

**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

[一、实验要求 2](#_Toc9342602)

[1.1任务 2](#_Toc9342603)

[1.2要求 2](#_Toc9342604)

[1.3说明 3](#_Toc9342605)

[二、系统方案 3](#_Toc9342606)

[2.1设计整体框图 3](#_Toc9342607)

[2.2主控制器件 5](#_Toc9342608)

[2.3图像采集与处理方案 5](#_Toc9342609)

[2.4滚球控制系统方案 5](#_Toc9342610)

[三、电路与程序设计 8](#_Toc9342611)

[3.1电路仿真设计 8](#_Toc9342612)

[3.2程序设计 9](#_Toc9342613)

[3.3 PID算法 9](#_Toc9342614)

[四、系统整体外观布局设计 12](#_Toc9342615)

[4.1整体架构 12](#_Toc9342616)

[4.2舵机连杆 12](#_Toc9342617)

[4.3主体架构 12](#_Toc9342618)

[4.4舵机的机械臂设计 13](#_Toc9342619)

[五、测试方案 13](#_Toc9342620)

[5.1硬件测试 13](#_Toc9342621)

[5.2硬件软件联调 13](#_Toc9342622)

[5.3测试条件与仪器 14](#_Toc9342623)

[六、实验感想及总结 14](#_Toc9342624)

[附录 15](#_Toc9342625)

[1、openmv程序 15](#_Toc9342626)

[2、stm32f4主控程序 15](#_Toc9342627)

**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：

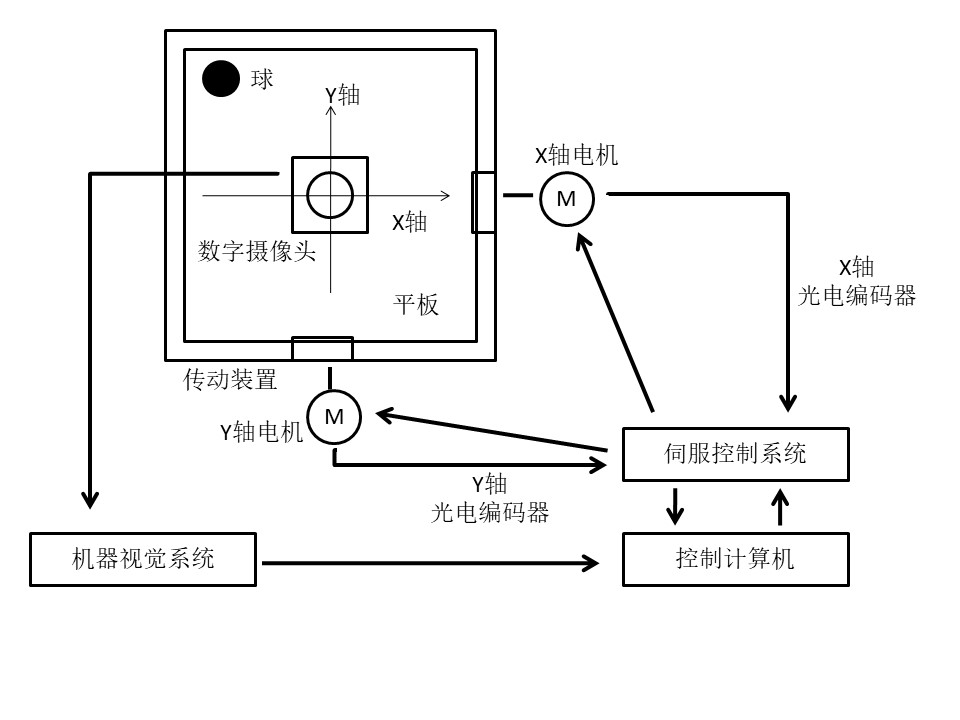


图 1系统结构图

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。

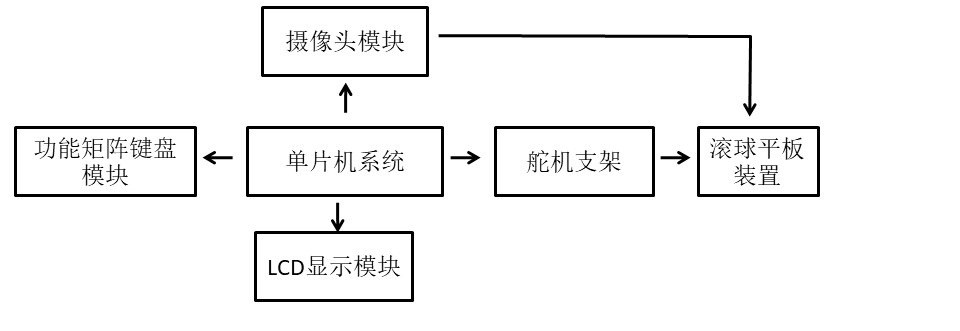


图 2总体方案框图

# 2.2主控器件

本组采用STM32F407单片机为主控芯片，具有速度快，片上资源丰富等优点，且具备丰富的外围借口，拓展性好。

# 2.3图像采集与处理方案

采用摄像头对小球进行过准确定位，根据实验要求，摄像机计算出小球当前坐标，通过通用串行异步输入输出接口（USART）发送给单片机，其波特率为76800baud。CPU计算出小球位置坐标与目的坐标的偏差，将其带入PID算法公式，从而控制舵机做出相应动作。外围电路增加LCD、矩阵键盘的人机接口，更有利于进行各实验参数的调试，这些参数主要包括PID两项系数：Kp和Kd。同时还能够实时显示小球的当前坐标等信息。

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RGB565，240x240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。

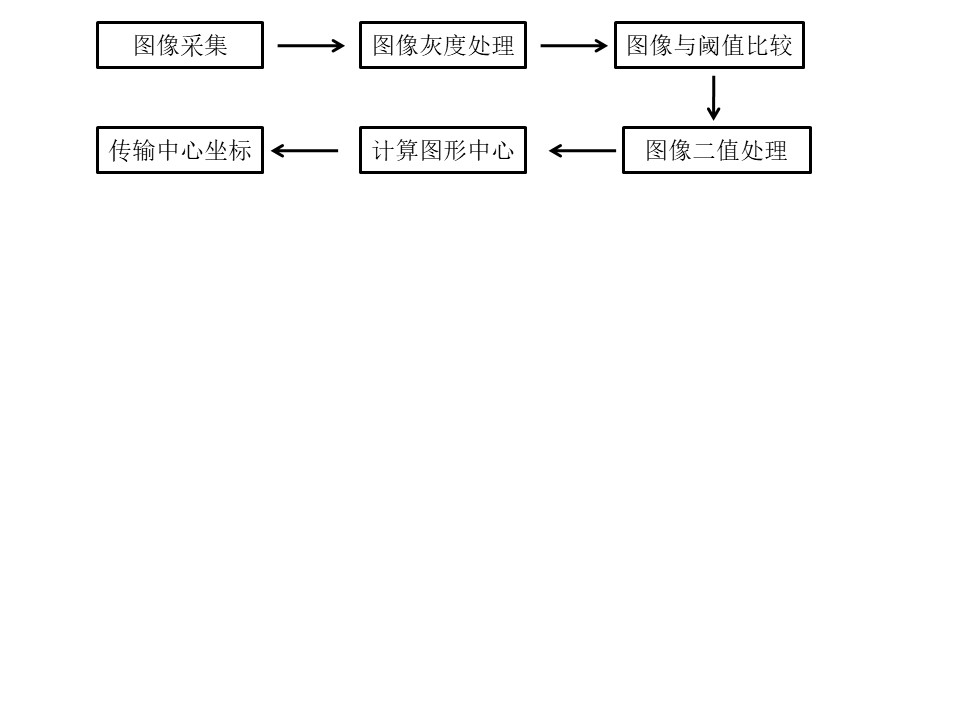
2.3.2图像处理

通过编程实现对保存的三个二维数组进行灰度处理，将三个二维数组组成的RGB图像转换为一个二维数组表示的灰度图像，将灰度图像中灰度值与所设定阈值进行比较，高于阈值为“1”，低于阈值为“0”，将图片转换为由一个二维数组表示的二值图像，并将表示二值图像的二维数组中值为“1”的像素点的横纵坐标最大值与最小值取平均值可以得到小球中心横纵坐标值，并将结果传送给控制系统进行相关处理。调试过程中可将满足条件的像素点范围与实际小球坐标范围进行比较来调整阈值，直到所得范围与实际范围相符合。

2.3.3图像采集与处理框图

2.4滚球控制系统方案

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。



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。

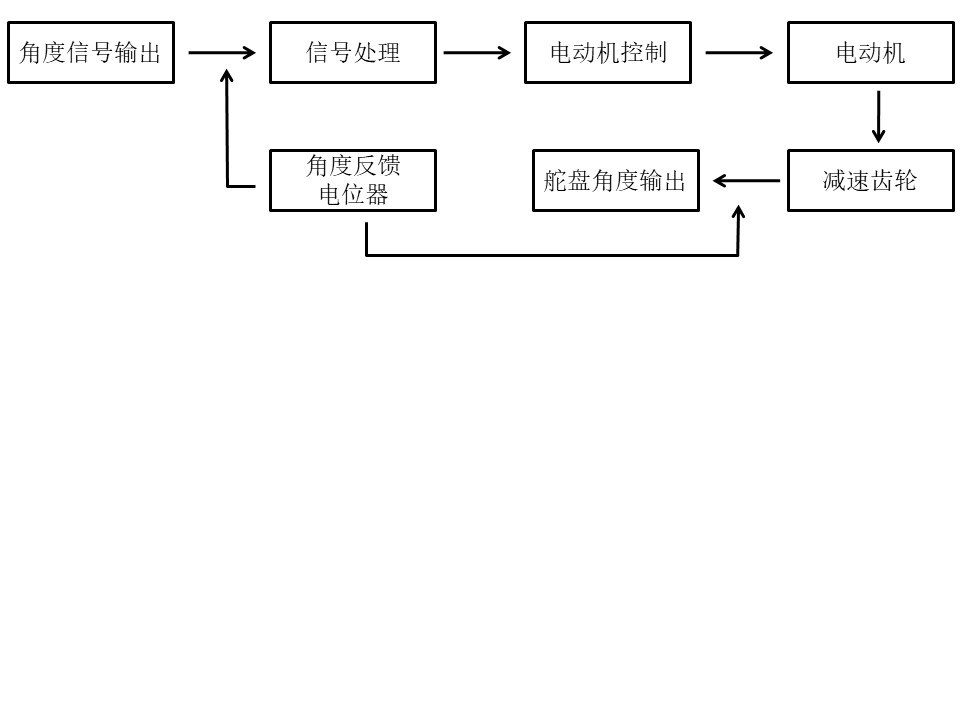
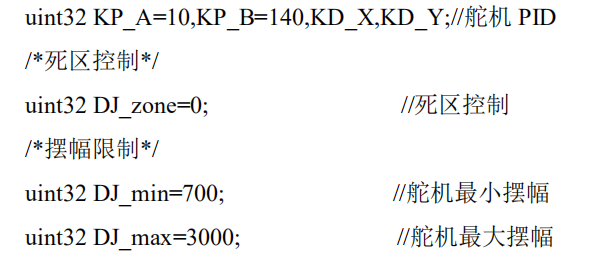


图3舵机控制流程图

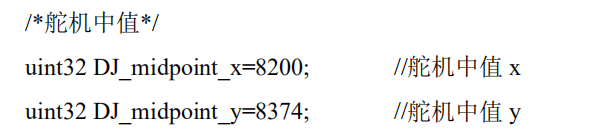
图4舵机控制PWM

1. 具体编程流程

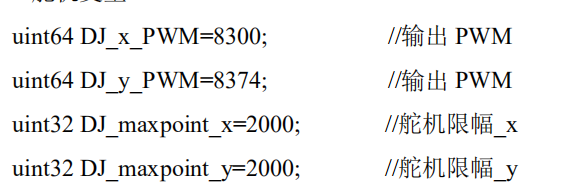
①摆幅控制及停止角度的精确度



②中值确定



③PWM



2.4.3机械臂

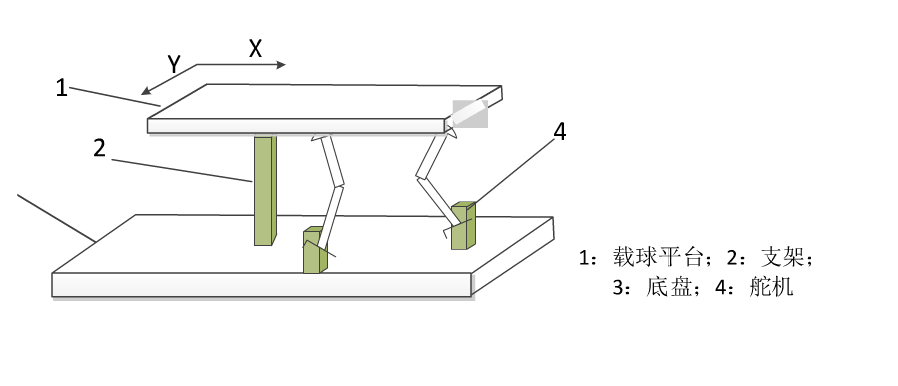


图5机械臂设计

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电路仿真设计

将单片机，摄像头，显示模块，电源模块进行布局，如下图所示：

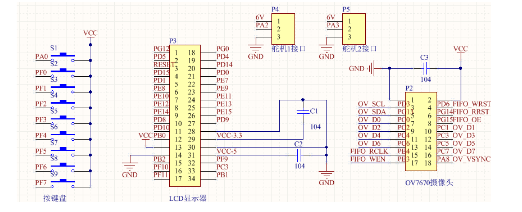


图 6电路仿真布局图

# 3.2程序设计

3.2.1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。

3.2.2程序流程图

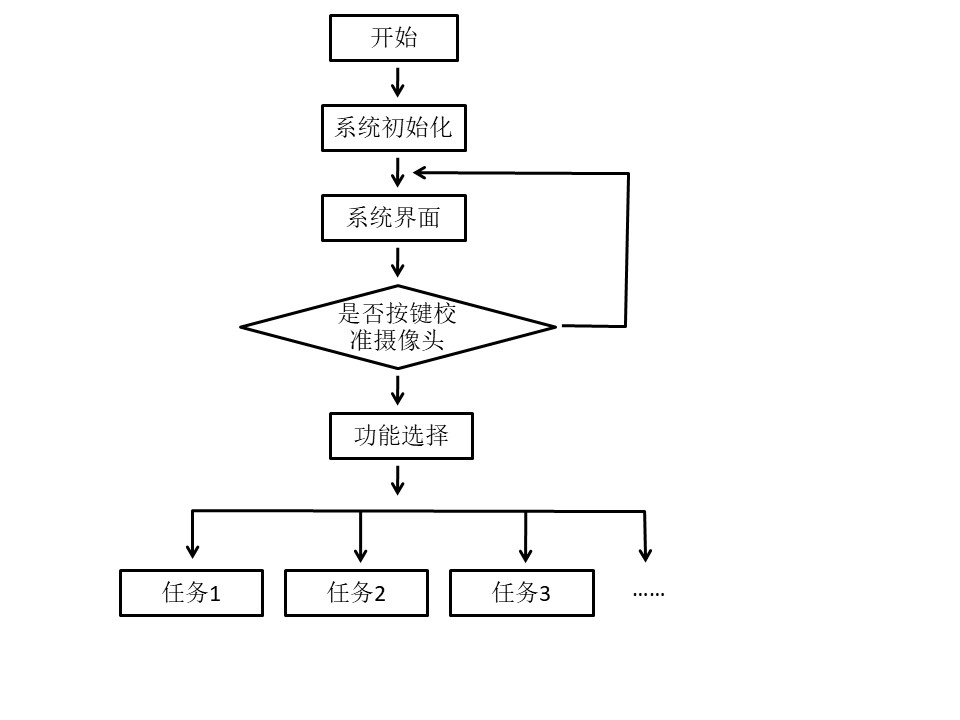


图 7程序流程图

# 3.3 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。

2.3.1 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。

3.3.2 PID位置式计算公式：

（式1-1）

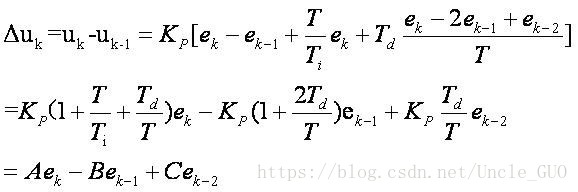
式1-1中为积分系数，为微分系数，u(n)为第k个采样时刻的控制，T为采样周期。

3.3.3增量式计算公式可有式1推到得到

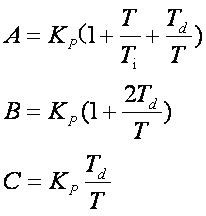
由式1可以得控制器的第k-1个采样时刻的输出值为：

（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：



式中,



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

3.3.4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。

1. 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

使用一米\*一米的三合板木板作底部，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。

# 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。

# 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。

# 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。

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

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。

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。

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。

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,0x0a])

uart.write(uart\_buf)

print(clock.fps())

# 2、stm32f4主控程序

1 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

pidx.Ek\_1=pidx.Ek;

pidx.OUT=pidx.Pout+pidx.Iout+pidx.Dout+pidx.OUT0;

pidx.OUT\_1=pidx.OUT;//保留上一次输出值

// if(my\_abs(pidx.OUT-pidx.OUT\_1)<0.08)

// pidx.OUT=pidx.OUT\_1;

PWMval\_x=(u16)pidx.OUT;

if(PWMval\_x<250)PWMval\_x=250;

if(PWMval\_x>1250)PWMval\_x=1250;

TIM\_SetCompare2(TIM3,PWMval\_x); //通道二为 x

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* y轴pid计算\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

pidy.Position\_expected = (double)y\_expected;

pidy.Position\_current = (double)y\_current;

pidy.Ek = pidy.Position\_expected - pidy.Position\_current;//比例偏差

pidy.Pout = pidy.Kp \* pidy.Ek;//比例输出项

pidy.SEk += pidy.Ek;//历史偏差总和

T\_Ti\_ = pidy.T / pidy.Ti;

pidy.Iout = pidy.Ki \* T\_Ti\_\*pidy.SEk;//积分输出项

deltaEk\_ = pidy.Ek - pidy.Ek\_1;//最近两次偏差之差

Td\_T\_ = pidy.Td / pidy.T;

pidy.Dout = pidy.Kd \* Td\_T\_ \* deltaEk\_;//微分输出项

pidy.Ek\_1=pidy.Ek;

pidy.OUT=pidy.Pout + pidy.Iout + pidy.Dout + pidy.OUT0;

pidy.OUT\_1=pidy.OUT;//保留上一次输出值

// if(my\_abs(pidy.OUT-pidy.OUT\_1)<0.08)

// pidy.OUT=pidy.OUT\_1;

PWMval\_y=(u16)pidy.OUT;

if(PWMval\_y<250)PWMval\_y=250;

if(PWMval\_y>1250)PWMval\_y=1250;

TIM\_SetCompare1(TIM3,PWMval\_y); //修改比较值，修改占空比 通道1为y;

pidx.caculate\_period=0;

}

2 主函数

u32 time=0;

u16 usart\_irq\_flag=0;

u8 mode=3;

extern u8 USART\_RX\_BUF[USART\_REC\_LEN];

extern u8 USART2\_RX\_BUF[USART\_REC\_LEN];

extern u8 x\_current,y\_current;

extern PID pidx,pidy;

int main(void)

{

u8 mode\_change\_flag=1;

u8 key;

u8 key2;

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);//设置系统中断优先级分组2

delay\_init(168); //初始化延时函数

uart\_init(76800);//初始化串口波特率为115200

uart2\_init(76800);//防止信号线太长 导致干扰 所以选择9600波特率

uart3\_init(76800);

LED\_Init();

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.

//对于舵机 比较值从250 到1250

TIM4\_Int\_Init(50-1,8400-1);//5ms中断 PID 更新周期5ms 84M/8400=10，000Hz，10，000/50=200Hz

KEY\_Init();

pid\_init(5);

MatrixKeyConfiguration();

OLED\_Init();

OLED\_Clear();

OLED\_ShowString(0,0,"x");

OLED\_ShowString(0,2,"y");

OLED\_ShowString(40,0,"Kp");

OLED\_ShowString(40,2,"Ki");

OLED\_ShowString(30,4,"Kd");

OLED\_ShowString(0,6,"pidx");

while(1)

{

key2=KEY\_Scan(0);

if(key2==KEY1\_PRES)

{

mode++;

mode\_change\_flag=1;

time=0;

if(mode==5)

mode=1;

}

key=GetMatrixKeyValue();

if(key==1)

{

pidx.Kp+=0.1;

pidy.Kp+=0.1;

key=0;

}

if(key==4)

{

pidx.Kp-=0.01;

pidy.Kp-=0.01;

key=0;

}

if(key==2)

{

pidx.Ki+=0.01;

pidy.Ki+=0.01;

key=0;

}

if(key==5)

{

pidx.Ki-=0.01;

pidy.Ki-=0.01;

key=0;

}

if(key==3)

{

pidx.Kd+=0.01;

pidy.Kd+=0.01;

key=0;

}

if(key==6)

{

pidx.Kd-=0.01;

pidy.Kd-=0.01;

key=0;

}

if(key==7)

{

pidx.OUT0+=1;

key=0;

}

if(key==0x0a)

{

pidx.OUT0-=1;

key=0;

}

if(key==9)

{

pidy.OUT0+=1;

key=0;

}

if(key==0x0c)

{

pidy.OUT0-=1;

key=0;

}

show\_parameters();

switch(mode)

{

case 1://停留在区域2

if(mode\_change\_flag==1)//每当改变了模式的时候 中值就需要改变

{

pid\_set\_OUT0(8);//设定中值

mode\_change\_flag=0;

time=0;

}

pid\_set\_position(0x70,0xc0);

break;

// case 1://做测试 停留在某区域

// if(mode\_change\_flag==1)//每当改变了模式的时候 中值就需要改变

// {

// pid\_set\_OUT0(1);//设定中值

// mode\_change\_flag=0;

// }

// pid\_set\_position(0x26,0x25);

// break;

case 2://从区域1 到 5

if(mode\_change\_flag==1)//每当改变了模式的时候 中值就需要改变

{

pid\_set\_OUT0(9);//设定中值

mode\_change\_flag=0;

pidx.Kd=6.78;

pidy.Kd=6.78;

time=0;

}

if(time==2\*threshold)//准备进入5

mode\_change\_flag=2;

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

{

pid\_set\_OUT0(5);//设定中值

mode\_change\_flag=0;

}

if(time<2\*threshold)//定在1

{

pid\_set\_position(0xb7,0xba);

}

else //定在5

pid\_set\_position(0x75,0x75);

break;

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

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

{

pid\_set\_OUT0(9);//设定中值

mode\_change\_flag=0;

time=0;

}

if(time==1.5\*threshold)//准备进入4

mode\_change\_flag=2;

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

{

pid\_set\_OUT0(6);//设定中值

pidx.Kd=6.78;

pidx.OUT0=851;

mode\_change\_flag=0;

}

if(time==5.5\*threshold)//准备进入5

mode\_change\_flag=3;

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

{

pid\_set\_OUT0(5);//设定中值

mode\_change\_flag=0;

}

if(time<1.5\*threshold)//停在1

{

pid\_set\_position(0xc4,0xbe);

}

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

pid\_set\_position(0xbc,0x77);

else //停在5

pid\_set\_position(0x75,0x75);

break;

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

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

{

pid\_set\_OUT0(9);//设定中值

mode\_change\_flag=0;

time=0;

}

if(time==1.5\*threshold)//准备进入5

mode\_change\_flag=2;

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

{

pid\_set\_OUT0(5);//设定中值

mode\_change\_flag=0;

}

if(time==2\*threshold)//准备进入5和9中间的位置

mode\_change\_flag=3;

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

{

pid\_set\_OUT0(5);//设定中值

mode\_change\_flag=0;

}

if(time==3.5\*threshold)//准备进入5和9中间的位置

mode\_change\_flag=4;

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

{

pid\_set\_OUT0(1);//设定中值

mode\_change\_flag=0;

pidx.Kd=6.78;

mode\_change\_flag=0;

}

if(time<=threshold)//停在1

pid\_set\_position(0xc4,0xbe);

else if(time<=2\*threshold&&time>threshold) //停在5

pid\_set\_position(0x7b,0x7b);

else if(time<=3.5\*threshold&&time>2\*threshold) //停在5和9中间

pid\_set\_position(0x4d,0x4c);

else //停在9

pid\_set\_position(0x30,0x2a);

break;

default: break;

}