燕山大学

课 程 设 计 说 明 书

题目：基于 STM32 的 X-Y 平台控制系统设计

学院（系）：电气工程学院

年级专业：14 级工自一班

学　　号：140103010021

学生姓名：魏士杰

指导教师：刘福才

教师职称：教授

燕山大学课程设计（论文）任务书

院（系）：电气工程学院　　基层教学单位：自动化系

学 号	140103010021	学生姓名	魏士杰	专业（班级）	14 工自 1 班
设计题目	基于 STM32 的 X-Y 平台控制系统设计				
设计技术参数	本系统基于 STM32 微处理器，以驱动模块驱动步进电机，直流电源为驱动器提供直流电源，USART 串口收发数据，以上位机控制设备的运行并用上位机模拟形成立体位置显示。				
设计要求	实现自动画方、画圆控制，限位的保护控制，立体位置显示以及定位，简单的立体控制。				
工作量	1、完成设计说明书一份(包括生产工艺对单片机控制系统的要求、单片机控制系统的总体设计方案、控制系统框图及各硬件接口电路工作原理，控制系统软件设计总流程图及子程序流程图，电源的设计等)；文稿用钢笔或圆珠笔书写，字迹应工整、清晰(打印也可)； 2、绘制单片机控制系统原理图(包括接口电路及电源)A2 图纸一张，可用铅笔绘制，应符合相关制图标准。				
工作计划	第 1 周：查阅相关资料，进行逐个环节的硬件电路设计，给出相应的主程序及相关子程序流程图，并请指导教师审核； 第 2 周：绘制硬件电路原理图（A2），并严格执行电气原理图绘图标准；编写主程序及相关子程序；撰写课程设计说明书，并注意文字、图表、及格式的规范化；最后进行答辩。				
参考资料	1. 阮毅. 电力拖动自动控制系统——运动控制系统. 第 4 版，北京：机械工业出版社，2009.8 2. 张淑清. 嵌入式单片机 STM32 设计及应用技术. 第 1 版， 北京：国防工业出版社，2016.7 3. 夏德铃 翁贻方. 自动控制理论. 第 4 版， 北京：机械工业出版社，2012.11				
指导教师签字			基层教学单位主任签字		

说明：此表一式二份，学生、指导教师各一份。

2018 年 1 月 12 日

摘要

步进电机在不精准的定位控制中有着一定的地位，在一定情况下可以通过步进电机的步数进行不精准的定位，我们所做的任务为使用三个步进电机实现 XYZ 三个坐标轴的立体移动和控制，并使用软件编程实现上位机控制，实现空间位置显示，并通过串口通信实现电机的前后、左右、上下的控制，同时在电机移动的轨道的最前端和最后端使用限位开关实现限位控制，防止电机超过量程，到不可控范围内。

步进电机能实现精确的角度和转数，具有良好的步进特性，最适合数字控制。在工控设备中得到了广泛的应用。而单片机具有芯片体积小，兼容性强，低电压地，低功耗等特点，使单片机成为驱动步进电机的最佳空盒子单元。所以单片机控制步进电机系统控制精度高，运行稳定，得以广泛运用。

关键词：步进电机，STM32 单片机，计算机，Qt，上位机

目录

1 项目任务整体简述

1.1 项目设计思路

本系统使用的是三个步进电机和三个驱动模块，使用 STM32 单片机进行电机控制，实现空间移动，从而达到绘制图像的目的；本系统使用上位机进行系统控制，实现电机的上下、前后、左右的移动和定位，并实现方形图形的绘制，在上位机可以显示电机的位置和运行轨迹。

电机的速度控制为频率控制，使用单片机生成可变的 PWM 波实现电机的调速，整体的逻辑部分由单片机和上位机共同完成。

1.2 设计方案框图

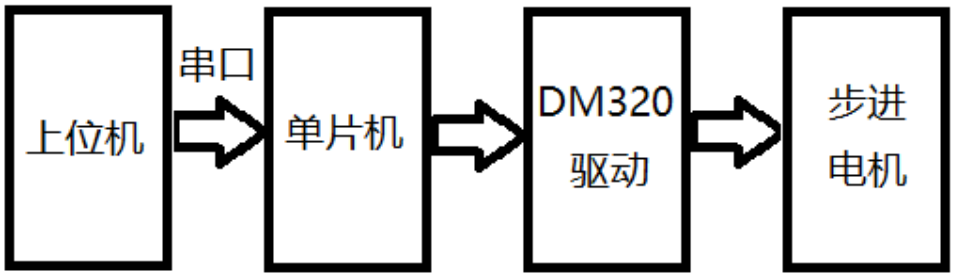


图 1

1.3 成果预期

使用上位机可控制单片机进行步进电机的控制，并实现方形图像绘制以及定位显示。

2 项目整体技术设计

2.1 元件清单

表 1．元件清单

器件种类	型号	个数
单片机	STM32F103VET6	1 个
电机驱动	DM320	3 个

变压器	220V/24V	1 个
排针及杜邦线		若干
插头及电源线		1 个
步进电机		3 个

2.2 系统原理框图

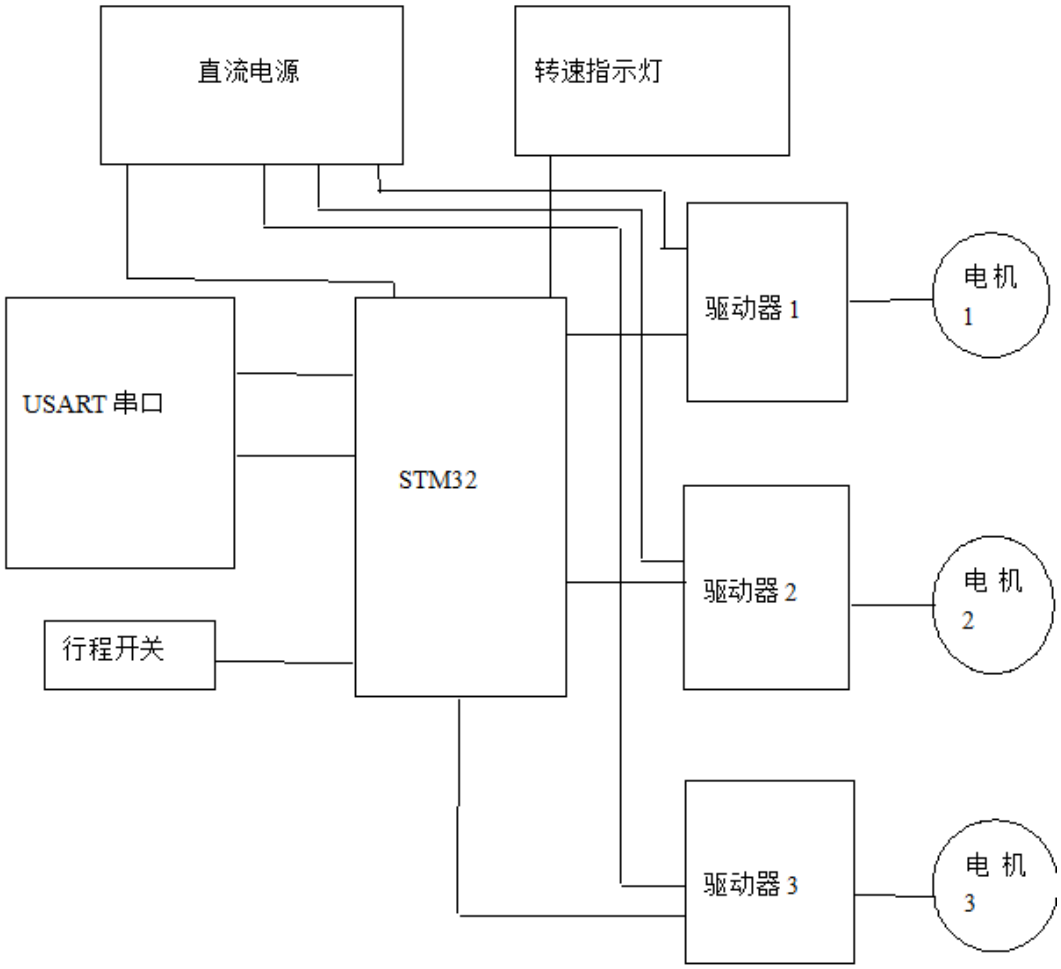


图 2

2.3 硬件组成与结构

根据构思与分析我们需要使用的硬件部分有：STM32、DM320 步进电机驱动、步进电机、串口线，计算机以及电源模块。原理图如下：

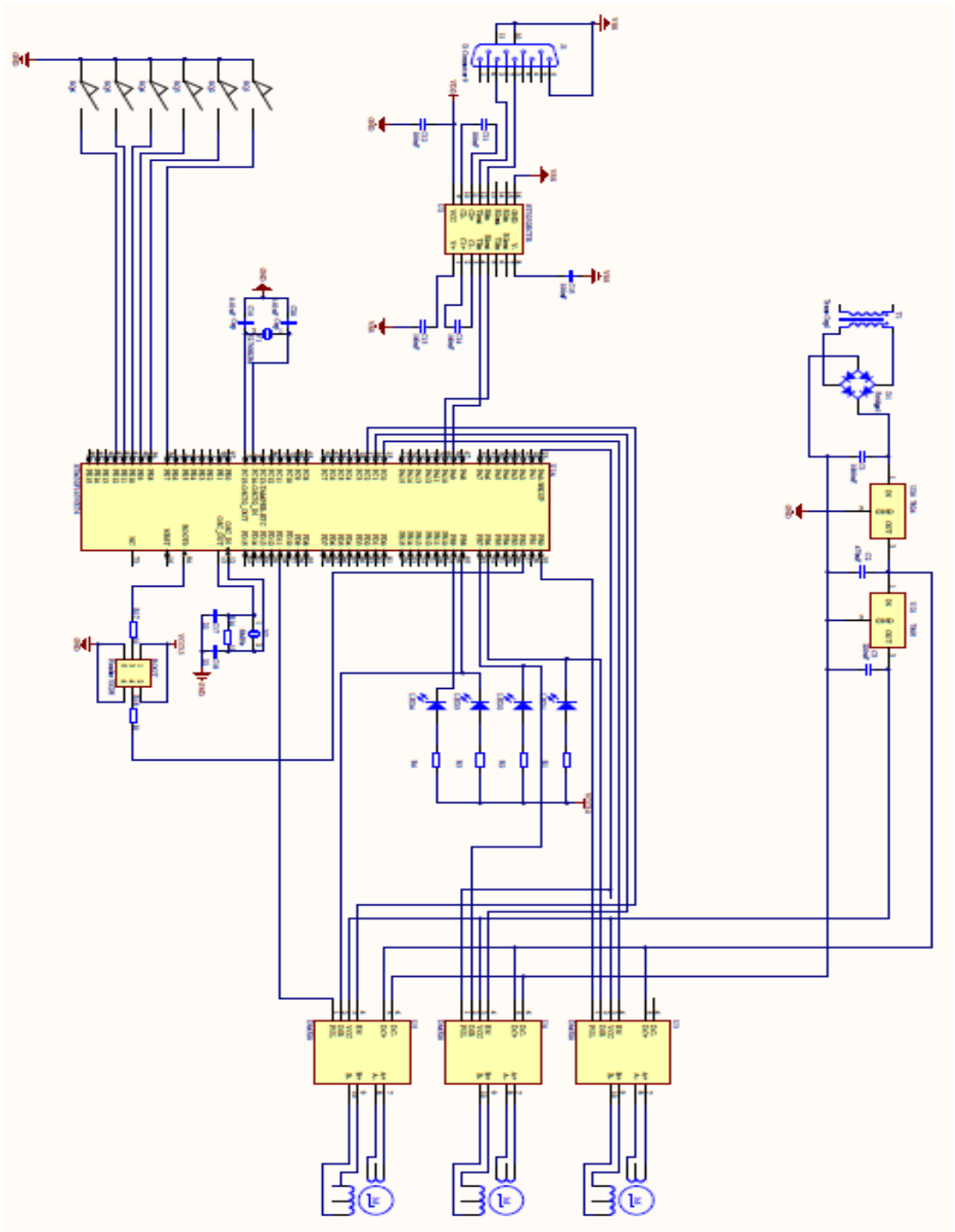


图 3

2.4 软件设计与编程

2.4.1 单片机系统主程序逻辑分析与设计

主程序中使用简单的轮询结构实现系统的基本控制以及限位控制和绘制轨迹控制，其中具体流程如下：

程序流程图如下：

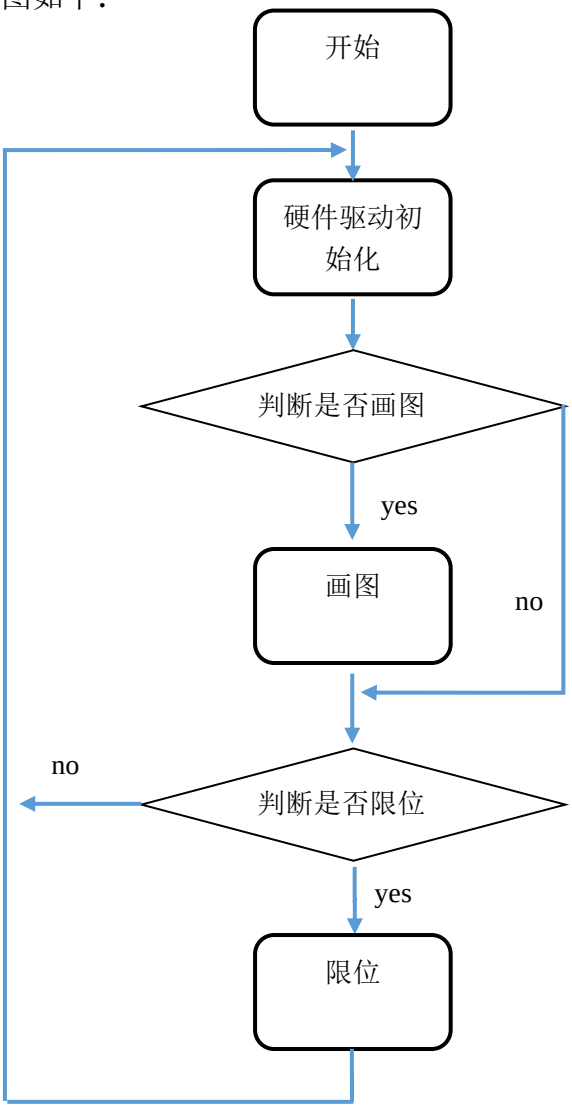


图 4

开机进入主函数，主函数首先进行硬件驱动的初始化，主要有 GPIO 初始化，串口初始化，定时器及 PWM 初始化，以及位置初始化。之后进入 while

循环进行绘图查询和限位查询，并进行响应的控制。

单片机主函数程序实现：

```
int main(void)
{
    Init_NVIC_PriorityGroupConfig();
    USART_Init_Config(115200);
    PWM_Define_Init();
    TIM_Init();
    LED_Config();
    Travel_Switch_Config();
    Motor_Config();
    Postion_Init();
    while(1)
    {
        if(GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_7)==RESET)
        Tr_Sw_Enable_1=true;
        if(GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_10)==RESET)
        Tr_Sw_Enable_2=true;
        if(GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_11)==RESET)
        Tr_Sw_Enable_3=true;
        if(play_square==true) Play_Square();
        if(Move_true==true) MoveToPostion();
        Play_Circle();
        Travel_switch_control();
        Motor_Enable_Control();
    }
}
```

2.4.2 方形绘制程序设计

通过四个行程开关进行定位，判断动作方向，从而控制电机实现方形轨迹的绘制。

```
void Play_Square()
{
if((GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_12)==RESET)&&(GPIO_ReadInputDataBi
t(GPIOE,GPIO_Pin_9)==RESET))
      {
            GPIO_SetBits(GPIOB,GPIO_Pin_7);
      }
if((GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_12)==RESET)&&(GPIO_ReadInputDataBi
t(GPIOE,GPIO_Pin_10)==RESET))
            {GPIO_ResetBits(GPIOB,GPIO_Pin_8);}
if((GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_11)==RESET)&&(GPIO_ReadInputDataBit
(GPIOE,GPIO_Pin_10)==RESET))
            { GPIO_ResetBits(GPIOB,GPIO_Pin_7);}
if((GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_11)==RESET)&&(GPIO_ReadInputDataBit
(GPIOE,GPIO_Pin_9)==RESET))
      {
            GPIO_SetBits(GPIOB,GPIO_Pin_8);
      }
      if((GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_7)==RESET)){
            GPIO_ResetBits(GPIOB,GPIO_Pin_9);
      }
      if((GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_8)==RESET))
      {
            GPIO_SetBits(GPIOB,GPIO_Pin_9);
      }
      Motor_2_Enable=true;
      Motor_0_Enable=true;
      Motor_1_Enable=true;
}
```

2.4.3 上位机信息处理函数程序设计：

本函数一共设定了三位控制位进行电机的动作控制，可实现电机的动作和曲线绘制控制。串口通讯使用了中断处理，在接收到信息之后进行处理并控制。

```
void USART1_IRQHandler(void)
{
      u8 Res;int count;
      static int n=0;
      int p_x,p_y;
      if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
      {
            USART_ClearITPendingBit(USART1, USART_IT_RXNE);
```

```
            Res =USART_ReceiveData(USART1);
            USART_SendData(USART1,Res);
            if(Res!='#'){
                  set_char[n]=Res;
                  n++;
                  if(n>9) n=9;
            }
            if(Res=='#')
            {
                  switch(set_char[0]){
                        case '0':
                              if(set_char[1]=='0'){if(TR_SW_6!=true)
                              GPIO_ResetBits(GPIOB,GPIO_Pin_7); }
                              else { if(TR_SW_5!=true) GPIO_SetBits(GPIOB,GPIO_Pin_7); }
                              if(set_char[2]=='0'){Motor_0_Enable=false;play_square=false;
                              Move_true=false;}
                              else {Motor_0_Enable=true;}
                        break;
                        case '5':
                              if(set_char[1]=='0') {
                                    play_square = false;
                                    Motor_Enable=false;}
                                    else { play_square = true;
                                    Motor_Enable=true; }
                        break;
                        case '6':
                              if(set_char[1]=='0') {
                              play_circle = false;
                              x=R;y=0;
                              Motor_2_Enable=false;
                              Motor_0_Enable=false;
                              Step_Control(true);
                              }
                              else { play_circle = true; x=R;y=0;}
                        break;
                        case 'p':
p_x=chartoint(set_char[1])*1000+chartoint(set_char[2])*100+chartoint(set_char[3])*10+charto
int(set_char[4]);
p_y=chartoint(set_char[5])*1000+chartoint(set_char[6])*100+chartoint(set_char[7])*10+charto
int(set_char[8]);
                        X_P=p_x;
                        Y_P=p_y;
                        Move_true=true;
                        break;
                        case 'g':
                              get_sure=true;
                        case 'm':
count=chartoint(set_char[1])*1000+chartoint(set_char[2])*100+chartoint(set_char[3])*10+chart
oint(set_char[4]);
                              Motor_0_Enable=true;
                              Step_Count_Control(0,count);
```

```
                    break;

                    }

                n=0;
            }
        }
}
```

2.4.4 行程开关限位程序设计

简单的控制逻辑即为遇见限位开关则停止电机运行，并使用判断值进行判断控制。

```
    if(GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_7)==RESET) {
    TR_SW_1=true;
    Z_TIM_3=0;
    if((TR_SW_Var_1==false)||(GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_9)==SET))
    GPIO_ResetBits(GPIOC,GPIO_Pin_2);
    TR_SW_Var_1=true;
    }
    else {TR_SW_1=false; TR_SW_Var_1=false;}

    if(GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_8)==RESET) {
    TR_SW_2=true;

if((TR_SW_Var_2==false)||(GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_9)==RESET))
    GPIO_ResetBits(GPIOC,GPIO_Pin_2);
    TR_SW_Var_2=true;
    }
    else {TR_SW_2=false; TR_SW_Var_2=false;}
```

2.4.5 圆形插补算法程序设计

根据步进电机的精确定位的特性，我们可以通过下一次各方向的步数设定进行位置定位以及形状的绘制，圆形插补算法是实现圆形绘制的一种方法，根据最小误差决定下一步的移动位置，具体位置控制实现如下，然后通过具体位置的控制电机的移动。

```
case 1:
    x2=x-1;
    y2=y+1;
    if(x==0) case_n = 2;
    break;
case 2:
    x2=x-1;
    y2=y-1;
    if(y==0) case_n = 3;

    break;
case 3:
    x2=x+1;
    y2=y-1;
    if(x==0) case_n=4;
    break;
case 4:
    x2=x+1;
    y2=y+1;
    if(y==0) case_n=1;
    break;
```

2.4.6 电机的步进脉冲控制程序实现

步进电机的控制通过脉冲的个数以及细分的程度进行控制，具体实现的方式如下：

```
if(period_0 < 5) {
    GPIO_SetBits(GPIOB,GPIO_Pin_0);
 }
 else{
    GPIO_ResetBits(GPIOB,GPIO_Pin_0);
 }
```

```
        if((PWM_Enable_0==true)||(PWM_Count_0>0)) {
            period_0++;
        }
        if(Motor_0_Enable==true)
        if(period_0>=10) {
            PWM_Count_0--;
            if(PWM_Count_0<=0) PWM_Count_0=0;
            period_0=0;
if((GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_12)!=RESET)&&
(GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_11)!=RESET)) {
        if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_7)==RESET) X_TIM_1++;
        else X_TIM_1--;
            }
        }
```

2.4.7 步进电机脉冲使能以及脉冲数控制

步进电机的位置设定在细分不变的情况下由脉冲数决定，具体的脉冲使能以及脉冲数控制如下：

```
void Step_Control(int num,bool Enable)
{
    switch(num)
    {
        case 0:PWM_Enable_0=Enable; break;
        case 1:PWM_Enable_1=Enable; break;
        case 2:PWM_Enable_2=Enable; break;
        default: break;
    }
}
void Step_Count_Control(int num,int Step_num)
{
    switch(num)
    {
        case 0: PWM_Enable_0=false; PWM_Count_0=Step_num; period_0 =0; break;
        case 1: PWM_Enable_1=false; PWM_Count_1=Step_num; period_1 =0; break;
        case 2: PWM_Enable_2=false; PWM_Count_2=Step_num; period_2 =0; break;
        default: break;
    }
}
```

2.4.8 上位机系统程序逻辑分析与程序设计

主程序类有三个继成类串口通讯类、步进电机控制类、步进电机位置显示类。这三个类分别实现了上位机所需的三个功能，即为实现与单片机通讯，并且控制电机同时在立体坐标系中显示电机的所在位置。

上位机主类实现程序实现

```
class Console;
class SettingsDialog;
class scattershow;
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
private slots:
    void openSerialPort();
    void closeSerialPort();
    void about();
    void writeData(const QByteArray &data);
    void readData();
    void handleError(QSerialPort::SerialPortError error);
    void scatterDataChange(int x,int y,int z);
    void scatterChangeData();
    void motorDataSend(int Motornumber,bool Ward,bool State);
private:
    void initActionsConnections();
signals:
    void datachange(int x,int y,int z);
private:
    void showStatusMessage(const QString &message);
    Ui::MainWindow *ui;
    QLabel *status;
    Console *console;
    SettingsDialog *settings;
    QSerialPort *serial;
    Q3DScatter *scatter;
```

```
    QScatter3DSeries *series;
    QWidget *container;
    QScatterDataArray data;
    MotorControl *motorcontrol;
};
```

2.4.9 单片机传输位置信息处理显示程序设计

上位机有位置信息处理以及位置显示的功能，即为通过串口接收数据后判断为位置信息后将信息处理为位置信息后发送信号给显示处理函数，然后显示位置。

位置信息处理程序实现：

```
void MainWindow::readData()
{
    int x,y,z;
    QByteArray data = serial->readAll();
    console->putData(data);
    QString string=data;
    if(data.mid(0,1)==QString("p"))
{
x=string.mid(1,string.indexOf("x")-1).toInt();
y=string.mid(string.indexOf("x")+1,string.indexOf("y")-string.indexOf("x")-1).toInt();
z=string.mid(string.indexOf("y")+1,string.indexOf("z")-string.indexOf("y")-1).toInt();
emit(datachange((x_wave_filter(x))/2597*2000,(y_wave_filter(y))/3088*2000,(z_wave_filte
r(z))/15883*2000);
    }
}
```

位置显示函数实现：

```
void MainWindow::scatterDataChange(int x, int y, int z)
{
    series->dataProxy()->removeItems(0,1);
    data.clear();
    data <<QVector3D(x, y, z);
    series->dataProxy()->addItems(data);
}
```

2.4.10 通讯协议简单设计

由于上位机与单片机的通讯协议同需要我们自己尾包从而实现数据的正常接收，不丢包。

通讯协议程序实现：

```
void MainWindow::writeData(const QByteArray &data)
{
    static QByteArray datanew;
    if(data=="\n")
    {
        serial->write(datanew.append("#"));
        datanew.clear();
    }
    if(data!="\n")
    datanew.append(data);
}
```

2.4.11 信息滤波程序设计

由于上位机与单片机的通讯时易出现信息出错现象，需要我们进行有效数据筛选。

信息滤波程序实现：

```
double x_wave_filter(int filter_num)
{
    static int wave[4]={0,0,0,0};
    int m; double count=0; int f;
    for(m=0;m<=2;m++) wave[m]=wave[m+1];
    wave[3]=filter_num;
    int a=wave[0],b=wave[1],c=wave[2],d=wave[3];
    if(a<b){f=a; a=b; b=f;}
    if(a<c){f=a; a=c; c=b; b=f;}
    if(a<d){f=a; a=d; d=c; c=b; b=f;}
    if(b<c){f=b;b=c;c=f;}
    if(b<d){f=b;b=d;d=c;c=f;}
    if(c<d){f=c;c=d;d=f;}
    count=a+b;
    count/=2;
```

```
    return count;
}
```

2.4.12 3D 立体模型显示程序设计

本程序采用 3D 绘图进行立体建模，其程序过长在此便不再赘述，定位移动根据实际与虚拟的模型的比例关系进行放缩显示。

```
cylinderTransform_2->setTranslation(QVector3D(X, -0.4f, 0.65f+3.5f));
cylinderTransform2_2->setTranslation(QVector3D(X, 0.4f, 0.65f+3.5f));
cylinderTransform3_2->setTranslation(QVector3D(X, 0.0f, 0.65f+3.5f));
cylinderTransform_3->setTranslation(QVector3D(X+0.6, 0.0f, -0.3+0.55+Z+0.5));
cylinderTransform2_3->setTranslation(QVector3D(X+0.6, 0.0f, 0.5f+0.55+Z));
cylinderTransform3_3->setTranslation(QVector3D(X+0.6, 0.0f, 0.55+Z+0.5+0.3));
cuboidTransform_Move->setTranslation(QVector3D(X, 0.0f, 0.0f));
cuboidTransform_2->setTranslation(QVector3D(X, 0.0f, 0.5f+7+0.55));
cuboidTransform_Move_2->setTranslation(QVector3D(X, 0.0f ,0.5f+0.55+Z));
cuboidTransform_Move_2_2->setTranslation(QVector3D(X-0.6-0.1, 0.0f ,0.5f+4));
cuboidTransform_3->setTranslation(QVector3D(X+0.2+0.5+0.1, -2.5f-0.15,0.5f+0.55+Z));
cuboidTransform_Move_3->setTranslation(QVector3D(X+0.2+0.5+0.1,
2.5f+0.45 ,0.5f+0.55+Z));
cuboidTransform_Move_2_3->setTranslation(QVector3D(X+0.2+0.05,
0.0f+0.3 ,0.5f+0.55+Z));
cuboidTransform_Move_3_3->setTranslation(QVector3D(X+0.2+0.5,
2.5f-Y-0.4 ,0.5f+0.55+Z));
```

3 软件及硬件调试

3.1 硬件调试

硬件部分主要有单片机，驱动设备以及电机。

在此以前从未接触过大型步进电机的控制，以及驱动器的使用，第一次做 XYZ 平台定位设计，所以硬件调试过程较长。

首先是电源模块的焊接与检测。完成检测后，将电源模块的 24V DC 电源接入驱动的 24V 电源连接端。检测是否焊接正常。

将软件部分的基础的电机驱动进行搭建，将单片机与驱动相连接，创建固定的

固定的轨迹进行电机的测试与整体的协调测试，并将电机的脉冲频率设定到最好的状态以为下一部分的软件测试搭建基础。

3.2 软件调试

软件调试分为上位机软件调试与单片机软件调试。

单片机软件调试分为驱动调试和系统整体逻辑调试，驱动调试分为 PWM 驱动模块选取，串口选取，使能端 IO 选取，方向控制端使能选取。整体逻辑调试分为定位程序调试，控制程序调试，轨迹绘制程序调试，以及上位机信息处理程序调试。

上位机软件调试分为显示程序调试，控制程序调试以及单片机通信信息处理调试。

上位机软件较为简单其涉及到的逻辑结构较为清晰，而单片机软件设计涉及到了硬件控制以及协调所以在一定程度上较难，其中最难的为图形绘制逻辑的结构设计，需要严谨的思路和逻辑结构。常常长时间卡在该点无法通过。

4 感想与总结

通过本次课程设计我学习到了步进电机控制的插补算法，并且将单片机知识在过程中得以实践。并且编写步进电机控制上位机，Qt 界面开发能力得到了进一步的巩固，并且学习到了工业控制上一些上位机控制的基本的基本思考方式。

步进电机在定位系统中有一定地位，当下流行的 3D 打印机，也采用步进电机控制，在过程中了解到了步进电机的控制方法以及步进电机驱动器的作用原理，在调试过程中，老师教授了我在书本中从曾学到的知识。

在项目的完成过程中，深知实践动手能力和团队合作的重要性，项目的完成分为硬件设计、软件设计两个部分，由队友和我二人完成，在过程中需要二人相互协调，相互配合，使得项目可以以高效高质完成。

参考文献：

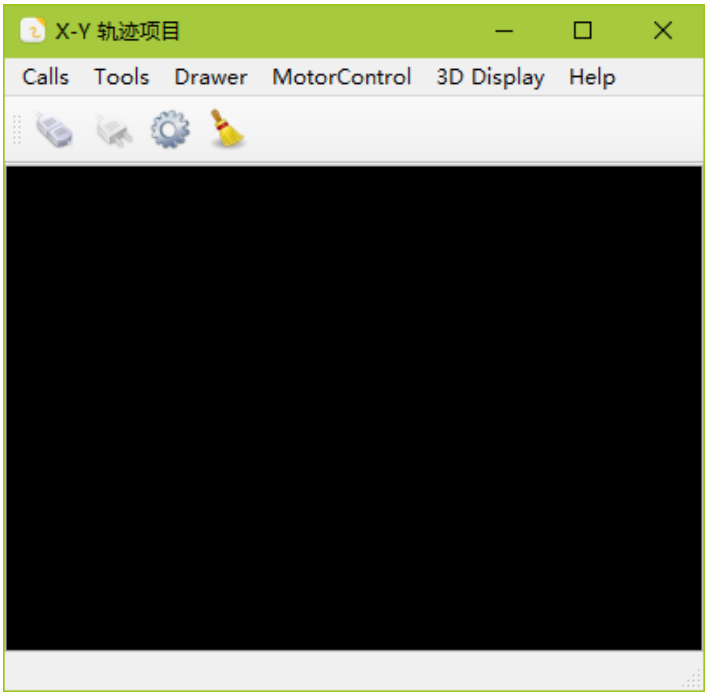
[1] 阮毅. 电力拖动自动控制系统——运动控制系统. 第 4 版，北京：机械工业出版社，2009.8

[2] 张淑清. 嵌入式单片机 STM32 设计及应用技术. 第 1 版， 北京：国防工业出版社，2016.7

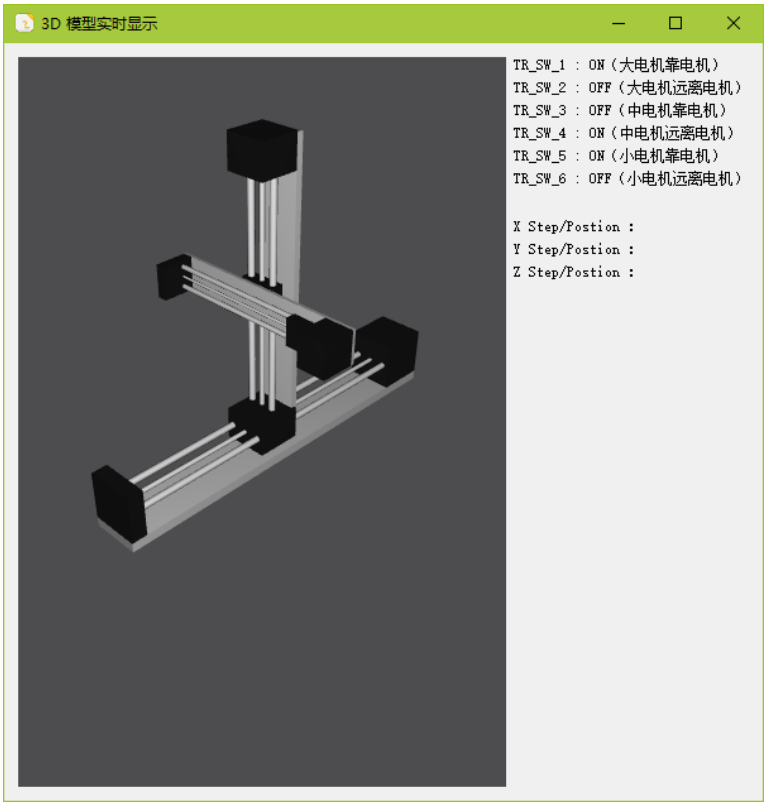
[3] 夏德铃 翁贻方. 自动控制理论. 第 4 版， 北京：机械工业出版社，2012.11

[4] 霍亚飞 Qt Creator 快速入门 第 1 版， 北京：北京航天航空大学出版社，2012.5

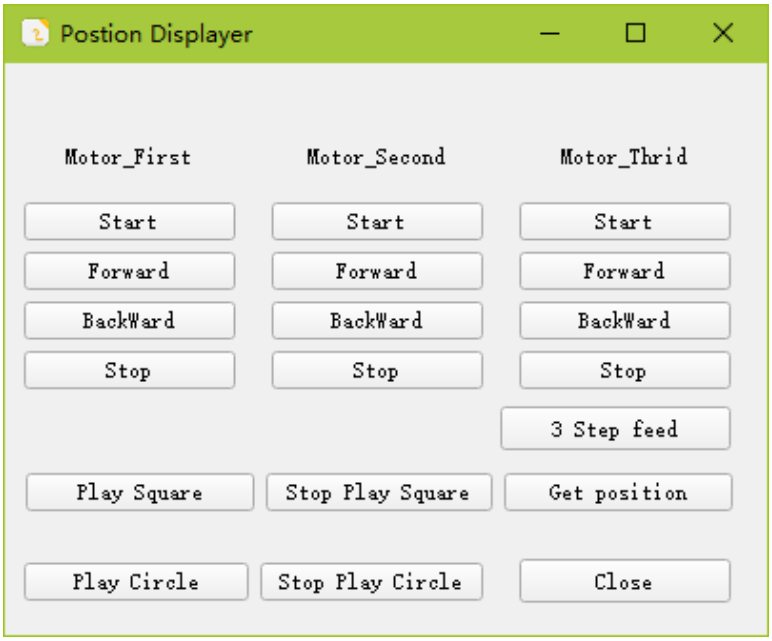
附录：



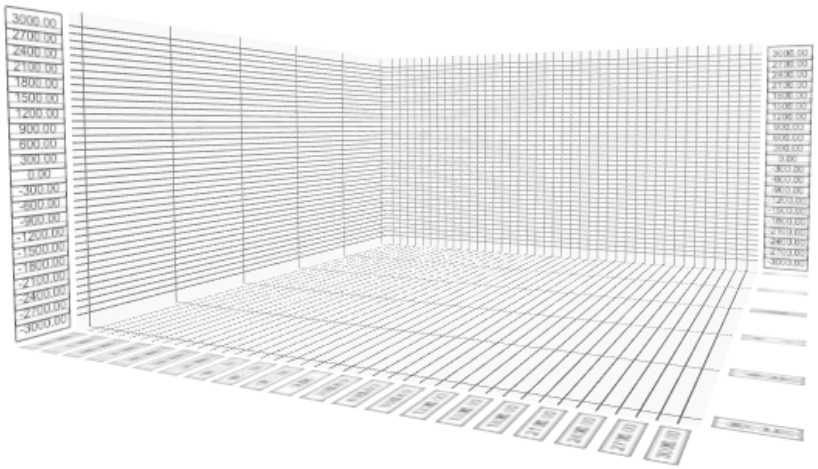
命令模式控制界面



3D 模型显示界面



按钮控制平台移动界面



具体位置显示界面

燕山大学自动化系课程设计评审意见表

平时成绩(10%)：_____

指导教师：_____

年　　月　　日

说明书成绩(20%)：_____

指导教师：_____

年　　月　　日

图面成绩(30%)：_____

指导教师：_____

年　　月　　日

答辩小组评语：

答辩成绩(40%)：_____

评阅人：_____

年　　月　　日

课程设计总成绩：

答辩小组成员签字：

年　　月　　日