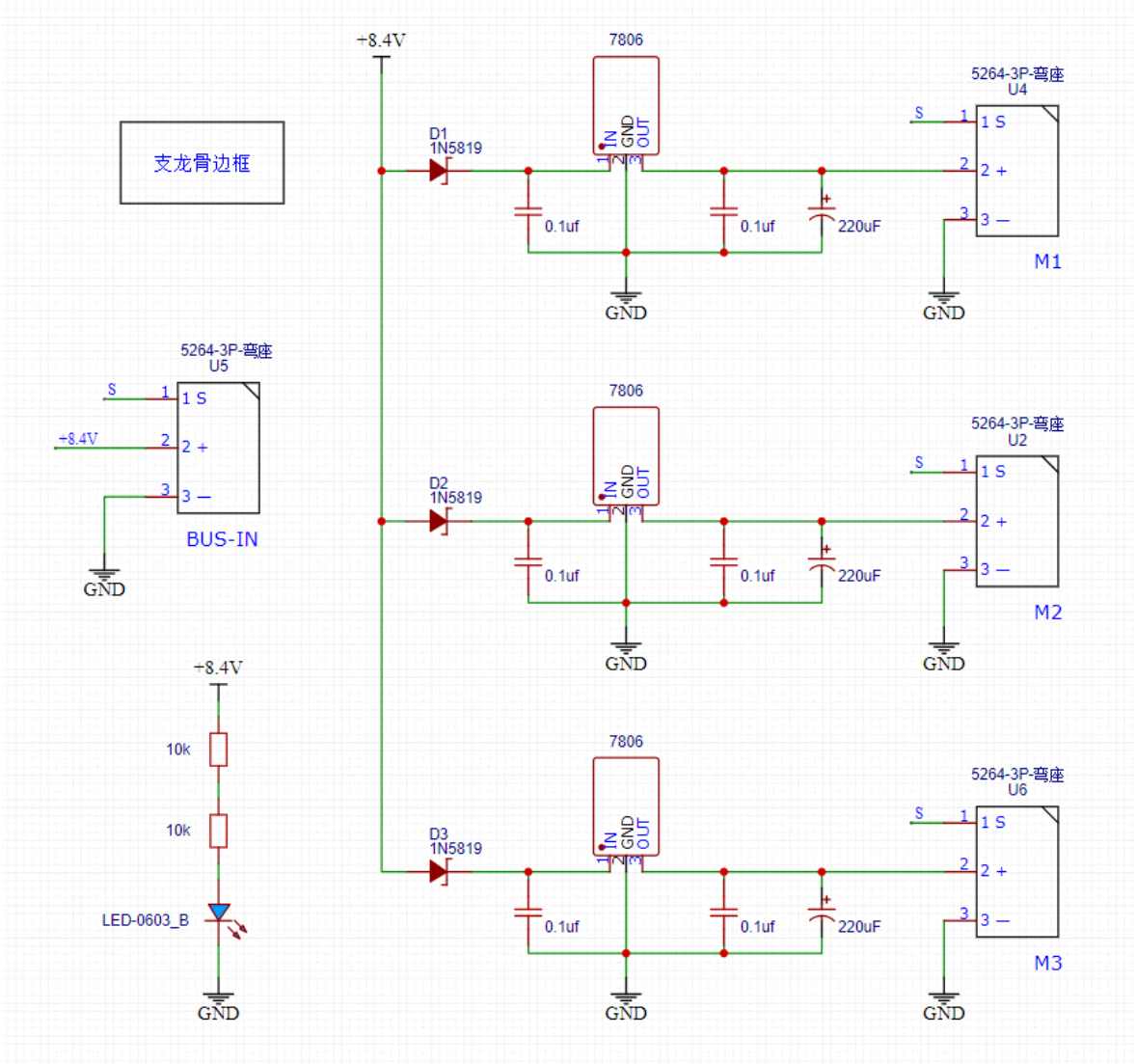
舵机电源管理

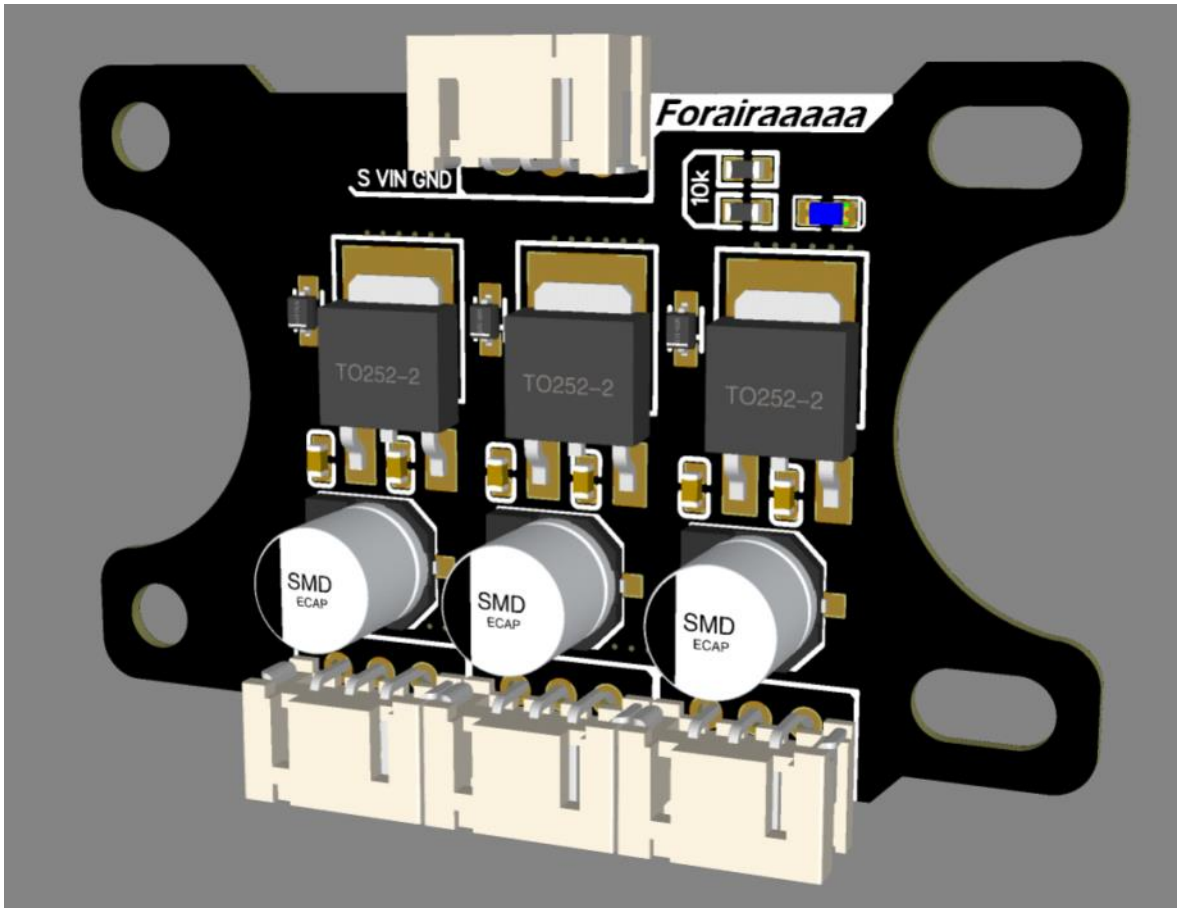
2020年9月24日　　21:15

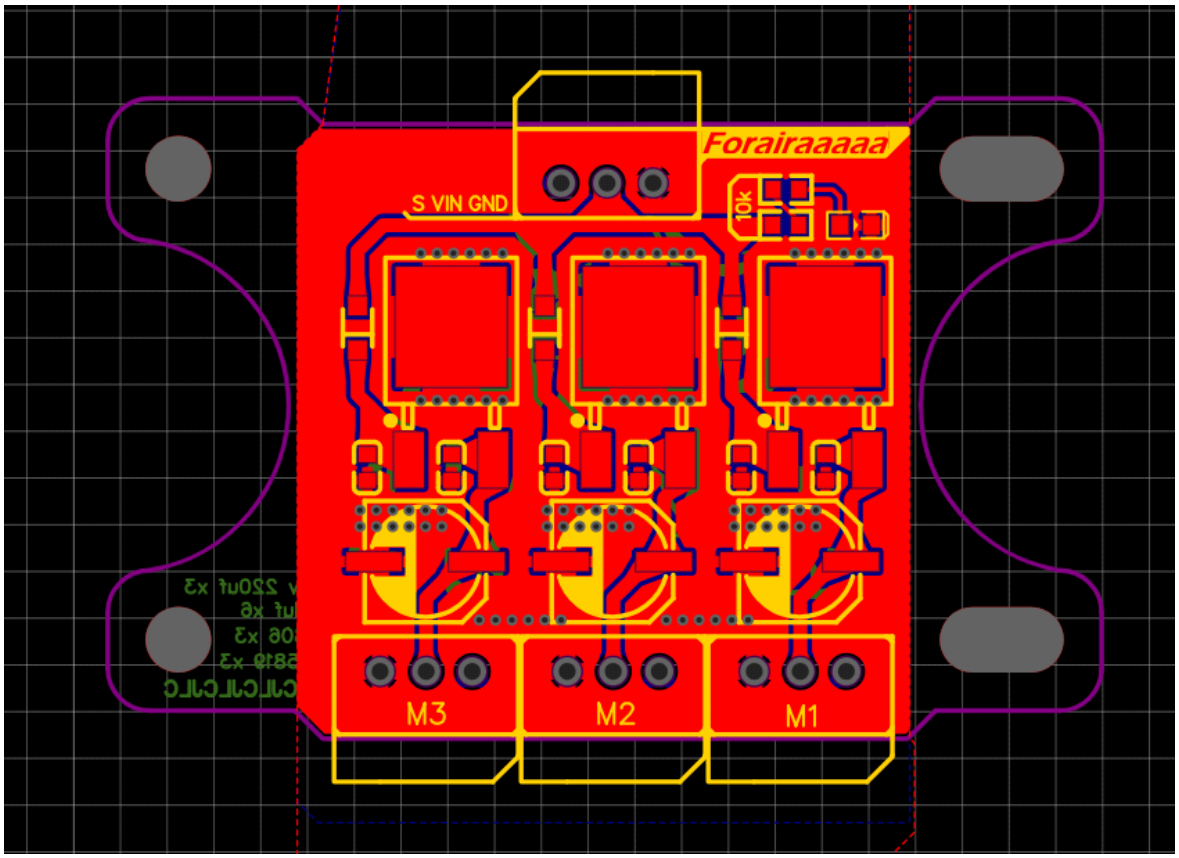
放弃大功率降压模块，分而治之。

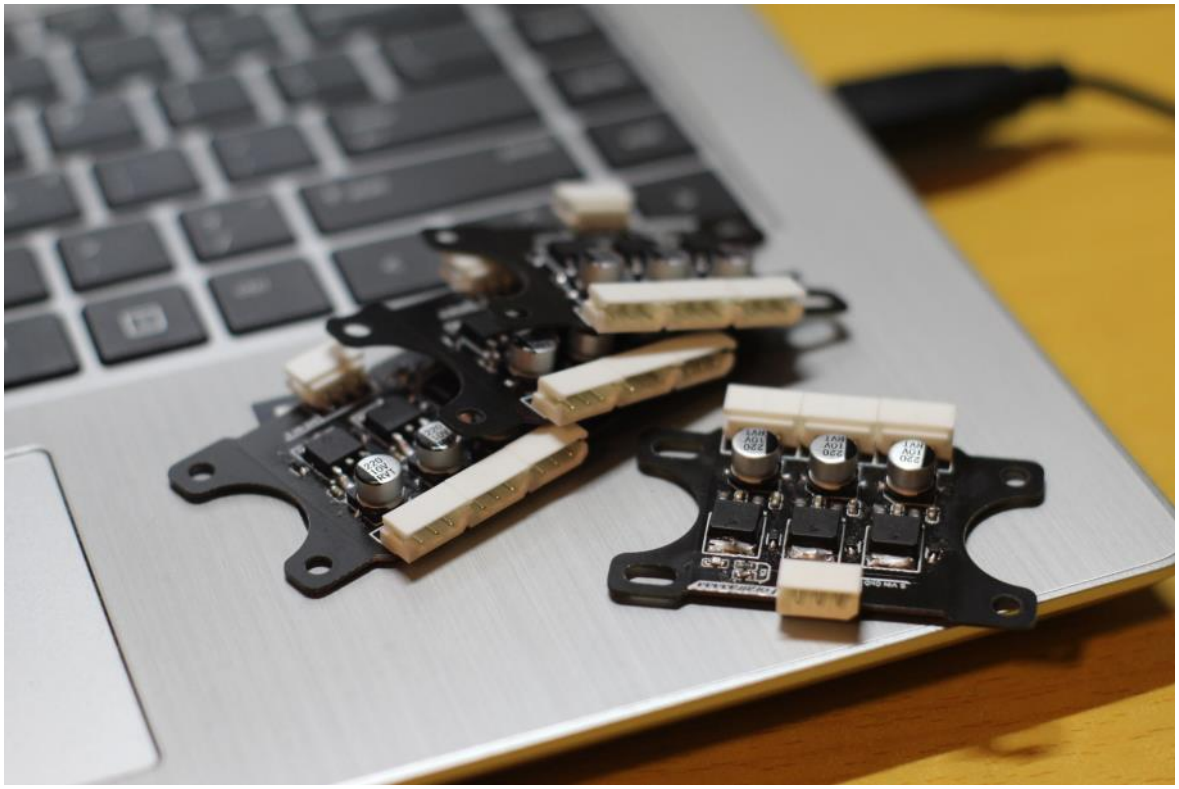
每组三只，固定在每条腿总成的龙骨间隙，总线连回总电源管理模块

并联输入，但输出相对独立，不知有无效果
二极管好像无卵用，反而压降0.2v，一定程度限制了负载功率，问题不大
大面积接地，不好焊接




Forairaaaaa
S VIN GND
10k
TO252-2
TO252-2
TO252-2
SMD
ECAP
SMD
ECAP
SMD
ECAP


Forairaaaaa
S VIN GND
10k
M3
M2
M1

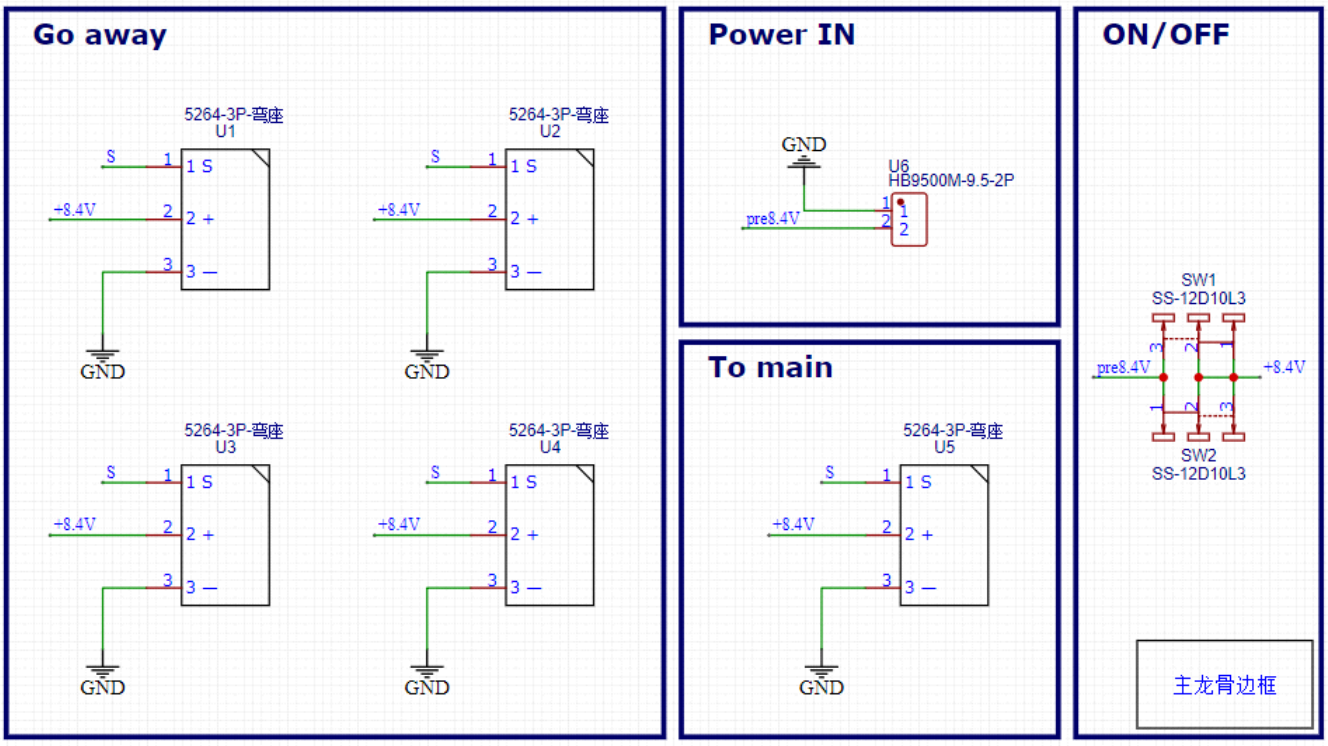


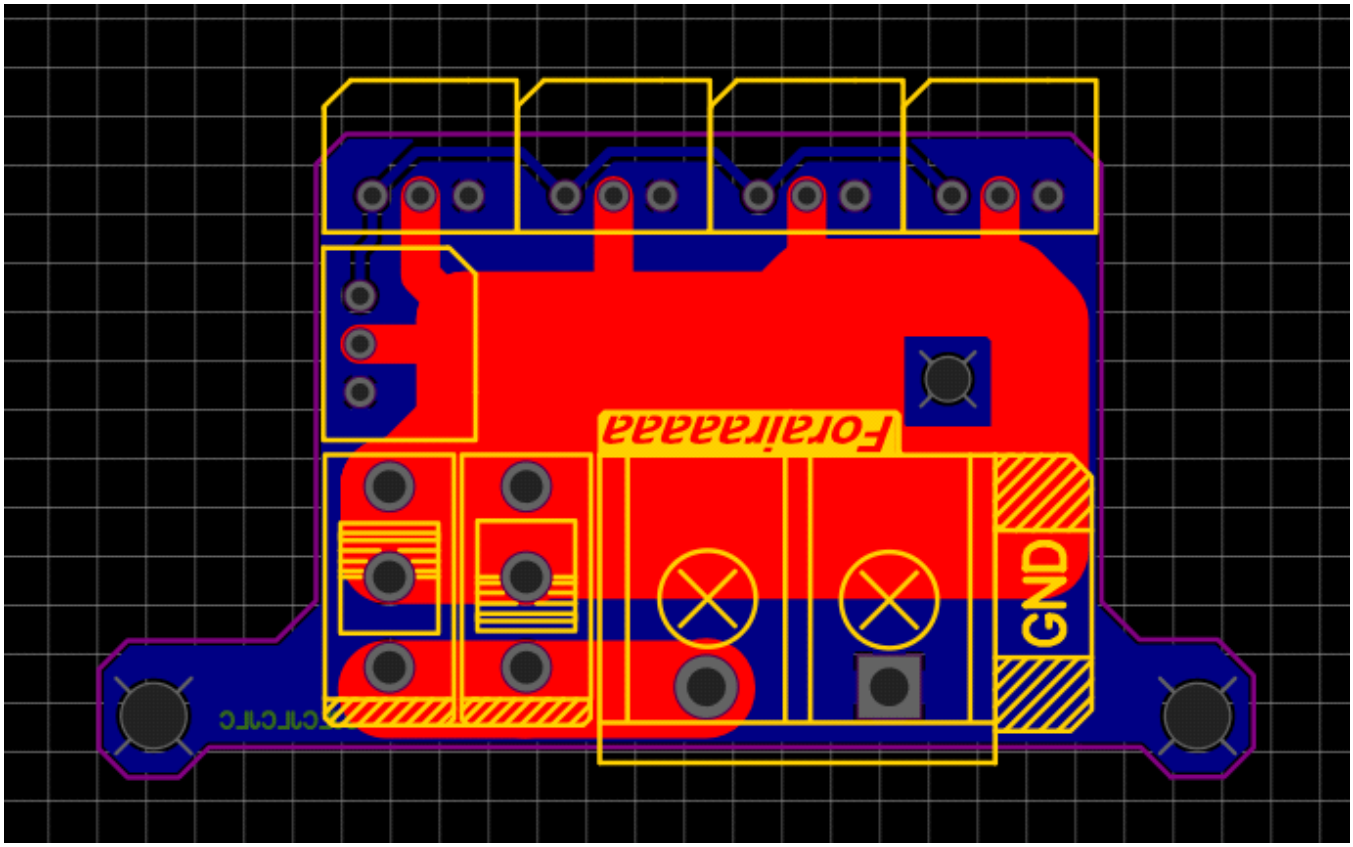
总电源管理

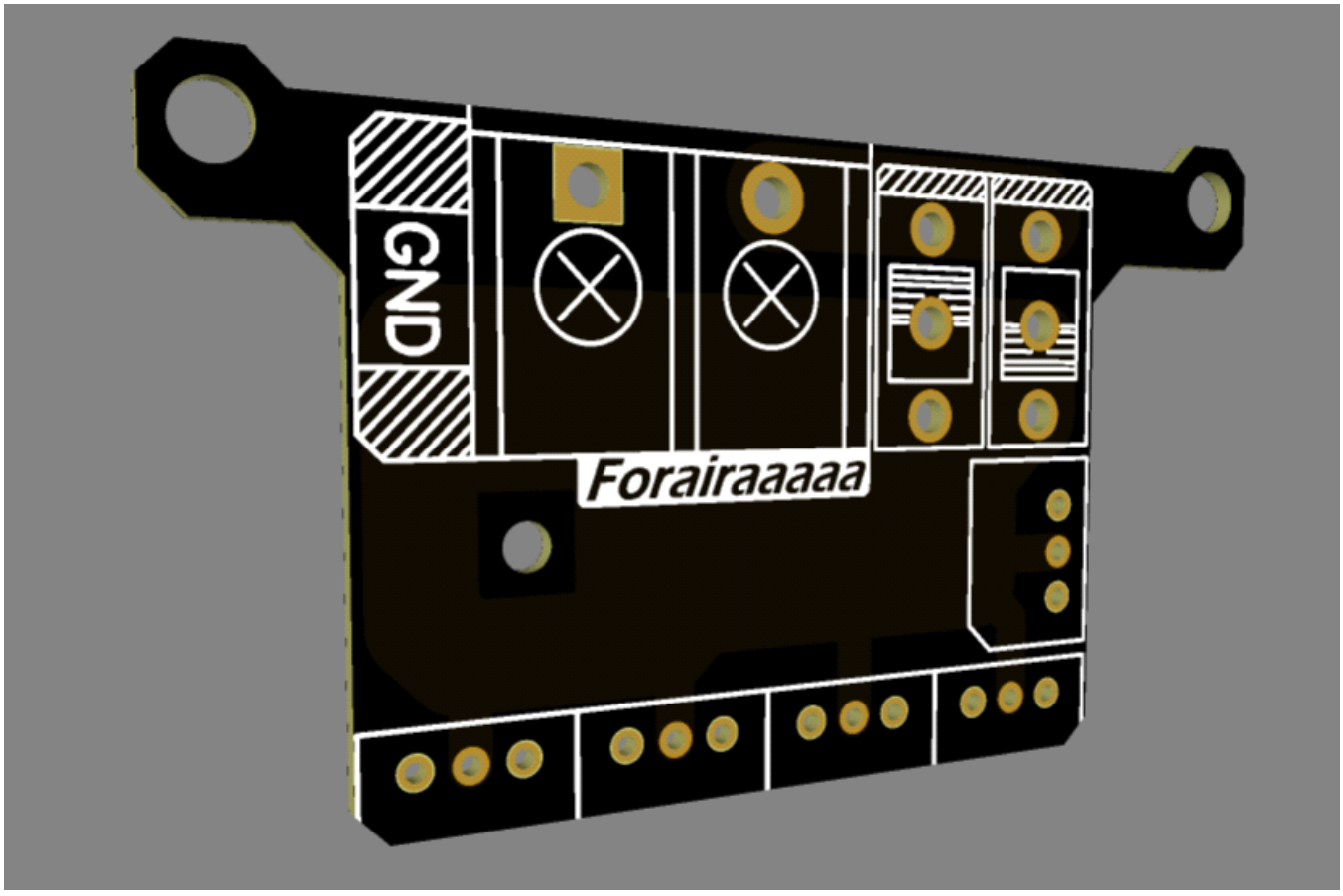
2020年9月24日　　21:45

总电源输入，分接舵机电源模块及主板模块

模块只作电源及信号走线，不作任何电压处理（体积限制，淦）
为什么两个开关？　钱多







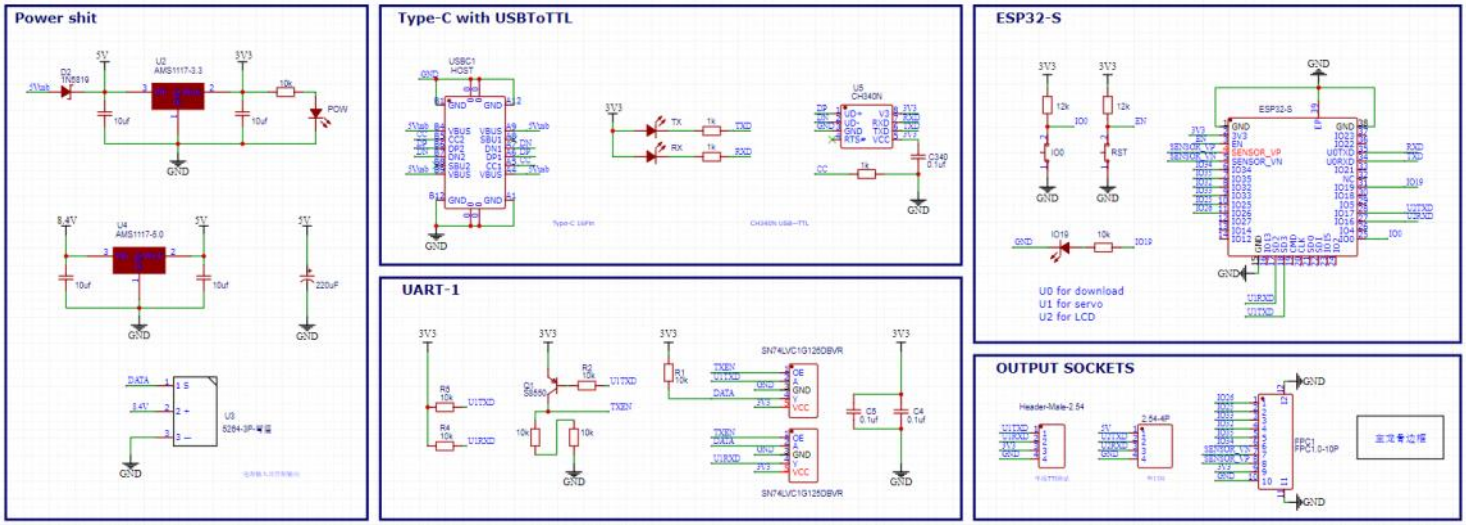
米老鼠？

主板

2020年9月25日　11:53

ESP32-S

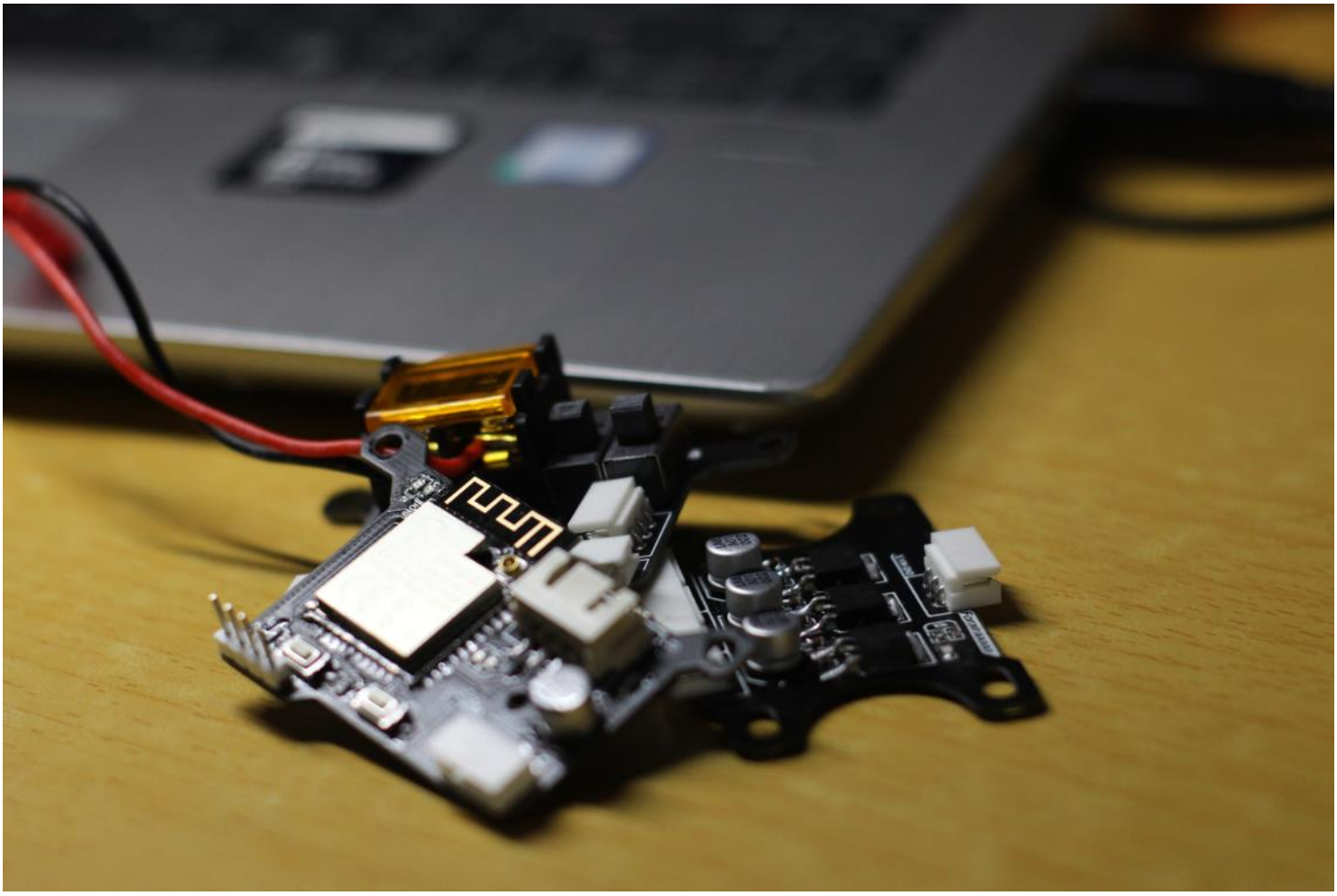
串口0用于下载，串口1转半双工控制舵机，串口2用于扩展
同时引出多个引脚于FPC1.0mm，但排线不好接



帅的一

设计缺陷：复位键基本按不到，IO0键不好摁

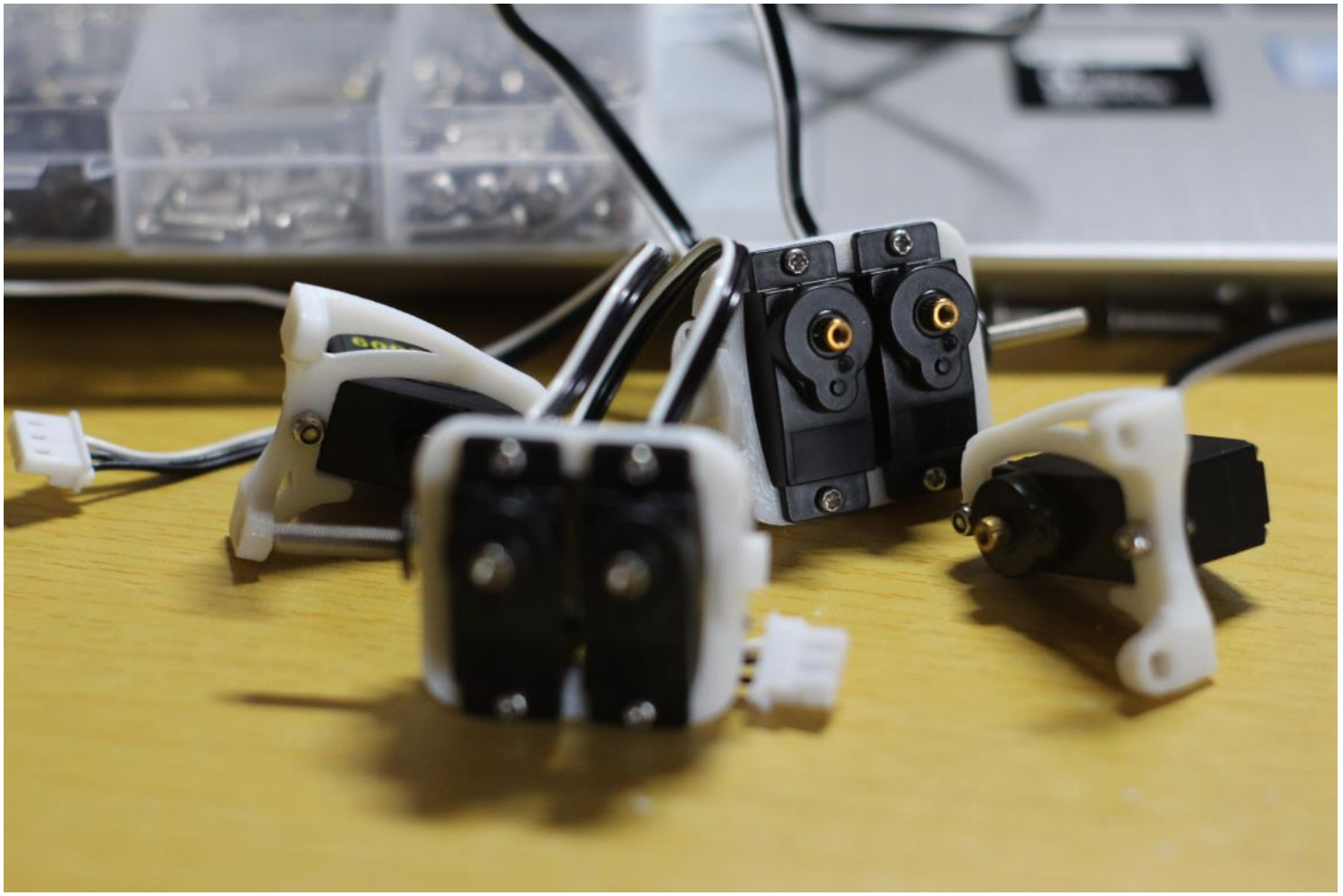

tab
caps lock
shift
fn lock
ctrl
fn
Q
A
Z
X
C
V
B
T
Y
G
H



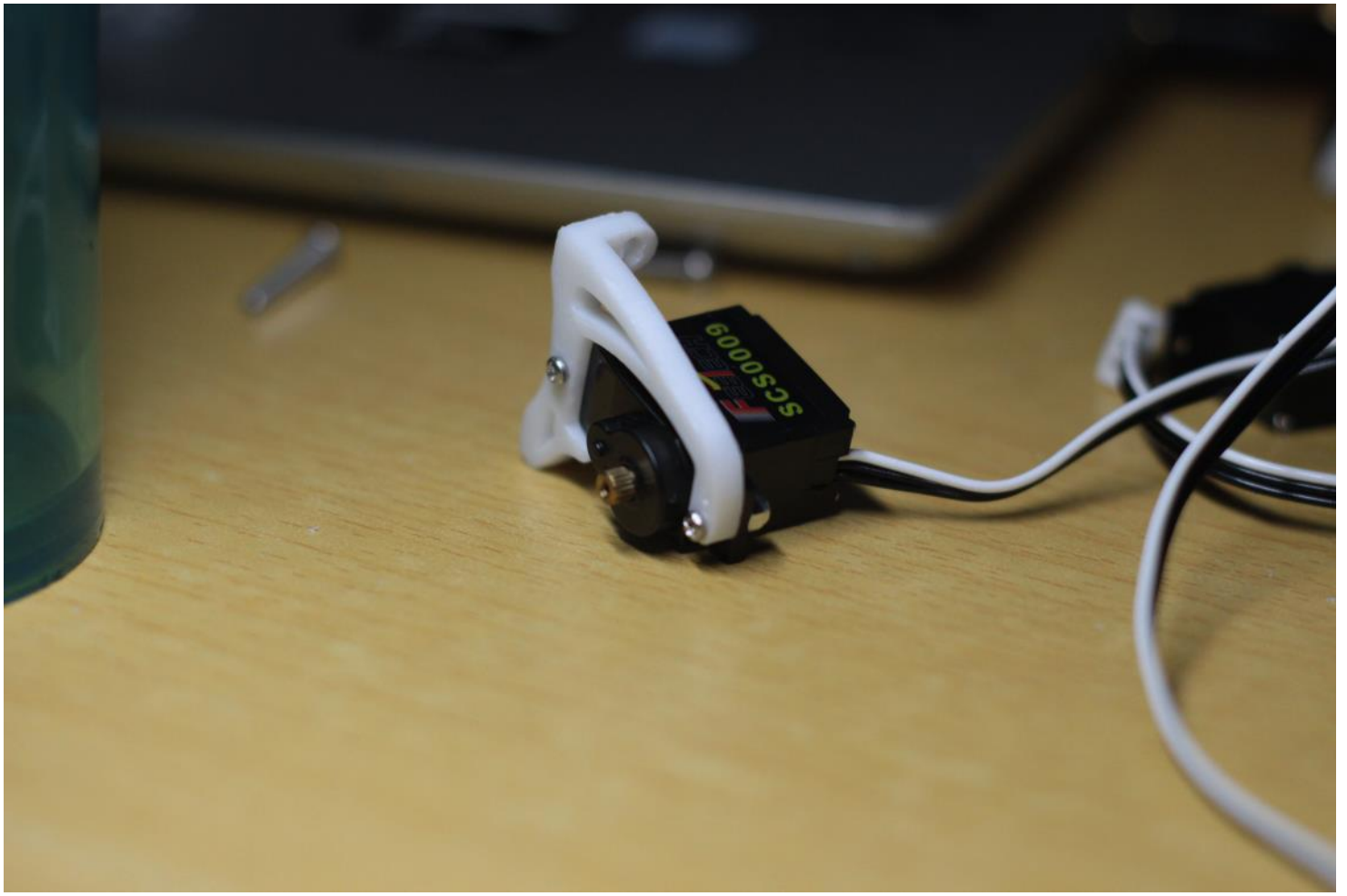
并联腿

2020年10月7日 11:30

碳纤维太贵了，下次亚克力



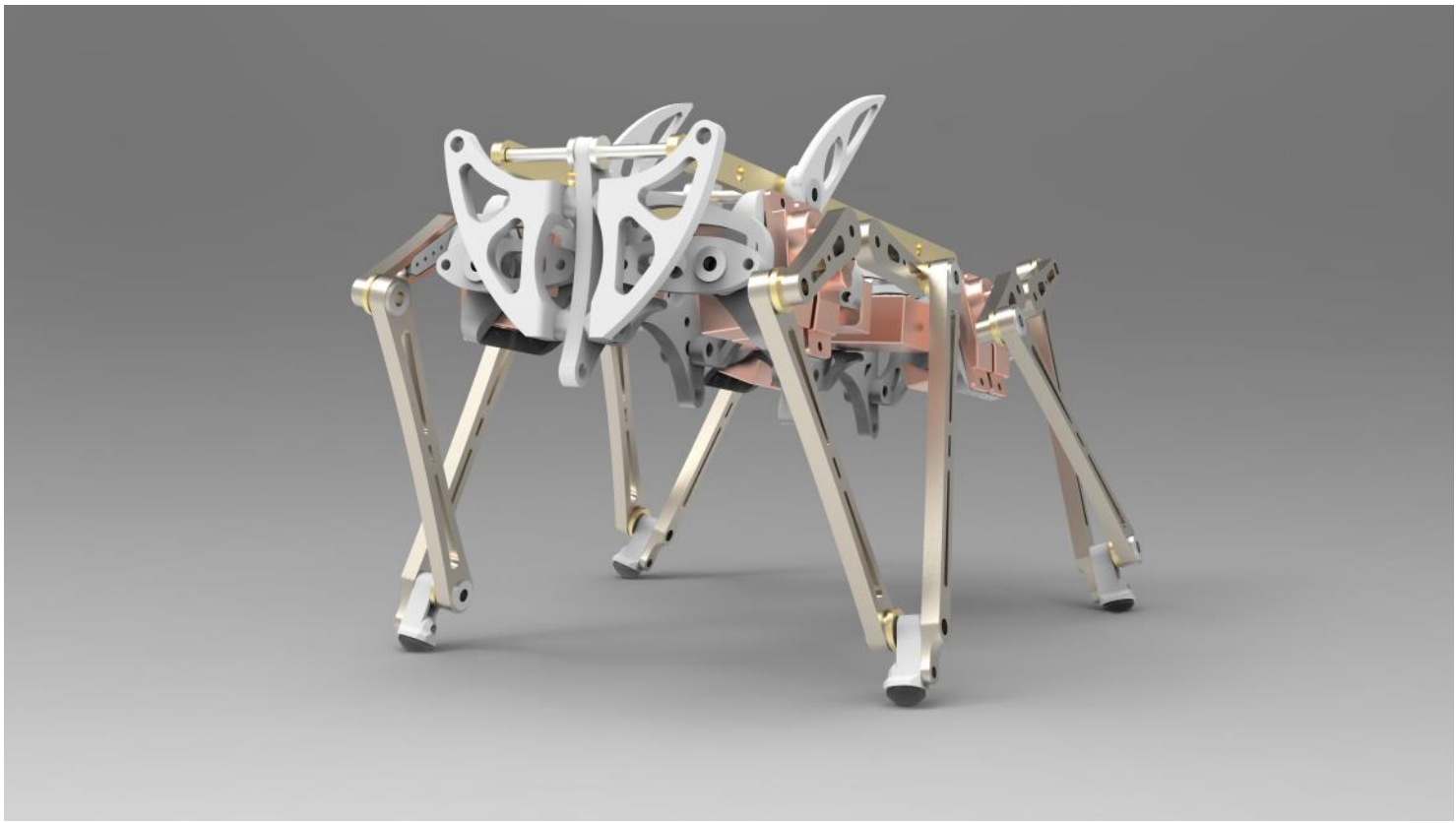
等有想法改成串联



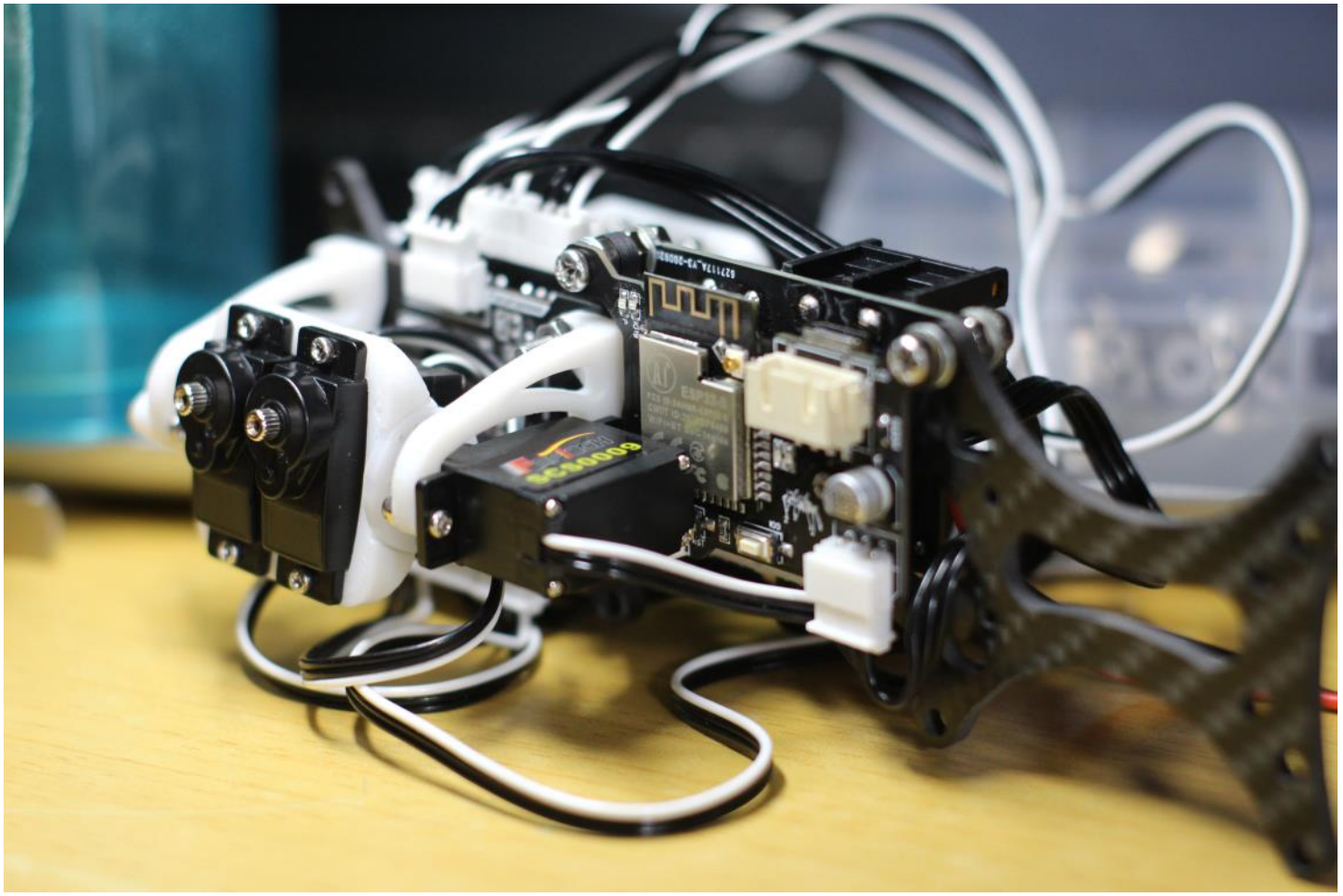
整体机构

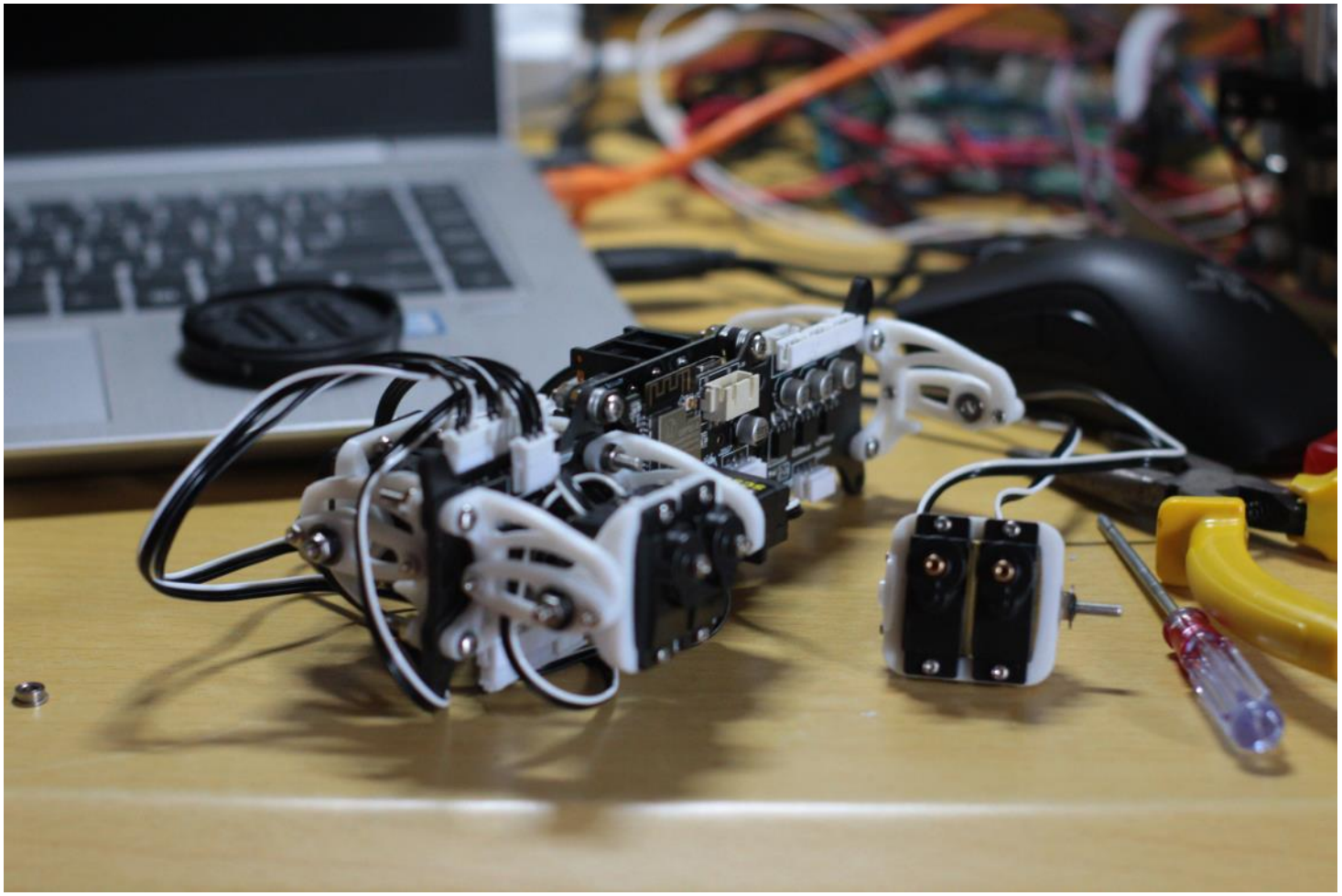
2020年10月7日 11:35

此机构组装复杂，维护麻烦，精度不好控制，空间小，小学生作品代表~

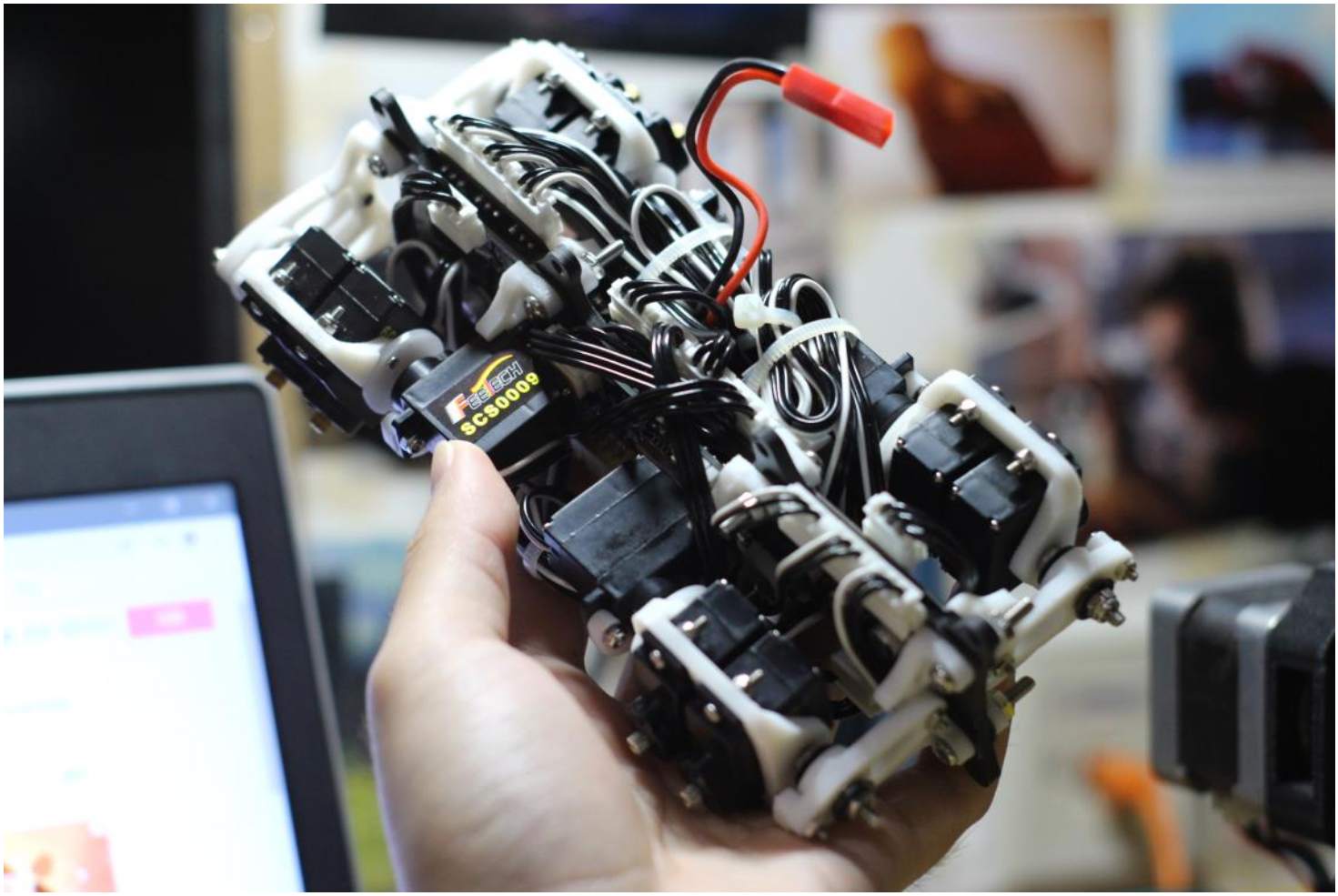


错综复杂

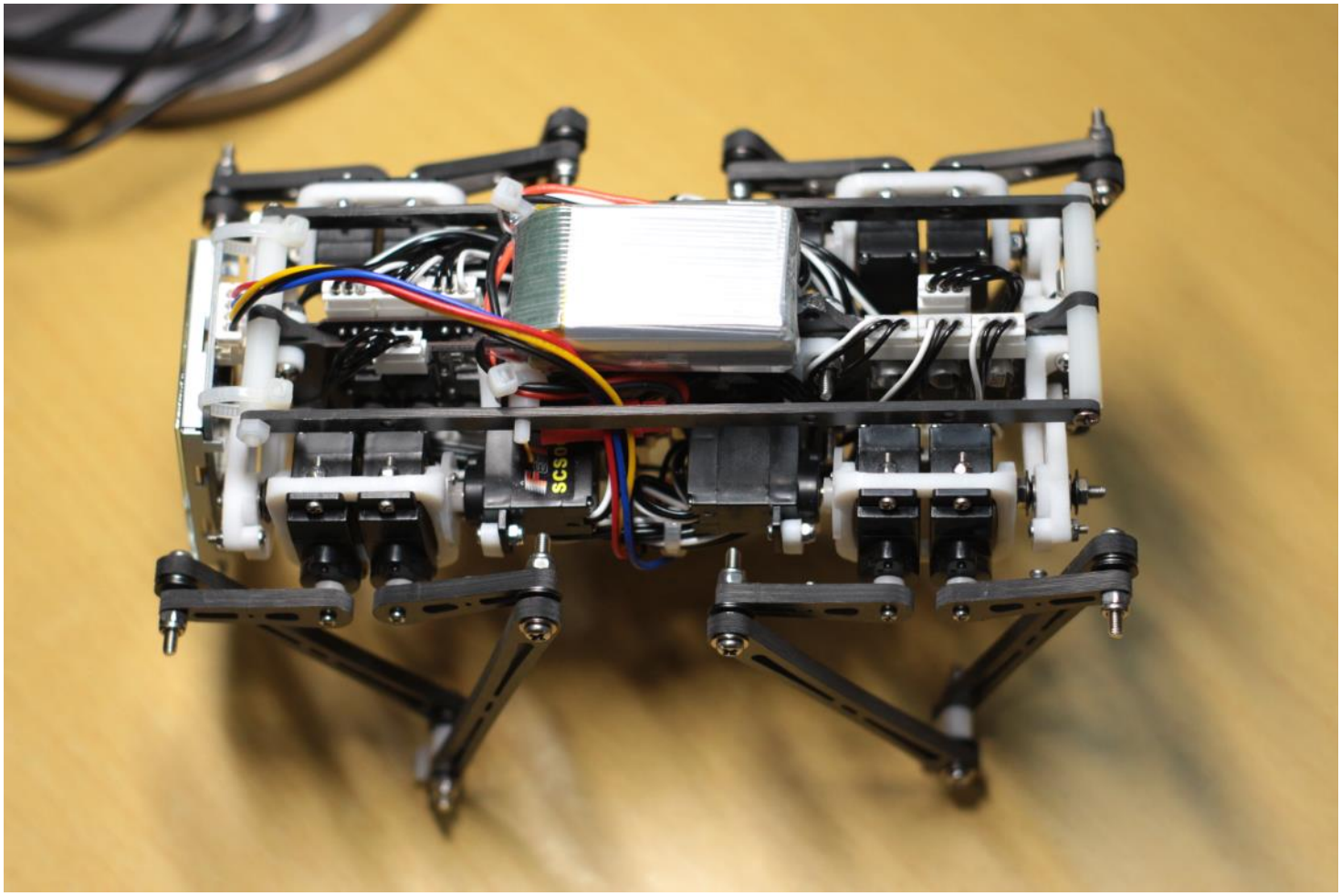


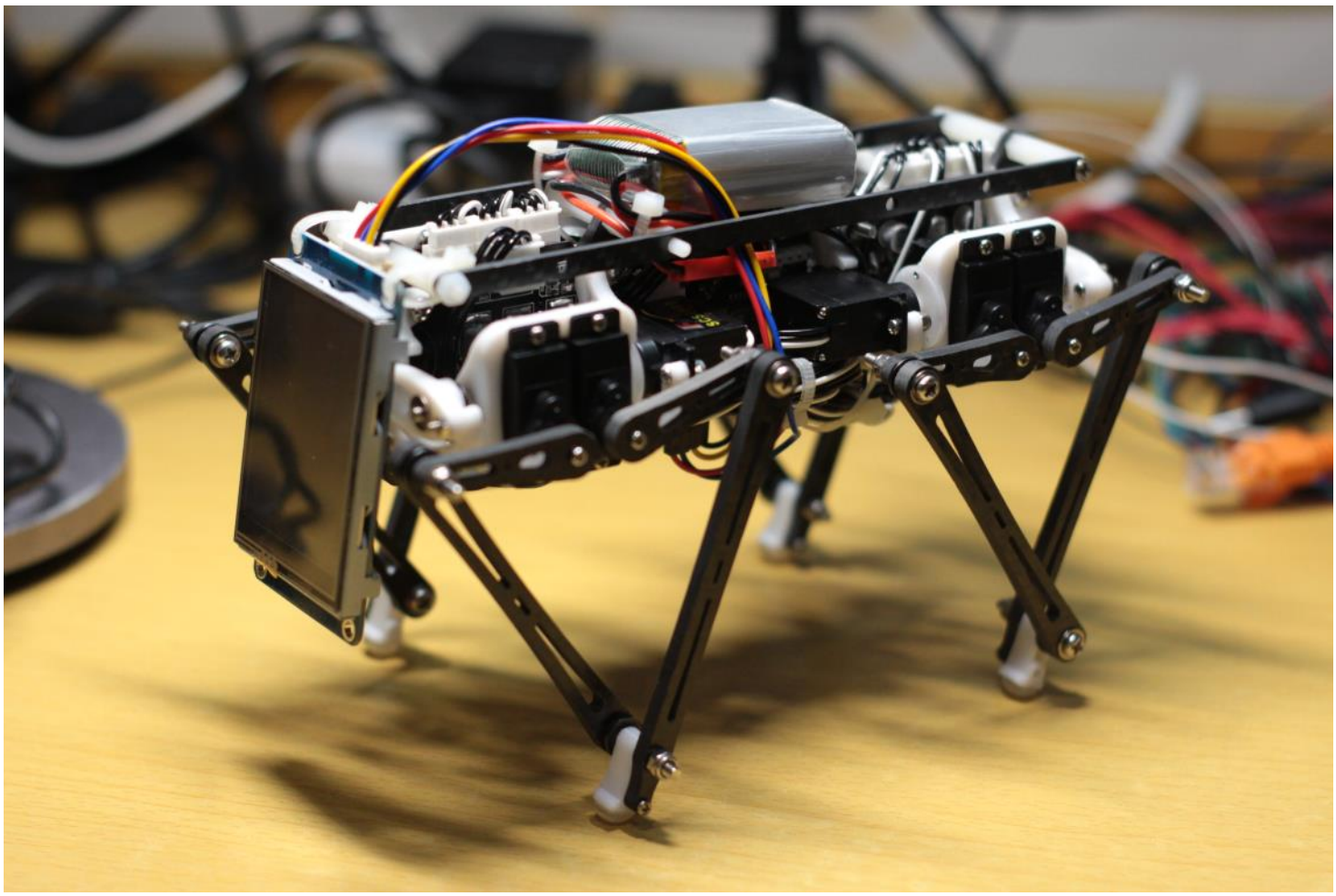


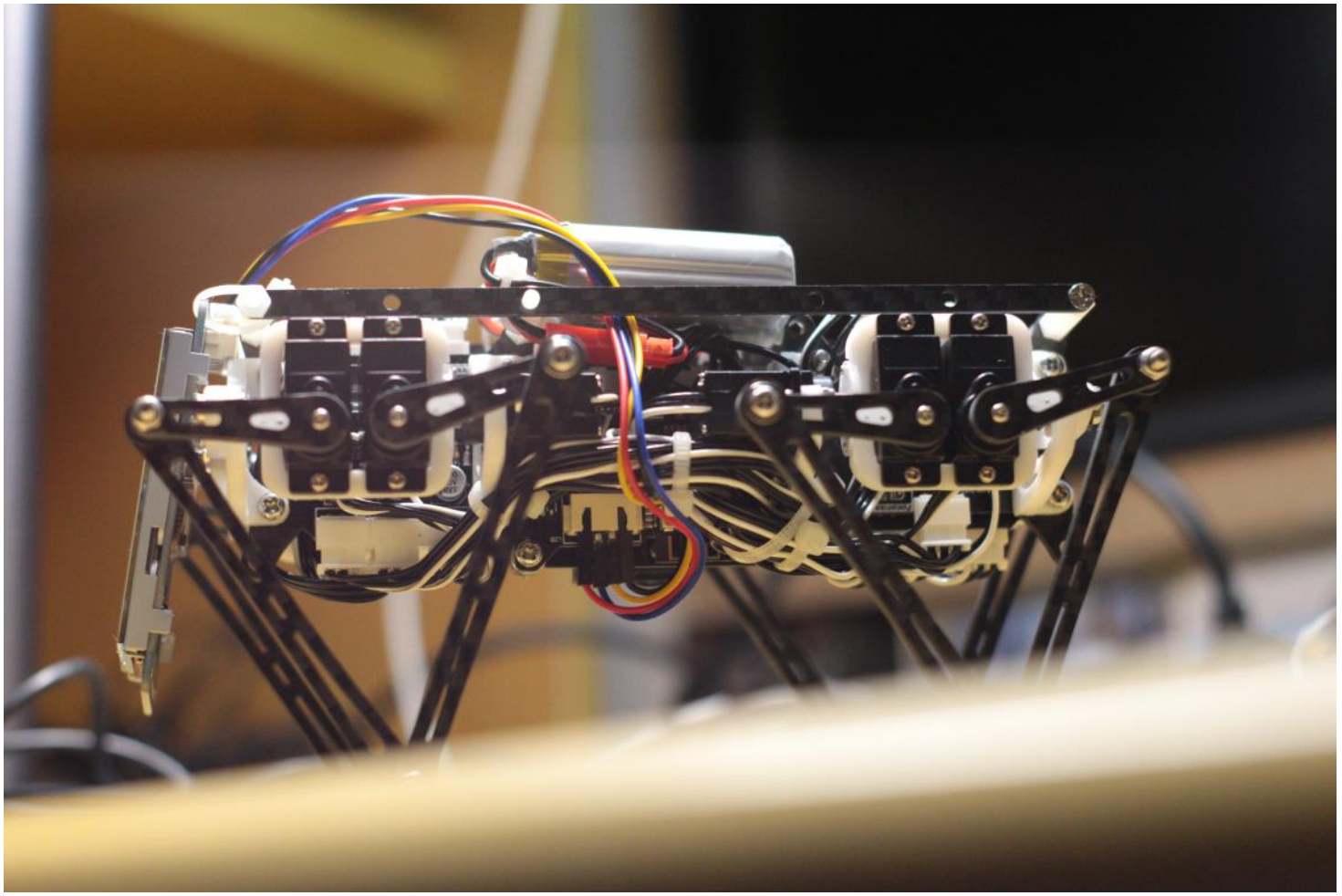
艰难理线



串口屏





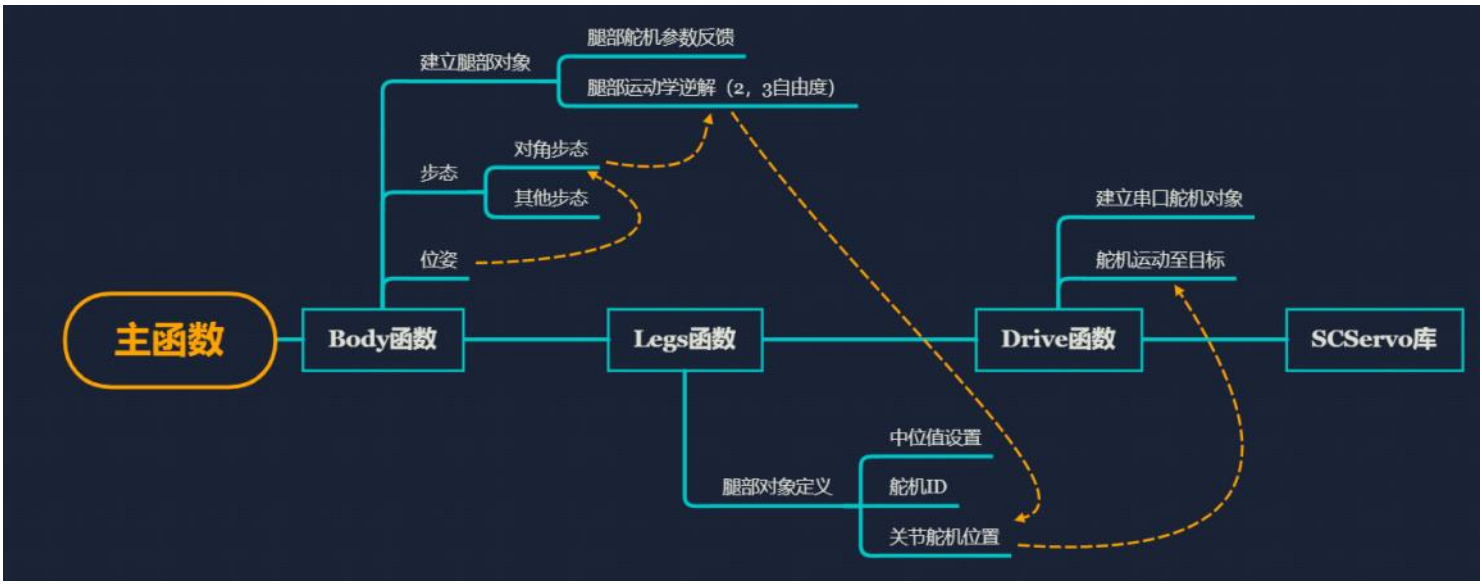


仍需设计防水挡板等，淦

程序结构

2020年10月7日 11:43

主函数提供位姿



逆解—平面2自由度并联臂

2020年10月7日 12:06

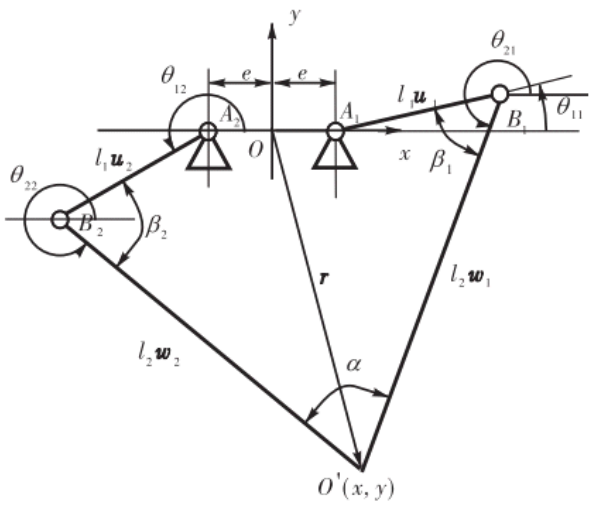
参考《一种二平动自由度并联机械手动力尺度综合》

$$\boldsymbol{u}_i = \begin{pmatrix} \cos\theta_{1i} & \sin\theta_{1i} \end{pmatrix}^{\mathrm{T}}$$

$$\boldsymbol{w}_i = \begin{pmatrix} \cos\theta_{2i} & \sin\theta_{2i} \end{pmatrix}^{\mathrm{T}}$$

$$\boldsymbol{e}_1 = \begin{pmatrix} 1 & 0 \end{pmatrix}^{\mathrm{T}}$$

$$\mathrm{sgn}(i) = \begin{cases} 1 & i = 1 \\ -1 & i = 2 \end{cases}$$



$$\theta_{1i} = 2\arctan\frac{-E_i + \mathrm{sgn}(i)\sqrt{E_i^2 - G_i^2 + F_i^2}}{G_i - F_i} \qquad (2)$$

式中：$E_i = -2l_1 y$；$F_i = -2l_1\left[x - \mathrm{sgn}(i)e\right]$；$G_i = x^2 + y^2 + e^2 + l_1^2 - l_2^2 - 2\,\mathrm{sgn}(i)ex$.

一种二平动自由度并...

python建立数学模型：

```python
# 主动臂偏转角
theta1 = 0
theta2 = 0

# 各臂长度
e  = 7
l1 = 30
l2 = 80

# 符号函数
def sgnShit(i):
    if(i==1):
        return 1
    elif(i==2):
        return -1
    else:
        return 0

# 逆解函数
def calculation(x=0, y=0, i=1):
    # 坐标系平移
    x = x+5
    y = y-70

    E = (-2)*l1*y
    F = (-2)*l1*(x-(sgnShit(i)*e))
    G = (x*x) + (y*y) + (e*e) + (l1*l1) - (l2*l2) - (2*sgnShit(i)*e*x)
    print(E, F, G)

    theta = 2*(np.arctan(((-1)*E + sgnShit(i)*(np.sqrt(E*E - G*G + F*F))) / (G - F)))
    print(theta)

    return (theta*180)/np.pi

x = 0
y = 30
print(calculation(x=x, y=y, i=2),calculation(x=x, y=y, i=1))
```

并联平面
2自由逆解

8自由度足端轨迹规划

2020年10月7日 12:11

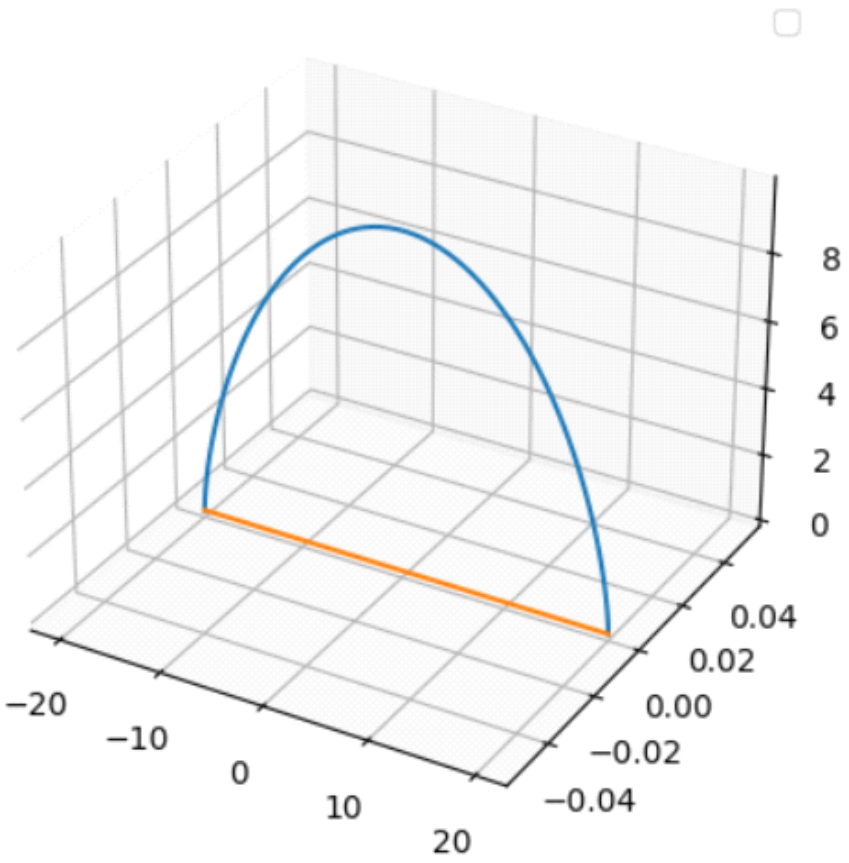
参考：https://zhuanlan.zhihu.com/p/69869440

其中 T_s 为步态周期， λ 为支撑相占空比。则通过调节增益参数期望落脚点坐标就会随着机器人实时速度的变化而改变，即速度越大步幅越大，速度越小步幅越小，速度为零时则为原地踏步。对于 z_f 来说则可直接等于该腿当前的期望高度 $z_f = z_{\exp}$ ，则摆线轨迹如下：

$$\begin{cases} x_{\exp} = (x_f - x_s)\frac{\sigma - \sin\sigma}{2\pi} + x_s \\ y_{\exp} = (y_f - y_s)\frac{\sigma - \sin\sigma}{2\pi} + y_s \\ z_{\exp} = h\frac{1-\cos\sigma}{2} + z_s \end{cases}$$

其中 $\sigma = \dfrac{2\pi t}{\lambda T_s}, 0 < t < \lambda T_s$ ，则在获取跨腿轨迹后可直接使用各时刻足尖位置采用运动学逆解计算出对应的摇臂角度。如果添加了着地传感器则在跨腿过程中检测到触地，该腿立刻停止并切换到支撑相参与VMC的反馈控制。

python数学模型：



摆线方程测试

8DOF应用层

2020年10月12日 20:30

应该不错的步态设置：

```
for(i=0;i<=1;i+=0.01){
    TROT_DataSet(1,-20,3,12);
    TROT_DataSet(2,-20,3,12);
    TROT_DataSet(3,-20,3,12);
    TROT_DataSet(4,-20,3,12);
    Gait_TROT_FromSet(i, 1);
}
```

无线串口

2020年10月13日 19:10

波特率：

AT+BAUD=7 // 115200

上位机端：

AT+RXA=0x01,0x02,0x03,0x04,0x05 // 接收地址
AT+TXA=0x01,0x02,0x03,0x04,0x06 // 发送地址

接收机端：

AT+RXA=0x01,0x02,0x03,0x04,0x06 // 接收地址
AT+TXA=0x01,0x02,0x03,0x04,0x05 // 发送地址