南京理工大学

NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

电子系统设计创新创业实践

实验报告

组　　号：73

指导老师：孙亚星

小组成员：常思克 9161040G0313

许晓明 9161040G0734

郭泽昊 9161040G0919

刘滔滔 9161040G1021

张锦涛 9161040G1034

2019.5

目录

滚球控制系统设计方案

一、实验要求

1.1 任务

在边长为65cm光滑的正方形平板上均匀分布着9个外径3cm的圆形区域，其编号分别为1～9号，位置如图1所示。设计一控制系统，通过控制平板的倾斜，使直径不大于2.5cm的小球能够按照指定的要求在平板上完成各种动作，并从动作开始计时并显示，单位为秒。

1.2 要求

1.基本部分：

(1) 将小球放置在区域2，控制使小球在区域内停留不少于5秒。

(2) 在15秒内，控制小球从区域1进入区域5，在区域5停留不少于2秒。

(3) 控制小球从区域1进入区域4，在区域4停留不少于2秒；然后再进入区域5，小球在区域5停留不少于2秒。完成以上两个动作总时间不超过20秒。

(4) 在30秒内，控制小球从区域1进入区域9，且在区域9停留不少于2秒。

2.发挥部分：

(1) 在40秒内，控制小球从区域1出发，先后进入区域2、区域6，停止于区域9，在区域9中停留时间不少于2秒。

(2) 在40秒内，控制小球从区域A出发、先后进入区域B、区域C，停止于区域D；测试现场用键盘依次设置区域编号A、B、C、D，控制小球完成动作。

(3) 小球从区域4出发，作环绕区域5的运动（不进入），运动不少于3周后停止于区域9，且保持不少于2秒。

(4) 其他。

1.3 说明

1.系统结构要求与说明

(1) 平板的长宽不得大于图 1 中标注尺寸；1～9 号圆形区域外径为 3cm，相邻两个区域中心距为 20cm;1～9 区域内可选择加工外径不超过 3cm 的凹陷；

(2) 平板及 1-9 号圆形区域的颜色可自行决定；

(3) 自行设计平板的支撑（或悬挂）结构，选择执行机构，但不得使用商品化产品；检测小球运动的方式不限；若平板机构上无自制电路，则无需密封包装，可随身携带至测试现场；

(4) 平板可采用木质（细木工板、多层夹板）、金属、有机玻璃、硬塑料等材质，其表面应平滑，不得敷设其他材料，且边缘无凸起；

(5) 小球需采用坚硬、均匀材质，小球直径不大于 2.5cm；

(6) 控制运动过程中，除自身重力、平板支撑力及摩擦力外，小球不应受到任何外力的作用。

2.测试要求与说明

(1) 每项运动开始时，用手将小球放置在起始位置；

(2) 运动过程中，小球进入指定区域是指小球投影与实心圆形区域有交叠；小球停留在指定区域是指小球边缘不出区域虚线界；小球进入非指定区域是指小球投影与实心圆形区域有交叠；

(3) 运动中小球进入非指定区域将扣分；在指定区域未能停留指定的时间将扣分；每项动作应在限定时间内完成，超时将扣分；

(4) 测试过程中，小球在规定动作完成前滑离平板视为失败；

二、系统方案

2.1 设计整体框图

系统结构如下图所示：

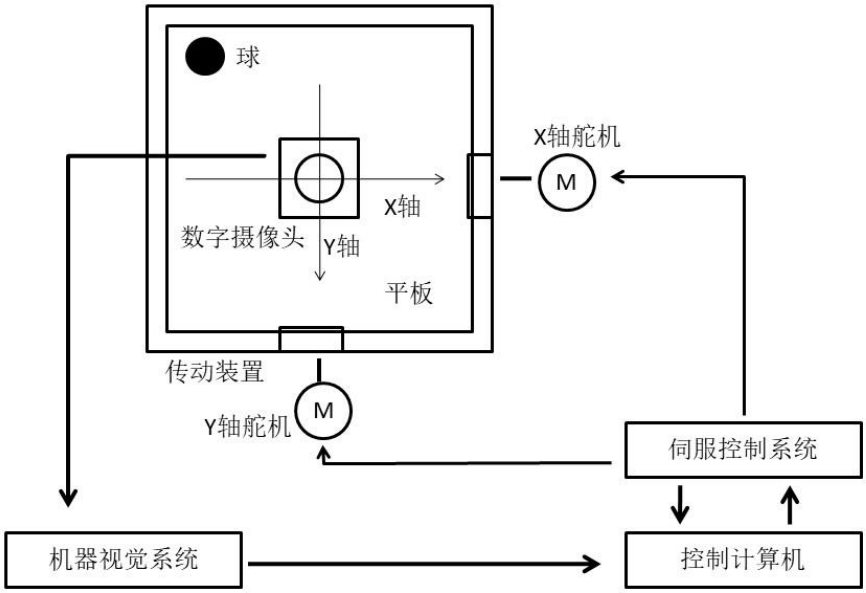


图 1 系统结构图

整个设计结构主要分为电源模块，摄像头 模块，控制模块，机械结构模块，显示与输入模块构成。单片机利用摄像头采集到的数据，确定小球的位置坐标，通过 PID 闭环控制舵机 PWM 输出打角拉动支撑杆，控制平板倾斜度达到小球滚到指定区域停留等状态。

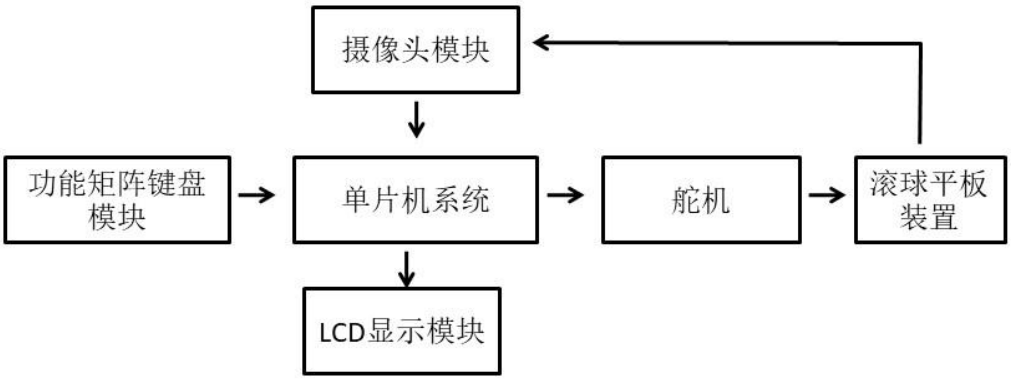


图 2 总体方案框图

2.2 主控制器件

一般采用 STM32F4VE 单片机为主控器，具有速度快，片上资源丰富，具有许多外围借口，拓展性好，具有很好的响应性。

2.3 图像采集与处理方案

用摄像头采集图片，并利用函数库进行图像处理，将得到的位置数据通过串口发给主控芯片。LCD 的显示，更有利于进行实验的调试。

2.3.1 图像采集

摄像头选用的是物致 DIY 家的 OpenMV3 可编程的摄像头模块，OpenMV3 摄像头模块是一款小巧，低功耗，低成本的模块，搭载 MicroPython 解释器，可以使用使用 Python 来编程，主控芯片 STM32f765vit6,摄像头 412KB RAM,2MB Flash，输出图像格式 RGB565，320x240 像素，焦距 2.8mm，角度 115°。将摄像头采集到的图像 R,G,B 分别保存到三个二维数组中。

2.3.2 图像处理

为了得到小球的位置，我们调用 openMV 中的 img.find_blobs()函数。

在调用之前，需给一个目标的阈值，比如红色的小球，则给的标准是(20, 75, 28, 127, -128, 127)——当像素点的 RGB 三个矩阵中的对应的数字在这个范围内，则认为是目标，不在范围，则不是。当遍历整个图像时，根据这个标准进行二值化，是目标则 1，否则 0。二值化以后可以得到目标的像素点的位置了。再调用 blob.x()，blob.y()来得到这个位置。此方法要求高对比度图像，即目标和背景的颜色差异大。为此，我们把纸板附上白纸，小球涂成红色。

调用双向串行通信协议的 uart.write(),把得到的位置数据发出去，波特率设置成 76800。

2.3.3 图像采集与处理框图

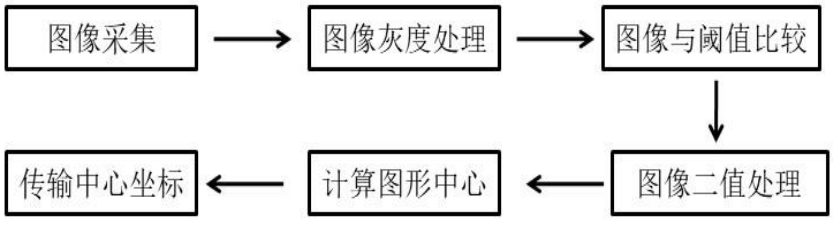


图 3 图像控制

2.4 滚球控制系统方案

采用 DG-995 舵机进行滚球的运动控制，舵机内置驱动，由此可以通过单片

机给予脉冲信号，通过舵柄转换成旋转运动来达到不同的角度，再加上支撑杆就可以控制平板的上下运动。

2.4.1 基本功能和要求

1）控制两路舵机

2）通过按键调整舵机的角度

3）显示两路舵机的角度

4）系统启动时要求舵机舵盘初始位置在中间（既能左转又能右转），能通过按键控制回到中间位置

2.4.2 硬件设计

实验设计主要采用 PWM 输出控制舵机

1）舵机基本控制原理

舵机是一种实现精确角度控制的伺服电动机，主要由小型直流电动机、变速齿轮组、电位器和控制电路板四部分组成，是一个典型的闭环控制系统。通过向舵机的信号线输入 PWM 信号，可以控制舵机输出角度。

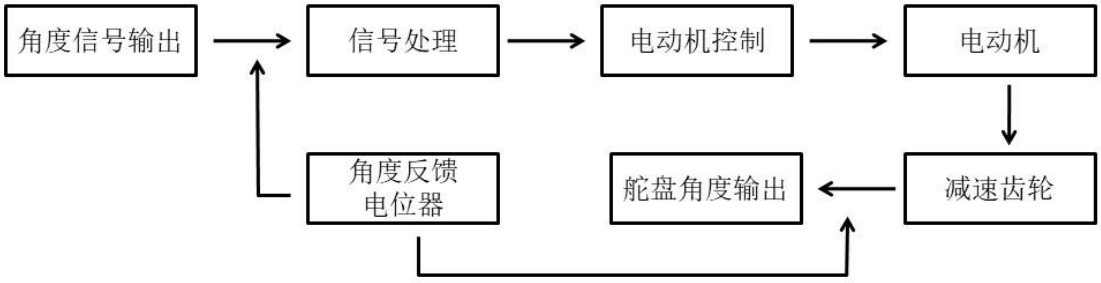


图 4 舵机控制流程图

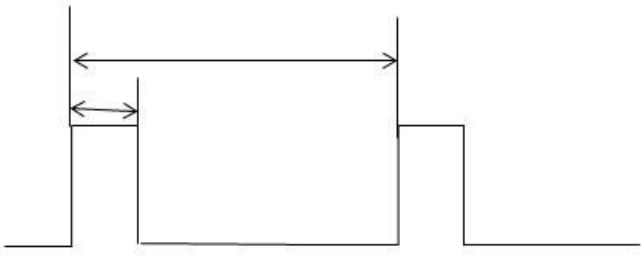


图 5 舵机控制 PWM

2）具体编程流程

①摆幅控制及停止角度的精确度

```
uint32 KP_A=10,KP_B=140,KD_X,KD_Y;//舵机 PID
/*死区控制*/
uint32 DJ_zone=0;                    //死区控制
/*摆幅限制*/
uint32 DJ_min=700;                   //舵机最小摆幅
uint32 DJ_max=3000;                  //舵机最大摆幅
```

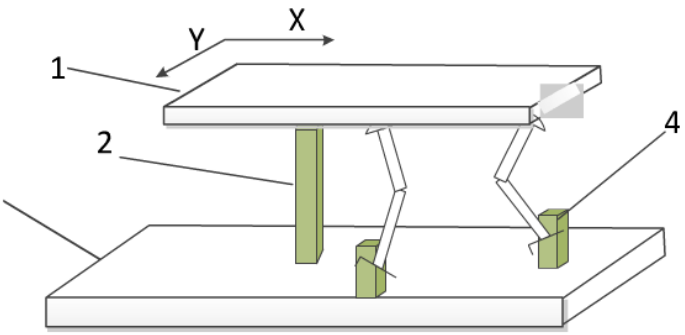
②中值确定

```
/*舵机中值*/
uint32 DJ_midpoint_x=8200;           //舵机中值 x
uint32 DJ_midpoint_y=8374;           //舵机中值 y
```

③PWM

```
uint64 DJ_x_PWM=8300;                //输出 PWM
uint64 DJ_y_PWM=8374;                //输出 PWM
uint32 DJ_maxpoint_x=2000;           //舵机限幅_x
uint32 DJ_maxpoint_y=2000;           //舵机限幅_y
```

2.4.3 机械臂



1：载球平台；2：支架；
3：底盘；4：舵机

图 6 机械臂设计

小球的运动控制是通过处理器的计算将结果反馈给舵机来控制 X 方向和 Y 方向的舵机转动，一实现载球平台不同方向的倾斜，从而使小球按照预期的轨迹和规定运动参数在平台上完成相应动作。采用 DS3119 舵机进行滚球的运动控制，舵机内置驱动，由此可以通过单片机给予脉冲信号，通过舵柄转换成旋转运动来达到不同的角度，再加上支撑杆就可以控制平板的上下运动。

在本方案中，我们使用试凑法来确定 PID 控制器的比例和微分参数。 试凑法是通过闭环试验，观察系统响应曲线，根据各控制参数对系统响应的 大致影响，反复试凑参数，以达到满意的响应，最后确定 PID 控制参数。对参数 调整实行先比例、后积分，再微分的整定步骤，试凑不是盲目的，而是在控制理论指导下进行的。 舵机 PD 试凑法的具体实施过程为： 整定比例部分，将微分部分置零，同时比例系数由小变大，并观察相应的系统响应，直至得到反应快、超调小的响应曲线，由此确定比例系数。加微分环节，随着速度的增加逐步增加微分环节的系数，最终得到最佳的微分系数。

2.5 显示器

采用显示器能够有利于人机界面的操作控制，适用于功能性较强的操作系统。

三、电路与程序设计

3.1 程序设计

3.1.1 设计思路

进入系统，先进行校准，调节舵机，是平板达到平衡，记录平衡值。通过摄像头采集平板的图像二值化，将二值化图像显示在 LCD 屏幕上，并找到 1,5,9 区域并保存重新定义区域位置来减少调节平板和摄像头因碰撞会产生的误差。经过二值化寻找到小球的位置，求出与目标位置的差值，经过 PID 算法，调节舵机输出值来调节平板的上下运动，使小球到达目标位置。

3.1.2 程序流程图

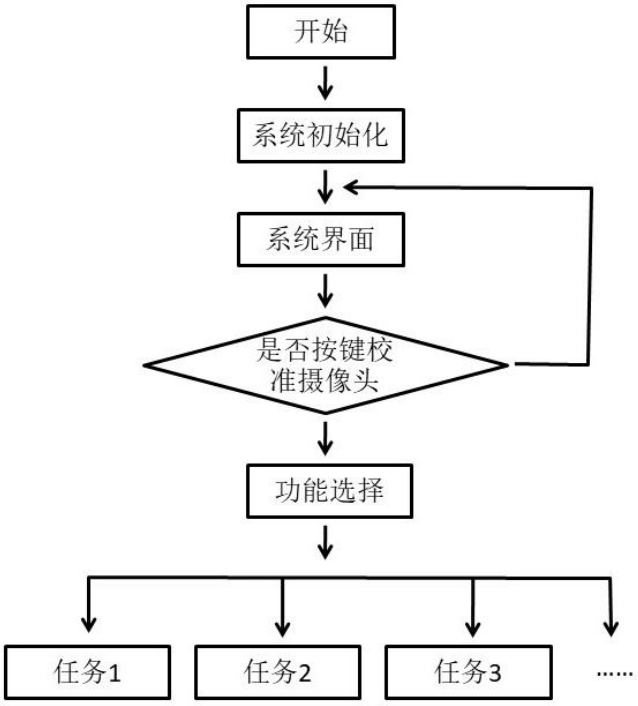


图 7 程序流程图

3.2 PID 算法

为了使舵机保持一个稳定的输出，需要采用 PID 算法对舵机进行调节，具体实现内容如下。

3.2.1 PID 含义

PID 是英文单词比例（Proportion），积分（Integral），微分（Differential coefficient）的缩写。PID 调节实际上是由比例、积分、微分三种调节方式组成，它们各自的作用如下：

比例调节作用：是按比例反应系统的偏差，系统一旦出现了偏差，比例调节立即产生调节作用用以减少偏差。比例作用大，可以加快调节，减少误差，但是过大的比例，使系统的稳定性下降，甚至造成系统的不稳定。

积分调节作用：是使系统消除稳态误差，提高无差度。因为有误差，积分调节就进行，直至无差，积分调节停止，积分调节输出一常值。积分作用的强弱取决与积分时间常数 Ti，Ti 越小，积分作用就越强。反之 Ti 大则积分作用弱，加入积分调节可使系统稳定性下降，动态响应变慢。积分作用常与另两种调节规律结合，组成 PI 调节器或 PID 调节器。

微分调节作用：微分作用反映系统偏差信号的变化率，具有预见性，能预见偏差变化的趋势，因此能产生超前的控制作用，在偏差还没有形成之前，已被微分调节作用消除。因此，可以改善系统的动态性能。在微分时间选择合适情况下，可以减少超调，减少调节时间。微分作用对噪声干扰有放大作用，因此过强的加微分调节，对系统抗干扰不利。此外，微分反应的是变化率，而当输入没有变化时，微分作用输出为零。微分作用不能单独使用，需要与另外两种调节规律相结合，组成 PD 或 PID 控制器。

3.2.2 PID 位置式计算公式

$$u(k)=K_p e(k)+K_i\sum_{j=0}^{k}e(j)+K_d[e(k)-e(k-1)]$$（式 1-1）

式 1-1 中 $K_i=\frac{K_pT}{T_i}$ 为积分系数， $K_d=\frac{K_pT_d}{T}$ 为微分系数，u(n)为第 k 个采样时刻的控制，T 为采样周期。

3.2.3 增量式计算公式

可有式 1 推到得到

由式 1 可以得控制器的第 k-1 个采样时刻的输出值为：

$$u_{k-1}=K_p[e_k-1+\frac{T}{T_3}\sum_{f=0}^{k-1}e_f+T_d\frac{e_{k-1}-e_{k-3}}{T}]$$（1-2）

将（1-1）与（1-2）相减并整理，就可以得到增量式 PID 控制算法公式：

$$\Delta u_k=u_k-u_{k-1}=K_p[e_k-e_{k-1}+\frac{T}{T_i}e_k+T_d\frac{e_k-2e_{k-1}+e_{k-2}}{T}]$$

$$=K_p(1+\frac{T}{T_i}+\frac{T_d}{T})e_k-K_p(1+\frac{2T_d}{T})e_{k-1}+K_p\frac{T_d}{T}e_{k-2}$$

$$=Ae_k-Be_{k-1}+Ce_{k-2}$$

式中，

$$A=K_p(1+\frac{T}{T_i}+\frac{T_d}{T})$$

$$B=K_p(1+\frac{2T_d}{T})$$

$$C=K_p\frac{T_d}{T}$$

由（1-3）可以看出，如果计算机控制系统采用恒定的采样周期 T，一旦确定 A、B、C，只要使用前后三次测量的偏差值，就可以由（1-3）求出控制量。

式中:

Δu(k)——控制器(也称调节器)的输出增量

e(k)——控制器的输入(常常是设定值与被控量之差,即 e(k)=r(k)-c(k));

Ki=(Kp*T)/Ti 为积分系数

Kd=(Kp*Td)/T 为微分系数

T: 计算机控制系统的采样周期，其在选定后就不再改变。其选用原则是在被控系统反馈信号的反应时间要求内，尽量小。但过小会增加运算量。

Kp——控制器的比例放大系数

Ti——控制器的积分时间

Td——控制器的微分时间

3.2.4 优缺点

1) 增量式算法优点：

①算式中不需要累加。控制增量 Δu(k)的确定仅与最近 3 次的采样值有关，容易通过加权处理获得比较好的控制效果；

②计算机每次只输出控制增量，即对应执行机构位置的变化量，故机器发生故障时影响范围小、不会严重影响生产过程；

③手动—自动切换时冲击小。当控制从手动向自动切换时，可以作到无扰动切换。

2) 位置式 PID 控制算法的缺点：

当前采样时刻的输出与过去的各个状态有关，计算时要对 e(k)进行累加，运算量大；而且控制器的输出 u(k)对应的是执行机构的实际位置，如果计算机出现故障，u(k)的大幅度变化会引起执行机构位置的大幅度变化。

四、系统整体外观布局设计

4.1 整体架构

使用三合板木板作底部，将木条固定成门型框架，再把 600x600mm 的亚克力切割打孔，固定在木质框架底部上，将摄像头固定在亚克力板的正上方，并且保证摄像头可以将全局面貌均拍摄完整。在摄像头位置安装要配合焦距,视野角度,

曝光等。

图 8 整体架构

4.2 舵机连杆

舵机部分，先上舵盘，也就是所谓的输出头，舵盘上一共四个螺丝点位，3mm 的，将舵盘固定。我使用舵机匹配的舵机臂，一头用千分尺和卖家的输出头图纸给出孔位参数，打四个安装螺丝孔，用来结合舵机的输出头，另一头直接打 12mm 直径的圆，使用了过盈配合，把一个直径 8mm 左右的轴承（内部 4mm 直径）牢牢嵌入。舵机竖着放舵机输出杆垂直冲着正中心。舵机连杆是时刻冲向地板中心的，有一定自锁作用，防止了自旋转。

图 9 舵机连杆

4.3 主体架构

中心支撑柱，采用 5mm 优质亚克力，高质量铜螺柱，经过三点式打孔，120 度分开安装螺丝螺柱，分散受力，每层与每层之间采用圆形亚克力固定连接，顶层固定法兰盘转接口安装碳纤杆，结实耐用受力均匀。关于整体高度问题，用铜螺柱和亚克力凑一定高度，大约 11cm 多，剩下距离的可通过碳纤来凑。

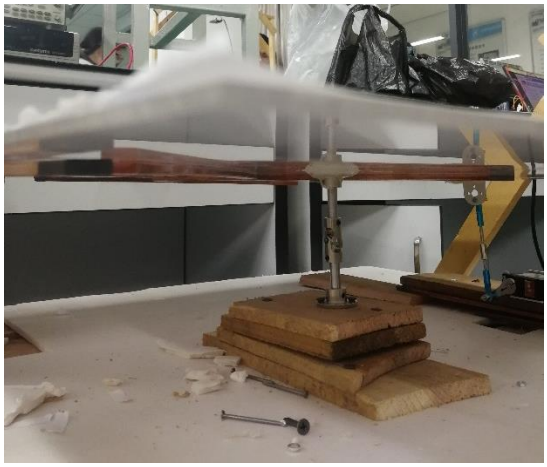


图 10 主体架构

4.4 舵机的机械臂设计

舵机使用 6s 锂电池降压 6v 供电最大 3A。由于板球板子可能过于沉且有惯性，在目标位置有些舵机工作功率小并不能维持稳定，舵机会因为板子的重力和速度惯性先目标位置被压下去，到了一定位置反馈增加力量开始将板子向上推，

推出速度到了目标位肯定又是拉不住，于是循环往复，处于临界状态，稍加扰动即可失控。

修正舵机扭力过载的问题有两种解决办法，和一种检测办法。检测方法是如果把舵机单独累增累减 pwm 发现没问题，放到装置上不加 pid，累增累减 pwm 让其摆动发现开始自震，那么就是目标位置过载了。解决的办法就是缩短力臂，并且尝试更换更大功率和扭力的舵机。

另外，两条舵机 PWM 信号线上加一个开关，单片机下程序或者上电的时候以及 pid 失控的时候关键时刻切开关，防止舵机出现失控等情况。

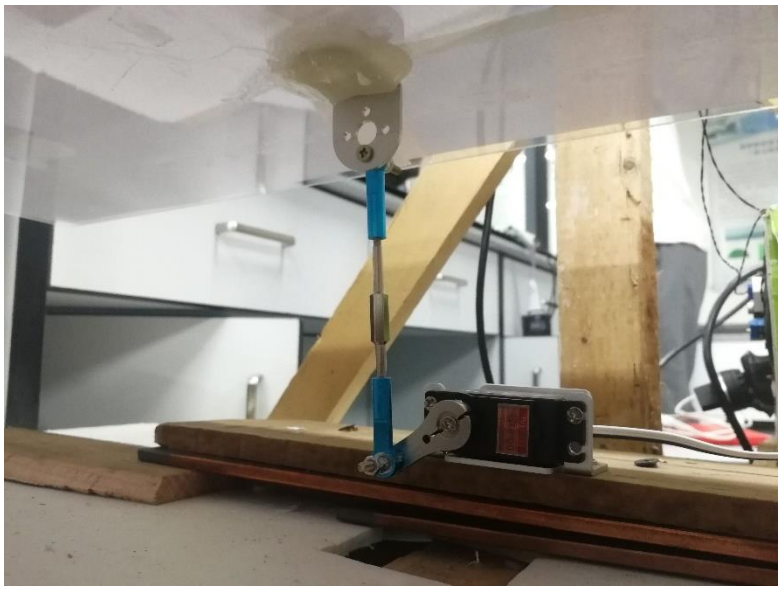


图 11 舵机机械臂

五、测试方案

5.1 硬件测试

(1) 检测焊接的单片机外接设备能否正常工作

(2) 摄像头采集显示到显示屏能否稳定判断平板和小球

(3) 检测电源提供的电压、电流模块使用能否达到稳压值

(4) 平板和的舵机装置静置是否平衡

(5) 检测按键能否实现不同任务

5.2 硬件软件联调

通过按键和显示器达到人机交互，测试滚球运动与实际偏差，通过调节按键选择 KP1、KD1 系数从而控制舵机 pwm 输出，调试滚球控制系统。

5.3 测试条件与仪器

测试条件：检查仿真电路、硬件电路和原理图是否一致，检查硬件电路是否存在虚焊等。

测试仪器：万用表、水平仪、秒表

六、实验感想及总结

6.1 实验感悟

此次的电子设计实验过程对于我们来说是一个巨大的挑战，由于组内大部分同学都没有相关的基础，因此从一开始的相关设计到后来的具体单片机等器件的选择都花费了很长的时间，尤其是实验过程中亚克力玻璃板的打孔，由于没有找到可以打孔的地方，因此我们小组自己去工训中心进行了手动打孔，但是正是由于没有经验以及不太了解具体做出来的情况，导致打孔出现失误，小球难以进入小孔当中，后期为了定准小球的坐标，我们也是换了将近 10 种纸垫在板子上，才得到一个较好的结果。

这次实验也很感激我们的指导老师——孙亚星老师的耐心指导，我们才能在实验过程中不断改进，并获得更好的实验结果。

6.2 实验结果与期望

在本次实验中，我们小组内能够团结一致，共同努力，不怕困难，这也才能让几乎没有基础的我们可以按时完成本次的实验。在实验过程中遇到问题时，没

有人抱怨，所有人都在积极地去查找资料或者询问老师，这也使得我们在设计过程中可以在个人能力上得到提高，在实验中收获知识的快乐。

本组目前已经完成所有基本要求　提高要求部分我们的组员正在努力。

附录

1、openmv 程序

```
import sensor, image, time
thresholds = (20, 75, 28, 127, -128, 127)
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.set_windowing((240, 240))
sensor.skip_frames(time = 1000)
sensor.set_auto_gain(False)
sensor.set_auto_whitebal(False)
clock = time.clock()
from pyb import Servo
from pyb import UART
uart = UART(3, 76800)
left_roi = [20,20,200,200]
while(True):
        clock.tick()
        img = sensor.snapshot()
        for blob in img.find_blobs([thresholds],roi=left_roi
):
                uart_buf = bytearray([blob[0],blob[1],0x0d,0
x0a])
                uart.write(uart_buf)
        print(clock.fps())
```

2、stm32f4 主控程序

PID 设定位置函数

```
void pid_set_position(u8 x_expected,u8 y_expected)
{

//**************  x 轴变量*************
    double deltaEk=0;
    double T_Ti;
```

```
    double Td_T;

    u16 PWMval_x;

//**************  y轴变量*************
    double deltaEk_=0;
    double T_Ti_;
    double Td_T_;

    u16 PWMval_y;

    if(pidx.caculate_period<pidx.T/5.0)//如果时间间隔还没到，则不进行计算
        return;
    if(usart_irq_flag>400)
    {
        TIM_SetCompare1(TIM3,pidy.OUT0);
        TIM_SetCompare2(TIM3,pidx.OUT0);
        return;
    }


    //***********************  x轴pid计算****************
    pidx.Position_expected=(double)x_expected;
    pidx.Position_current=(double)x_current;

    pidx.Ek=pidx.Position_expected-pidx.Position_current;//比例偏差
    pidx.Pout=pidx.Kp*pidx.Ek;//比例输出项

    if(my_abs(pidx.Ek)<6.0)///////////////停止的判定
        pid_error_count++;
    else
        pid_error_count=0;
    if(pid_error_count>36)
        return ;

    pidx.SEk+=pidx.Ek;//历史偏差总和
    T_Ti=pidx.T/pidx.Ti;
    pidx.Iout=pidx.Ki*T_Ti*pidx.SEk;//积分输出项


    deltaEk=pidx.Ek-pidx.Ek_1;//最近两次偏差之差
    Td_T=pidx.Td/pidx.T;
    pidx.Dout=pidx.Kd*Td_T*deltaEk;//微分输出项

```

```
51.     pidx.Ek_1=pidx.Ek;
52.
53.     pidx.OUT=pidx.Pout+pidx.Iout+pidx.Dout+pidx.OUT0;
54.     pidx.OUT_1=pidx.OUT;//保留上一次输出值
55. //  if(my_abs(pidx.OUT-pidx.OUT_1)<0.08)
56. //      pidx.OUT=pidx.OUT_1;
57.
58.     PWMval_x=(u16)pidx.OUT;
59.     if(PWMval_x<250)PWMval_x=250;
60.     if(PWMval_x>1250)PWMval_x=1250;
61.     TIM_SetCompare2(TIM3,PWMval_x); //通道二为 x
62.
63.
64.     //***********************  y 轴 pid 计算****************
65.     pidy.Position_expected = (double)y_expected;
66.     pidy.Position_current = (double)y_current;
67.
68.     pidy.Ek = pidy.Position_expected - pidy.Position_current;//比例偏
    差
69.     pidy.Pout = pidy.Kp * pidy.Ek;//比例输出项
70.
71.
72.     pidy.SEk += pidy.Ek;//历史偏差总和
73.     T_Ti_ = pidy.T  / pidy.Ti;
74.     pidy.Iout = pidy.Ki * T_Ti_*pidy.SEk;//积分输出项
75.
76.
77.     deltaEk_ = pidy.Ek - pidy.Ek_1;//最近两次偏差之差
78.     Td_T_ = pidy.Td / pidy.T;
79.     pidy.Dout = pidy.Kd * Td_T_ * deltaEk_;//微分输出项
80.
81.     pidy.Ek_1=pidy.Ek;
82.
83.     pidy.OUT=pidy.Pout + pidy.Iout + pidy.Dout + pidy.OUT0;
84.     pidy.OUT_1=pidy.OUT;//保留上一次输出值
85. //  if(my_abs(pidy.OUT-pidy.OUT_1)<0.08)
86. //      pidy.OUT=pidy.OUT_1;
87.
88.     PWMval_y=(u16)pidy.OUT;
89.     if(PWMval_y<250)PWMval_y=250;
90.     if(PWMval_y>1250)PWMval_y=1250;
91.     TIM_SetCompare1(TIM3,PWMval_y); //修改比较值，修改占空比  通道 1 为 y;
92.
93.
```

```
94.     pidx.caculate_period=0;
95. }
```

主函数

```
1.  u32 time=0;
2.  u16 usart_irq_flag=0;
3.  u8 mode=3;
4.  extern u8 USART_RX_BUF[USART_REC_LEN];
5.  extern u8 USART2_RX_BUF[USART_REC_LEN];
6.  extern u8 x_current,y_current;
7.  extern PID pidx,pidy;
8.  int main(void)
9.  {
10.     u8 mode_change_flag=1;
11.     u8 key;
12.     u8 key2;
13.
14.     NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置系统中断优先级分
    组 2
15.     delay_init(168);  //初始化延时函数
16.     uart_init(76800);//初始化串口波特率为 115200
17.     uart2_init(76800);//防止信号线太长 导致干扰 所以选择 9600 波特率
18.     uart3_init(76800);
19.     LED_Init();
20.     TIM3_PWM_Init(10000-1,168-1);    //84M/168=500Khz 的计数频率,重装载值
    10000，所以 PWM 频率为 500K/10000=50hz.
21.                                      //对于舵机 比较值从 250 到 1250
22.     TIM4_Int_Init(50-1,8400-1);//5ms 中断 PID 更新周期 5ms  84M/8400=10，
    000Hz，10，000/50=200Hz
23.     KEY_Init();
24.     pid_init(5);
25.     MatrixKeyConfiguration();
26.
27.     OLED_Init();
28.     OLED_Clear();
29.
30.     OLED_ShowString(0,0,"x");
31.     OLED_ShowString(0,2,"y");
32.     OLED_ShowString(40,0,"Kp");
33.     OLED_ShowString(40,2,"Ki");
34.     OLED_ShowString(30,4,"Kd");
35.   OLED_ShowString(0,6,"pidx");
36.     while(1)
```

```
37.     {
38.         key2=KEY_Scan(0);
39.         if(key2==KEY1_PRES)
40.         {
41.             mode++;
42.             mode_change_flag=1;
43.             time=0;
44.             if(mode==5)
45.                 mode=1;
46.         }
47.
48.         key=GetMatrixKeyValue();
49.         if(key==1)
50.         {
51.             pidx.Kp+=0.1;
52.             pidy.Kp+=0.1;
53.                 key=0;
54.         }
55.         if(key==4)
56.         {
57.             pidx.Kp-=0.01;
58.             pidy.Kp-=0.01;
59.                 key=0;
60.         }
61.
62.         if(key==2)
63.         {
64.             pidx.Ki+=0.01;
65.             pidy.Ki+=0.01;
66.                 key=0;
67.         }
68.         if(key==5)
69.         {
70.             pidx.Ki-=0.01;
71.             pidy.Ki-=0.01;
72.                 key=0;
73.         }
74.
75.         if(key==3)
76.         {
77.             pidx.Kd+=0.01;
78.             pidy.Kd+=0.01;
79.                 key=0;
80.         }
```

```
81.         if(key==6)
82.         {
83.             pidx.Kd-=0.01;
84.             pidy.Kd-=0.01;
85.                 key=0;
86.         }
87.         if(key==7)
88.         {
89.             pidx.OUT0+=1;
90.                 key=0;
91.         }
92.         if(key==0x0a)
93.         {
94.             pidx.OUT0-=1;
95.                 key=0;
96.         }
97.         if(key==9)
98.         {
99.             pidy.OUT0+=1;
100.                key=0;
101.         }
102.         if(key==0x0c)
103.         {
104.             pidy.OUT0-=1;
105.                 key=0;
106.         }
107.
108.         show_parameters();
109.
110.         switch(mode)
111.         {
112.             case 1://停留在区域 2
113.                 if(mode_change_flag==1)//每当改变了模式的时候 中值就需要改
    变
114.                 {
115.                     pid_set_OUT0(8);//设定中值
116.                     mode_change_flag=0;
117.                     time=0;
118.                 }
119.                 pid_set_position(0x70,0xc0);
120.             break;
121. //          case 1://做测试 停留在某区域
122. //              if(mode_change_flag==1)//每当改变了模式的时候 中值就需要改
    变
```

```
123. //                {
124. //                    pid_set_OUT0(1);//设定中值
125. //                    mode_change_flag=0;
126. //                }
127. //                pid_set_position(0x26,0x25);
128. //            break;
129.
130.
131.              case 2://从区域 1 到 5
132.                  if(mode_change_flag==1)//每当改变了模式的时候 中值就需要改
     变
133.                  {
134.                      pid_set_OUT0(9);//设定中值
135.                      mode_change_flag=0;
136.                      pidx.Kd=6.78;
137.                      pidy.Kd=6.78;
138.                      time=0;
139.                  }
140.                  if(time==2*threshold)//准备进入 5
141.                      mode_change_flag=2;
142.                  if(mode_change_flag==2)//在同一个模式中，小球在不同的地
     方 中值也需要改变
143.                  {
144.                      pid_set_OUT0(5);//设定中值
145.                      mode_change_flag=0;
146.                  }
147.                  if(time<2*threshold)//定在 1
148.                  {
149.                      pid_set_position(0xb7,0xba);
150.                  }
151.                  else                    //定在 5
152.                      pid_set_position(0x75,0x75);
153.                  break;
154.
155.
156.              case 3:  //从 1 到 4 在 4 内停留不少于 2 秒  然后进入 5。总时间不超过
     20   9 - 6 - 5
157.                  if(mode_change_flag==1)//每当改变了模式的时候 中值就需要改
     变 计时需要清零
158.                  {
159.                      pid_set_OUT0(9);//设定中值
160.                      mode_change_flag=0;
161.                      time=0;
162.                  }
```

```
163.
164.                 if(time==1.5*threshold)//准备进入 4
165.                     mode_change_flag=2;
166.                 if(mode_change_flag==2)//在同一个模式中，小球在不同的地
    方 中值也需要改变
167.                 {
168.                     pid_set_OUT0(6);//设定中值
169.                     pidx.Kd=6.78;
170.                     pidx.OUT0=851;
171.                     mode_change_flag=0;
172.                 }
173.
174.                 if(time==5.5*threshold)//准备进入 5
175.                     mode_change_flag=3;
176.                 if(mode_change_flag==3)//在同一个模式中，小球在不同的地
    方 中值也需要改变
177.                 {
178.                     pid_set_OUT0(5);//设定中值
179.                     mode_change_flag=0;
180.                 }
181.
182.                 if(time<1.5*threshold)//停在 1
183.                 {
184.                     pid_set_position(0xc4,0xbe);
185.                 }
186.                 else if(time<5.5*threshold&&time>1.5*threshold)//停在
    4 一个 threshold 是 2s 区间为 4 个 threshold 为 8s
187.                     pid_set_position(0xbc,0x77);
188.                 else                                    //停在 5
189.                     pid_set_position(0x75,0x75);
190.                 break;
191.
192.
193.             case 4://从区域 1 到 9   可以分段实现 先到 5 附近   再到 5 和 9 中间
    某处  再到 9
194.                 if(mode_change_flag==1)//每当改变了模式的时候 中值就需要改
    变 计时需要清零
195.                 {
196.                     pid_set_OUT0(9);//设定中值
197.                     mode_change_flag=0;
198.                     time=0;
199.                 }
200.                 if(time==1.5*threshold)//准备进入 5
201.                     mode_change_flag=2;
```

```
202.                    if(mode_change_flag==2)//在同一个模式中，小球在不同的地
    方 中值也需要改变
203.                    {
204.                        pid_set_OUT0(5);//设定中值
205.                        mode_change_flag=0;
206.                    }
207.                    if(time==2*threshold)//准备进入 5 和 9 中间的位置
208.                        mode_change_flag=3;
209.                    if(mode_change_flag==3)//在同一个模式中，小球在不同的地
    方 中值也需要改变
210.                    {
211.                        pid_set_OUT0(5);//设定中值
212.                        mode_change_flag=0;
213.                    }
214.                    if(time==3.5*threshold)//准备进入 5 和 9 中间的位置
215.                        mode_change_flag=4;
216.                    if(mode_change_flag==4)//在同一个模式中，小球在不同的地
    方 中值也需要改变
217.                    {
218.                        pid_set_OUT0(1);//设定中值
219.                        mode_change_flag=0;
220.                        pidx.Kd=6.78;
221.                        mode_change_flag=0;
222.                    }
223.                    if(time<=threshold)//停在 1
224.                        pid_set_position(0xc4,0xbe);
225.                    else if(time<=2*threshold&&time>threshold)    //停在
    5
226.                        pid_set_position(0x7b,0x7b);
227.                    else if(time<=3.5*threshold&&time>2*threshold)    //停
    在 5 和 9 中间
228.                        pid_set_position(0x4d,0x4c);
229.                    else                                              //停在
    9
230.                        pid_set_position(0x30,0x2a);
231.                    break;
232.
233.                default: break;
234.            }
```