**

## HUNAN UNIVERSITY

毕 业 论 文

|  |  |
| --- | --- |
| **论文题目** | 基于PCI的单自由度振动 |
|  | 平台上位机软件管理系统设计 |
| 学生姓名 | 曾庆宏 |
| 学生学号 | 201407030414 |
| 专业班级 | 自动1404 |
| 学院名称 | 电气与信息工程学院 |
| 指导老师 | 江未来 |
| 学院院长 | 王耀南 |

2018 年 5月 20日

**湖 南 大 学**

**毕业论文原创性声明**

本人郑重声明：所呈交的论文是本人在老师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

学生签名： 日期：2018 年 5月 20日

**毕业论文版权使用授权书**

本毕业论文作者完全了解学校有关保留、使用论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权湖南大学可以将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本论文。

本论文属于

1、保密 ，在\_\_\_\_\_\_年解密后适用本授权书。

2、不保密√。

（请在以上相应方框内打“√”）

学 生签名： 日期：2018 年 5月 20日

指导教师签名： 日期：2018 年 5月 20日

**摘要**

VR（虚拟现实）技术，是一种通过搭建一个模拟真实环境的人机交互系统，从而高真实度再现真实环境的技术。本课题计划并实现一个能够真实高精度再现实际环境的模拟振动平台，通过模拟真实环境中的海浪、地震、滑雪、飞行等，一方面可以为用户提供身临其境的浸入感，另一方面，可以促进VR（虚拟现实）技术在各种结构测试中的应用，例如道路模拟机，房屋结构抗震设计，海况模拟平台，飞行模拟器等等。本文基于单自由度模拟振动平台系统，主要是开发一套用于与系统下位机（控制器）交互、调试的上位机软件系统。

本文首先根据控制精度、控制需求分析了整个系统的结构，之后简单介绍模拟振动平台系统的机械结构、算法设计、硬件电路、以及下位机的软件系统。对于上位机软件系统，本文采用跨平台开发工具Qt进行开发，使其不仅仅可以在Windows系统下运作，还能在Linux、Mac等系统下工作，首先介绍了其整体设计框架以及用户界面，然后对于上位机软件系统的各个模块：系统管理模块、数据处理模块、通讯模块、数据显示与保存模块，分析了其设计与实现。其中，重点介绍了动态绘图中的生产者-消费者矛盾以及双缓冲解决办法、以及基于TCP/IP应用层的通信协议制定。最后我们对振动平台上位机软件管理系统各个功能进行了调试和详细测试

**关键词：**模拟振动平台、跨平台、用户界面设计、双缓冲机制、生产者-消费者矛盾、通信协议、C++、Qt

**ABSTRACT**

In the field of Virtual Reality(VR) , people can experience virtual worlds through a man-machine interactive system .This topic plan to design a vibration platform, a high precision actual simulation can be reproduced on this platform .By simulating the real environment of sea waves , earthquakes, ski, flight, etc. Through this technology ,on the one hand ,It can provide users with immersion .On the other hand ,It can promote the application which based Virtual Reality(VR) of all kinds of structure test ,such as road simulator ,anti-seismic design of building structure ,sea state simulation platform , the flight simulator , etc. Based on the single-degree-of-freedom simulation vibration platform system ,this paper mainly develops a set of upper computer software system for the interaction and debugging of the system’s lower computer(the controller).

At first, this paper analysis the whole system structure according to the requirements of control accuracy, after this ,the paper briefly introduces the mechanical structure of the simulated vibration platform system, algorithm design, hardware circuit and the software of lower computer system .For upper software system, first of all ,we introduces the design framework and user interface, then for upper computer system modules: system management module, uppercom-lowercom communication module, data management module, data displaying and saving module, analyses the design and implementation of that .Among them, the author focuses on the producer-consumer contradiction in dynamic drawing and the double buffering solution, and the communication protocol based on TCP/IP application layer .Finally, the software system modules are tested in detail.

**Keywords**: Simulated vibration platform, cross platform ,UI design, producer-consumer conflict , double buffering, communicating protocol , C++, Qt

**目录**

[摘要 1](#_Toc9949)

[ABSTRACT 2](#_Toc5529)

[第一章 绪论 3](#_Toc32573)

[1.1 课题背景及研究意义 3](#_Toc19635)

[1.2 国内外研究现状 3](#_Toc349)

[1.3 本设计的主要内容和章节安排 4](#_Toc6060)

[第二章 模拟振动平台系统**总体介绍** 5](#_Toc21899)

[2.1 总体框架 5](#_Toc120)

[2.2 机械设计 6](#_Toc23877)

[2.3 硬件系统 7](#_Toc19599)

[2.4 算法研究 7](#_Toc21068)

[2.5 软件系统 7](#_Toc21068)

[第三章 上位机软件系统设计 8](#_Toc22579)

[3.1 软件系统需求分析 14](#_Toc2904)

[3.2 上位机软件系统整体架构设计 15](#_Toc17612)

[3.2.1 基本功能框图 15](#_Toc31607)

[3.2.2 总体架构设计 16](#_Toc23378)

[3.3 软件开发平台简介 16](#_Toc19165)

[3.4 通信模块总体设计 17](#_Toc19165)

[3.4.1 通信框架概述 17](#_Toc31607)

[3.4.2 通信过程 18](#_Toc23378)

[3.4.2 应用层帧结构 19](#_Toc23378)

[第四章 系统实现 21](#_Toc18070)

[4.1 系统管理模块 21](#_Toc19288)

[4.2 系统通信模块 24](#_Toc14241)

[4.2.1 以太网TCP/IP通信基本参数的初始化 24](#_Toc19847)

[4.2.2上位机通信设计 25](#_Toc23621)

[4.3 数据处理模块 26](#_Toc23726)

[4.3.1 默认配置信息的保存 26](#_Toc15288)

[4.3.2 试验参数和控制 27](#_Toc2875)

[4.3.3 试验数据软件的存储和试验报告 28](#_Toc28285)

[4.3 数据显示模块 29](#_Toc23726)

[4.4.1 三参量数据图表分析 29](#_Toc15288)

[4.4.2 生产者-消费者矛盾 30](#_Toc15288)

[第五章 系统调试及测试 33](#_Toc1366)

[第六章 总结与展望 30](#_Toc12100)

[6.1总结 30](#_Toc3322)

[6.2 展望 30](#_Toc9543)

[参考文献 31](#_Toc15426)

[致谢 33](#_Toc15130)

1. **绪论**

**1.1 课题背景及研究意义**

近些年来，VR（虚拟现实）技术不断完善高速发展，其不仅可以用于用户的日常娱乐，而且可以用于在工业生产，基础设施建设中模拟再现一些真实极端环境，从而对各种装置的工程性能进行检验和评判[2]（基于电液振动台的模型参考自适应控制的研究） 。模拟振动平台通过高精度控制算法，可以真实再现地震、道路状况、海浪、以及一些飞行真实场景，有助于实现房屋结构抗震理论研究和实际设计，亦可以实现机动车零部件和整车可靠性实验，对轮船和海洋工程装备制造也有莫大帮助。模拟振动平台高精度实时再现实际环境技术，目前国内外已有公司开始相关研究，但是成品大多来自国外进口。若能将此技术国产化，不仅可以用于我国虚拟现实中实现更深的浸入感，对于国家的国计民生、工业发展也有着重要的意义，有助于推动我国产业的升级

**1.2 国内外研究现状**

**国内外现状:**

模拟振动台是舰船、飞机、海工装备在研发和使用过程中最重要的检验和模拟使用设备，也是汽车底盘、车身、车载电子设备的实验室研究中最重要的一环，同时，在土木工程领域，是建筑抗震性能研究中最重要的设备，但是关键技术一直被美国的MTS公司、英国的Servotest公司垄断，国家每年都要花费十几亿美元外汇从国外购买。由于上述设备集成了机械、电子、液压、传感器、计算机软硬件等多门类的关键技术，使用和维护都比较复杂，从国外购买的设备的使用和售后都非常不便，国防和相关关键部门的设备又不能请美国等西方国家的公司去现场维护，这一切，都极大制约了我国在相关领域的研发进度[8]。

**发展趋势:**

近年来，国内投入到电液伺服模拟平台的专业厂家，研究机构也在逐步增多，中科院力学所、哈尔滨工业大学、华中科技大学、哈尔滨工程大学和浙江大学液压所，都做过类似的科研,发表过相关论文，不过目前市场上只有小型的电机驱动的模拟振动平台出售，一般只用于游戏、科普等领域。

通过继续的深入研究，提高振动台控制精度以及增加自由度，可以模拟再现实际场景，这项技术不仅仅可以用于虚拟现实，也可以对国计民生、工业发展有着积极作用，比如应用至地震模拟、道路模拟和海浪模拟中，而目前振动台多由国外进口，关键技术尚未国产化，价格昂贵，一旦研究实现其技术国产化，有较好的市场前景，同时对于社会的经济效益非常大。

**1.3 本设计的主要内容和章节安排**

本课题研究设计一套便于与模拟振动平台下位机通讯，调试，便于用户操作的上位机软件系统。本文通过对振动平台的工作需求分析，从而得到上位机软件系统的总体框架，对动态绘图和应用层通信协议进行设计，完成上位机界面和数据库开发。全文章节结构如下：

第一章介绍了振动平台的研究现状背景，目前为止，国内国外对于振动平台的研究进展，以及本文的章节安排。

第二章分析了系统的需求，概括说明模拟振动平台的整体结构。然后，介绍了振动平台的机械结构设计、算法研究、硬件电路、以及软件系统。

第三章是上位机软件系统设计部分，通过对整个系统的控制需求进行分析，根据不同功能，系统由以下四个部分组成——系统管理模块、数据处理模块、通讯模块、数据显示与保存模块分析界面。然后通过分析上位机动态绘图中的生产者-消费者矛盾，得到本系统的数据结构，并应用双缓冲机制解决问题。最后是应用层的通信协议，重点介绍了基于TCP/IP的通信协议以及上下位机的通信过程。

第四章首先介绍了数据库访问技术，并把数据库的添加、查找、更新、删除操作封装成函数。然后设置串口通信参数，介绍了通信实现过程。最后是本系统的用户界面实现过程，重点论述了C#绘制折线图的方法和数据库操作。

第五章是系统的测试与调试，通过对各个界面进行调试，分析康复训练管理系统的各个模块功能。

第六章总结全文，概括本文创新点，并展望本课题的未来。

1. **模拟振动平台系统总体介绍**

**2.1 总体框架**

本项目研究的主要内容为高精度大型模拟振动平台的随机波形控制，而研究内容主要包括四个部分：第一部分从机械系统的角度出发，如何最大限度减小振动台的非线性，研究不同建模对振动台进行控制的影响；第二部分从算法角度出发，在机械角度的基础上，通过算法实现高精度实时复现；第三部分从硬件电路的角度出发，通过全数字工作方式的控制器，实现多通道模拟振动台的随机波实时波形复现，第四部分从振动平台控制上位机软件出发，通过多功能的上位机控制系统，可以很有效的实现操作指令的下达、数据图像的实时绘制，试验状态的实时监控，从而达到所有数据可视化的效果。【地震模拟振动台的时域复现控制策略研究】

模拟振动平台主要由振动台体、振动台台面机械系统、位移，加速度，速度传感器、电源、和控制系统组成。模拟振动平台利用传感器采集三参量控制信号，并将其反馈回控制系统，通过不同的控制算法，使得振动平台可以高精度跟踪随机输入波形，从而完美复现波形。系统框图如下所示：

参考

信号

控制

系统

驱动

信号

电动

激振系统

台体及台面机械系统

试件夹具

传感器

控制

信号

上位机软件

调节

信号

整个系统控制框图如下所示:

43 ye

+

输出

上位机给定信号

前级调理控制电路

电动激振器

振动台台体

控制量反馈

振动台主要技术指标和预期目标如下：

（1）振动台电动激振器  
⚫ 最大激振力：±20kN  
⚫ 最大行程：±100mm  
⚫ 最大速度： 500mm/s  
⚫ 最高频响： 70Hz  
（2）台面机械系统  
⚫ 最大正弦推力：50kN  
⚫ 最大加速度:2g  
⚫ 最大速度:500mm/s  
⚫ 最大位移:200mm  
⚫ 最大负载： 3000kg  
⚫ 台面尺寸： 800\*1200mm  
⚫ 最大力矩:3kN·m  
⚫ 频率范围:0--50Hz

（3）整体系统

⚫ 加速度最大相对误差≤ 10%  
⚫ 位移段波形失真度≤ 5%，加速度段波形失真度≤25%  
⚫ 加速度横向比≤ 25%  
⚫ 时域复现精度：时域相关系数≥ 90%,最大峰值误差≤ 10%

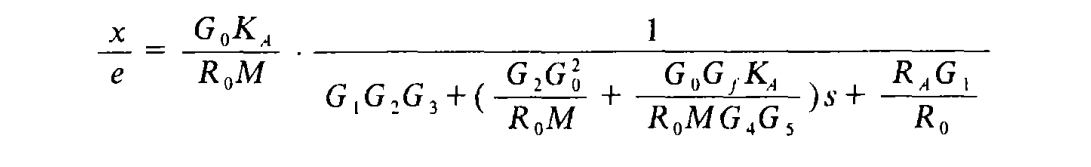
模拟振动平台运动分析如下：

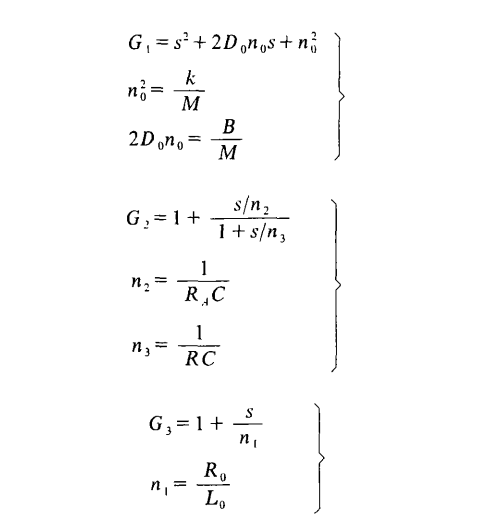
整个振动台最高频率响应为70Hz，在这种情况下，振动平台控制系统高频噪声可以忽略。此时整个系统的运动方程为：

此式中：为振动平台系统质量； 为振动台电动激振器的转换系数； 为整个系统阻尼系数比。为电动激振器产生的动力，为振动平台的水平位移。

对电动伺服激振器进行建模，可以得到其表达式为：

由此，可以得出，振动平台输出位移和其输入电压之间的关系为：



式中：

由此上分析，可以知道,通过三参量反馈、迭代控制等等算法，可以大大提高系统的动态性能，达到更优的控制指标。

**2.2 机械设计**

机械与动力部分的设计可以参考市场上成熟额进口产品的设计原则，需要解决的是高频电动伺服动作器的国产化，主要的技术难点存在以下三个：

* 提高电动伺服动作器的最高响应频率；
* 优化机械结构，减小系统摩擦带来的非线性影响；
* 实现高频响应电动伺服动作器的国产化

本部分的技术关键包括：

1. 振动平台的运动学、动力学建模：

推导振动台运动学、动力学表达式，具体研究振动台位姿随时间的变化规律以及振动台所受力与其运动规律之间的关系

1. 动力结构设计：

利用振动台的动力学模型，用最大功率点相等的方法求取动力机构的参数，选出液压控制元件和液压驱动元件，并检验设计方案是否满足工况要求

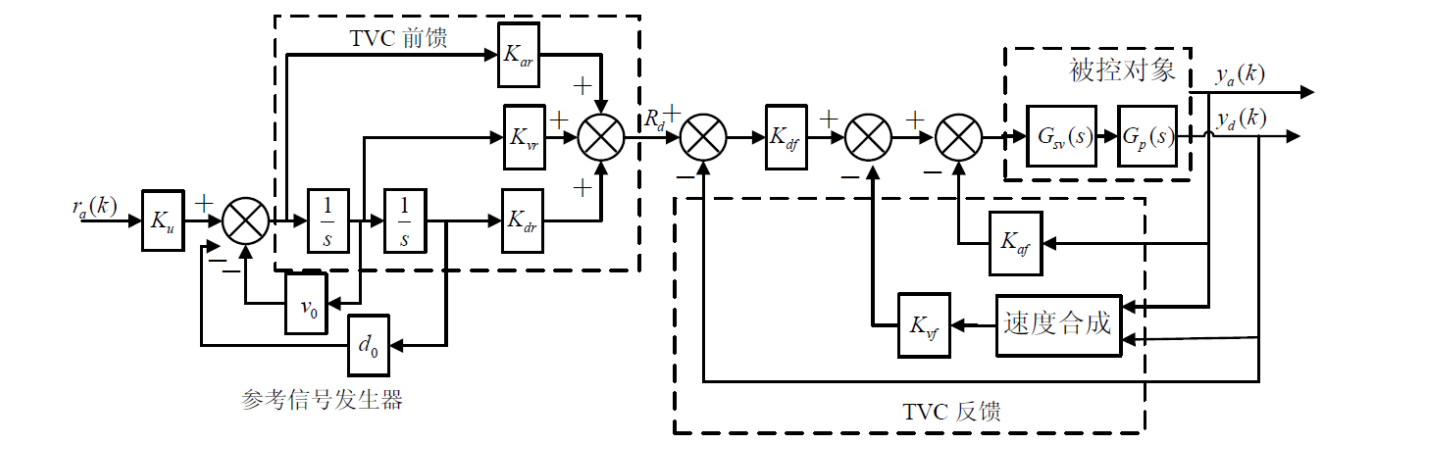
**推导振动台运动学、动力学表达式，运用MATLAB和ADAMS软件分别建立振动台的动力学的数学模型，对模型进行仿真验证。运动学分析是从几何角度去描述和研究物体位置岁时间变化规律，本项目中具体是要研究振动台的位姿随时间变化的规律。振动台是一个复杂的并联机构，对于并联机构在空间的位姿，一般引入有限个变量进行描述，在运动学中对于物体位姿的描述有多种方法，比如：欧拉四元数法、方向余弦坐标、欧拉角坐标、卡尔丹坐标等。本项目拟采用欧拉角坐标方法对振动台的位姿进行描述。**

**2.3 算法研究**

由于模拟振动平台系统存在强耦合，高滞后，采用传统的控制算法效果并不能很好地满足跟踪精度，本项目采用下列控制算法来改善控制效果：

1. 三参量控制算法：

**在设计反馈控制器的过程中，为了补偿振动台在单纯的位移控制情况下工作频宽较窄阻尼较小的缺陷，研究采用三参量控制算法来更好的实现不同频段的加速度、速度、位移控制。而三参量控制中的位移控制用于低频段控制、速度控制用于中频段控制，加速度控制用于高频段控制，从而能够实现增大系统阻尼，拓宽系统工作频宽的目的。其控制框图如下：**



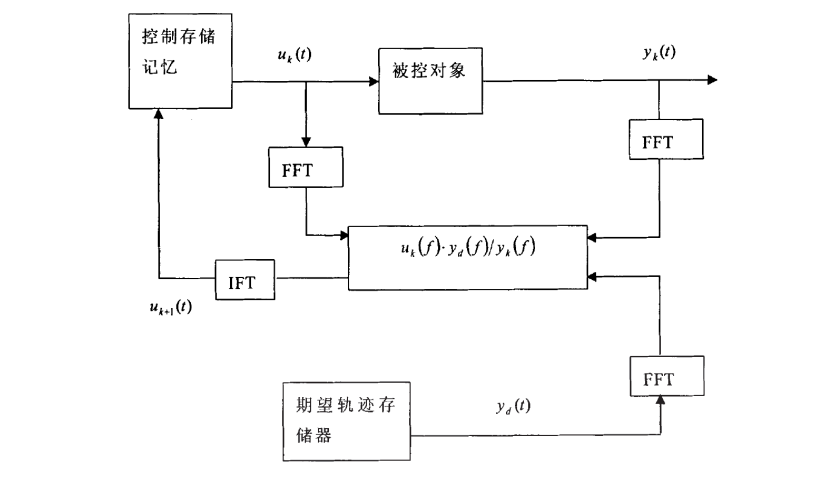
**本项目根据模拟振动平台控制系统的结构和工作性能，对振动台系统进行一定的线性化处理，搭建系统的数学模型。在位移控制的基础上，增加加速度和速度环节，设计三参量反馈控制器和三参量前馈控制器。三参量前馈控制的目的是对消系统位置闭环传递函数中距离虚轴较近的极点从而拓展系统的频宽；三参量反馈控制将在基于位置闭环的基础上，通过加速度反馈和速度反馈来提高系统的阻尼比和固有频率，从而在保证系统稳定性的基础上拓展振动平台系统的工作频宽。**

1. 迭代学习控制算法

**在振动台的控制过程中，由于本系统存在着很大的非线性因素，仅采用伺服控制方法虽然可以控制平台运动，但是系统的输出响应信号与输入的期望信号之间仍然存在较大偏差。因此我们拟在伺服控制系统的外环再增加基于迭代学习控制算法的外闭环构成双闭环控制系统，对驱动信号进行迭代补偿，使系统的响应能够高精度的复现期望的波形信号。**

**迭代学习控制具有严格的数学描述，在对具有较高位置的精度要求、较强的非线性耦合和精确轨迹跟踪控制要求的动力学系统研究中有广泛应用。迭代学习算法在应用于期望轨迹跟踪时，利用前一次的初始信息和不断累积的控制经验得到跟踪控制误差，通过重复学习不断修正下一次的控制量，得到一个有效的输入控制信号，使得被控系统在有限时间和区间上能过比较精确的跟踪期望轨迹。**

**迭代学习控制在结构上包括开环学习和闭环学习，闭环学习在收敛条件、收敛速度和抗干扰能力等方面都优于开环，但是运算量要大于开环，而优于伺服控制系统已经能使系统的稳定性得到保障，因此本项目拟采用开环迭代学习控制律，通过对输入控制信号不断迭代修正以使系统的输出响应跟踪期望的输出波形来模拟预期的振动环境。其原理如图所示：**

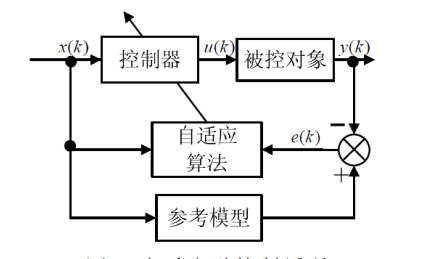


1. 自适应逆控制算法

**在迭代学习控制算法中，需要对系统的频响函数进行估计，并采用逆频响函数对系统进行输入信号的更新，因此振动台系统的频响函数的测量和辨识的精度对于迭代收敛的速度和波形复现的精度会有很大的影响，使得该控制方法可能存在以下问题：(1） 由于系统的频响函数是离线辨识的，在进行试验时认为系统的动态特性是不变的，但是实际中系统可能存在时变；（2）迭代学习控制方法需要进行大量的 FFT和 IFFT 变换来辨识系统的频响函数并产生修正的输入信号，而时域波形复现是瞬态变化的是瞬态变化的，采用 H1 估计法无法实现实时估计系统的频响函数。**

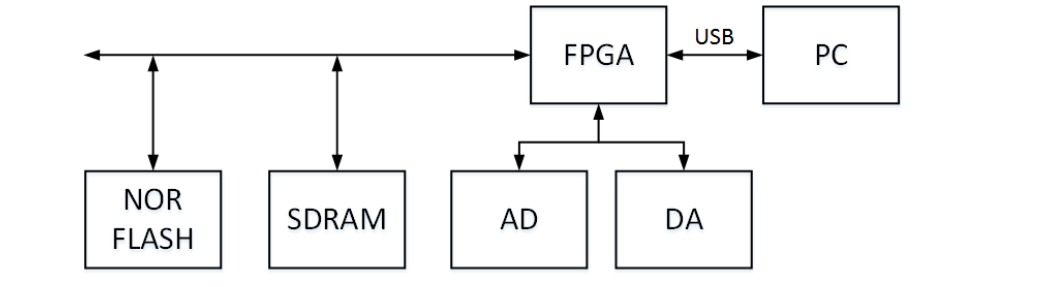
**对此，本项目拟采用自适应逆控制方法，将期望的输出波形信号通过一个补偿滤波器产生被控系统的输入信号，使系统的输出波形收敛于参考波形，从而使振动台输出期望波形。**

**常用的自适应滤波算法包括最小均方误差（LMS）算法和递推最小二乘（RLS）算法两类。最小均方误差（LMS）算法计算量小，易于实现，但存在着收敛精度和收敛速度之间的矛盾，通常采用变步长的策略来解决这一矛盾。递推最小二乘（RLS）算法，具有快速收敛性，但是在计算复杂度和稳定性方面表现较差。本项目将结合两种算法的优缺点具体分析，对两种算法进行算法实现和比较。自适应逆控制原理图如下：**

****

**2.4 硬件系统**

全数字控制器的硬件电路和组成方案如图所示，下面介绍主要模块和工作原理。



1. FPGA

采用高性能的 FPGA 作为整个系统的控制器， FPGA 作为可编程逻辑控制芯片，可以实现系统的 AD、 DA 和 USB、以太网转换接口的需要，同时可以高速进行大量并行计算，为数据处理和信号传输提供硬件支持。

1. 信号的输入与输出采集卡

信号的输入与输出通道包括模拟量通道和数字量通道，使用 A/D 输入通道接受指令参考电压信号以及传感器输出的振动台位移、加速度信号，接收到的信号经 FPGA 的控制算法处理后，由 D/A 模块转换为控制电流/电压信号输出给伺服阀。

1. 硬件部分主要工作原理

如图所示，控制系统中的 FPGA 实现实时闭环控制算法、模拟输入/输出通道的控制、数字输入/输出的控制、并实现了与 USB 接口和以太网口的通讯。系统中具有 NOR FLASH 芯片，从而全数字控制器不但可以作为分布式控制器，由 PC 下载 FPGA 代码，还可以作为嵌入式控制器使用，从 NOR FLASH 中自动加载代码至 FPGA 中。

* 1. **软件系统**
  2. 上位机

全数字控制器的上位机软件提供了友好的人机交互界面，能实现参数设置，实验过程监控，实验数据保存等。 上位机软件按照功能可划分为 4 个模块，即：系统管理模块、数据处理模块、通讯模块、数据显示与保存模块。

系统管理模块功能主要包括:软件系统初始化、控制参数设置、通道参数设置、系统文件管理；

数据处理模块功能主要包括：对PC的消息，参数地址接口进行定义，并通过以太网口与下位机进行消息通讯，实验数据信号的读取和分配等功能；

通讯模块功能主要包括：采用USB和TCP/IP通讯协议和从机互相收发数据，进行通讯。

数据显示与保存模块功能主要包括：实现实验数据信号的实时显示和保存，使用户能够方便地对实验过程进行监控。

* 1. 下位机

下位机实现信号的采集、传输、处理与输出，并利用控制算法实现控制功能，是全数字控制器的核心部分。下位机分为三个层次，即系统管理层、信号层、算法层。

系统管理层是整个全数字控制器的框架，主要调配信号层和算法层完成相应的运算。

信号层的主要功能是在系统管理层的调配下完成信号采集，为算法运算提供数据，同时将算法输出的数据转化为电信号输出，同时将反馈信号和指令信号通过系统管理层发送至 PC。

算法层位于整个程序的最上层，它在信号层提供的数据接口基础上，在系统管理层的调配下完成算法运算，实现控制算法。

1. **上位机软件系统设计**

**3.1 软件系统需求分析**

振动平台需要通过上位机软件系统，对整个控制系统进行控制管理，参数调节，和达到控制状态数据可视化。通过对第二章的振动平台系统各部分的分析，我们得到振动台上位机软件系统需要实现以下功能：

系统与用户的交互，实现试验参数设置、操作命令发送、控制过程监控、数据显示与存储、实验结果产生等功能。

振动平台软件应具有控制功能和辅助功能，可以根据不同的用户需求配置响应的控制器和软件功能。具体功能如下：

1. **控制功能**

* **随机系列**
* **随机试验**
* **随机地震波冲击试验**
* **正弦加随机试验**
* **随机加随机试验**
* **…**
* **冲击系列**
* **典型冲击试验**
* **冲击响应谱试验**
* **瞬态冲击试验**
* **…**
* **正弦系列**
* **正弦试验**
* **谐振搜索与驻留试验**
* **…**

1. **辅助功能**

* 系统校准
* 离线浏览
* 波形编辑器
* 信号报警
* 波形显示
* …

**3.2 上位机软件系统整体架构设计**

**3.2.1 基本功能框图**

通过上一节对系统需求的分析，可以知道振动平台的上位机软件管理系统需要实现以下四个基本功能：

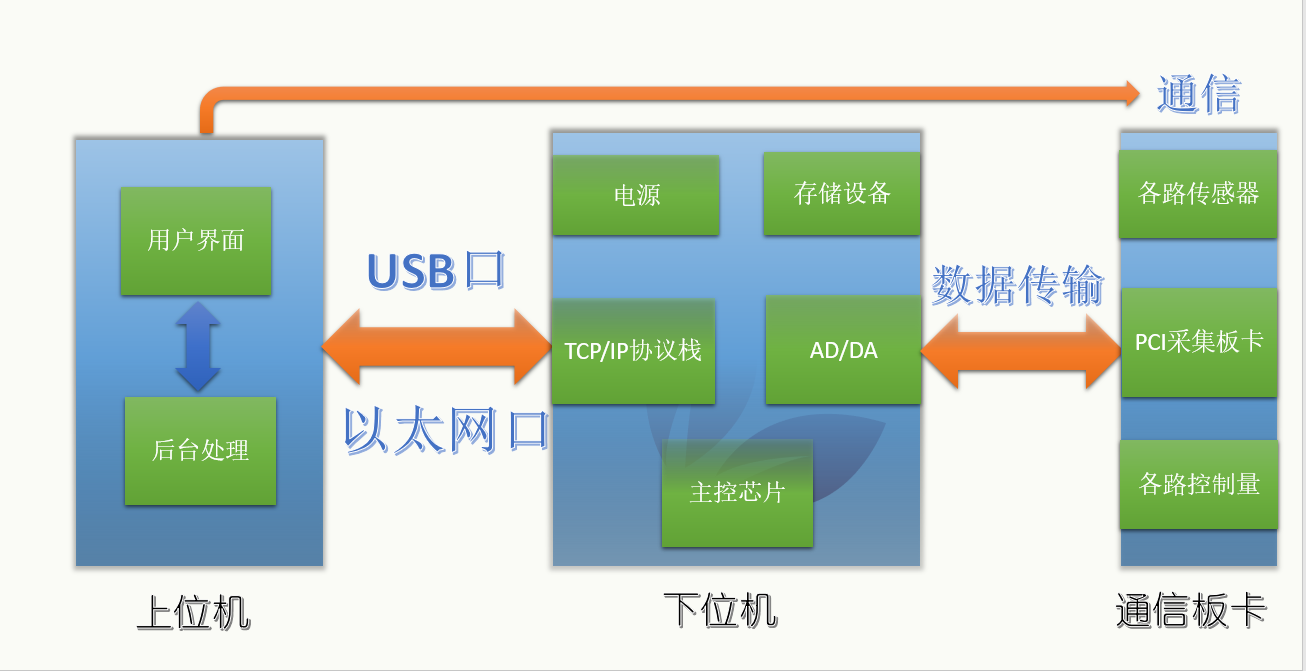
* 读取系统的基本初始化文件，设置本次试验参数等等。
* 本系统通过以太网或USB口，以通信协议的方式将指令下达到对应的控制器中，下位机接收到指令后，立马产生控制动作
* 下位机将振动平台的状态实时上传到上位机，并保存至数据库
* 将振动平台的实时状态以动态曲线绘制成时间表图，根据波形进行试验状态分析

对于以上需要实现的基本功能，需要设计以下四个模块：模拟振动平台基本信息界面、控制参数信息界面、数据图表分析界面以及系统设置界面。

整体框图如下所示：

**3.2.2 总体架构设计**

宏观上来看，振动平台软件系统的主体结构分为下位机设计和上位机设计两个部分，框架如图所示:

****

上下位机用以太网和USB技术实现连接，为了保证数据的无差错传输，我们采用基于TCP/IP的以太网通信协议进行数据通讯，严格定义帧结构。上位机是PC端，在**Qt**中采用**C++**开发，具有良好的跨平台特性：

**3.3 软件开发平台简介**

常见的用于上位机界面开发的语言有C++、C#和VB。采用C++语言开发，程序运行高效，长度也相对简短，可移植性高，。基于C++语言的设计在Qt5开发平台上，利用Qt5固有的开发库，可以方便的进行桌面应用程序的开发设计。除此之外，C++语言还有以下两大优点：C++的基本语法和C语言度相似，对于新手来说上手较快；C++的多态和封装，使得整个系统具有很高的可移植性。

Qt是的桌面应用IDE集成工具，具有很良好的跨平台特。Qt提供了自动补全，语法高亮，前端、后端设计分离等等方便地特性，而且Qt有着丰富的API函数，拥有250个以上已经封装好的C++的类，还提供了基于模板的类。Qt在网上也有着大量的资料和开发文档，上手容易。Qt不仅仅支持2D图形绘制，也能渲染3D图形，为以后对振动台进行3D建模、实时监测也提供了便利。

**3.4通信模块总体设计**

**3.4.1 通信框架概述**

通信技术是模拟振动平台上位机软件系统的关键技术，即上位机如何发送请求到下位机，以及下位机如何反馈数据到上位机。本节将概述上位机和控制器的通信过程，并且分析其原理及其实现，已达到上位机对整个系统实时监控的效果。

模拟振动平台上位机软件系统需要的监控多个控制器状态，同时不同的控制器均需要将数据上传至上位机进行实时显示，这就涉及到多机通讯问题。在工程设计中，利用基于以太网的组网系统已经成为控制系统的标准配置，本文也将采用基于以太网的TCP/IP协议进行通讯（引用 基于工业以太网实现上位机对远程PLC的监控）。

TCP/IP协议可以实现客户端-服务器通讯模式，其中上位机作为服务器，控制机作为客户端，对服务器发起连接，从而进行端对端的通信。服务器只需要开放一个端口，客户端就能通过服务器IP地址的端口号，建立起与服务器之间的连接，并不受限于一个主节点多个子节点的通信模式。而且连入以太网的各个客户端与服务器设备各自独立运行，数据收发互不影响，而且通信质量稳定，不容易受到外界干扰。

故本文采用TCP/IP以太网组网方式，结构图如下所示：

以太网总线

TCP服务器

TCP服务器

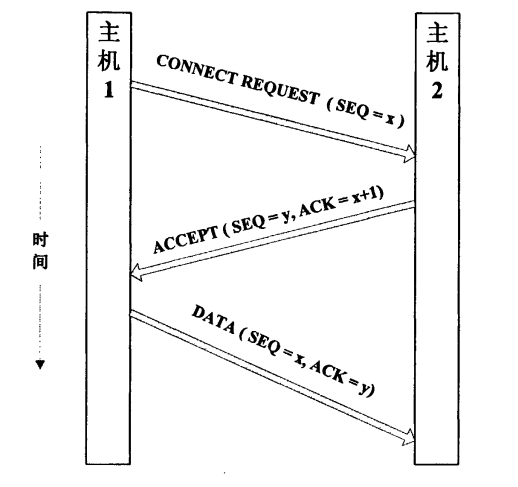
TCP客户端

TCP客户端

TCP服务器

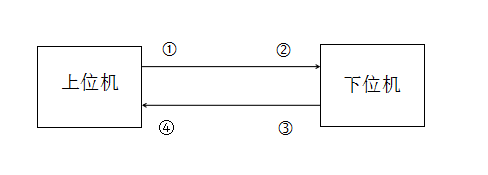
**3.4.2 通信过程**

采用以太网TCP/IP通讯方式，在上位机和下位机通讯过程中，一个典型的事物请求与处理流程为：

1. 客户端通过服务器的IP地址以及端口号与服务器连接；
2. 客户端构造带有应用层协议的TCP请求，其中包含了应用层的帧头等数据；

正常情况下TCP连接的建立

1. 客户端发送TCP请求到服务器，并等待服务器响应；
2. 服务器接收到请求，解析请求并处理，同时构造响应并发送至客户端；
3. 客户端收到响应后，一次事务处理完成。客户端应主动关闭到服务器的连接，方便服务器为其他客户端提供服务。



具体说明如下：

（1）上位机界面输入控制参数等数据。软件系统将用户输入的数据封装成帧，通过以太网发送到下位机。在封装的过程中，除了用户输入的数据，还需要加上应用层协议的起始字符、功能码、校验字符、帧头和帧尾等。

（2）下位机为了提高效率，可以采用中断控制方式，将上位机发送来的数据先存到“接收缓冲区”。然后对接收缓冲区中的数据进行扫描判定，判断是否存在符合条件的“正确帧”。

（3）如果找到符合规定的正确帧，下位机将根据功能码将数据及时反馈到上位机。

（4）上位机通过以太网口接收来自下位机的数据，先把数据放到上位机的“接收缓冲区”，然后再判断是否存在符合条件的正确帧。如果存在正确帧，将根据反馈的数据判断帧的类型（功能，绘图，错误），从而接下来做出不同的指令反应。

**3.4.3 应用层帧结构**

TCP通信是端对端，面向字节流的通信，所以为了能使上位机和控制器相互确定所发当前数据的含义，需要制定应用层的通信协议。为了保证通信过程的无差错传输，我们参照传统的TCP/IP通信协议，定义完整的帧结构，如表3-5所示。如果接收到的应用层数据帧严格符合上述定义，则称之为“正确帧”。

表3-5 帧结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 起始字符 | 数据长度 | 功能码 | 数据区域 | 校验字符 |
| 1字节 | 2字节 | 1字节 | N字节 | 2字节 |

具体说明如下：

1：收码和发码格式均设为十六进制格式。

2：控制器发送自定义数据，格式为：

* 起始数据帧头 + 功能字 + 数据长度 +校验和
* 数据长度为所发送数据的长度(字节数)（不包括起始数据帧头、功能码、数据长度标志还有校验和
* 校验和采用TCP/IP中方式，是整个数据帧的二进制反码和。

3：网络通信中数据采用大端模式传输，即多字节数据时，高字节数据在前，低字节数据在后传输，传输中的数据类型没有限制，unsigned int ，long，int等均可。

4：下位机传输给上位机显示绘图的帧，（帧格式：帧头+功能字+数据长度+ACC DATA（加速度）+VELOCITY DATA（速度）+POS DATA（位移）+ 0x00 0x00 + 0x00 0x00+校验和，共32字节，ACC DATA/ VELOCITY DATA/ POS DATA/数据为int格式，LEN = 0x10,其中将数据乘以一个系数化成整数后再进行传输（例如位移0.02\*100=2）

5：下位机发送PID数据给上位机的格式为：帧头+功能字+数据长度 + PID数据 + 0x00+0x00 + 校验和

PID数据为acc\_p,acc\_i,acc\_d,vel\_p..i..d,pos\_p,,i,,d，共9个uint16(unsigned short int)型数据也是将数据乘以系数后传输 (LEN=0x14)。

6：上位机发送PID数据给下位机的格式为：

PID数据格式和下位机发送给上位机的格式一样。

7：实验参数帧的格式为: 帧头+功能字+数据长度+ 波形 + 振幅 +频率 +0x00+0x00+ 校验和

其中数据为波形，振幅，频率3个u16数据，加上两字节填充数据,LEN=0x08

8：限制参数帧的格式为：帧头+功能字+数据长度 +限幅+限频+校验和

其中数据为 限幅 限频2个u16数据，LEN=0x04

所有PID参数扩大之后再进行传输

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 控制器->上位机 | | | | | | |
| 帧功能 | 帧头 | 功能字 | 长度 | 数据 | 校验 | 说明 |
| 状态帧  STATUS | 0x88 | 0x01 | LEN | int ACC\_DATA(加速度)\*100  int VEL\_DATA（速度）\*100  int POS\_DATA(位移）\*100 | SUM | 向上位机汇报当前状态，上位机接受并显示 |
| PID帧 | 0x88 | 0x02 | LEN | Short int ACC\_P  Short int ACC\_I  Short int ACC\_D  Short int VEL\_P  Short int VEL\_I  Short int VEL\_D  Short int POS\_P  Short int POS\_I  Short int POD\_d | SUM | 传输当前PID参数，所有PID参数扩大1000倍成为整数后传输 |
| 实验参数帧 | 0x88 | 0x03 | LEN | U16 waveform  U16 amplitude  U16 frequency | SUM | 汇报当前实验参数 |
| 限制参数帧 | 0x88 | 0x04 | LEN | U16 amp\_cons  U16 fre\_cons | SUM | 汇报当前限制参数 |
| 警告信息帧 | 0x88 | 0x05 | LEN | U16 warning | SUM | 向上位机发起警告（例如超过量程、连接断开） |
| 错误帧 | 0x88 | 0x06 | LEN | 错误信息 | SUM | 发生错误时，发送当前错误 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 上位机->控制器 | | | | | | |
| 帧功能 | 帧头 | 功能字 | 长度 | 数据 | 校验 | 说明 |
| 控制帧  STATUS | 0x80 | 0x01 | LEN | 控制命令   1. 启动停止 2. 传输数据 3. 错误处理 4. 请求返回数据 | SUM | 表明上位机接下来要做的动作吗，具体格式还未定 |
| PID帧 | 0x80 | 0x02 | LEN | Short int ACC\_P  Short int ACC\_I  Short int ACC\_D  Short int VEL\_P  Short int VEL\_I  Short int VEL\_D  Short int POS\_P  Short int POS\_I  Short int POD\_d | SUM | 上位机传输给控制器的三参量PID参数 |
| 实验参数帧 | 0x80 | 0x03 | LEN | U16 waveform  U16 amplitude  U16 frequency | SUM | 上位机设定的实验参数传给控制器 |
| 实验数据帧 | 0x80 | 0x04 | LEN | U16  position[array] | SUM | 上位机读取波形文件，并将其数据传入至控制器中 |
| 限制参数帧 | 0x80 | 0x05 | LEN | U16 amp\_cons  U16 fre\_cons | SUM | 上位机设定的限制参数 |
| 紧急停止帧 | 0x80 | 0x06 | LEN | U16 stop | SUM | 接受到错误后，要求振动台立即停止 |

整个应用层的通讯数据包应如下所示：

Struct data\_pack

{

Char head;

Char function;

Int length;

U8 \* data //指向数据的指针

U16 check\_sum;

U16 warning\_no;

U16 error\_no;

}

1. **系统实现**

模拟振动平台上位机软件系统是在Qt 5.9平台上，结合以太网TCP/IP通讯技术实现的。在图形用户界面设计中，采用C++编程语言，可以高效地开发交互性强的应用程序,并且具有很好地跨平台特性。

**4.1 系统管理模块**

系统管理模块分为前端设计和后端设计，前端设计主要包括UI人机交互界面，采用**Qt**中的**Qt designer**进行前端设计和布局。通过创建一个Qt中MainWindow Widget类，从而创建出整个上位机系统的主界面，在分析出系统管理模块的需求之后，可以在MainWindow中放置如toolbar、statusbar、pushbutton等等控件，从而搭建出满足功能需求的界面。

后端设计则涉及到处理各种按钮的信号逻辑，以及每个控件需要实现的功能。Qt中使用信号槽机制，通过connect函数，可以检测某个事件发生后，连接的槽函数会自动被回调，例如菜单栏中退出试验的信号槽代码如下：

QObject::connect(&button,&QPushButton::clicked,&QApplication::quit);

C++语言具有面向对象设计的特点，其继承、封装、多态可以很好的实现UI界面中的各种功能。

本上位机软件系统软件进入试验功能主界面后，系统管理模块主要包括七个部分：

* 菜单栏：

列出所有可供用户操作的命令和设置，有文件菜单、设置菜单、视图菜单、命令菜单、窗口菜单、显示菜单、存储菜单、报告菜单、帮助菜单等等。

* 工具栏：

工具栏列出用户将频繁使用的操作、设置和命令的快捷按钮。包括有文件管理工具栏、参数设置工具栏、显示操作工具栏和窗口工具栏等。

* 信号面板

信号面板位于功能主界面的左侧，默认是隐藏的，单击“信号”按钮，信号面板就会显示出来，信号面板中列出本试验的所有信号，包括系统信号和用户定义信号。

* 窗口显示

试验状态显示了试验运行命令、部分试验控制命令，显示本次试验的目标、控制状态信息和运行时间等。每个试验的运行状态项稍有不同。

试验信息显示试验报警、中断信息、历史记录以及谐振搜索与驻留记录。

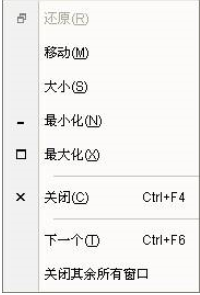
* 试验状态

试验状态显示了试验运行命令、部分试验控制命令，显示本次试验的目标、控制状态信息和运行时间等。每个试验的运行状态项稍有不同。

* 试验信息

试验信息显示试验报警、中断信息、历史记录以及谐振搜索与驻留记录。

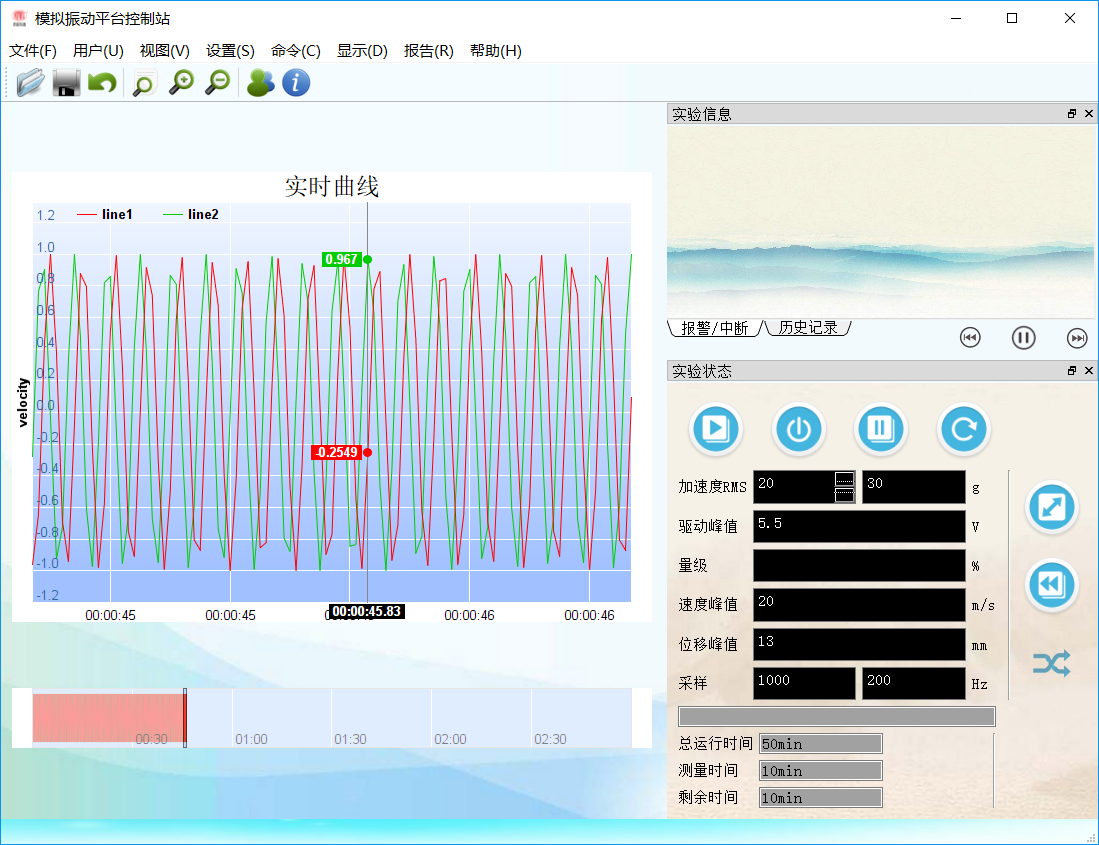
* 窗口列表

窗口列表显示当前所有打开的窗口，单击窗口名称按钮，可以使该窗口显示在最上层。在窗口名称按钮上右击，弹出如下浮动菜单：

* 状态栏

状态栏显示试验的存储路径和本次试验的运行状态，如初始化、运行预试验、正在执行第1个计划表的第1项等。

系统界面如下所示：



整个系统初始化关键代码如下:

ConfigureParameterAO tmpAO;

tmpAO.deviceName="PCI-1716,BID#1";

tmpAO.channelStart=0;

tmpAO.channelCount=1;

tmpAO.valueRange=*V\_Neg10To10*;

ConfigureParameterAI tmpAI;

tmpAI.deviceName="PCI-1716,BID#1";

tmpAI.channelStart=0;

tmpAI.channelCount=1;

tmpAI.valueRange=*V\_Neg10To10*;

tmpAI.clockRatePerChan=1000;

tmpAI.sectionLength=1024;

ConfigureParameterPCI tmp;

tmp.iniSettingAI=tmpAI;

tmp.iniSettingAO=tmpAO;

IniHelper \*tmpHelper=*new* IniHelper;

bool result=tmpHelper->writeToPCIIni("../IniSetting/PCI1716.ini",tmp);

*if* (result)

{

qDebug()<<"PCI1716.ini配置文件写入成功！";

}

*else*

qDebug()<<"PCI1716.ini配置文件写入失败";

*delete* tmpHelper;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*实时曲线绘制UI----LeftFrame\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

leftFrame = *new* QFrame(*this*);

leftFrame->setGeometry(4, 62, 660, 500);

leftFrame->setStyleSheet("background-color:white");

*//* *Chart* *Viewer*

m\_ChartViewer = *new* QChartViewer(leftFrame);

m\_ChartViewer->setGeometry(10, 25, 640, 480);

connect(m\_ChartViewer, *SIGNAL*(viewPortChanged()), *SLOT*(onViewPortChanged()));

connect(m\_ChartViewer, *SIGNAL*(mouseMovePlotArea(QMouseEvent\*)),

*SLOT*(onMouseMovePlotArea(QMouseEvent\*)));

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*控制---RightFrame\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

ui->labeStatusl->setStyleSheet("font:bold;font-size:12px;background-color:#7FFFD4");

ui->labeStatusl\_2->setStyleSheet("font:bold;font-size:12px;background-color:#7FFFD4");

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*状态栏\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

currentLabel=*new* QLabel;

currentLabel->setMinimumSize(200,25);

*//currentLabel->setFrameShadow(QFrame::Sunken);*

currentLabel->setFrameShape(QFrame::*WinPanel*);

ui->statusBar->addWidget(currentLabel);

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*颜色样式\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

setWindowIcon(QPixmap(":/Icon/Icon/dashboard.png"));

ui->menuBar->setStyleSheet("background-color:#EEE9E9");

ui->toolBar->setStyleSheet("background-color:#C0C0C0");

ui->statusBar->setStyleSheet("background-color:#CCFFFF");

*//leftFrame->setStyleSheet(QString::fromUtf8("border:1px* *solid* *red"));*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ENC7480\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

int d7480rtn= Enc7480\_Init();

qDebug()<<"leisai number:"<<d7480rtn;

*if*(d7480rtn < 1)

{

qDebug()<<"初始化ENC7480计数卡失败!";

QMessageBox::information(NULL,"提示","初始化ENC7480计数卡失败",QMessageBox::*Ok*|QMessageBox::*Cancel*);

*this*->close();

}

Enc7480\_Set\_Encoder(0,0);

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PCI1716初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

instantAoCtrl = InstantAoCtrl::Create();

startFlag=*false*;

dataScaled[0]=0;

dataScaled[1]=0;

*//将从ini文件中导入的配置初始化*

Initialize();

waveformAiCtrl = WaveformAiCtrl::Create();

waveformAiCtrl->addDataReadyHandler(OnDataReadyEvent, *this*);

waveformAiCtrl->addOverrunHandler(OnOverRunEvent, *this*);

waveformAiCtrl->addCacheOverflowHandler(OnCacheOverflowEvent, *this*);

waveformAiCtrl->addStoppedHandler(OnStoppedEvent, *this*);

ConfigureDevice();

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*参数初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*//*

*//* *Initialize* *member* *variables*

*//*

m\_currentIndex = 0;

*//* *Initially,* *auto-move* *the* *track* *line* *to* *make* *it* *follow* *the* *data* *series*

trackLineEndPos = 0;

trackLineIsAtEnd = *true*;

*//* *Initially* *set* *the* *mouse* *to* *drag* *to* *scroll* *mode.*

pointerPB->click();

*//* *Enable* *mouse* *wheel* *zooming* *by* *setting* *the* *zoom* *ratio* *to* *1.1* *per* *wheel* *event*

m\_ChartViewer->*setMouseWheelZoomRatio*(1.1);

*//* *Configure* *the* *initial* *viewport*

m\_ChartViewer->setViewPortWidth(initialVisibleRange / (double)initialFullRange);

*//* *Start* *the* *random* *data* *generator*

*//dataSource* *=* *new* *RandomWalk(OnData,* *this);*

*//dataSource->start();*

*//* *Set* *up* *the* *chart* *update* *timer*

m\_ChartUpdateTimer = *new* QTimer(*this*);

connect(m\_ChartUpdateTimer, *SIGNAL*(timeout()), *SLOT*(onChartUpdateTimer()));

*//* *The* *chart* *update* *rate* *is* *set* *to* *100ms*

m\_ChartUpdateTimer->start(100);

timer=*new* PerformanceTimer(*this*);

connect(timer,*SIGNAL*(timeout()),*this*,*SLOT*(slotFuction()));

timer->start(PERFORMANCEINTERVAL); *//20为毫秒*

msCount=0;

}

**4.2 系统通信模块**

**4.2.1 以太网TCP/IP通信基本参数的初始化**

本次课程设计采用FPGA作为下位机的CPU。作为高性能的全数字控制器可以在FPGA中移植轻量级TCP/IP协议：LwIP协议栈，从而与上位机进行网络通信。

基本通信参数设置如下：

表4-1 网络局域网通信参数初始值

|  |  |
| --- | --- |
| IP地址 | 192.168.0.X网段 |
| 子网掩码 | 255.255.255.0 |
| 默认网关 | 192.168.0.1 |
| 端口 | 8800 |
| TCP服务器Socket | 上位机 |
| TCP客户端Socket | 控制器 |

在上下位机通信过程中，需要使PC端与控制器端的IP地址在同一局域网内，并且控制器客户端连接上位机服务器打开的有效端口，双方通过socket套接字通信，Qt中模拟振动平台上位机端服务器初始化程序如下： MYTcpServer\_v1::MYTcpServer\_v1(QWidget \*parent)

: QDialog(parent)

, ui(*new* Ui::TcpServer)

{

ui->setupUi(*this*);

*this*->setFixedSize(350,180);

tcpPort = 8800;

tcpServer = *new* QTcpServer(*this*);

connect(tcpServer,*SIGNAL*(newConnection()),*this*,*SLOT*(sendMessage());

initServer();

}

*//开始发送数据*

void TcpServer::sendMessage()

{

ui->serverSendBtn->setEnabled(*false*);

clientConnection = tcpServer->*nextPendingConnection*();

connect(clientConnection,*SIGNAL*(bytesWritten(qint64)),*SLOT*(updateClientProgress(qint64)));

ui->serverStatusLabel->setText(QStringLiteral("开始传送数据%1 ！").arg(theFileName));

localFile = *new* QFile(fileName);

*if*(!localFile->*open*((QFile::*ReadOnly*)))

{

QMessageBox::warning(*this*,

QStringLiteral("应用程序"),

QStringLiteral("无法接受数据 %1:\n%2").arg(fileName).arg(localFile->errorString()));

*return*;

}

TotalBytes = localFile->*size*();

QDataStream sendOut(&outBlock,QIODevice::*WriteOnly*);

sendOut.setVersion(QDataStream::*Qt\_4\_6*);

time.start(); *//开始计时*

QString currentFile = fileName.right(fileName.size() - fileName.lastIndexOf('/')-1);

sendOut << qint64(0) << qint64(0) << currentFile;

TotalBytes += outBlock.size();

sendOut.device()->*seek*(0);

sendOut << TotalBytes << qint64((outBlock.size()-*sizeof*(qint64)\*2));

bytesToWrite = TotalBytes - clientConnection->write(outBlock);

qDebug() << currentFile << TotalBytes;

outBlock.resize(0);

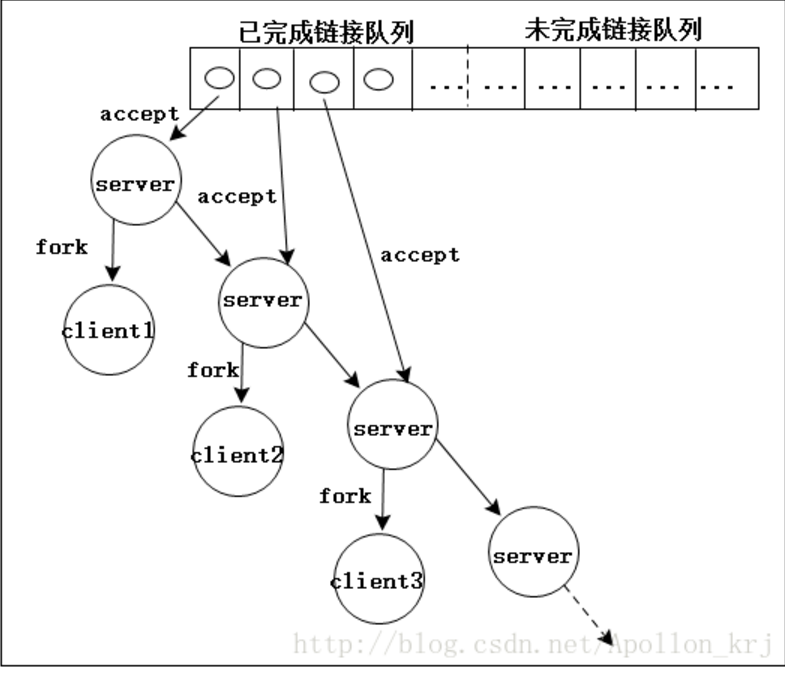
}

**4.2.2上位机通信设计**

1.并发服务器设置

在进行数据通讯时，模拟振动台上位机端服务器可能同时需要连接多个控制器客户端，此时因受限其基于打开端口-读取端口数据-写入数据-关闭连接的固定编程模式，服务器同时支持多个客户端的连接变得困难。此时考虑并发服务器

的实现，可以为每个客户端连接的TcpSocket分别分配一个专门的线程来处理。从而可以实现同时与多个客户端通信的功能,结构图如下所示：

 并发服务器结构图

2.上位机在处理接收数据的过程中要进行以下几个步骤：

1. 把数据存到“环形缓冲区”

下位机使用中断来接收数据，提高系统的实时性。与此相类似，上位机通过将下位机发来的数据及时存储在事先开辟的环形缓冲区，等待后续的处理。

1. 找正确帧

在环形缓冲区中寻找符合应用层 定义的正确帧，按照如下步骤严格判定。

①起始字符必须是0x88；

③功能码必须在0x00~0x10之间；

④帧的总长度≥数据长度+4；

⑤利用CheckSum函数计算校验字符，和接收帧的最后一个字节进行比对。

以上4条，只要有一条不符合，即判断帧错误，需要从当前位置往后重新扫描。

1. 进入相应的处理函数

如果找到正确帧，就根据不同的功能码从而进行不同的处理流程。

**4.3 数据处理模块**

模拟振动台上位机软件系统提供试验数据浏览和分析处理功能。浏览实验数据不需要硬件的支持，所以在设计过程中需要考虑在试验运行和脱离硬件状态下都可以浏览和处理由正式运行保存下来的实验数据：

**4.3.1 默认配置信息的保存**

实验配置信息在模拟振动台中具有很重要的作用，关系到整个系统的运行效果。在此上位机软件系统的界面设计中，需要实现一种加载和保存默认配置文件的功能，以便保存一些已经调节好的参数，这个问题可以通过将一些重要参数写入一个ini文件，和每次软件系统初始化时，均读入ini配置文件解决，关键实现代码如下：

void ConfigureDialog::Initailization()

{

InstantAoCtrl \* instantAoCtrl = InstantAoCtrl::Create();

Array<DeviceTreeNode> \*supportedDevices = instantAoCtrl->getSupportedDevices();

*if* (supportedDevices->getCount() == 1)

{

QMessageBox::information(*this*, tr("Warning Information"),

tr("No device to support the currently demonstrated function!"));

QCoreApplication::quit();

}

*else*

{

*for* (int i = 0; i < supportedDevices->getCount(); i++)

{

DeviceTreeNode *const* &node = supportedDevices->getItem(i);

qDebug("%d, %ls\n", node.DeviceNumber, node.Description);

ui.cmbDevice->addItem(QString::fromWCharArray(node.Description));

}

ui.cmbDevice->setCurrentIndex(0);

}

configure.profilePath = L"";

instantAoCtrl->Dispose();

supportedDevices->Dispose();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*将配置文件读入到configureAO 和 configureAI中\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

IniHelper \*hh=new IniHelper;

ConfigureParameterPCIiniSettingFile=hh->readFromPciIni("../IniSetting/PCI1716.ini");

configureAO=iniSettingFile.iniSettingAO;

configureAI=iniSettingFile.iniSettingAI;

qDebug()<<"from the iniFile AO is :"<<configureAO.deviceName<<configureAO.channelStart<<configureAO.channelCount<<configureAO.valueRange;

sPIDInfo=hh->readFromCtrlIni("../IniSetting/CtrlIni.ini");

delete hh;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*将配置文件读入到configureAO 和 configureAI中\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IniHelper \*hh=new IniHelper;

ConfigureParameterPCIiniSettingFile= hh->readFromIni("../IniSetting/PCI1716.ini");

configureAO=iniSettingFile.iniSettingAO;

configureAI=iniSettingFile.iniSettingAI;

qDebug()<<"the configureParameterAO is :"<<configureAO.deviceName<<configureAO.channelStart<<configureAO.channelCount<<configureAO.valueRange;

*//将Ini文件中默认的配置显示出来*

void ConfigureDialog::SetConfigureParameterAO(ConfigureParameterAO tmpConfigure,ConfigureParameterAI tmpConfigureAI)

{

configure=tmpConfigure;

ui.cmbDevice->setCurrentIndex(0);

ui.cmbChannelStart->setCurrentIndex(0);

ui.cmbChannelCount->setCurrentIndex(0);

ui.cmbValueRange->setCurrentIndex(2);

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

configureAI=tmpConfigureAI;

ui.cmbChannelStart\_AI->setCurrentIndex(0);

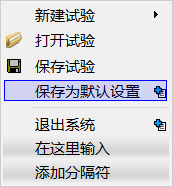
ui.cmbChannelCount\_AI->setCurrentIndex(0);

ui.cmbValueRange\_AI->setCurrentIndex(4);

ui.edtClockRatePerChan\_AI->setText(QString::number(configureAI.clockRatePerChan));

ui.edtSectionLength\_AI->setText(QString::number(configureAI.sectionLength));

}

功能界面如下所示,以及保存文件如下所示：

**4.3.2 试验参数和控制**

保存的试验工程文件记录了当前实验的各项系统参数的设置情况，包括通道参数、控制参数、限制餐宿等等，为数据处理提供参考依据。

这些试验参数仅供分析参考使用，不允许用户修改，所以个参数对话框的“确定”button都处于禁止按下状态。

此外，我们需要浏览一些数据，所以需要一些控制浏览的控件实现，本上位机软件系统中采用了5个pushbutton按钮实现功能，并且重写了他们的信号槽函数：

****

显示第一帧数据；

显示当前数据的前一帧数据；

显示当前数据的后一帧数据；

显示最后一帧数据；

系统会自动从第一帧数据开始浏览显示，直至 用户按下“停止”命令或运行到最后一帧数据，系统才停止浏览；

停止浏览后面的数据；

关键实现代码如下所示:

void MainWindow::on\_action\_ChannelParameters\_triggered()

{

*//首先进入构造函数*

ConfigureDialog dialog;

dialog.SetConfigureParameterAO(configureAO,configureAI);

int resultDialog = dialog.*exec*();

*//将主程序中的configureAO传入到设置界面*

*//qDebug()<<"the* *configureParameter* *is* *:"<<configureAO.deviceName<<configureAO.channelStart<<configureAO.channelCount;*

*if* (resultDialog == QDialog::*Rejected*)

{

*//exit(0);*

}

*else* *if*(resultDialog == QDialog::*Accepted*)

{

SetConfigureParameterAO(dialog.GetConfigureParameter());\

SetConfigureParameterAI(dialog.GetConfigureParameterAI());

qDebug()<<"the data from configureDialog is :";

qDebug()<<"haha"<<configureAO.deviceName<<configureAO.channelStart<<configureAO.channelCount<<configureAO.valueRange;

qDebug()<<"haha"<<configureAI.deviceName<<configureAI.channelStart<<configureAI.channelCount<<configureAI.valueRange<<configureAI.clockRatePerChan<<configureAI.sectionLength;

Initialize();

}

}

void MainWindow::on\_btnStart\_clicked()

{

double AoValue;

*if* (ui->btnStart->isFlat())

{

ui->btnStart->setFlat(*false*);

ui->btnStart->setStyleSheet("background-color:green");

}

*else*

ui->btnStart->setFlat(*true*);

*//PID静态位移控制开始*

msStartCount=0;

*//AO*

AoValue=ui->lineEdit->text().toDouble();

*//qDebug()<<"输出电压"<<AoValue;*

*if*(AoValue<=-10||AoValue>=10)

{

QMessageBox::information(NULL,"提示","Too large Voltage!",QMessageBox::*Ok*|QMessageBox::*Cancel*);

exit;

}

*for* (int i = 0; i < 2; i++)

dataScaled[i]=AoValue;

*//AI*

ErrorCode errorCode = waveformAiCtrl->Start();

CheckError(errorCode);

startFlag=*true*;

}

void MainWindow::on\_btnStop\_clicked()

{

*if* (ui->btnStop->isFlat())*//没按*

{

ui->btnStop->setFlat(*false*);

ui->btnStop->setStyleSheet("background-color:green");

}

*else*

{

ui->btnStop->setFlat(*true*);

}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*AO\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

timer->stop();

dataScaled[0]=0;

dataScaled[1]=0;

ErrorCode errorCode = *Success*;

errorCode = instantAoCtrl->Write(configureAO.channelStart,configureAO.channelCount, dataScaled);

CheckError(errorCode);

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*AI\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

errorCode = waveformAiCtrl->Stop();

CheckError(errorCode);

ui->lineEdit->setText("0");

}

void MainWindow::on\_action\_ControlParameters\_triggered()

{

CtrlDialog dialog;

dialog.setPIDParameter(sPIDInfo);

int resultDialog = dialog.*exec*();

*if* (resultDialog == QDialog::*Rejected*)

{

*//exit(0);*

}

*else* *if*(resultDialog == QDialog::*Accepted*)

{

sPIDInfo=dialog.getPIDParameter();

qDebug()<<"PID 参数是:"<<sPIDInfo.SP<<sPIDInfo.SI<<sPIDInfo.SD;

}

}

void MainWindow::on\_pushButton\_clicked()

{

*for* (int i = 0; i < 2; i++)

dataScaled[i]=ui->lineEdit->text().toDouble();

ErrorCode errorCode = *Success*;

errorCode = instantAoCtrl->Write(configureAO.channelStart,configureAO.channelCount, dataScaled);

CheckError(errorCode);

}

**4.3.3 试验数据软件的存储和试验报告**

在浏览试验数据过程中和浏览结束后，都可以按照一定的格式和内容产生相应的Word试验报告，用户可以定义报告信息和报告内容。振动台上位机软件系统提供标准的报告形式，可以在线产生报告或历史结果报告等,并且可以保存当前实验数据曲线，以便之后分析，实现代码如下所示：

*//*

*//* *The* *Save* *button* *is* *pressed*

*//*

void MainWindow::onSave(bool)

{

QString fileName = QFileDialog::getSaveFileName(*this*, "Save", "chartdirector\_demo",

"PNG (\*.png);;JPG (\*.jpg);;GIF (\*.gif);;BMP (\*.bmp);;SVG (\*.svg);;PDF (\*.pdf)");

*if* (!fileName.isEmpty())

{

*//* *Save* *the* *chart*

BaseChart \*c = m\_ChartViewer->*getChart*();

*if* (0 != c)

c->makeChart(fileName.toUtf8().constData());

}

}

**4.4 数据显示模块**

**4.4.1 三参量数据图表分析**

在试验过程中，模拟振动平台时的实时位移、加速度、速度数据都反馈到上位机中，为了能实时、方便地观察浏览试验情况，需要将其动态曲线绘制出来。浏览试验数据曲线时，可以改变窗格大小、设置窗格中信号的 、添加和删除标记注释、缩放图形等等。

本上位机软件系统采用Qtcharviewer控件进行动态绘图，关键代码如下所示：

*//* *Initialize* *member* *variables*

*//*

m\_currentIndex = 0;

*//* *Initially,* *auto-move* *the* *track* *line* *to* *make* *it* *follow* *the* *data* *series*

trackLineEndPos = 0;

trackLineIsAtEnd = *true*;

*//* *Initially* *set* *the* *mouse* *to* *drag* *to* *scroll* *mode.*

pointerPB->click();

*//* *Enable* *mouse* *wheel* *zooming* *by* *setting* *the* *zoom* *ratio* *to* *1.1* *per* *wheel* *event*

m\_ChartViewer->*setMouseWheelZoomRatio*(1.1);

*//* *Configure* *the* *initial* *viewport*

m\_ChartViewer->setViewPortWidth(initialVisibleRange / (double)initialFullRange);

*//* *Start* *the* *random* *data* *generator*

*//dataSource* *=* *new* *RandomWalk(OnData,* *this);*

*//dataSource->start();*

*//* *Set* *up* *the* *chart* *update* *timer*

m\_ChartUpdateTimer = *new* QTimer(*this*);

connect(m\_ChartUpdateTimer, *SIGNAL*(timeout()), *SLOT*(onChartUpdateTimer()));

*//* *The* *chart* *update* *rate* *is* *set* *to* *100ms*

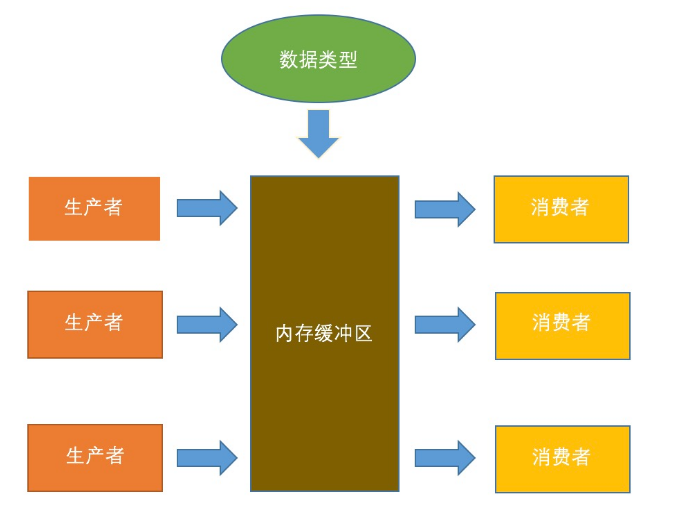
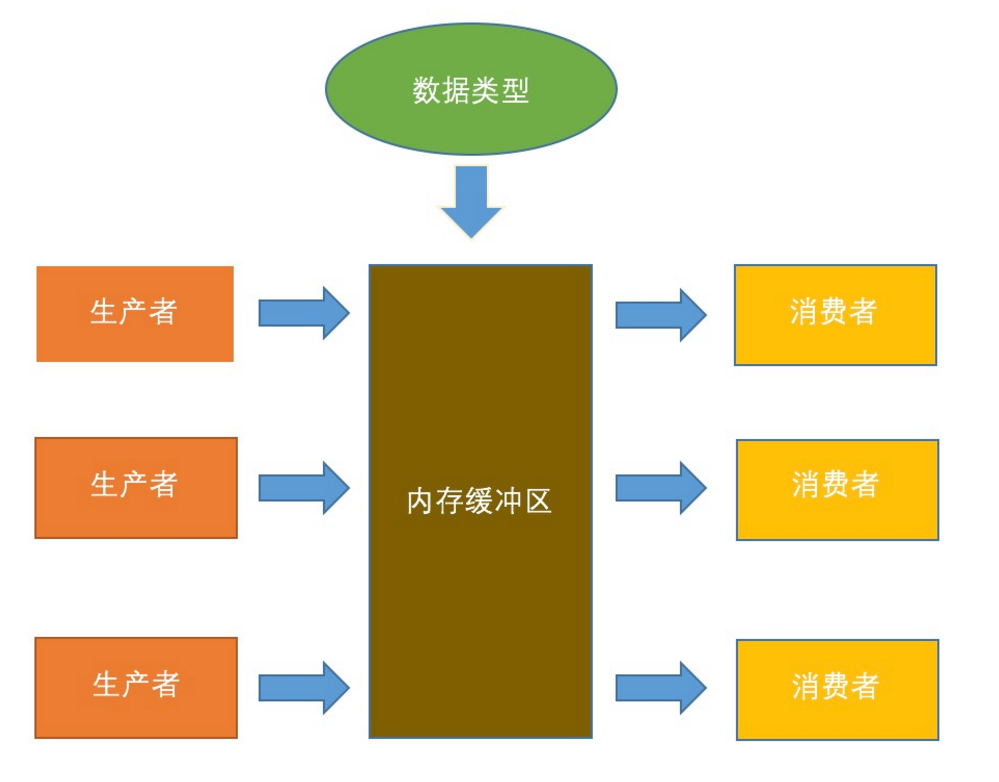
m\_ChartUpdateTimer->start(100);

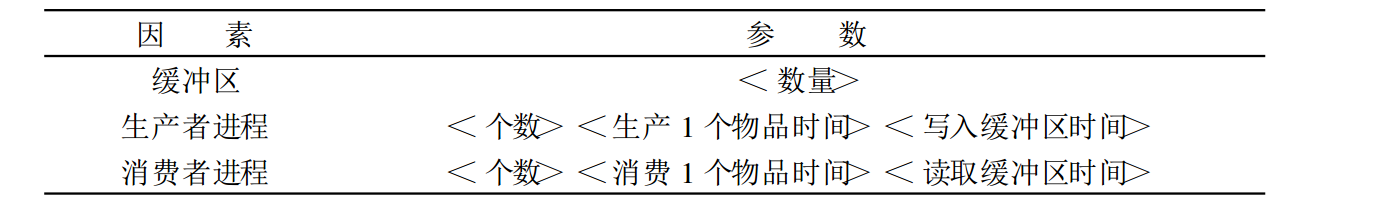
timer=*new* PerformanceTimer(*this*);

connect(timer,*SIGNAL*(timeout()),*this*,*SLOT*(slotFuction()));

timer->start(PERFORMANCEINTERVAL); *//20为毫秒*

**4.4.2 生产者-消费者矛盾**

 进行动态绘图的过程中，会碰到一个问题，一种情况是下位机在不停产生数据并上传，此时上位机在处理一些其他的任务，那么会导致数据溢出，并在绘图过程中产生闪烁中断等问题。而如果下位机上传数据的速度不够快，则会导致上位机不能正确绘图，并且容易引起系统崩溃。我们把这个称为“生产者-消费者矛盾”。而动态绘图过程中必须解决这个生产和消费速度不匹配的矛盾，从而解决这个并发读写的问题，生产消费者模式如下图所示：

生产者-消费者算法参数

解决这个问题可以采用开辟双缓冲区，实现绘图过程中的读写分离。通过添加互斥信号量对缓冲区进行读写保护，当某一个缓冲区作为下位机发送数据的缓冲区时，请求信号量，进行互斥保护，此时上位机读取另外一个缓冲区进行绘图，完成操作后，释放信号量，读写缓冲区交换，由于数据的读写分离，并且有互斥信号量的读写保护，所以可以解决其速度不匹配的问题,关键实现代码如下：

*class* QDoubleBufferedQueue

{

*public*:

*//*

*//* *Constructor* *-* *construct* *a* *queue* *with* *the* *given* *buffer* *size*

*//*

QDoubleBufferedQueue(int bufferSize = 10000) :

bufferLen(0), bufferSize(bufferSize)

{

buffer0 = buffer = *new* T[bufferSize];

buffer1 = *new* T[bufferSize];

}

*//*

*//* *Destructor*

*//*

~QDoubleBufferedQueue()

{

*delete*[] buffer0;

*delete*[] buffer1;

}

*//*

*//* *Add* *an* *item* *to* *the* *queue.* *Returns* *true* *if* *successful,* *false* *if* *the* *buffer* *is* *full.*

*//*

bool put(*const* T& datum)

{

mutex.lock();*//mutex是互斥量，把buffer已经指针锁起来，不允许其他现成进行操作*

bool canWrite = bufferLen < bufferSize;

*if* (canWrite) buffer[bufferLen++] = datum;

mutex.unlock();

*return* canWrite;

}

*//*

*//* *Get* *all* *the* *items* *in* *the* *queue.* *The* *T\*\** *argument* *should* *be* *the* *address* *of* *a* *variable* *to*

*//* *receive* *the* *pointer* *to* *the* *item* *array.* *The* *return* *value* *is* *the* *size* *of* *the* *array.*

*//*

int get(T\*\* data)

{

mutex.lock();

\*data = buffer;

int ret = bufferLen;

buffer = (buffer == buffer0) ? buffer1 : buffer0;*//交换*

bufferLen = 0;

mutex.unlock();

*return* ret;

}

*private*:

*//* *Disable* *copying* *and* *assignment*

QDoubleBufferedQueue & operator=(*const* QDoubleBufferedQueue&);

QDoubleBufferedQueue(*const* QDoubleBufferedQueue&);

T\* buffer0;

T\* buffer1;

T\* buffer;

int bufferLen;*//当前指针*

int bufferSize;*//缓冲区总的长度*

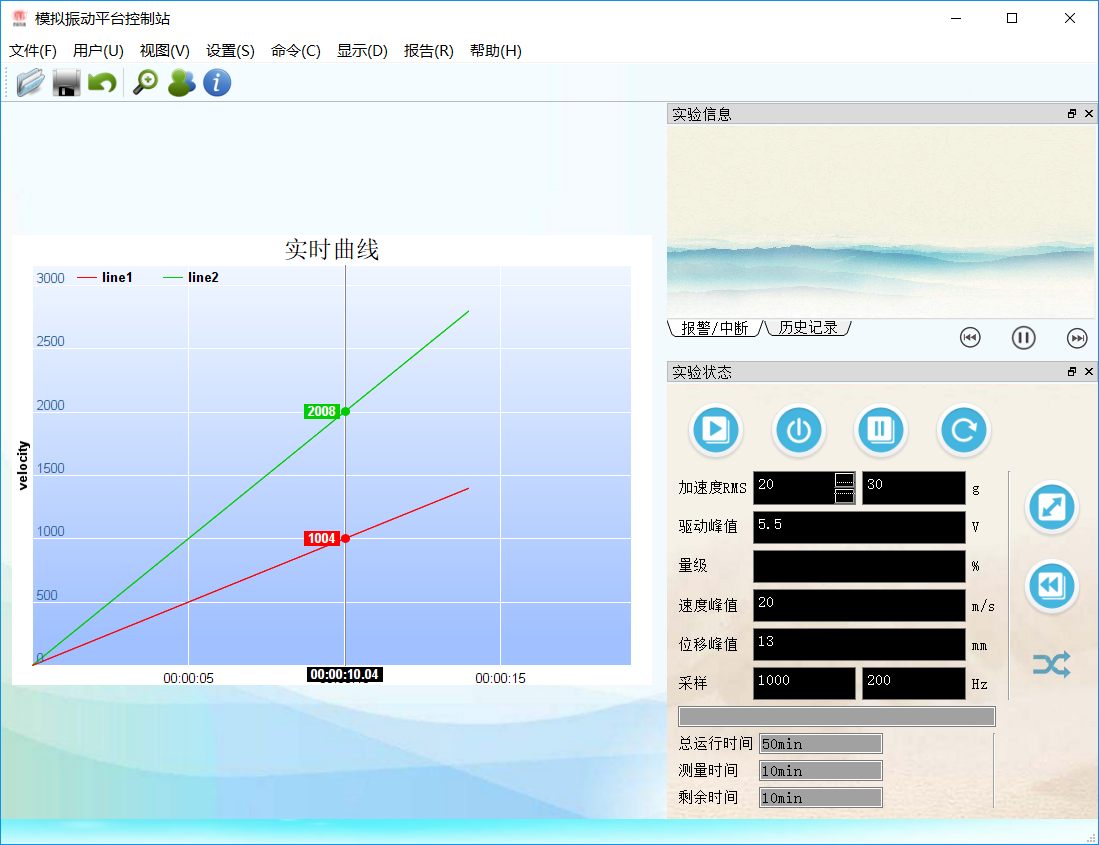
QMutex mutex;

};

1. **系统调试及测试**

模拟振动平台上位机软件系统主要包括系统管理模块、数据处理模块、通讯模块、数据显示与保存模块等四个模块。现从软件的实际使用中，测试本系统各个模块功能的实现情况。

在启动模拟振动平台上位机软件系统后，首先初始化PC端与PCI板卡的驱动链接。之后读入之前保存的默认配置文件，加载一些默认配置参数，之后可以选择“文件”菜单中的“新建实验”命令或点击工具栏上的按钮开启一次新的实验。

图5-1 振动平台上位机管理系统主界面

在试验选择界面,主要划分为4个功能大类：典型试验、混合试验、波形再现、其他试验。

分别如下所示：

* 典型试验
* 随机试验
* 典型冲击试验
* 正弦试验
* 地震随机波形试验
* 混合试验
* 正弦加随机试验
* 随机加随机试验
* 地震波形加随机波形试验
* 波形再现
* 瞬态冲击试验
* 地震波模拟再现试验
* 其他试验

首先单击标签选择功能大类，然后双击功能按钮，软件会创建相应的试验项目，之后进入相应类型的试验主界面,如下图所示:

试验选择界面

通过点击不同的button控件，从而实现加载不同的实验数据，进行所选择的试验。

模拟振动平台上位机控制管理软件提供了用户管理机制，可以限制用户使用权限，防止软件被擅自使用。

可以在主界面中单击“用户”菜单—‘用户登录’，来进行用户管理,用户登录的菜单界面如下所示

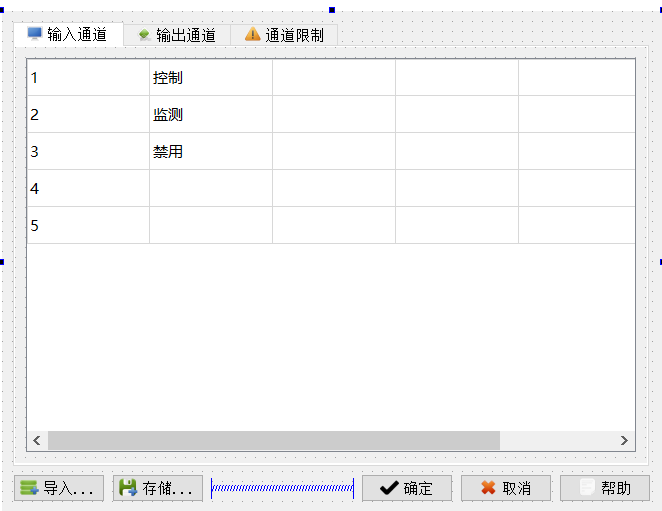
 用户登录菜单

用户登录界面如下:

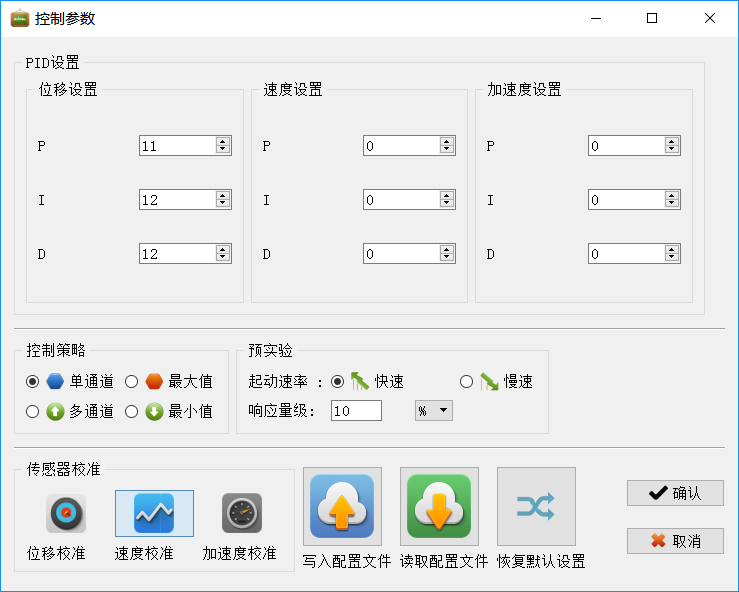


通过用户管理，可以设置用户的权限，管理员可以修改用户权限：开始试验、通道参数、控制参数等等。通过管理员对用户的权限管理，可以提高振动平台实验时的安全性。

主界面中点击“设置”---“通道参数”，即可打开通道参数设置界面，通道参数用来设置所有连接到输入端的传感器或者前置调理电路的相关参数，如输入通道的量程、耦合方式、传感器的类型等、还有输出通道的参数，以及通道限制参数等等。如下图所示：

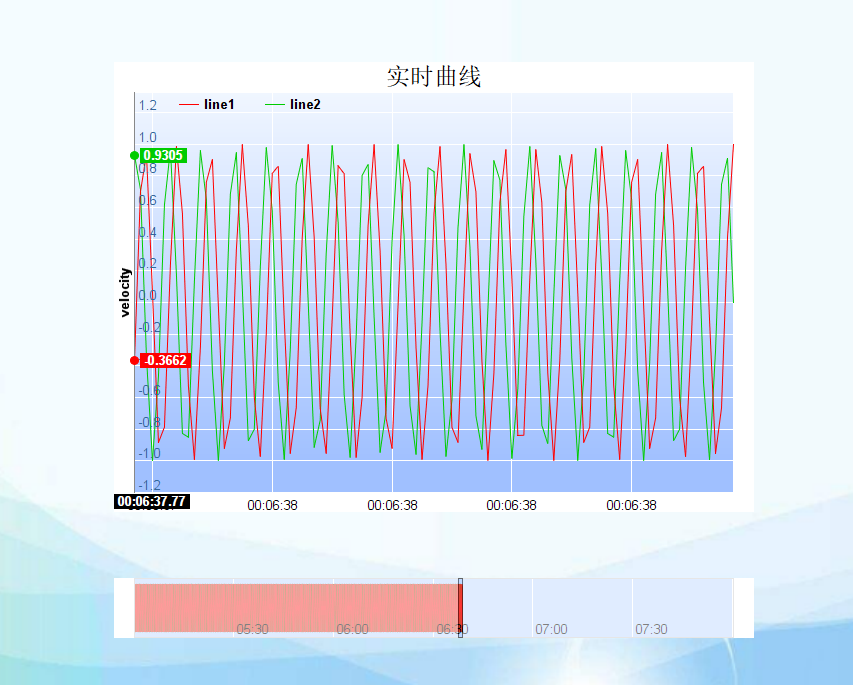
 设置菜单

通道参数界面

 主界面中点击“设置”—-“控制参数”，即可打开控制参数设置界面，控制参数是用来定义与试验运行、控制相关的参数。例如，控制策略、量级、预实验和高级设置等等，如下图所示:

控制参数设置界面

在主界面的左侧，为信号实时曲线显示区域，采用Qt charviewer控件开发，利用多媒体定时器实时绘图，并且具有坐标自适应缩放，实时显示数据幅值等功能，在窗口任意处单击并拖动鼠标可以上、下、左、右拖动曲线，双击窗口将弹出数据显示大窗口以供清楚查看数据,绘图区域下方有全局绘图状态条，可以全局总览整个试验过程中的数据波形从而进行分析。信号曲线实时绘制界面如图所示：



三参量曲线实时显示界面以及全局状态条

1. **总结与展望**

**6.1总结**

本文设计的康复训练管理系统，为国内相关方面的研究提供了更加丰富详实的工程经验积累。首先详细介绍了足底压力测量国内外研究进展，探讨了本课题研究的意义和价值。然后通过需求分析，确立了本系统的整体研究框架，并简单介绍了硬件平台和软件开发环境。接着从用户界面、数据库和串口通信三个方面，对康复训练管理系统进行详细的设计。紧接着是以上三个模块的具体实现过程，重点论述数据库访问技术和C#折线图绘制方法。最后对整个系统的模块功能进行测试。

本文特色和创新点主要有：

采用基于ModBus的通信协议，在数据接收过程中开辟“环形缓冲区”，严格定义帧的结构，实现数据的无差错传输。

在VS2010中结合使用SQL Server数据库，实现病人信息和训练方案的新建、查询、删除等功能。在“压力数据图表分析界面”提供实时压力数据折线图和日K线图，直观地显示病人的训练情况，让医生一目了然。

**6.2 展望**

本文所设计的康复训练管理系统，实现了一些基本的预期功能。但是，还有不少地方需要改善，有很多的提升空间。后续的开发若要有所突破，可以从以下三个方面加以改善：

在“压力数据图表分析界面”不仅实现图表的绘制，还应该给出提醒，针对病人的压力曲线，智能地帮助医生给出医嘱。

本课题是PC端与下位机直接连接，未来可以考虑加入手机移动客户端。单片机的数据首先借助蓝牙上传到手机，再发送到远方的数据库。

康复训练管理系统是一项有价值的技术，相信经过本设计的不断完善，最终能够最大程度地发挥其在医学临床诊断、可穿戴设备等方面的作用。

**参考文献**

1. 彭开怀. 基于嵌入式系统的足底压力分布检测仪[D]. 电子科技大学, 2015
2. 刘世法. 足底压力分布综合分析系统的开发与应用[D]. 中国科学技术大学, 2009.
3. 向仁英, 罗运莲, 张静荣,等. 骨科康复训练实施卡的设计与应用[J]. 中华现代护理杂志, 2009, 15(33):3566-3566.
4. 祝光志. 基于足底压力检测装置的数据分析系统设计[D]. 电子科技大学, 2014
5. 周定康. 关系数据库理论及应用[M]. 华中科技大学出版社, 2005.
6. 顾潇. 基于SysML的应急管理系统建模技术研究[D]. 南京理工大学, 2014.
7. 苗雪兰, 刘瑞新, 邓宇乔,等. 数据库系统原理及应用教程[M]. 机械工业出版社, 2014.
8. 张齐 杜群贵. 单片机应用系统设计技术: 基于C语言编程[M]. 电子工业出版社, 2004.
9. 方开红. ADO.NET数据库访问技术在ASP.NET开发中的研究与应用[J]. 软件工程师, 2015(7):19-21.
10. 张凯华, 王兰美, 李安国. 基于LabVIEW的鞋垫式足底压力测试系统的设计与实现[J]. 传感器世界, 2009, 15(3):31-33.
11. 梁玉. 基于C#的数据采集上位机软件设计[D]. 西安电子科技大学, 2014.
12. 胡锃, 李文杰. ADO.NET脱机数据的开放式并发控制研究[J]. 计算机与现代化, 2010(3):162-165.
13. Lord M. Foot pressure measurement: a review of methodology[J]. Journal of Biomedical Engineering, 1981, 3(2):91-99.
14. Orlin M N, Mcpoil T G. Plantar pressure assessment.[J]. Physical Therapy, 2000, 80(4):399-409.
15. Elftman H. A cinematic study of the distribution of pressure in the human foot[J]. Anatomical Record, 2005, 59(4):481-491.
16. Praet S F, Louwerens J W. The influence of shoe design on plantar pressures in neuropathic feet[J]. Diabetes Care, 2003, 26(2):441.
17. Abdul R A H, Aladin Z, Begg R K, et al. Foot Plantar Pressure Measurement System: A Review[J]. Sensors, 2012, 12(7):9884.
18. Chockalingam N, Healy A, Naemi R, et al. Comments and Reply to: Foot Plantar Pressure Measurement System: A Review. Sensors 2012, 12, 9884-9912[J]. Sensors, 2013, 13(3):3527.