2012 年英特尔杯大学生电子设计竞赛嵌入式系统专题邀请赛

2012 Intel Cup Undergraduate Electronic Design Contest

- Embedded System Design Invitational Contest

作品设计报告

Final Report

Intel Cup Embedded System Design Contest

报告题目： Guitar Robot—吉他机器人

学生姓名： 仇成 许逸航 阎锡民

指导教师： 唐新民

参赛学校： 北京师范大学-香港浸会大学联合国际学院

2012 年英特尔杯大学生电子设计竞赛嵌入式系统专题邀请赛

参赛作品原创性声明

本人郑重声明：所呈交的参赛作品报告，是本人和队友独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果，不侵犯任何第三方的知识产权或其他权利。本人完全意识到本声明的法律结果由本人承担。

参赛队员签名：

日期：　　年　　月　　日

吉他机器人

摘要

随着科技与时代的发展，机器人技术已日趋成熟，且越来越受到人们的关注与喜爱。与此同时，音乐机器人也逐步登上历史舞台，既带给人们音乐的美好享受，又解放了部分劳动力。为此，我们设计了一款吉他机器人表演系统，该系统将现代计算机技术与吉他弹奏相结合，并基于 Intel@ Atom 处理器的嵌入式平台，结合自主设计的通信控制系统，实现了对 MIDI 音乐文件的解析，弹奏吉他以及实时生成 3D 伴奏图形等功能。

为了使机器能够识别 MIDI 音乐文件，我们使用了 MIDI 文件解析算法，将 MIDI 文件解析成音符，之后转换为 16 进制码作为控制命令传递给控制系统进行识别，控制系统通过气动装置进行识别，以达到精准按压和拨弦的效果。同时，为了充分发挥 Intel@ Atom 嵌入式平台的强大功能，我们设计了 3D 实时变换效果，使得人们在欣赏音乐的同时能够有不一样的视觉体验。

本系统适用于机场，饭店以及室内等场所，可代替人类进行大量繁琐或重复性大的演奏任务。

关键词： 吉他机器人，通信控制，MIDI 解析算法，气动装置，3D 图形

GUITAR ROBOT

ABSTRACT

This project aims to develop a programmable instrument which works like an acoustic guitar. It is a combination work of mechanical design, software development and hardware integration. In this work, we built all the hardware of this instrument by ourselves. In order to get a better design, we reviewed some existed work first. And then we designed the 3D model of the instrument by using AutoCAD. The implementation work is based on Intel@ Atom CPU platform. We use pneumatic system to play the guitar, use PLC and demo board work as the control module, and use the COM serial port as the communication interface. MIDI file is used as the music source. It has been decoded to generate the control signal and passed to the control module. Meanwhile, 3D graphic has been generated according to notes that decoded from the MIDI file for visualization effect of the music.

This system is suitable to be placed at airport, restaurant and indoor place etc. And it can replace human to conduct music performance.

Key words: Programmable Instrument, Automated Control, MIDI Decoding, Pneumatic Device, 3D Graphic

目　录

第一章 绪论----------6
1.1 音乐机器人历史----------6
1.2 文献回顾----------6
1.2.1 LEMUR 机器人简介----------6
1.2.2 Team Dare 吉他机器人----------8
1.3 本章小结----------9
第二章 系统总体设计方案----------9
2.1 系统设计----------9
2.1.1 3D 模型设计----------10
2.1.2 原始设计的优劣---------- 11
2.2 系统实现原理----------11
2.3 系统功能及特色----------12
2.4 系统指标----------12
第三章 硬件框图----------13
3.1 硬件配备---------- 13
3.2 硬件原理----------13
第四章 软件流程----------16
4.1 解析 MIDI----------16
4.1.1 MIDI 简介---------- 16
4.1.2 选用 MIDI 的原因----------17
4.1.3 MIDI 构成 ----------17
4.1.4 解析算法----------19
4.2 串口通信----------21
4.2.1 通信协议----------22
4.2.2 程序原理---------- 23
4.3 3D 实时图形----------24
4.3.1 音乐可视化---------- 24
4.3.2 使用 OpenGL 以及 3DSMAX 原因---------- 25
4.3.3 模型制作及操作----------25
第五章 系统测试----------26
5.1 硬件系统测试----------26
5.1.1 空气压缩机测试----------26
5.1.2 继电器及电磁阀模块测试----------26
5.1.3 小气缸模块测试----------27
5.1.4 整体测试----------27
5.2 软件系统测试----------28
5.2.1 MIDI 解析测试----------28
5.2.2 串口通信测试---------- 28
5.2.3 3D 伴奏图形测试---------- 29
第六章 结论---------- 30
参考文献----------31
谢辞----------32

第一章 绪论

随着时代与科技的发展，机器人也愈来愈成为人类的宠儿。机器人的发展大致可以分为三个阶段：第一阶段工业机器人，第二阶段带有“感觉”的机器人，第三阶段智能机器人。机器人可以帮助人类完成很多繁重甚至危险的任务，大幅度的解放劳动力，提高工作效率。虽然机器人的发展迅猛，但是音乐机器人的概念并不为人所知，基于此，我们大胆设计并开发了吉他机器人这一项目，使得机器人可以弹奏不同的吉他乐曲，服务于人类。

1.1 音乐机器人历史

最初的音乐机器人是使用自动齿轮这样的机械部件来创造音乐。1863 年，第一个名叫 "Pianista" 的音乐机器人在 Fourneaux 问世[1]；上世纪 90 年代早期，Trimpin 制造出名为 "Krantkontrol"的连续 12 个类似吉他的乐器，而每把吉他都有一个马达和 H-桥构成的拨弦系统，及四音调[2]。之后在 2003 和 2008 年，LEMUR 和 Team Dare 制造出了新款的吉他机器人。

1.2 文献回顾

下面将介绍 LEMUR 和 Team Dare 的吉他机器人，我们的系统设计灵感大部分来自这两种设计，并综合了每种设计的优点，在有限的时间及资源下完成我们的作品。

1.2.1 LEMUR 机器人简介

LEMUR（2003）的吉他机器人可分为三个系统：支撑系统，拨弦系统，和音调控制系统。支撑系统主体是一块"36 X 4"(cm)的铝合金底座，配以 Saddle（鞍），nuts（一种固定物）以及钢轨桥，如图 1：

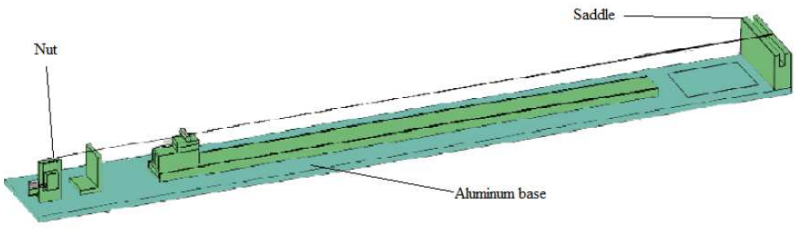


图 1　LEMUR 支撑系统

拨弦系统是由随动马达，配以 3 片划桨，一个电磁阀瓣，一个低音弦以及一个选择轮构成。随动马达通过驱动 3 片划桨模拟人类拨弦的动作；而数字信号控制电磁阀模拟人类压品的动作。皮带和轮子则可以将马达的力传递到整个系统。如图 2 所示[3]：

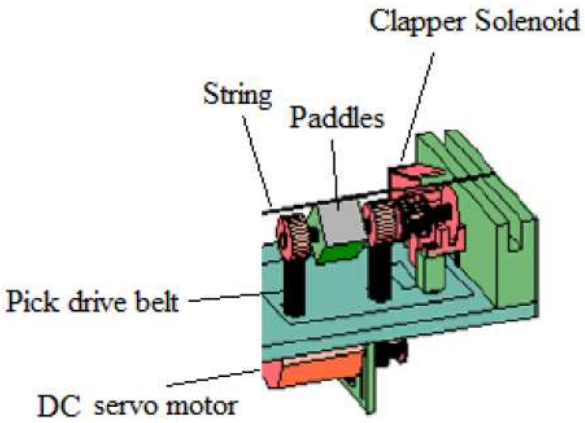


图 2　LEMUR 马达系统

音调控制系统是由 DC 随动马达，十匝旋转电位计和可移动钢轨桥构成。DC 随动马达可控制钢轨桥移动到准确的品位，时间极快，从首到尾只需 250 毫秒。而电位计则可以用自己的自选角记录每个品的位置，告诉 DC 马达应该按压那个品。如图 3 所示[3]：

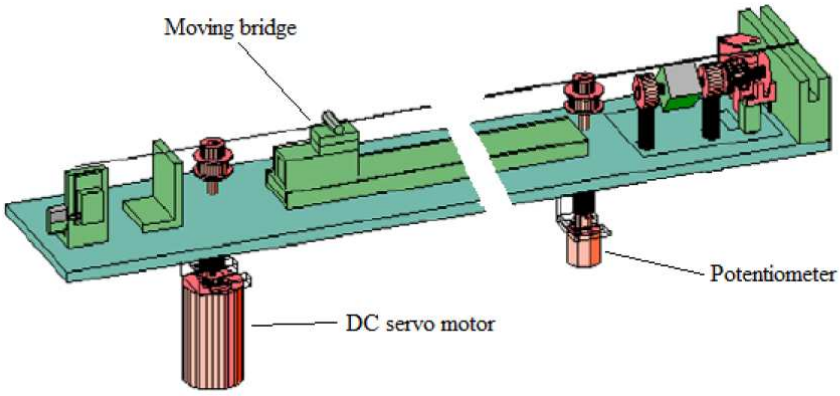


图 3　LEMUR 音调控制系统

LEMUR 的吉他机器人简单，高效。首先，与机械手臂以及拨弦棒相比，LEMUR 的拨弦系统设计新颖。我们知道，机械手臂和拨弦棒的运作是依靠轴承去传递能量的，这就意味着能量在马达和桨（paddles）之间会有延迟和消散；如果要保证系统的正常运行，马达的扭力需要更高的标准，但与此同时投入的成本会更高。而 LEMUR 的系统则不同，它会直接操作 paddles，降低了能量的损耗。假设拨弦系统马达是 300r/m,那么拨弦的速度就会超过每分钟 900 次，也就是说，每秒钟拨弦 15 次！现在没有音乐可以这么快，LEMUR 的设计足以弹奏各种音乐。LEMUR 的设计还有一个好处，易于拆卸，并且金属的材质不易于扭曲，可用时间长。

但是 LEMUR 的设计缺点也是很明显的。第一，4 根弦不能弹奏所有的音符；第二，金属材质的调音容易走调，速度虽快，但不够稳定，弹出的音可能与预期的不同。

1.2.2 Team Dare 机器人简介

Team Dare 的吉他机器人于 2008 年获得阿尔特密斯管弦演奏金奖。他的机器人也是分为三个部分：支撑部分，音调控制部分以及拨弦部分。不同之处是 Team Dare 的支撑部分十

分复杂，因为他的目的是将这套装置安放在真正的吉他上面，如图 4 所示[4]：

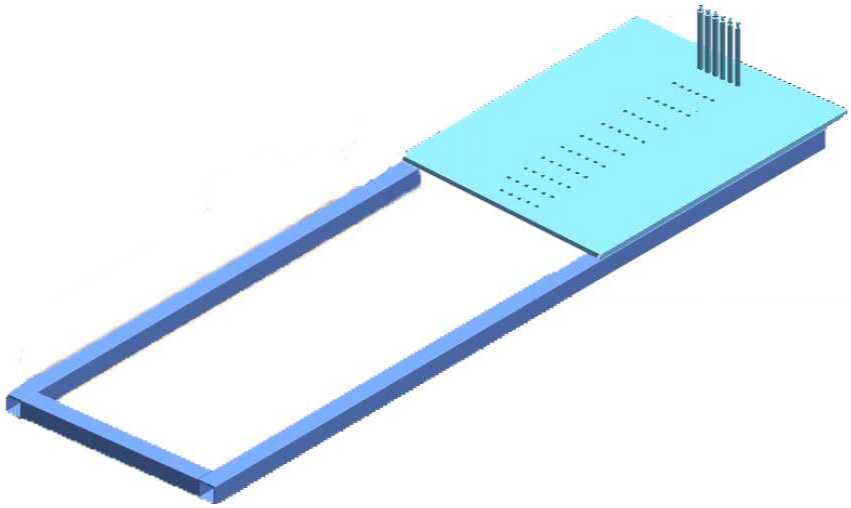


图 4　Team 部分固定系统

至于拨弦系统，Team Dare 使用了机械手臂模拟人类的手指来弹奏 6 根吉他弦，如图 5 所示[4]：

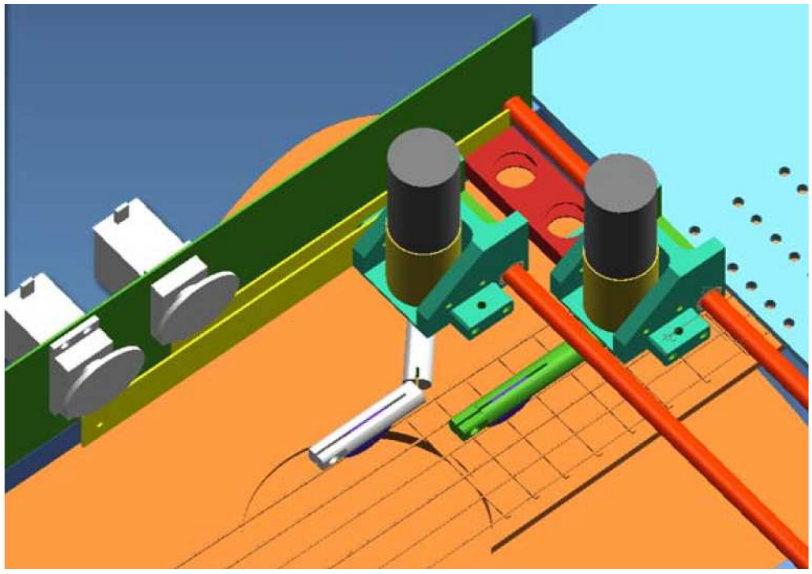


图 5　Team Dare 拨弦系统

音调控制系统是由 72 个小气缸构成，来模拟人手按压吉他品的动作，并且所有的动力都来自一个空气压缩机，并由气流控制，如下图[4]：

图 6 气压计

图 7 小气缸接线板

Team Dare 的吉他机器人可以弹奏出准确的音符，而且十分流畅。但是他所用的部分材料我们无法配备，能量要求高。

1.3 总结

LEMUR 和 Team Dare 的设计各有优缺点。LEMUR 的吉他机器人耗能少，充分利用马达；但仅仅 4 根弦无法弹奏所有的音符。Team Dare 的吉他机器人堪称完美，由于它使用了真吉他，所有的音符都可以涵盖，而且速度快，无延迟，可弹奏和弦，但是机械手臂对马达要求高，以我们现有的技术以及资金，很难买到或制作这样的机械手臂以及马达。最终，我们的基础设计大部分采用了 LEMUR 的设计，只是在控制系统那里选取了 Team Dare 采用的方法，即使用小气缸完成吉他品的按压。

第二章 系统总体设计方案

2.1 系统设计

我们的目标是制作出一款全新的，由机器控制的吉他。基于此，我们没有采用 Team Dare 那样的设计（使用真吉他），而是偏向于 LEMUR 的设计，使用铝合金作为底板，制作一个全新的吉他。对于硬件设计部分，我们的支撑系统主体是一块 100 X 6（cm）的铝板，上面事先刻好了 12 条不等间距品线，用来放置 12 个品；铝板上方固定有机玻璃，有机玻璃上固定 11 个小气缸负责按品。每个底座会配以一组电磁阀控制小气缸的按压（每组 10 个电磁阀）。额外的 3 个电磁阀将控制三个拨弦小气缸，保证拨弦动作顺利完成。4 组电磁阀和 2 个 PLC-Relay（一个 32 路继电器和一个 16 路继电器）相连。这两块继电器板是控制系统的核心，它们将串口传递过来的 16 进制信号转换成数字信号进而控制电磁阀的开合。软件部分是由 3 部分构成：MIDI 解析，窗口通信以及 3D 图形实时生成构成。首先 MIDI 解析算法将 MIDI 文件解析，得到音符信息，时间强度等有效信息，进而通过串口将信息传递到 PLC，同时生成 3D 实时图形。系统框图如下：

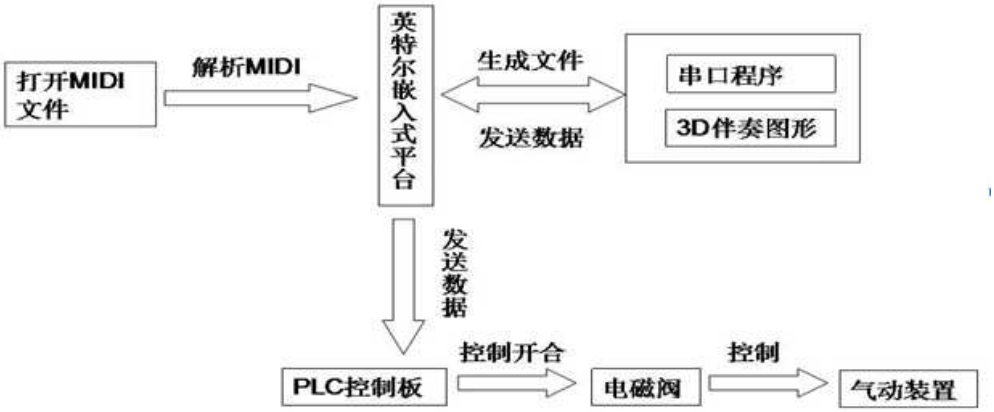


图 8　系统框图

2.1.1 3D 模型设计

硬件部分的最初设计使用 AutoCAD 完成， 依照我们的设计，整体 3D 模型如图 9 所示：

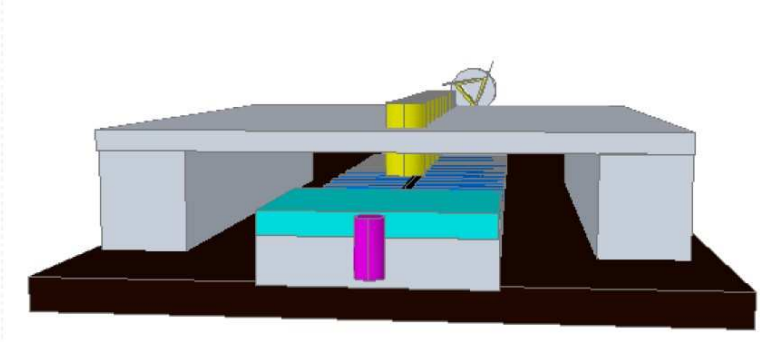


图 9　整体 3D 模型设计图

支撑系统如图 10 所示：

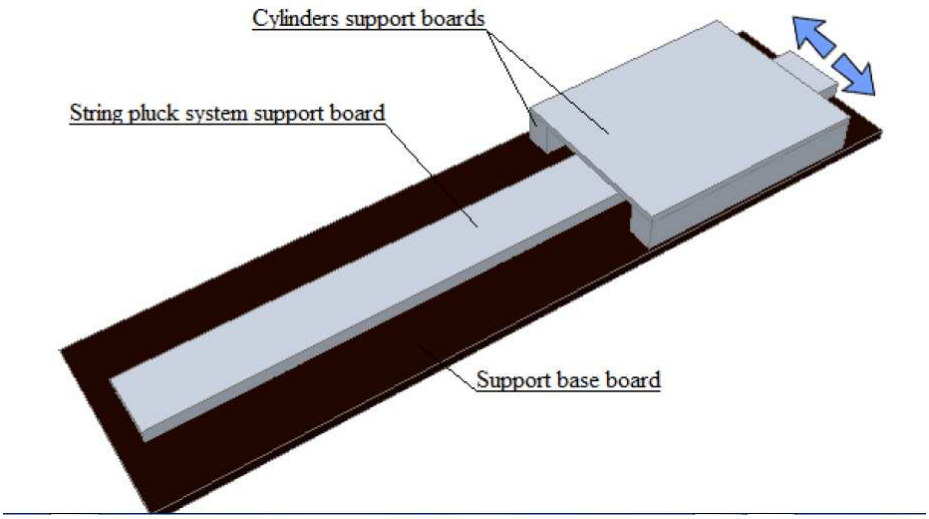


图 10　铝合金底座设计图

拨弦系统最初设计是由一个随动马达控制，每次波动 120°，但实际测试后，发现该设计存在缺陷：每次旋转角度不精准，导致拨弦有误差。后来我们改成了“直来直去式”，由小气缸直接控制拨弦，速度快，较为精准。如图 11 所示：

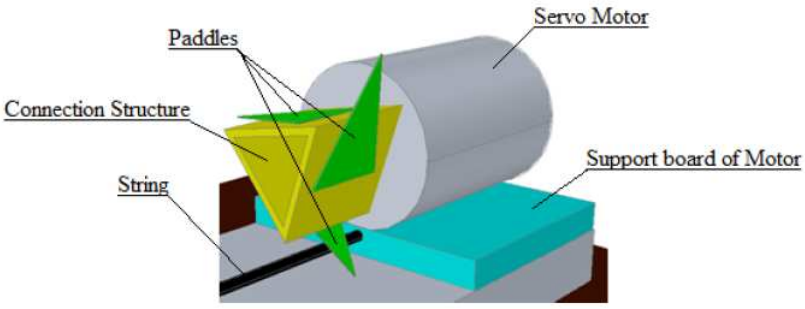


图 11 原始马达设计图

2.1.2 原始设计优劣

按照最初的设计，该系统易组装，可拆卸，方便携带，适用于流动表演；而且音准高，控制准确，能够较好地将解析好的 MIDI 文件弹奏出来。劣势是外观不美观，更接近机器。

而且目前成本较高，有轻微噪音（金属与金属，金属与有机玻璃碰撞声音）。由于经费以及硬件限制，我们的吉他机器人只有 3 根弦，但是可以弹奏绝大多数吉他曲，并且容易扩展，只要有符合条件的硬件，我们的吉他机器人就可以弹奏出任何吉他曲谱。

2.2 系统实现原理

本系统开发基于 C++语言环境，使用 Visual Studio 2010 开发，基于 Windows Platform SDK 的框架，充分应用 windows 下 API 编程，以及 Microsoft 提供的串口控件 MSCOMM。硬件部分由底座，电磁阀组以及 PLC 构成；软件部分分为 MIDI 解析，串口通信以及 3D 图形实时生成 3 部分。首先，MIDI 解析算法将 MIDI 文件解析成音符，时间，强度等组别，之后发送到串口处理部分，由串口控件将相关的 16 进制码发送到 PLC，并由 PLC 将 16 进制转换为电磁阀能够识别的数字型号，同时解析好的 MIDI 也会被发送到 3D 图像处理模块，实时生成动态音符供观众欣赏，以达到视觉体验。电磁阀接到数据后会进行相应的动作：开或合，同时控制小气缸（air cylinder）下压或缩回，以达到按品的目的。每个底座还会配备一个拨弦器，也是有小气缸控制，并连接电磁阀，原理同上。详细的各个部分原理会在稍后的章节一一介绍。流程图如下：

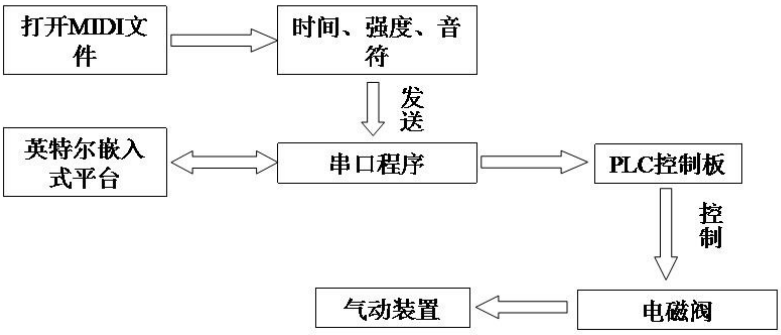


图 12　系统原理图

2.3 系统功能及特色

本系统实现吉他机器人弹奏 MIDI 吉他乐曲，同时生成 3D 实时图像，给观众即时的视觉体验。吉他机器人系统题材新颖，很巧妙的将计算机技术与吉他弹奏结合在一起，创造出一种全新的音乐体验方式。本系统应用领域广泛，有潜在的市场价值，例如，机场或饭店可以摆放一台吉他机器人，进行大量重复的演奏或伴奏工作，既可以减少人力劳动，又可以有效的降低成本。本系统经过完善后可以弹奏任意音符，任意乐曲，技术性良好，且稳定。

2.4 系统指标

吉他机器人主要有 4 个指标，分别对应 3 个子系统，分别是 MIDI 解析，串口通信以及音调控制。首先是 MIDI 解析，我们的 MIDI 解析算法将 MIDI 分为头块，与若干轨道块，轨道块部分又会分为若干事件（详细内容将在第五章介绍）。对 MIDI 解析算法的要求是：提取到音符间隔时间，音符名，以及每个音符持续时间；串口通信需要将准确的 MIDI 文件信息以 16 进制的形式传递到 PLC（继电器板），并由继电器板进行 16 进制与数字信号的转换。最后是音调控制，机器弹出的音必须符合 MIDI 原来的音准，噪音尽可能降低。

第三章 硬件框图

首先我们使用一台空气压缩机为整个系统提供原始动力，并使用口径 M6 的气管将电磁阀和启动装置分别和空气压缩机相连。电磁阀和 2 块继电器电路板（PLC）通过导线以及面包板相连，使得电路接通；Intel 嵌入式平台和继电器板通过串口相连，保证数据传输通畅。

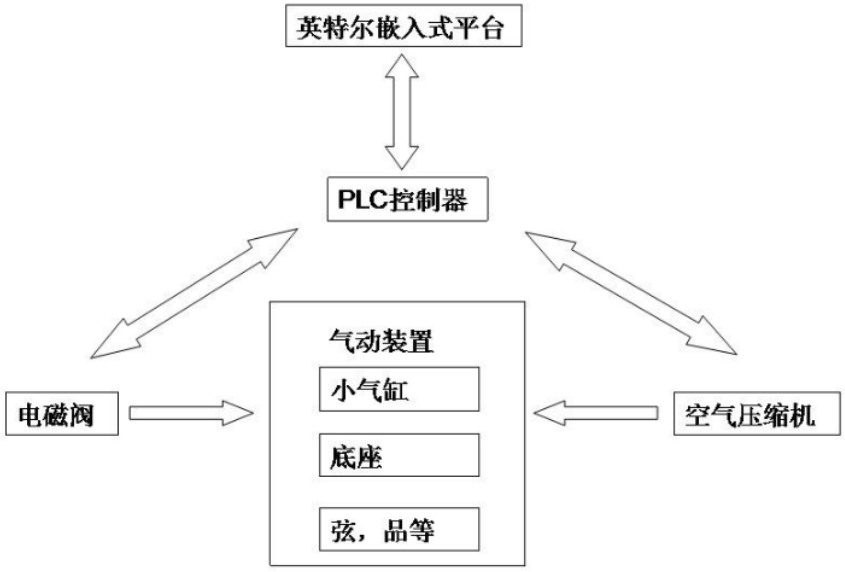


图 13　硬件原理图

3.1 硬件配备

我们使用到的硬件设备有一台空气压缩机，一块 32 路继电器控制板，一块 16 路继电器控制板，36 个小气缸(1.0mpa,10.2kgf/cm2)，3 块 100 X 6(cm)的铝板，36 个电磁阀(H5221-08D,0.15-0.8MPA)，3 根吉他弦，36 个吉他品等硬件。下面详细介绍部分硬件的原理及使用。

3.2 硬件原理

吉他机器人的动力源是一台空气压缩机，它主要提供小气管大气压力以保证其能准确按压吉他品。有压力调节旋钮，可调节空气压强，与小气管匹配。如图 14 所示;

图 14　空气压缩机实物图

本系统使用 36 个小气缸完成对品的按压及拨弦动作。小气缸的工作原理如下:它分为两个状态，弹出和缩回。当 port 1 的气压大于 port 2 的气压的时候小气缸会压下；当 port 2 的气压大于 port 1 的气压的时候小气缸便会缩回，如图 15 所示：

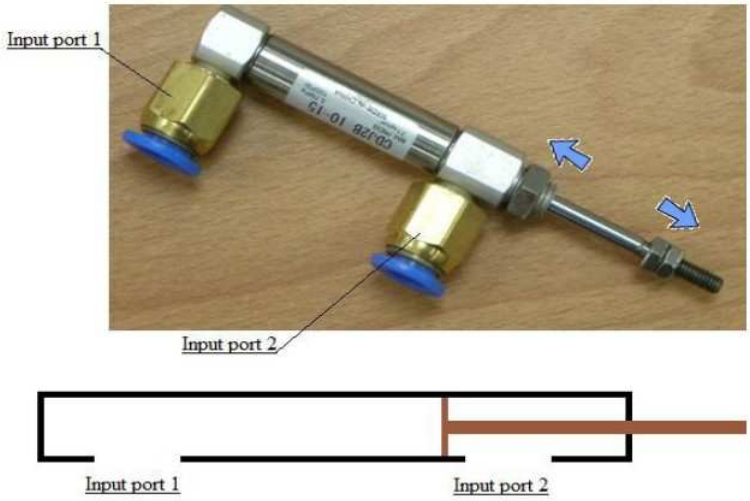


图 15　小气缸实物及原理图

与小气缸直接相连的是一组电磁阀，电磁阀控制小气缸（air cylinder）的弹出和缩回。它有一个输入端，两个输出端和两个出气端。电磁阀有开和关两个状态，当电磁阀打开的时候，输入端会和输出端 1 相连通，而输出端 2 和出气端 2 相连通；当电磁阀关闭的时候，输入端会和输出端 2 相连通，而输出端 1 会和出气端 1 相连通。在这样的原理下，当电磁阀通电时，小气缸（air cylinder）port1 的气压会大于 port 2 的气压，小气缸（air cylinder）便会下压；而当电磁阀关闭的时候，port 2 的气压又会大于 port1 的气压，这样小气缸（air cylinder）就会缩回了。原理图如图 16 所示：

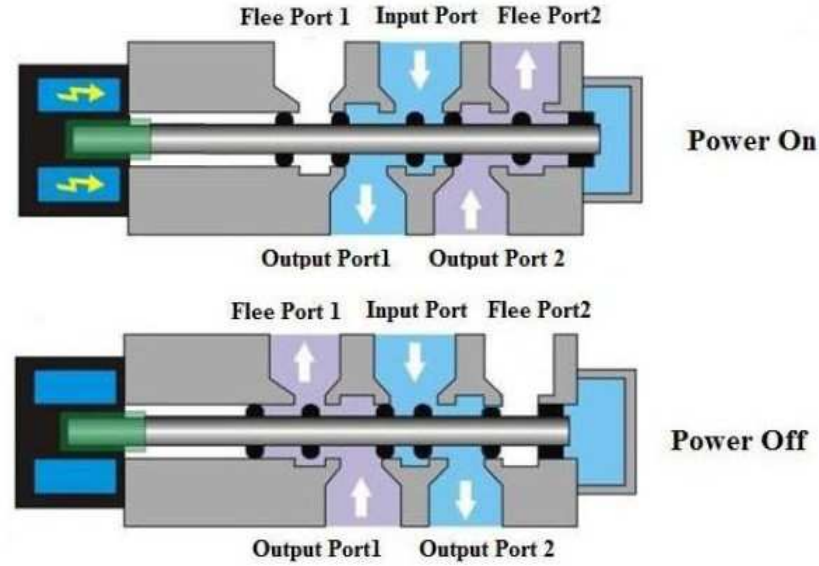


图 16　电磁阀原理图

核心的控制元件是两块继电器板（PLC），分别是 16 路继电器板，和 32 路继电器板。它们的作用是将串口传来的 16 进制信号转换成数字信号，方便电磁阀识别。继电器板上每个继电器配备一个 LED 灯，用于识别。16 路继电器板最小反应时间位 210 毫秒，32 路继电器板最小反应时间位 130 毫秒，如图 17,18 所示：

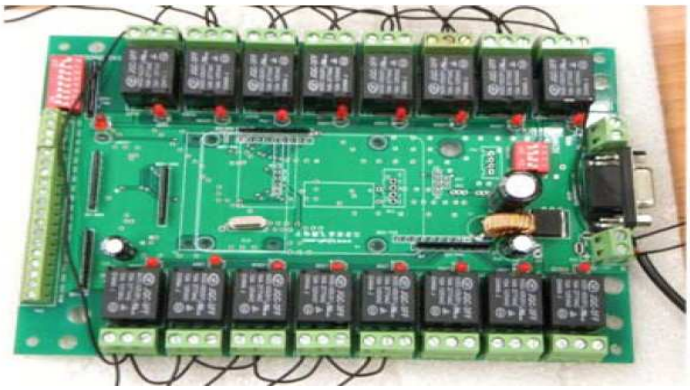


图 17　　16 路继电器板

图 18　　32 路继电器板

第四章 软件流程

本系统软件部分详分为 3 部分:MIDI 解析部分,串口通信部分和 3D 图形实时生成部分。MIDI 解析程序首先将 MIDI 音乐文件解析，之后串口程序将信息发送到 PLC，同时生成 3D 实时图形，软件流程图如下：

图 19 软件流程图

4.1 解析 MIDI

我们设计的吉他机器人使用 MIDI 文件作为音乐源，而要弹奏 MIDI 就需要将 MIDI 音乐文件解析成机器能够识别的指令来控制音乐的弹奏。下面我们就来详细介绍什么是 MIDI，我们为什么选用 MIDI 以及如何设计解析算法。

4.1.1 MIDI 简介

MIDI 是 Musical Instrument Digital Interface 的简称，意为音乐设备数字接口。它是一种电子乐器之间以及电子乐器与电脑之间的统一交流协议。MIDI 是数字技术带来的又一便利工具，和音频不同的是，它不记录声音，只是一种控制电子乐器和录音室设备的数字协议，换句话说，它向电子乐器和设备发出命令，让后者发出特定的声音。 世界上有无数软硬件厂商生产软硬件设备和音乐软件，这些设备和软件必须要有一个统一的 MIDI 标准来实现相互连接，比如说演奏电子合成器的一个键，从合成器 MIDI 输出口就发出了一个 MIDI 事件，这个事件包括一系列的 MIDI 信息通过 MIDI 线传送给其他设备。 音乐软件可以录制下 MIDI 事件编辑它们和回放。MIDI 文件格式也称为频率合成音频文件，它是由频率合成音频文件中储存的码元（控制信息）控制声卡上的音频合成器输出音频信号的编码和时间，简单的说，就是本身不能发生，需要经 A D 驱动声卡转换发声。所以说 MIDI 并不是一个实在的东西，而是一个国际通用的标准接口。通过它，各种 MIDI 设备都可以准确传送 MIDI 信息[5]。

4.1.2 选用 MIDI 的原因

第一，MIDI 指定了统一的数字化乐器接口规范。1983 年 8 月，YAMAHA，ROLAND,KAWAI 等著名电子乐器制造商联合指定了统一的数字化乐器接口规范，名为 MIDI1.0 技术规范[5]。这意味着我们可以用同一种方式解析 MIDI 文件，并将其应用在不同乐器上，大大减少了重复性劳动，适用范围广。

第二，MIDI 格式不包含流媒体信息，而是用一种语言将乐器声音记录下来，使得计算机可以轻易的读取它。

第三，MIDI 文件所需存储空间小。MIDI 文件的扩展名是*.mid,他和另外一种计算机中常用的声音波形文件（*.wav 文件）有什么不同呢？表面上两种文件都可以产生音响效果或音乐，但本质是完全不同的。普通的声音文件（*.wav 文件）是计算机直接把声音信号的模拟信号经过取样--量化处理，变成与声音波形对应的数字信号，记录在计算机的存储介质中。

通常声音文件都比较大，如记录一分钟的声音（立体声，CD 音质），大概需要 10.5M 存储空间。一首几分钟的歌曲甚至需要几十兆的硬盘存储空间，一张 CD 光盘只能容纳十几首歌曲。为了减少声音文件存储的空间，近年来计算机技术上采用了压缩技术，将声音文件经过处理，在不影响播放质量的前提下，把文件大小压缩到原来的 10~12 分之一，这就是近年来流行的 MP3 文件格式。而 MIDI 文件则不是直接记录乐器的发音，而是记录了演奏乐器的各种信息或指令，如用哪种乐器，什么时候按下哪个键位，例如怎样等等，至于播放发出的声音，是通过软件或者音源转换而成的。因此，MIDI 文件通常比声音文件小得多，一首乐曲只有十几 K 或者几十 K，只有声音文件的千分之一左右，便于存储和携带[6]。

4.1.3 MIDI 的构成

一个 MIDI 文件基本上是由两个部分组成，头块（ Header Chunk）和轨道块（Track Chunk）。头块用来描述文件格式、轨道块个数等内容。轨道可以想象为一个大型多音轨录音机，可以为某种声音，乐谱，乐器或者任何需要的东西分配一个轨道。

每个 Chunk（块）的组成是特定的，如表 1：

表 1　每个 Chunk 的组成

类型	长度	数据
4 字节（ASCII）	4 字节（Byte）	特定

有两种类型的Chunks:

Header Chunk：标志为“MThd”；

Track Chunks : 标志为“MTrk”；（紧接着头块，一个或多个）

头块是形如：4D 54 68 64 00 00 00 06 ff ff nn nn dd dd这样的结构，4D 54 68 64代表ASCII码MThd；00 00 00 06代表大小（总是6，大于则被忽略）；ff ff是文件类型，有以下3中文件类型：

0----单轨

1---- 多轨，同步

2---- 多轨，异步

单轨道顾名思义，只有一个音轨；同步多音轨则是曲目都是垂直同步的，换言之，他们都在同一时间开始，所以可以代表一首歌的不同部分；异步多音轨意味着曲目不在同一时间启动，并可以完全异步。nn nn代表了轨道块的个数，dd dd代表每四分音符tick的个数[6]。简单来说，头块数据部分包含3个16位信息，描述了MIDI格式，轨道数量以及时间设置。长度为6字节。无论何种软件必须遵守这个原则，即使大于预期的，任何意料之外的数据都会被忽略。

表2　头块数据格式

Header Chunk				
Chunk 类型	长度	数据（6字节）		
4字节（ASCII）	4字节（32位二进制数）	16位	16位	16位
MThd	<长度>	<格式>	<tracks>	<division>

MIDI文件格式是一个16位二进制数有效格式：

0----单轨

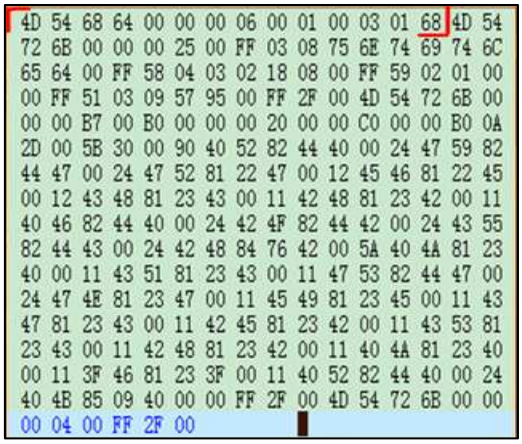
1---- 多轨，同步

2---- 多轨，异步

着重介绍division： 这个是定义在文件中每个4分音符delta-time节奏数，也是一个16位二进制数，分为两种格式，如下表。

表3 division 格式

位	15	14....8	7.....0
<division>	0	1/4音符tick数	
	1	帧/秒	Ticks/帧



示例一：

4D 54 68 64： MThd；

00 00 00 06: 头块大小；

00 01： 同步多音轨；

00 03： 文件中共3个音轨，

1个头块加2个轨道块；

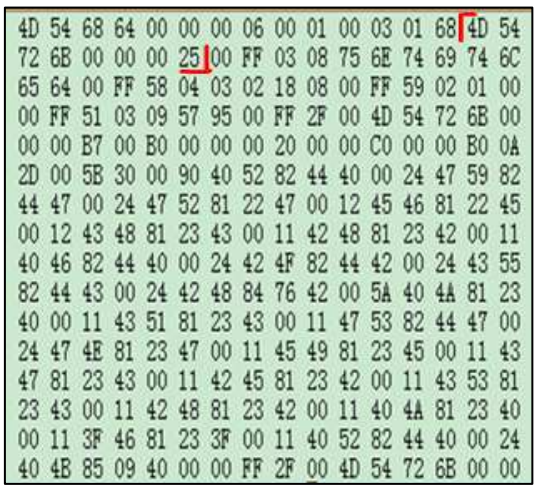
01 68： 360 ticks；

【0168（HEX）=360（DEC）】

轨道块开头是形如 4D 54 72 6B XX XX XX XX 的格式。4D 54 72 6B 代表 MTrk；后面的 4 个字节代表该轨道块长度。轨道块数据部分由一对和多对<delta-time><event>组成。<delta-time>代表间隔时间，0 是有效地间隔时间。

表4 轨道块格式

Track Chunk		
类型	长度	数据
4 字节（ASCII）	4 字节（32 位二进制数）	<--长度-->
MYrk	<长度>	<delta-time><event>



示例二(轨道块头部)：

4D 54 72 6B：表示 MTrk；

00 00 00 25：表示长度；

MIDI 事件

MIDI 包含 3 个不同的事件：分别是 midi 事件，sysex 事件和 meta 事件。我们主要通过分析 midi 事件完成对 MIDI 音乐文件的解析。

4.1.4 解析算法

解析算法首先解析 midi 音乐文件头部，分析音轨；之后将根据系统的宏定义去匹配各个事件，分门别类进行解析，并将持续时间，音符，强度等有效信息写入文件。

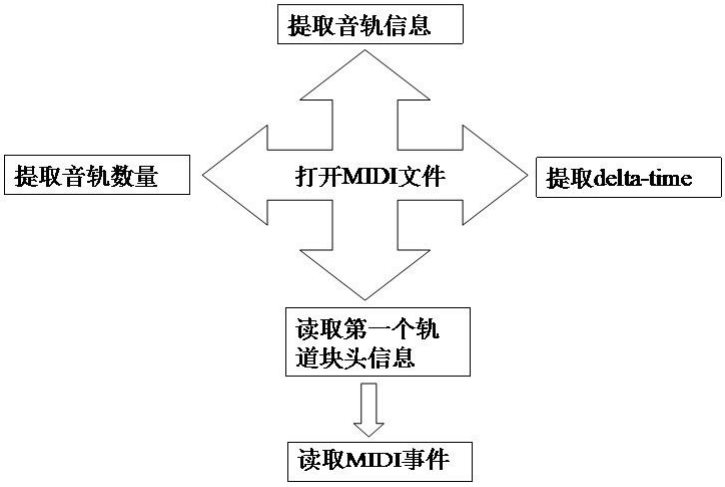


图 20　入口函数流程图

图 20 展示了分析文件的入口函数，负责打开 MIDI 文件及判断。通过读取头块分析文件结构，例如同步，异步，旋律设置，轨道块数并且读取第一个轨道块头信息，之后进入循环，读取数据并分析,由于 delta-time 的组成不是确定的，是可变的数据，最多可以用 4 个字节来表示，最少用一个字节（为了节省空间）。这就需要函数在读取出来时判断由几个字节组成，然后进行处理。用到的函数有：

R_DeltaTime_Two(int a, int b)　　将两个字节的数据转化为整数
R_DeltaTime_Three(int a, int b, int c)　将三个字节的数据转化为整数
R_DeltaTime_Four(int a, int b, int c, int d)　将四个字节的数据转化为整数

对于MIDI 事件的分析是本软件中最重要的一部分，由于MIDI 事件分为多种，有系统消息事件、Meta-event 和控制事件。要分别对这些事件进行处理，在这里只对常用的信息进行处理，因为有些信息是不会再MIDI 文件中出现，上面提到的就是处理用到的所有函数，我们通过判断一个字节的高四位来区分是属于什么信息，然后再把分类后的信息做具体分析，这样就可以把所有事件都分析出来。在处理可变信息的时候采用链表的形式存储，比如文字信息，歌词等等，这些信息以一定的标志开始和结束，则把数据存储到一条链表中，处理完之后再加到结果字符串中。下图是MIDI事件处理流程[8]：

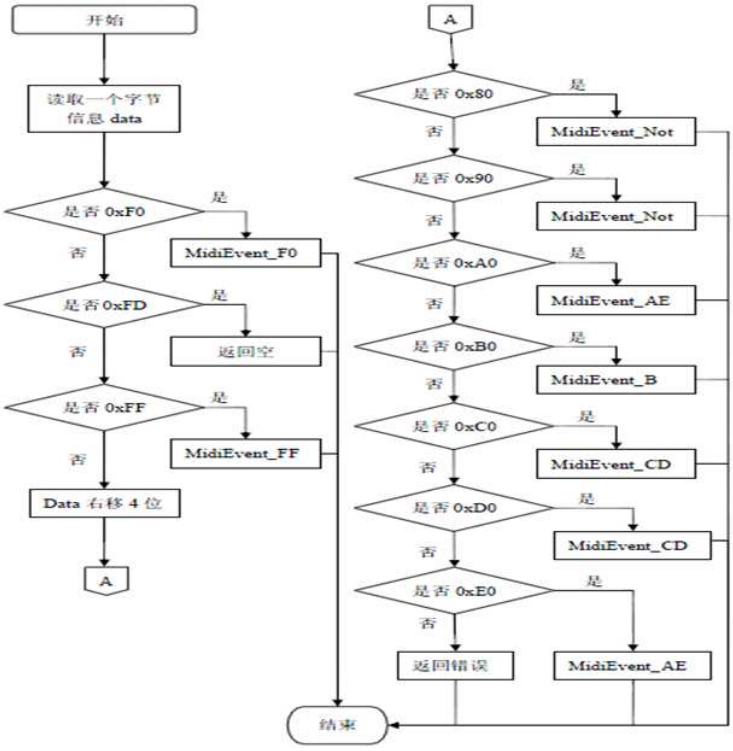


图21　MIDI事件流程图

结果分析（存放于文本文件）：

每行记录3个数据，第一个是delta-time，表示音符之间的间隔时间，0为有效值，单位为毫秒；第二个和第三个数据是一个3维数组的前两维，分别代表弦和品，例如0；0代表第一弦，第一品，0；4代表第一弦，第五品。这种对应关系是在程序中定义的，midi解析后，会将各种音符转换成相对应的10进制数值，这些不同的十进制数值又分别对应吉他的相应弦和品。下图为分析结果，例如第一行：0；0；0，第一个0代表该音符前没有停顿时间；第二个0代表拨动第一弦；第三个0代表按压第一品。

0; 0; 0
0; 0; 0
30; 4; 1
30; 4; 1
0; 4; 3
30; 4; 3
0; 5; 0
30; 5; 0

图22　解析结果

4.2 串口通信

本系统使用 RS232 接口进行通信。硬件部分主要用到两块继电器板，一块是 32 路继电器板，另一块是 16 路继电器板。32 路继电器板负责控制小气缸的按压，而 16 路继电器板则负责拨弦动作。不同的 PLC 的通信协议是有差异的，下面分别介绍这两块板子的通信协议。

4.2.1 通信协议

16路继电器板指令包含10个字节，发送16进制数据，基本格式为帧头、地址、功能码、数据、帧尾、校验。如图23：

图 23　16 路继电器板通信指令

CC DD代表指令头部，标志着控制命令的输入，即将控制电路板；A0/B1是模块控制数据，标记读与写；接下来的00代表地址模块（自处地址为0）；之后的两个字节是数据字节，为16进制，转换为2进制则代表继电器的控制数据位，为1则吸合某通道的继电器，为0则关闭该继电器。例如：数据为0006,其二进制为0000 0000 0000 0110 ，则表示数据为第2、3 路继电器吸合，其他都断开（注意此数据只有在后面的通道选择位为1 时，才有效）；再后面两位是控制有效字节，此两个字节的数据，分别表示，16 个继电器（DO），有效选择中位，相应的位为1，则选中对应的继电器动作例如：CH1=FFH, CH0=FFH，则表示此次要控制的继电器是16 路，而这16 路继电器的具体动作，需要根据前面的D1D0 来进行。[9]

表5　16路继电器板通信协议

数据	D1								D0								CE1								CE0							
16进制	FF								FF								05								F0							
2进制	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	1	1	1	1	1	0	0	0	0
	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√						√		√	√	√	√	√				
表示意义	数据位表示所有的继电器都断开																选中的继电器为第 11、9、8、7、6、5 路															
执行结果：　断开第 11、9、8、7、6、5 路继电器，其他继电器保持上次的动作不变																																

校验字节：9D 3A

校验和字节1 = 除去帧头以后的6 个字节累加和

例如：9D = A1 + 00 + FF + FF+ FF+ FF

校验和字节2 = 除去帧头以后的6 个字节以及校验字节1 的累加和

例如：3A = A1 + 00 + FF + FF+ FF+ FF+9D

32 路继电器板通信协议指令是由 7 个字节构成，同样发送 16 进制数据，第一个字节是0x10，表示命令，第二个字节是 0，（此字节是扩展用，此板子没有用到，可不用管），第三到第六字节，发送的数据，单片机根据此字节来执行操作。第七个字节是结束码。每通过上位机执行一次吸合和断开操作，都会发送 7 个字节，每一次都会更新数据字节中一个字节的某一位，来传达给单片机，执行操作[10] 。

表 6　32 路继电器板通信协议

功能	地址（扩展用）	数据内容	方向	结束码
实时控制	10H	D0(1-8 路)、D1（9-16 路）、D2（17-24 路）、D3（25-32 路） 0 表示开，1 表示关	向下	50H

其中有一点和 16 路继电器不同，32 路继电器的 0 代表接通，1 代表关闭。例如打开第一路继电器，命令为 10 00 FE FF FF FF 50.

4.2.2 程序原理

串口通信程序基于 VS2010的 MFC 对话框程序，使用了 MSCOMM 串口通信控件。它是 Microsoft 公司提供的简化 Windows 下串行通信编程的 ActiveX 控件，它为应用程序提供了通过串行接口收发数据的简便方法，使得程序员不必花大量时间去了解较为复杂的 API 函数，而且在 VC、VB、Delphi 等语言中均可使用，十分方便。

Intel 嵌入式平台提供了2个串口，所以我们使用了2个串口控件，分别控制16路和32路继电器板。下面是其常用属性：

CommPort　　设置并返回通讯端口号。

Settings　　以字符串的形式设置并返回波特率、奇偶校验、数据位、停止位。

PortOpen　　设置并返回通讯端口的状态。也可以打开和关闭端口。

Input　　从接收缓冲区返回和删除字符。

Output　　向传输缓冲区写一个字符串。

首先，我们在程序中事先定义好继电器与吉他品的对应关系，当读取文件时，读到相应的品位时，就触发对应的继电器，进而实现对整个系统的控制。

程序运行效果图：

图24　串口程序运行效果图

4.3 实时图形

3D 图像实时生成程序使用 MIDI 解析出来的参数通过分析音符以及出现的时间，并且使用 OpenGL 动态生成 3D 动画。

4.3.1 音乐可视化

音乐可视化，是指一种以视觉为核心，以音乐为载体，以大众为诉求对象，借助多种新媒体技术等传播媒介，通过画面、影像来诠释音乐内容的、视听结合的大众化传播方式。它能为理解、分析和比较音乐艺术作品形态的表现力和内外部结构提供的一种直观视觉呈现的技术[10] 。

很多时候，音乐并不一定能为人所理解，但是当音乐配上合适的画面的时候，人能够更深切地理解以及体会音乐，另外，这更是另外一种艺术创作形式，通过视觉元素不同的构图、色彩影调、场景、音乐元素与音乐元素的音高，节奏，曲调，风格等形成的互动而把音乐的美感同化为视觉的感受。因音乐本身的自由性导致可产生的动画也是十分丰富的，其灵活的形式使得它具有十分高的美学价值。

4.3.2 OpenGL 及 3DSMAX

OpenGL，全称为 Open Graphics Library，是一个跨平台和编程语言的强大的 2D/3D 底层图形库并为诸如微软、IBM 等业界主导公司所用。OpenGL 虽然是底层图形库，可是我们可以通过一些转换程序很方便的将诸如 AutoCAD、3DSMAX 等 3D 图像设计软件的 DXF 和 3DS 模型文件转换成 OpenGL 的定点数组。另外 OpenGL 也可以很紧密地和 Visual C++结合以便实现我们的实时动画显示。OpenGL 能提供强大的点线面以及复杂曲线和曲面的绘制函数， 基本变换如平移、旋转、变比、镜像， 投影变换如平行投影和透视投影，RGBA 模式或者颜色索引（Color Index），材质光照如辐射光（Emitted Light）、环境光（Ambient Light）、漫反射光（Diffuse Light）和镜面光（Specular Light），能提供逼真物体细节的纹理映射功能，位图显示和图象增强图象功能除了基本的拷贝和像素读写外，还提供融合（Blending）、反走样（Antialiasing）和雾（fog）的特殊图象效果处理，双缓存动画（Double Buffering）以提供流畅的动画功能，和深度暗示（Depth Cue）、运动模糊（Motion Blur）等特殊效果。另外我们可以使用 GLU\GLUT\GLUI\GLEW\GLEE 等库来实现原本 OpenGL 没有的功能[13]。

4.3.3 模型制作及操作

Autodesk 3D Studio Max，也称为 3ds Max 或 3DMAX，它是一款由 Kinetix 公司出品，专为建模、动画而设计的三维制作软件。它相对一些专业软件来说对 PC 性能的要求比较低，界面设计易用，并且具有强大的插件以及脚本控制功能。功能强大的 3DMAX 可以制作出工业级的动画[12]。由于 3DMAX 能较容易地建立复杂模型，但是不容易通过程序控制。由此我们先在 3DMAX 中进行模型的编辑，随后再在程序中导入模型数据。下图中的音符模型由基本模型通过扭曲、变形、点面编辑之后得到：

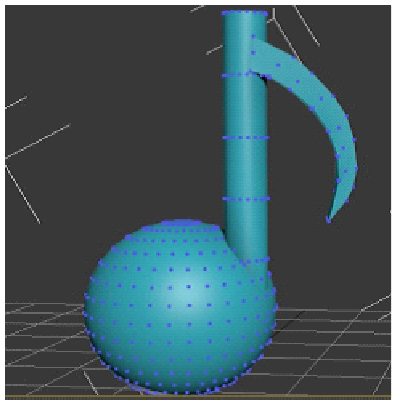


图 25　3dsmax 下的音符模型

4.3.4 实时图形程序

本程序设计运行在 Windows 7 平台上。 OpenGL 程序建立在 Windows 程序框架上。

Windows 框架：

1. WinMain() 函数为程序入口，并且初始化 Windows 窗口。

2. MsgProc() 为窗口回调函数，MsgProc() 基于事件机制负责在运行周期中对如时间、鼠标点击、窗口变化、键盘按键等事件的进行响应。

3. SetDCPixelFormat() 对显示设备进行初始化设置：检测设备支持 OpenGL，设置显示相关的参数。

OpenGL 框架:

1. InitializeRC() 对 OpenGL 进行初始化设置，添加环境光照等。

2. DrawScene() OpenGL 主函数，进行模型技术以及场景渲染。

3. SelectMaterial() 根据索引对模型进行材质映射。

其中模型数据包括以下内容：

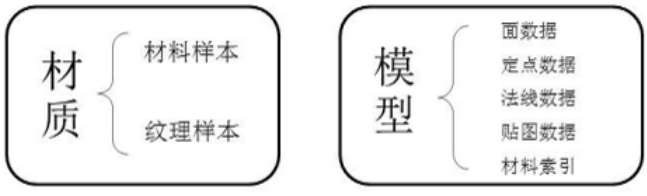


图 26　模型数据储存内容

OpenGL 的渲染流程

1.加载模型数据。

2.绘制模型轮廓：通过 OpenGL 点、线、多边形的绘制函数实现。

3.模型变换：通过投影变换、几何变换、裁剪变换、视口变换获得所需视角与模型。

4.着色：使用颜色索引模式对模型进行着色。

5.纹理映射：使用材料以及纹理样本对模型进行映射，可使三维模型更贴近理想材质。
6.其他特效：如光照、反走样等。
7.交互通讯与动画：OpenGL 提供了消息响应函数，由此实现获取 midi 音符信息即时产生新动画效果。

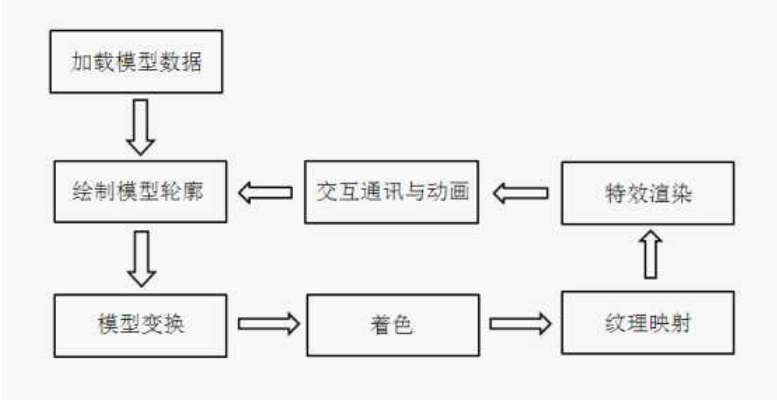


图 27　渲染流程

以模型为视图，然后加载模型的顶点数组并移动视图到一个合适的位置然后进行旋转以及放大缩小操作。同时通过捕捉信号来进行创建，调整生成窗口大小，根据时间轴调整动画参数，销毁等功能。

下图展示了单个音符模型导入后，在 OPENGL 程序下显示的效果。我们后期会制作多个音符模型。这些不同的音符模型会根据输入 MIDI 文件的不同进行音符的显示，渲染，跳跃及旋转，以可视化的方式展现出音乐。

图 28　音符效果图

第五章 系统测试

5.1 硬件系统测试

硬件系统测试分为 3 部分，空气压缩机测试，继电器及电磁阀测试和小气缸测试。在分别测试后，还应进行系统的整体测试。

5.1.1 空气压缩机测试

测试组件：一台空气压缩机。

测试指标：测试空气压缩机提供的压强是否符合本系统的需求，并找出最小驱动本系统启动装置的空气压强。

测试方案：将空气压缩机通过直径为 6 毫米的气管与 12 个小气缸相连，接通空气压缩机以及 PLC 控制模块，并不断调节提供的空气压强，测试小气缸是否能顺利下压。

测试结果：当空气压缩机的压强变化在 10~30kg/cm^2 的范围时，小气缸可以顺利的实现下压动作，力度均匀。但当空气压缩机的压强小于 10kg/cm^2 时，部分小气缸不能实现下压动作，其余小气缸虽然能够下压但力度较小，并且速度减慢；当空气压缩机的压强小于 5kg/cm^2 时，小气缸大部分不能下压，其余速度明显变慢，力度很小，无法完成按品的动作。

表 7　空气压缩机测试结果

压力/kg/cm^2	完成按压的品数	速度	力度
20	12	快，符合标准	大
8	8	较慢，较为符合	较小
4	2	慢，不符合标准	小

结论：空气压缩机的压强需保持在 10~30kg/cm^2 范围内。

5.1.2 继电器及电磁阀模块测试

测试组件：一块 16 路继电器，一块 32 路继电器，以及 1 组电磁阀。

测试指标：是否能够流畅，正确的连续打开继电器模块，而不会出现停滞或部分 LED 灯不亮的现象。将电磁阀和继电器板接通到导线后，观察电磁阀是否可以正确的开合，是否与继电器模块相对应。

测试方案：首先使用 PLC 串口通信调试助手测试继电器模块，确保继电器模块是能够正常工作的。之后需要测试电磁阀是否能够正常工作，首先接通电源，将电磁阀的正负极接到电源上，观察电磁阀 LED 灯是否可以打亮，里面的电磁铁是否吸合。接下来使用我们自己编写的 C++串口通信测试程序连续打开继电器模块，观察继电器是否可以正常工作；连接电磁阀后，是否可以正常工作。

测试结果：使用 PLC 串口通信调试助手测试每个继电器模块正常；单独每个测试电磁阀结果正常。但当我们使用自己的串口通信测试程序的时候，继电器没有反应，所以我们进入了 debug 模式测试程序，发现在 debug 模式下继电器模块可以正常打开。于是我们修改了测试程序部分代码，使其暂停一段时间，发现继电器模块可以正常工作。连接电磁阀后，继电器和电磁阀都可以正常工作。

表 8　继电器、电磁阀模块测试

继电器/电磁阀模块	间隔时间（毫秒）	继电器亮灯速度	观测亮灯个数
16 路继电器	160	极快	无
16 路继电器	210	较快	都亮
32 路继电器	100	极快	无
32 路继电器	130	较快	都亮

结论：需在程序中控制间隔时间，避免继电器板反应不过来。16 路继电器板需要至少 0.2 秒的反应时间；32 路的继电器板需要至少 0.13 秒的反应时间。

5.1.3 小气缸模块测试

测试组件：小气缸。

测试指标：小气缸能否顺利下压，力度及速度是否符合系统标准，最重要的是当小气缸下压时，与有机玻璃撞击产生的噪音不影响人们欣赏音乐。

测试方案：首先分别测试小气缸能否正常工作，连接空气压缩机测试即可。对于噪音测试，设置对照组，第一组在有机玻璃和小气缸之间不加减噪垫圈，第二组加上厚度为 1 毫米的垫圈，第三组加上厚度为 2 毫米的垫圈，第四组加上厚度为 4 毫米的垫圈。接通空气压缩机，PLC 控制板以及电磁阀，测试小气缸下压时产生的噪音。

测试结果：小气缸都能正常下压。当小气缸不加垫圈时，其与有机玻璃撞击的声音很大，严重影响了音乐的声音。当加上厚度为 1 毫米的垫圈时，噪音明显减少，音乐声音洪亮；当加上厚度为 2 毫米的垫圈时，减噪效果与厚度为 1 毫米的垫圈相同，但使得小气缸不易固定；当加上厚度为 4 毫米的垫圈时无法测试，因为厚度太高，使得小气缸几乎与有机玻璃脱离。

表 9 小气缸模块测试

垫圈厚度（毫米）	减噪效果	可固定性
0	无，噪音很大	可以
1	较好，噪音明显减少	可以
2	较好，噪音明显减少	可以
4	无法测试	不可以

结论：最终，我们使用了厚度为 1 毫米的垫圈，减噪效果不错，而且小气缸易于固定。

5.1.4 整体测试

测试组件：空气压缩机、继电器以及电磁阀模块、小气缸、吉他弦、吉他品。

测试指标：整个系统每部分能够正常运作，最终能够弹奏出一首完整的吉他曲。

测试方案：将 2 块继电器板通过 2 个 COM 口与 Intel 嵌入式平台相连，同时连接一组电磁阀，并将空气压缩机、小气缸、电磁阀通过气管相连。使用吉他调音器将吉他弦的音调至标准音，将吉他品按照标准安放在底座上。

测试结果：系统各部分工作正常。整体效果较好，测试者能够清楚地听出音乐的曲调，虽然弹奏伴随着些许噪音，但并不影响测试者欣赏音乐。

表 10　整体系统测试

测试模块	测试结果
空气压缩机	正常
继电器以及电磁阀模块	正常
小气缸	正常
整体系统	正常

5.2 软件系统测试

软件部分测试主要分为 3 个部分，MIDI 解析部分，串口通信部分以及 3D 伴奏图形实时生成部分。

5.2.1 MIDI 解析测试

测试模块：MIDI 解析模块。

测试指标：MIDI 解析模块能够正确解析 MIDI 文件，并将正确的结果写入文件中。

测试方案：首先使用 “Guitar Pro”这款软件写出一份测试 MIDI，导出成（.mid）格式再将该 MIDI 音乐文件导入我们的 MIDI 解析模块，检查 MIDI 解析模块生成的解析结果是否与“Guitar Pro”中我们自己定义的结果相同。

测试结果：结果与预期相同。在“Guitar Pro”中，我们直接定义的是弦和品，而解析文件中的数值会比“Guitar Pro”中的小 1。

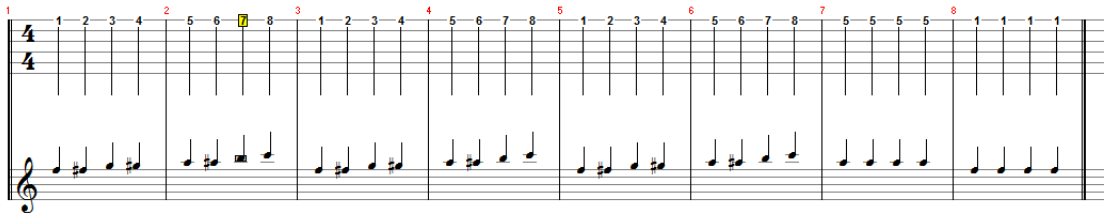


图 28　Guitar Pro 中的 MIDI

其中，六条线代表吉他的六线谱，1、2、3、4 等数值代表该弦上的品位。上图代表弹奏一弦的 1-7 品。下面是使用我们的算法解析的结果：

0; 0; 0
0; 1; 0
0; 0; 2
0; 0; 3
0; 0; 4
0; 0; 5
0; 0; 7
0; 0; 0
0; 1; 0
0; 0; 2
0; 0; 3
0; 0; 4
0; 0; 5
0; 0; 7
0; 0; 0

每行的第三位代表品位，如 0-7 代表着吉他的 1-8 品，结果正确。

5.2.2 串口通信测试

测试模块：串口通信组件。

测试指标：该串口通信程序能够发出正确的数据，两块继电器板能否正确被控制。

测试方案:将 Intel 嵌入式平台通过 2 个 COM 口与 2 块继电器模块相连，运行程序，观测是否能够正确操作继电器。

测试结果：正常运行，继电器板能够正确打亮。（测试数据与硬件相同）

5.2.3 3D 伴奏图形测试

测试模块：消息队列，模型参数，3D 动态图形

测试指标：消息队列能否接收正确参数，模型参数计算是否正确，3D 动态图形生成是否随音乐同步变化。

测试方案：进入 debug 调试模式，打开 MIDI 音乐文件，测试消息队列接收到的信息，检查模型参数的正确性，观察 3D 伴奏图形变换。

测试结果：消息队列接收了正确参数，模型参数计算准确，3D 伴奏图形随音乐同步变化。

第六章 结论

我们的吉他机器人实现了所有预期的功能，能够通过 Intel 嵌入式平台进行与 32 路，16 路继电器板的通信；可以正确操作小气缸的按压及拨弦，最重要的是，我们的吉他机器人能够弹奏出大部分吉他乐曲，音准度高。同时，我们的系统还配以 3D 实时伴奏图形，当机器弹奏音乐时，配以跟随音乐变化的 3D 画面让观众欣赏，既有听觉效果，又配以视觉享受，让欣赏者拥有最好，最美的体验。

由于时间以及硬件的限制，我们的作品还有一些不足，并没有做到十全十美。一是没能完全消除噪音，小气缸和有机玻璃之间的摩擦碰撞未能完全消除，还是会有碰击声发出；二是我们的系统只有 3 根弦，而标准的吉他是有 6 根弦的。虽然 3 根弦已经能够弹奏常见的吉他乐曲，但是并不能完全覆盖所有音符，这是我们需要改进的部分。三是机器的外观不是很好看，需要更好的设计。

本系统可扩展性非常好，易于改进。比方说升级成 6 根弦，我们只要再打造 3 个一样的底座，并配备上相应的材料就可以了。改进的系统能够覆盖所有音符，能够弹奏所有吉他乐曲，非常适用于机场，饭店等人流量大的场所，进行重复性大，强度高的的伴奏及演奏任务。而且本系统弹奏和弦有得天独厚的优势，人手弹奏和弦不可能同时弹奏几根弦，但是机器可以，也就是说再某种程度上，我们的系统可以弹出更准确的音乐。

参考文献

[1] Eric Singer, Kevin Larke, David Bianciardi "LEMUR GuitarBot: MIDI Robotic String Instrument", LEMUR, 2008

[2] Ajay Kapur, "A HISTORY OF ROBOTIC MUSICAL INSTRUMENTS", University of Victoria, Music Intelligence and Sound Technology, Interdisciplinary Centre (MISTIC)

[3] Ajay Kapur, "A HISTORY OF ROBOTIC MUSICAL INSTRUMENTS", University of Victoria, Music Intelligence and Sound Technology, Interdisciplinary Centre (MISTIC)

[4] TeamDARE , Artemis Orchestra Competition, Athens, Greece, 2008.

[5] 手机铃声，百度百科，http://baike.baidu.com/view/19507.htm

[6] MIDI转MP3方法 ，http://hi.baidu.com/qimuqiang/item/a89935108d828c9e99ce3390

[7] "http:\\fydoc.tripod.com\formats\mid.htm",(.mid)Stander MIDI File Format,Header Chunk.

[8] MIDI 文件解析及播放 辽宁科技大学毕业论文 2011

[9] FT-RS16 开关量输入输出模块说明书 北京宏志飞腾电子科技发展有限公司 2012

[10] 工业级品质 32 路 RS232 RS485 串口继电器控制板 http://item.taobao.com/item.htm?id=8714718646

[11] 音乐可视化 百度百科 http://baike.baidu.com/view/7793471.htm

[12] 维基百科 OpenGL http://zh.wikipedia.org/zh-cn/OpenGL

[13] 3D MAX 百度百科 http://baike.baidu.com/view/123913.htm

谢辞

在吉他机器人的开发过程中，我们首先要感谢唐新民老师，他提出了许多宝贵意见，还不辞辛劳的带着我们去各个电子市场购买硬件及配件，在各方面给与了很大支持。其次，我们还要感谢徐超学长，他对 MIDI 音乐研究的前期工作为我们设计 MIDI 解析算法提供了很大的帮助。

同时，我们也要感谢这次大赛组委会为我们提供了一个绝佳的平台展示自我，也让我们了解和学习了 Intel 嵌入式平台下的开发，积累了宝贵经验。无论获奖与否，我们都视这次比赛为一个绝好的学习的过程，再次感谢这次大赛！

最后感谢各位专家评委审阅我们的报告。也感谢曾经给我们关心和帮助的朋友们。