浙 江 大 學

《机电系统实验》

题　　目：六自由度机械手控制系统

姓名学号：曾逸轲 **3160104377**

专　　业：机械工程

指导教师：高宇

上课时间：周二 下午

目录

1.实验目的及实验要求

1.1 实验目的

1）了解 ROS 系统的工作原理；

2）了解机械臂的机械系统和控制系统的组成、工作原理；

3）了解机械臂正反解计算及具体位置标定方法；

4）掌握机械臂轨迹驱动方法；

5）掌握相关软件编程方法及硬件连接原理。

1.2 实验要求

1）使用 SolidWorks 设计合适的工位以固定零件和检测元件；

2）选择和设计合理的机械臂抓手；

3）编写 PC 端节点程序，完成与 arduino 端的通讯以及对机械臂的控制；

4）通过编写 arduino 端节点程序，对继电器的控制以及对传感器信号的采集；

5）撰写实验报告。

2.实验器材

1）seed 六自由度机械臂（含示教器）；

2）气泵；

3）电磁阀；

4）继电器；

5）arduino mega 2560 主控板以及拓展板；

6）三爪气动卡盘；

7）上位 PC 机；

8）其他自制机械结构件。

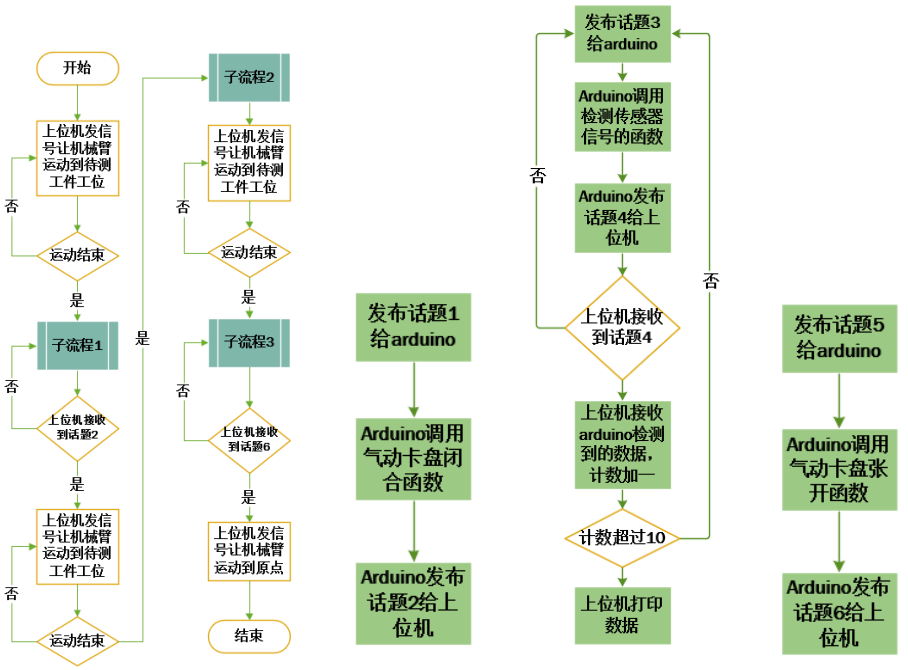
3.预期实验结果

通过达成各项实验目标，完成上位机对六轴机械臂的控制，并且使其按照特定的轨迹运行到待测工件工位处，夹取待测工件后运行至检测工位，由 arduino 读取传感器信号并传递给上位机打印，以判断待测工件是否合格。

整个实验的控制思路和控制流程图如下：上位机发信号让机械臂运动到第一个工位准备抓取工件——上位机发信号给 Arduino 主板进而控制电磁阀，通过三爪气动卡盘抓起待测工件——上位机发信号让机械臂运动到检测传感器所在的工位正上方——一共设定 10 个测定角度，每到达一个角度之后，将 Arduino 主板接收到的来自传感器的信号传给上位机——待 10 个位置测量完毕之后，上位机发送信号使机械臂运动到第一个工位，松开三爪气

动卡盘，将待测工件放回原位——上位机发送信号让机械臂运动到初始位置，完成一次测量。

实验控制流程图如图 1 所示：



(a)控制流程　(b)子流程 1　(c)子流程 2　(d)子流程 3

图 8 控制流程图

4.实验内容

4.1 了解机械臂

4.1.1 机械臂介绍

六自由度机器手是由六个关节组成，每个关节上安装一个电动机，通过控制每个电动机旋转，就可以实现机械手臂的空间运动。本实验使用的是 Seed 六自由度机械臂，它能实现物品的抓取和移位的机械自动控制。该六自由度机械手臂的底座能进行大角度转动，实现机械抓取物体的移位；关节的俯仰和摆动能实现机械手臂在不同位置抓取物体。

在手腕的末端我们自行设计和安装了一个气动抓手，抓手与气泵相连后具有开闭能力，通过控制装置能实现物体的抓取和放下。

机械手的结构如下图 2 所示：

系统共有 6 个自由度，分别是：机体的回转、连杆一转动、连杆二转动、手腕一转动、手腕二转动和机械抓手旋转。

前面三个关节确定手部的空间位置，后面三个关节确定手部的姿态。

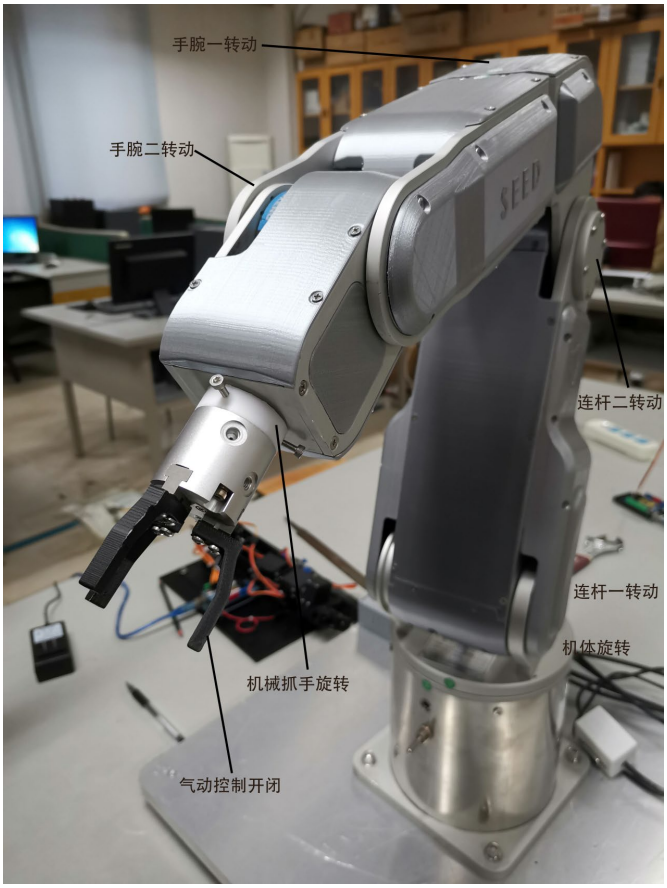


图 2 机械手结构示意图

4.1.2 示教器介绍

示教器的基本控制以及介绍如下图 3 所示：

图 3 示教器使用简要说明

4.2 辅助机械结构设计

本实验的机械设计部分的主要工作是设计待检工位、确定夹持方案、设计检测工位和设计整体定位方案。具体的设计要求和解决方案如下图 4 所示：

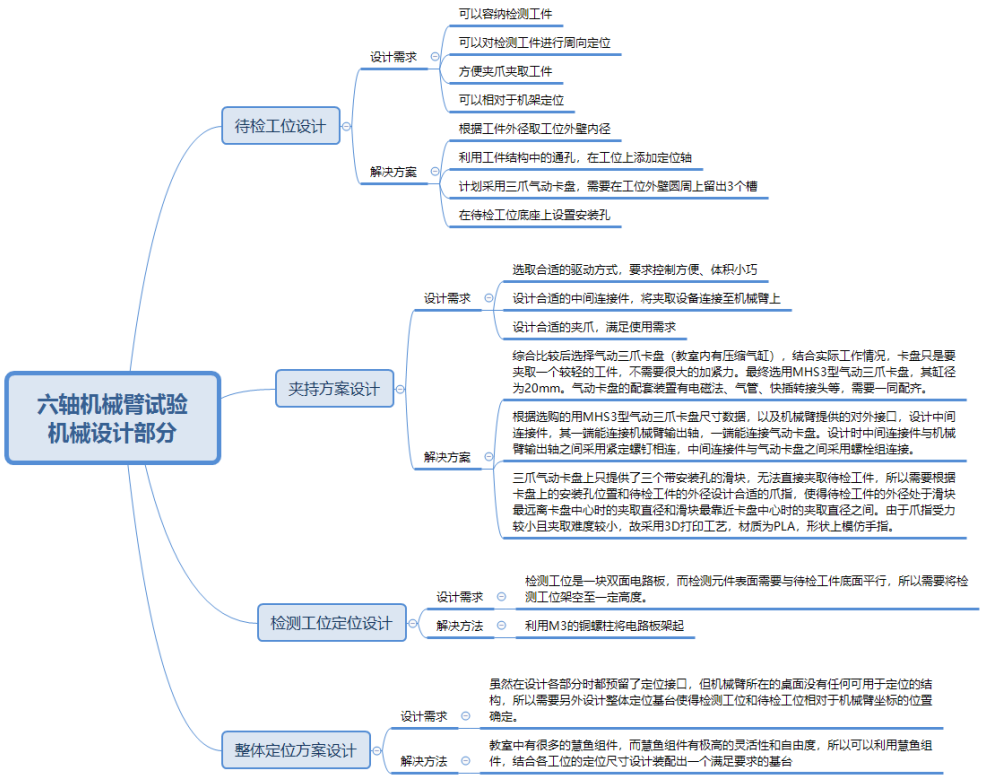


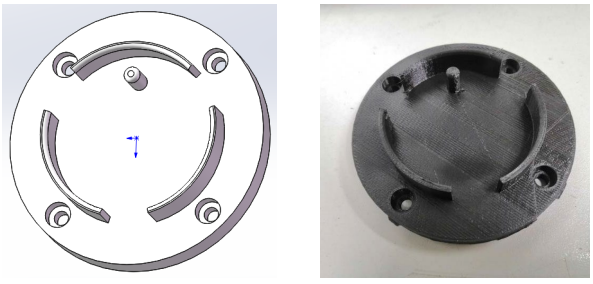
图 4 辅助机械结构设计要求及解决方案

4.2.1 待检工位设计

该工位的设计需求为：1）可以容纳工件；2）可以对工件进行轴向与周向的定位；3）方便夹爪夹取工件；4）可以相对于机架定位。

为满足上述需求，制定的解决方案如下：1）根据工件外径取工位内壁直径；2）利用工件结构中的通孔，在工位上添加定位轴；3）计划采用三爪气动卡盘，需要在工位外壁圆周上留出 3 个槽；4）在待检工位底座上设置安装孔。

具体实施情况如下图 5 所示：



（a）三维模型图　　（b）实物图

图 5 待测工件工位图

待测工件的设计并非一步到位，期间也出现了各种问题，在解决问题的过程中有以下心得：待测工件的工位进行了两次设计，第一次的设计在实际测试时无法正确容纳待检工件，有两点原因，其一是待检工位内壁直径过小，其二是待检工位中的定位轴直径过大无法穿过待检工件。在老师的建议下缩小了待检工位的轴向尺寸，并在待检工位内侧及定位轴上倒圆角，使待检工件能更加顺利地放入工位。

在以后的设计中一定要结合工艺精度设置余量，否则容易出现制造出来的工件尺寸与设计需求不符的情况，同时，添加一些工艺结构会提升使用性能。

4.2.2 三爪卡盘辅助结构设计

三爪卡盘辅助结构的设计需求为：1）选取合适的驱动方式，要求控制方便、体积小巧；2）设计合适的中间连接件，将夹取设备连接至机械臂上；3）设计合适的夹爪，满足使用需求。

针对上述需求提出的解决方案如下：

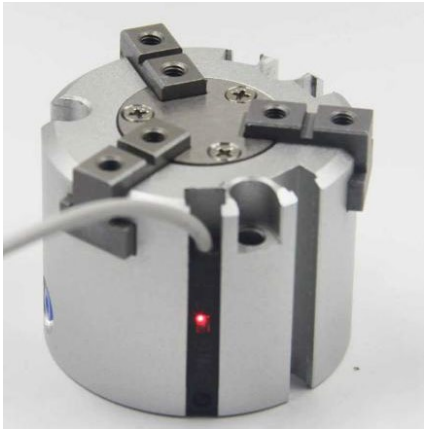
1）综合比较后选择气动三爪卡盘（教室内有气泵），结合实际工作情况，卡盘只是要夹取一个较轻的工件，不需要很大的加紧力。最终选用 MHS3 型气动三爪卡盘，其缸径为 20mm。气动卡盘的配套装置有电磁阀、气管、快插转接头等，需要一同配齐；

2）根据选购的用 MHS3 型气动三爪卡盘尺寸数据，以及机械臂提供的对外接口，设计中间连接件，其一端能连接机械臂输出轴，一端能连接气动卡盘。设计时中间连接件与机械臂输出轴之间采用紧定螺钉相连，中间连接件与气动卡盘之间采用螺栓组连接；

3）三爪气动卡盘上只提供了三个带安装孔的滑块，无法直接夹取待检工件，所以需要根据卡盘上的安装孔位置和待检工件的外径设计合适的爪指，使得待检工件的外径处于滑块最远离卡盘中心时的夹取直径和滑块最靠近卡盘中心时的夹取直径之间。由于爪指受力较小且夹取难度较小，故采用 3D 打印工艺，材质为 PLA。

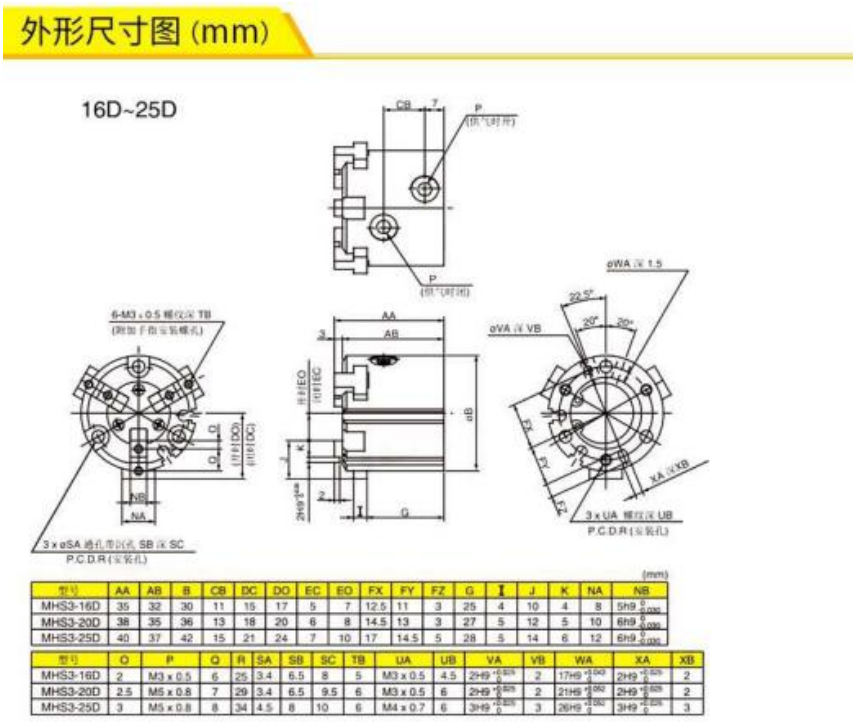
具体实施情况如下：

1） 气动三爪卡盘实物、配套的快接插头和气管以及气动卡盘结构数据如图 6 所示：



（a）气动三爪卡盘实物

（b）配套快接插头和气管



(mm)

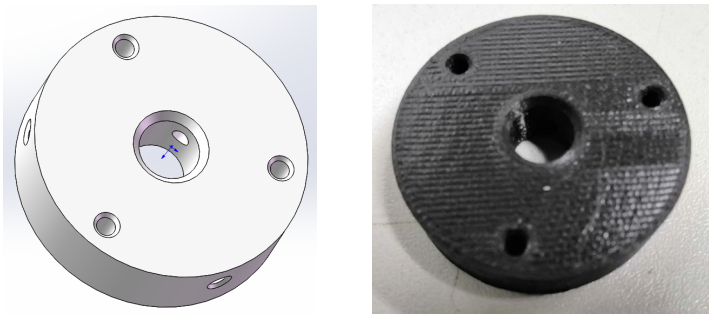
型号	AA	AB	B	CB	DC	DO	EC	EO	FX	FY	FZ	G	I	J	K	NA	NB
MHS3-16D	35	32	30	11	15	17	5	7	12.5	11	3	25	4	10	4	8	5h9
MHS3-20D	38	35	36	13	18	20	6	8	14.5	13	3	27	5	12	5	10	6h9
MHS3-25D	40	37	42	15	21	24	7	10	17	14.5	5	28	5	14	6	12	6h9

型号	O	P	Q	R	SA	SB	SC	TB	UA	UB	VA	VB	WA	XA	XB
MHS3-16D	2	M3 x 0.5	6	25	3.4	6.5	8	5	M3 x 0.5	4.5	2H9	2	17H9	2H9	2
MHS3-20D	2.5	M5 x 0.8	7	29	3.4	6.5	9.5	6	M3 x 0.5	6	2H9	2	21H9	2H9	2
MHS3-25D	3	M5 x 0.8	8	34	4.5	8	10	6	M4 x 0.7	6	3H9	3	26H9	3H9	3

（c）气动卡盘结构数据

图 6 气动卡盘及其结构数据图

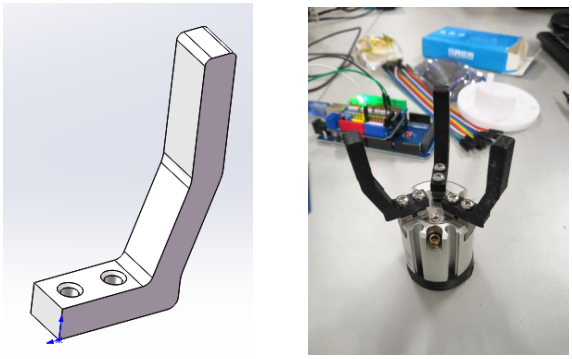
2） 中间连接件的三维模型图以及实物图



（a）中间连接件三维图　　（b）中间连接件实物图

图 7 中间连接件模型及实物图

3） 卡盘爪指三维模型图以及实际装配图



（a）卡盘爪指三维模型图　　（b）卡盘爪指装配图

图 8 卡盘爪指三维模型图及实际装配图

4） 实际装配效果图

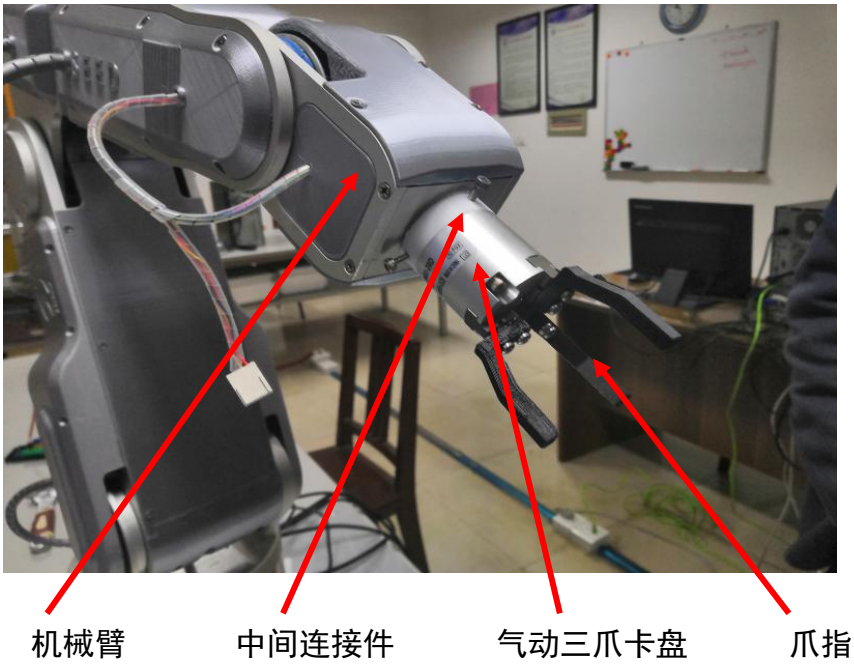


图 9 实际装配效果图

三爪卡盘的选型以及辅助结构的设计都由本组其他同学完成，根据设计需求选型后，实物能够正常使用。在观察三爪卡盘的工作原理以及辅助结构的设计情况后，对于气动三爪卡盘的原理、工作过程以及配套设备有了深刻的理解与认识。

4.2.3 检测工位设计

检测工位的设计需求是：检测工位是一块双面电路板，而检测元件表面需要与待检工件底面平行，所以需要将检测工位架空至一定高度。

对应的解决方案采取了最简单的实际使用效果很好的办法：利用 M3 的铜螺柱将电路板架起。

4.2.4 整体定位方案设计

虽然在设计各部分时都预留了定位接口，但机械臂所在的桌面没有任何可用于定位的结构，所以设计需求为：设计整体定位基台使得检测工位和待检工位相对于机械臂坐标的位置确定。

对此我们采取的解决方案是利用具有极高自由度和灵活性的慧鱼组构件设计了一个给待测工位、检测工位以及系统中其它部分定位用的基台。利用这个定位基台，只要确定了它与桌面的位置关系，其余组件的位置关系也就随之确定，不用每次拆装后重新标定。

具体实施如下图 10 所示：

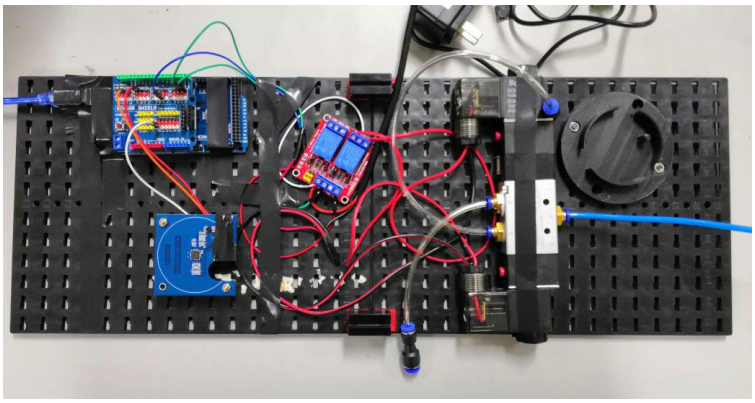


图 10 定位方案实施图

关于定位方案设计，本组刚开始决定采用激光切割亚格力板的方案，但这要求先确定系统中各个部件的安装尺寸，然后在绘制出 CAD 图纸并在网上下单制作。但是这种方案存在四个问题：1）检测板的安装尺寸需要自己量取，这一步引入的测量误差很可能导致最终无法正确配合；2）制作工期无法保证，可能会拖延 1-2 周，这对我们的实验进程很不利；3）可能存在的各组件之间的干涉问题，很难考虑全面；4）这个方案经济性不高，会造成较大的浪费。

所以我们最终放弃了这个方案，同时结合春学期的课程中使用慧鱼组件的经验。提出了第二种方案。经过试验验证，慧鱼组件搭建的定位基台能够很好地满足使用需求。

4.3 气路控制与传感器信号采集

气路控制与传感器信号的采集主要与 arduino 有关，所以在此放在一起进行总结。气路控制的最终目的是实现气动卡盘的开合，完成对待测工件的抓取动作；传感器信号的采集主要是获得在不同角度下待测工件被传感器检测到的电平反馈值，以此来判断工件的质量。接下来将详细介绍在实验中这两部分所遇到的问题、解决问题的方法以及心得体会。

4.3.1 气路控制

首先介绍气路控制所涉及到的部件：三爪气动卡盘（如图 11）、电磁阀（如图 12）、继电器（如图 13）、arduino mega 2560 主板（如图 14）、气泵。

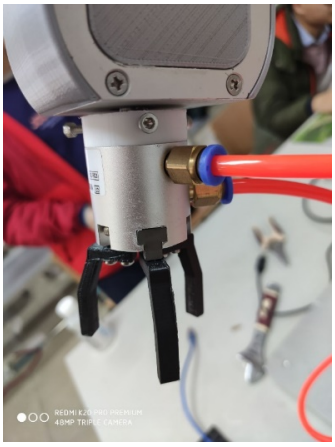


图 11 气动卡盘

如图 11 所示，当气动卡盘的两个通气口气路分别连通时，气动卡盘将出现夹紧或放松动作，配合由本组同学自行设计的夹爪，可以实现对待测工件的抓放动作。

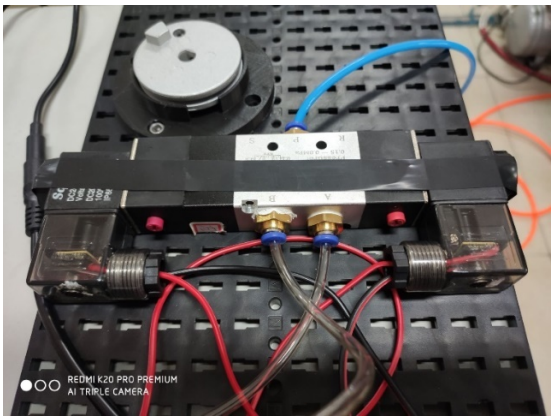


图 12 电磁阀

如图 12 所示，电磁阀的左右部分可以分别连接两条气路，后端连接气泵，通常状态下由气泵引出的气路与前端的两气路是断开状态，只有当相应侧的电路接通时，气路也会被接通，进而实现由电路控制气路的目的。同时，可以按下对应侧的红色按钮来连通气路，方便调试时气路的开断。

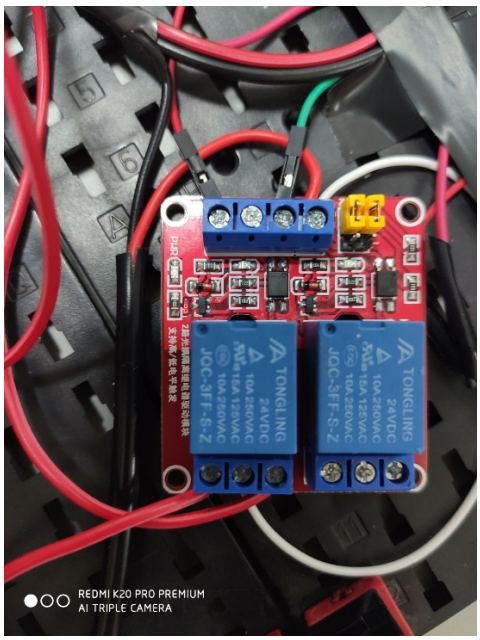


图 13 继电器

图 13 所示的为本组所使用的继电器，用以实现 mega 主板对电磁阀电路的开断。在选择电磁阀型号时，考虑到需要两路电路控制，所以继电器也选用了两路类型的，但在使用时发现两路电路必定是一通一断的状态，可以对应连接继电器的常开常闭端，所以在使用时只用到了一路。本继电器可以选择高电平触发、低电平触发模式，为设计者提供了多种方案选择，但也需要在设计时注意选择的电平触发模式与代码控制时的对应关系。

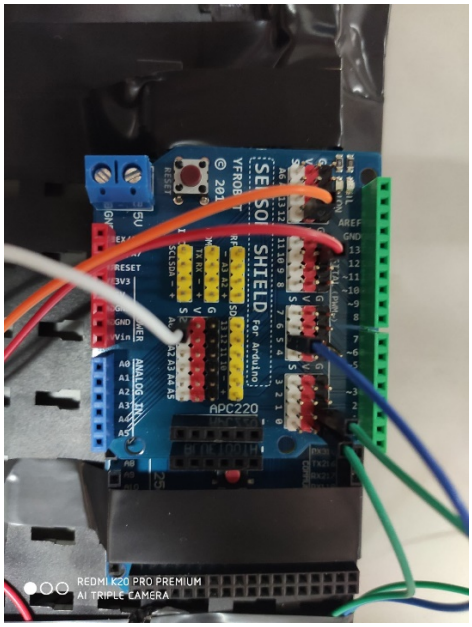


图 14 arduino mega 2560 主板

如图 14 所示为本次实验使用的主控板，该主板本身可以作为上位机在系统中出现，但是本实验将笔记本电脑作为上位机，在 ROS 环境当中，把 arduino 设置为 ROS 的一个结点，通过发布和订阅话题的方式与上位机进行交流，当上位机指令执行到特定行时，发送对应话题给 arduino 并由 arduino 运行设定好的代码，完成对继电器的控制、传感器反馈信号的记录等操作，完成操作后 arduino 发布对应话题给上位机，当上位机接收到此话题时将继续执行后续指令。

接下来介绍进行气路控制实验时遇到的问题以及解决方案：

由于是第一次使用光耦合继电器，对其工作方式的了解还不是特别清楚，所以在第一次做实验时出现了一些失误，以下是来自淘宝卖家的光耦和继电器电路接线原理图：

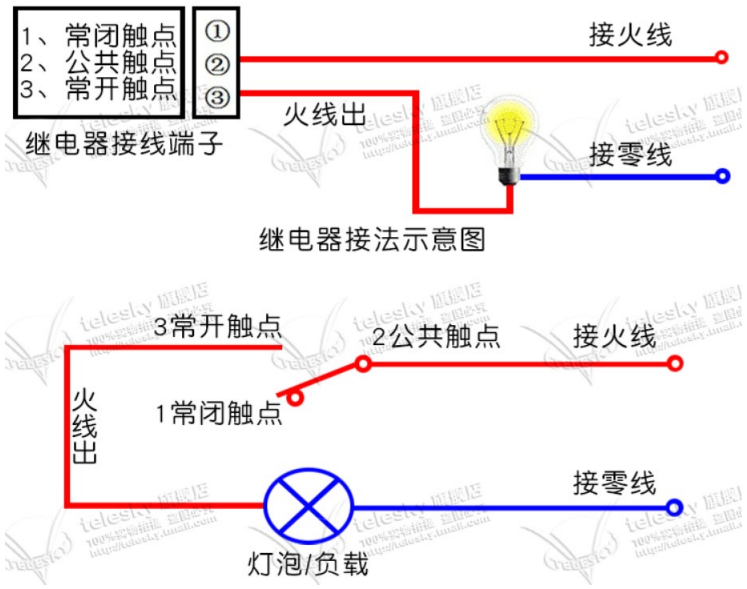


图 15 光耦合继电器电路接线原理图

第一次接线时，忽略了这是一款光耦合的继电器，错误地把继电器等效于一个接在电路中间的可以由 arduino 板控制开断的开关，进行了如图 16 所示的接线：

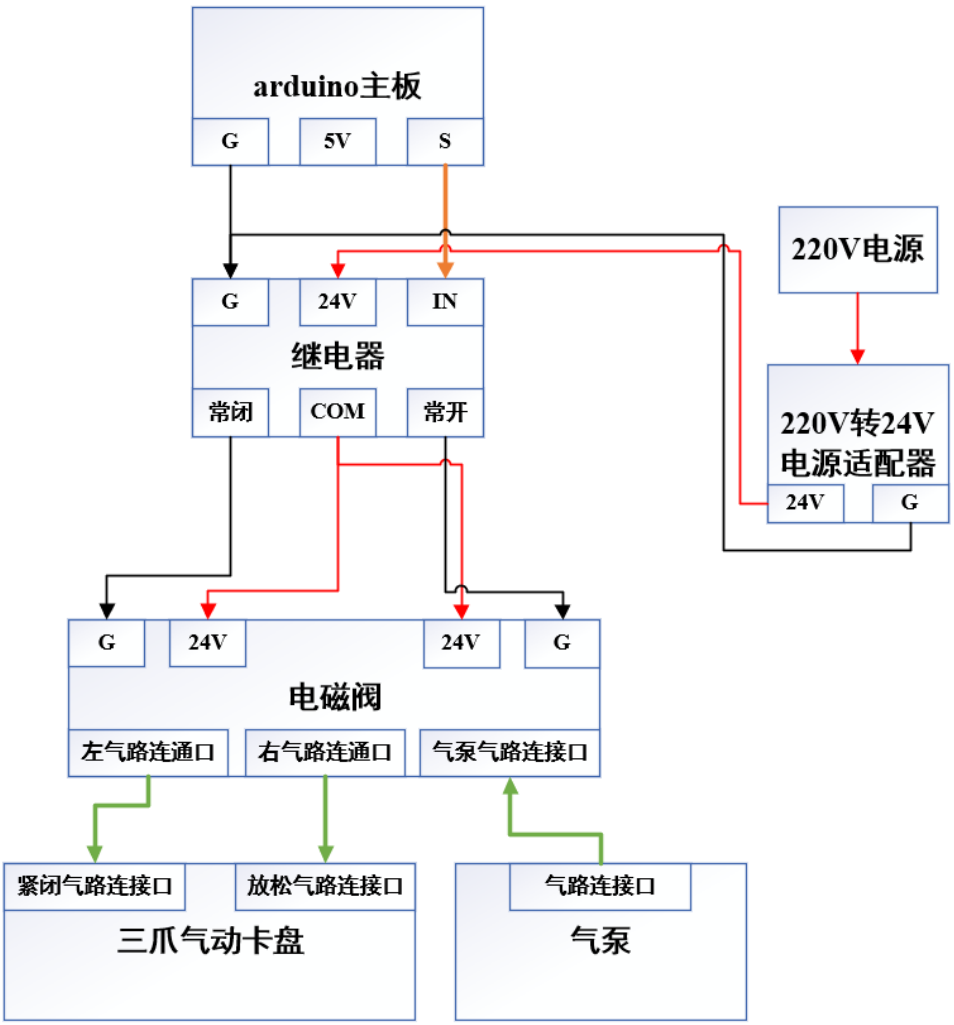


图 16 气路与电路连线错误示例

如图所示绿色连线为气路连接、橙色为信号线、黑色为地线、红色为火线，由于选用的继电器为 24V 供电，所以直接从电源适配器取电。在错误的理解下，将继电器理解为一个接在电路中间的可以由 arduino 板控制开断的开关，所以只需要将供电从继电器的输入端输入即可，经过相当于接入电路中的开关后从 COM 端引出的线相当于从电源适配器引出的火线，所以直接与电磁阀的火线连接，常闭常开接口分别于电磁阀左右端的地线连接，当 arduino 对应引脚的电平改变时分别接通常闭或常开电路，进而接通左气路或右气路。

但实际上光耦合继电器在输入端的电压不会通过继电器连接到 COM 端，所以需要在 COM 端额外供应电磁阀工作时所需要的电压，并且与电磁阀连接成一条独立的回路，经过改正后的连接方式如图 17 所示，在该连接方式下继电器可以正常工作。

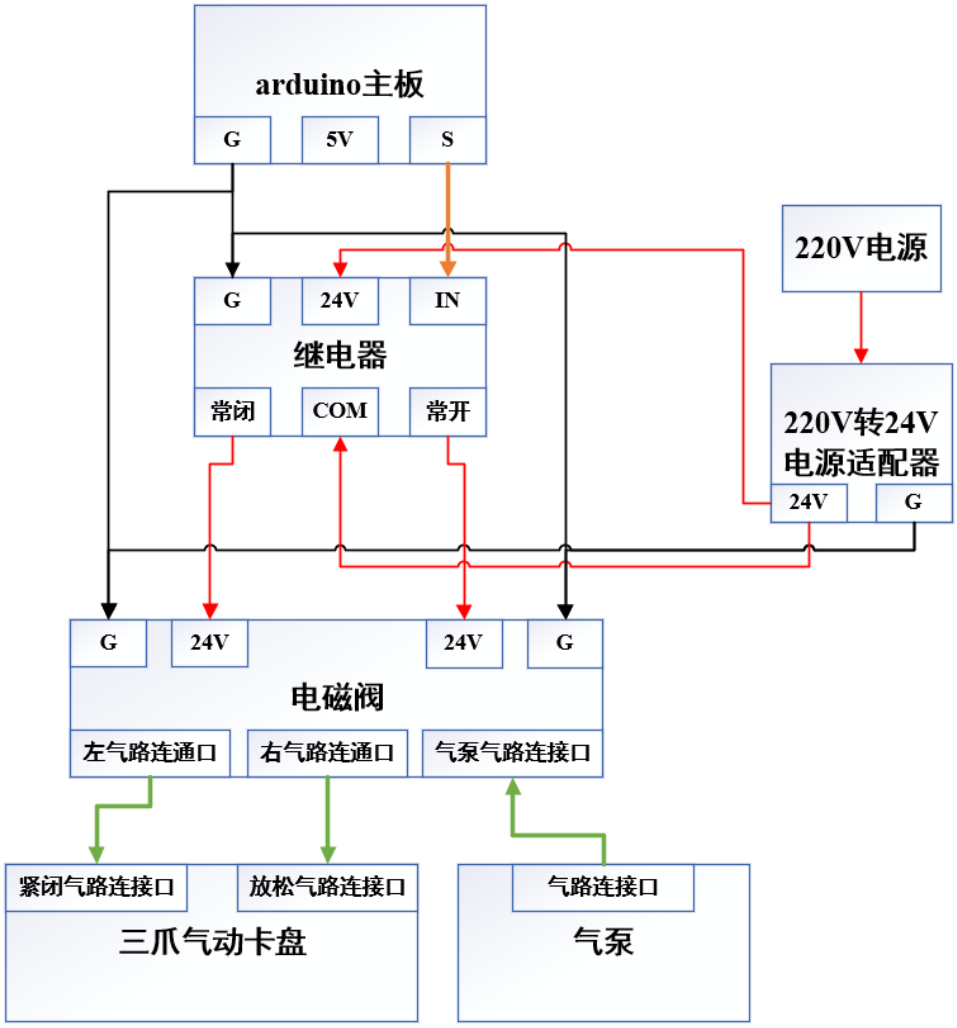


图 17 气路与电路连线示意图

在连线完成后，气路的控制代码非常简单，只需要按卡盘的开合要求将连接的 arduino 引脚电平写入高电平或低电平即可，测试代码如下所示：

```
int info = A0;
int opens = 5;
int closes = 6;

void setup() {
Serial.begin(9600);//串口初始化
pinMode(info,INPUT);
pinMode(opens,OUTPUT);
pinMode(closes,OUTPUT);
}

void loop() {
digitalWrite(opens,HIGH);
delay(5000);
digitalWrite(opens,LOW);
delay(5000);
}
```

测试结果良好，在系统整合的时候只需要将该段代码填入订阅到对应话题时调用的函数体中即可。

4.3.2 传感器信号采集

对传感器信号进行采集的测试过程如下：

首先将传感器的信号口（txd_b 引脚）接到 arduino 的模拟信号接口 a0 上，并且在 IDE 中定义其为输入量，接下来只要在 loop 函数中写下每隔一定时间读取 a0 口的电平值，并且将其打印出来，即可完成测试。

测试结果显示，需要将待测物件放在传感器上相隔很近的位置，即 2mm 以内，并且将待测物件与传感器中心对心，才可以平稳地采集信号。

以下是测试用代码截图。

```
int info = A0;

void setup() {
Serial.begin(9600);//串口初始化
pinMode(info,INPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
double ttl_info = analogRead(info);

Serial.println(ttl_info);
delay(1000);
}
```

需要指出的是，此处两次读取 a0 引脚反馈信号的条件是时隔 1 秒，而在最后系统地进行实验时，相邻两次读取 a0 引脚的信号的标志应为机械臂完成一次转角动作之后反馈的信号值，而不是以时间作为是否进行下一次读取 a0 引脚电平的判定。

4.4 代码编写

软件具体实现分为两个部分，一个是 PC 端的节点，负责与机械手进行通讯并管理任务流程，另一个是 Arduino 端的节点，负责管理 IO。

4.4.1 PC 端节点

1）机械手控制

机械手采用串口控制，用到了 ROS 的 serial 功能包。主要涉及到一条运动指令——轴角度插补指令，和一种状态反馈信息——坐标值状态信息。

轴角度插补指令：

a[0]	a[1]	a[2]	a[3] –a[6]	a[7] –a[10]	a[11] –a[14]	a[15] –a[18]	a[19] –a[22]	a[23] –a[26]	a[27] –a[30]	a[39] –a[42]	a[43] –a[46]	a[47]
0xee	'3'	0–6	a0	a1	a2	w0	w1	aw	PWM	N/A	speed	0xef
帧头	指令 a	指令 b	浮点数 1 mm	浮点数 2 mm	浮点数 3 mm	浮点数 4 度	浮点数 5 度	浮点数 6 度	浮点数 7 1us/步	浮点数 10 0	浮点数 11 度/秒	帧尾
a[31]–a[34]是外部轴 E0 角度值，a[35]–a[38]是外部轴 E1 角度值												

坐标值状态信息：

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]
0xce	x_h	x_l	y_h	y_l	z_h	z_l	0	0xcf
帧头	d0	d1	d2	d3	d4	d5	指令标志	帧尾

PC 机向机械手发送轴角度插补指令，该指令包含了通过示教得到的目标点位置的关节坐标信息 a_0、a_1、a_2、w_1、w_2、aw 和转动角速度信息 speed。机械手接收到运动指令后，结合当前关节坐标，插补得到运动轨迹，并沿着这条轨迹运动到目标位置。在机械手运动过程中，PC 机等待，并不断读取机械手反馈的坐标值状态信息，从中提取机械手当前的坐标位置，当获得的坐标值等于目标坐标值时，PC 机停止等待，进行下一个任务。

结构体 axis_angle_msg 包含了轴角度插补指令所需要的信息：

```
struct axis_angle_msg
{
    float angle1;
    float angle2;
    float angle3;
    float angle4;
    float angle5;
    float angle6;
    float speed;
    axis_angle_msg();
    axis_angle_msg(float a1,float a2,float a3,float a4,
float a5,float a6,float s);
};
```

指令类 order，包含运动指令初始化接口、控制指令初始化接口（本项目中没有用到控制指令）和指令发送接口：

```
class order
{
public:
    void move_order_init(axis_angle_msg msg);
    void control_order_init();
    size_t send(serial::Serial &sp);
private:
    void trans(float angle,uint8_t a[]);
    uint8_t order[48];
};
```

目标点位置的关节空间坐标表示和操作空间坐标表示需要提前通过示教得到。为了防止机械手最后运动到的位置与目标值有误差，设定一个极小的容许误差 T_：

```
bool is_right(position_msg &pos_msg,int pos_choose)
{
    if(pos_choose==1)
        return (abs(pos_msg.x-origin_pos.x)+abs(pos_msg.y-origin_pos
.y)+abs(pos_msg.z-origin_pos.z))<T_;
    if(pos_choose==2)
        return (abs(pos_msg.x-wp_up_pos.x)+abs(pos_msg.y-wp_up_pos.y
)+abs(pos_msg.z-wp_up_pos.z))<T_;
    if(pos_choose==3)
        return (abs(pos_msg.x-wp_down_pos.x)+abs(pos_msg.y-wp_down_p
os.y)+abs(pos_msg.z-wp_down_pos.z))<T_;
    if(pos_choose==4)
        return (abs(pos_msg.x-detect_pos.x)+abs(pos_msg.y-detect_pos
.y)+abs(pos_msg.z-detect_pos.z))<T_;
    return false;
}
```

以下是使机械手运动到原点并等待运动结束的整个过程，其中，wait 函数中的参数“1”代指储存的第一个目标点（原点）：

```
myorder.move_order_init(origin);
myorder.send(sp);
wait(sp,mystate,1);//运动至原点
```

2）话题的订阅与发布——与 Arduino 进行通讯

PC 与 Arduino 之间的 ROS 通讯依赖于 rosserial_arduino。PC 端节点与 Arduino 端节点通过话题进行消息传递，在 PC 端创建一个话题发布器 chatter_pub 用于发布指令信息 chatter_pub，创建一个话题订阅器 chatter_sub 用于订阅动作完成信息 chatter_sub：

```
    ros::Publisher  chatter_pub = n.advertise<std_msgs::String>("chatter
_pub", 100);
    ros::Subscriber chatter_sub = n.subscribe<std_msgs::String>("chatter
_sub", 100, Callback);
```

发布和订阅的话题内容具体如下：

```
Publish:
    "loosen claw"      ------>        松爪
    "shrink claw"      ------>        收爪
    "detect"           ------>        检测一次
Subscrib:
    "has loosened"           ------>    已松爪
    "has shrinked"           ------>    已收爪
    "has detected xxxxxx"    ------>    已检测一次 + 六位检测值
```

PC 端发布执行任务话题后等待应答，Arduino 端订阅话题后执行相应的任务并产生应答，收到应答后 PC 端进行下一个任务。

4.4.2 Arduino 端节点

1）话题的订阅与发布——与 PC 进行通讯

Arduino 端节点创建一个话题订阅器 sub 用于订阅 PC 端节点发布的话题 chatter_pub，创建一个话题发布器用于发布 PC 端需要订阅的话题 chatter_sub：

```
ros::Publisher pub("chatter_sub", &str_msg);
ros::Subscriber<std_msgs::String> sub("chatter_pub", messageCb );
```

发布和订阅的话题内容具体如下：↵

```
Subscrib:
    "loosen claw"      ————>        松爪
    "shrink claw"      ————>        收爪
    "detect"           ————>        检测一次
Publish:
    "has loosened"           ————>     已松爪
    "has shrinked"           ————>     已收爪
    "has detected xxxxxx"    ————>     已检测一次 + 六位检测值
```

2）气动卡盘控制

气动夹手用 Arduino 数字口控制：

```
pinMode(CLAW,OUTPUT);
void loosen_claws(void)
{
  digitalWrite(CLAW,LOW);
  delay(1000);
}

void shrink_claws(void)
{
  digitalWrite(CLAW,HIGH);
  delay(1000);
}
```

3）检测传感器读数

检测板读数利用 Arduino 的模拟口，最大读数范围为 0~1023。以下过程为：从对应的模拟口读出检测板读数，并将数据组织成“has detected xxxxxx”的格式：

```
pinMode(DETECT,INPUT);
void detecting(char msg[])
{
  char temp_str[10];
  int digit;
  digit=analogRead(DETECT);
  strcpy(msg,"has detected ");
  sprintf(temp_str,"%-6d",digit);
  for(int i=0;i<7;i++)
    msg[13+i]=temp_str[i];
}
```

5.实验总结与归纳

这是一个为期 7 周的实验，在与本组同学的团结合作、不懈努力下终于能交上一份较为满意的答卷，以下对本次实验中所遇到的问题、我们采用的解决方案以及获得的实验心得进行叙述：

5.1 机械臂上电时遇到的问题

机械臂上电后有一个自检复位的动作，目的是使机械臂上电后重新找到自己的姿态和坐标。复位运动到定位点后触发红外光电开关，使机械臂回到标准的启动状态。红外光电开关对普通光线不是很敏感，但如果在阳光充足的环境，或者高亮度灯光直射，还是会干扰到光电开关，产生不触发现象。如果光电开关不触发，机械臂受干扰的轴会继续向复位方向旋转，产生错误。如果转过头太多，会损坏线路。机械臂在初始安装过程要注意环境光，上电测试时谨慎观察，有问题马上断电。

另外机械臂在断电后，某些轴有可能会停在复位触发点之外。机械臂正常运行时有软限位保护，不会转到复位开关以外。由于此机械臂是没有刹车的，断电后机械臂多数轴处于自由状态，如果手推动机械臂，大部分轴会被推转。如外力推动机械臂转到了极限状态，这时不要上电，先手动把机械臂推转回到正常范围。

械臂在调试过程要特别小心，注意观察，如果动作与预期不符就马上断电。

5.2 机械臂操作时遇到的问题

通过示教器操作使得机械臂运动到指定位置时记录下各轴角度，用该组轴角度数据通过上位机控制机械臂时，得到的机械臂运动终点位置与示教器操作时不同。通过不断测试后，我们组认为造成该结果的原因可能有以下三点：1）机械臂各轴运动的步进角与示教器显示的精度存在一定的差距，即只有当示教器的数值变化一定档位后机械臂对应轴的转角才会出现一个跳跃式的变化，所以通过示教器的显示数值输入上位机的控制程序来控制机械臂时，并不能达到理论上示教器显示的精度；2）如果是使用示教器的 XYZ 模式控制机械臂运动到指定位置，则可能因为 XYZ 轴移动时三轴并不独立，例如仅移动 X 轴时，YZ 轴的位置也会发生变化，这与示教器的内部坐标转换算法有关，属于我们难以减小的误差；3）通过示教器控制机械臂到指定位置并记录后，将机械臂重新上电，并通过上位机控制机械臂，会在 XY 方向出现一定的偏差，但经过多次测试后，发现该偏差值是特定的，只与示教器控制机械臂的终点位置有关，所以可以通过反向移动一定距离来修正。

实验的最后，当我们把其他所有能消除的误差都消除后发现，由于机械臂关键轴的步进角太大，而使得我们不管如何调整该角度或调整其他轴的角度来修正，都无法使得工件处于传感器的可检测范围内，最终我们通过垫加一组螺母来增加传感器距离定位平台的数值高度的办法解决了这一问题，垫加后检测工位的细节图如图 18 所示：

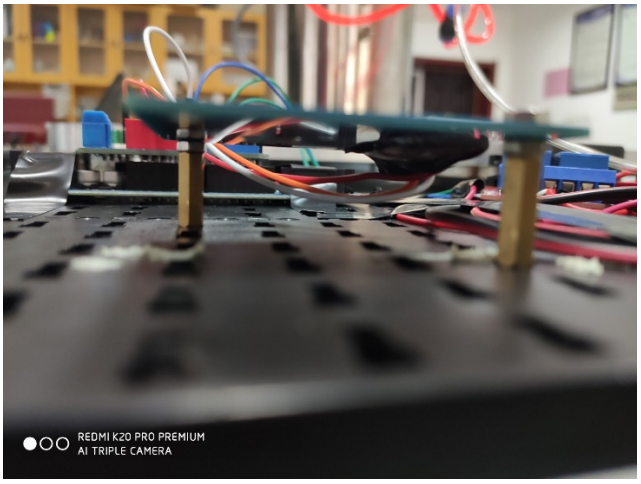


图 18 检测工位修正

5.3 PC 与 Arduino 通讯时遇到的问题

从 ROS 发布的信息如果不写在循环体里面，便无法被 Arduino 接收到，具体原因尚未确定。通过将需要发布的信息嵌套在无限循环的循环体中，能够解决这个问题，但是对于代码的运行效率会有很严重的影响。通过不断尝试后发现，将需要发布的信息嵌套在有限循环次数的循环体中并且在循环体最末尾加上短暂延时之后，能够解决这个问题，且对代码整体运行效率不会造成太大的影响，并且我们最后选择了该方案编写代码。但这也留下了一个问题，需要我们更深入了解 PC 与 Arduino 的通讯机制后去解答。

实验到此就暂告一段落了，在这七周内我们综合运用了自己大学前三年所学到的知识，包括 solidworks 建立三维模型、选择加工方式并且加工实物、类 C 语言的 Arduino 代码的编写等，以及之前没有接触过的 ROS 系统，使我们在巩固基础时更学习到了更多有用的知识，受益良多，也为后续的毕业设计提前预热，使自己更容易进入到研究的状态当中。