,vangle)
```

6 扣篮算法

需要两个电机，一个用来解锁弹簧腿弹起的电机，一个用来将升降杆拉伸的电机。先将升降杆延长，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
```

7 main函数

```
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
```