

Interactive Teaching Guitar

Second Semester Report
Spring Semester 2009

by
Michael Ullmann
Tim Stansbury
Joseph Molaskey

Prepared to partially fulfill the requirements for
ECE402

Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, Colorado 80523

Report Approved: _____
Project Advisor

Senior Design Coordinator

Abstract

Guitar is one of the most popular instruments to play, however most people that play guitar as a hobby can only play a few songs. As with any instrument, people would like the ability to play any song they wanted, but it is very difficult to teach yourself how to play any song. There are many methods out there on how to teach yourself, but most require the ability to read music which for many is difficult to learn on your own. For our project we wanted to develop any easier method and product for learning how to play any song.

Our project accomplished this feat by developing a guitar that has the ability to teach a person to play any song they want. This guitar is self contained so that the user does not need to be connected to any external components, such as a computer, in order to learn. We also believed that this guitar should be able to sound and feel similar to any other guitar so that the user is not confused by the feel when using a normal guitar. The way in which we believed this to be best accomplished was to use lights to show the user where to put their fingers while playing.

To make the lighting of the guitar effective, there must also be a way to slow the speed of the lights. In order to accomplish all of these tasks, we used a microcontroller in the body of the guitar. The microcontroller is the central hub that connects the lights, songs, and additional features. Once we were able to get a simple light set up to display to the user where to put their fingers we were able to expand the guitars teaching capabilities to the many features of the microcontroller.

We believe the lighting of the fingers will be an effective teaching tool, similar to how some current electric keyboards light the keys that are supposed to be played. If the guitar works well for users learning the guitar, we believe it would be good to expand this technology to other stringed instruments like a bass.

Table of Contents

ABSTRACT	2
TABLE OF CONTENTS	3
LIST OF FIGURES	5
LIST OF TABLES	5
CHAPTER I - INTRODUCTION	6
CHAPTER II – EXISTING PRODUCTS AND LEARNING METHODS	7
CHAPTER III – HARDWARE	8
III.1 MICROCONTROLLER	8
III.3 LEDs	9
CHAPTER IV – SOFTWARE	11
IV.1 SUPPORT SOFTWARE	11
IV.1.1 TUX GUITAR	11
IV.1.2 POWER TAB FILES	12
IV.1.3 POWER TAB CONVERTER	12
IV.1.4 ADDITIONAL SOFTWARE	14
IV.2 MICROCONTROLLER SOFTWARE	14
IV.2.1 FILE INPUT SYSTEM	14
IV.2.2 BUTTON PROGRAM	15
IV.2.3 LED PROGRAM	16
IV.2.4 LCD PROGRAM	16
CHAPTER V – GUITAR CONSTRUCTION	18
V.1 FINGERBOARD	18
V.2 NECK	19
V.3 BODY	20
V.4 FINISHING TOUCHES	20
CHAPTER VI – MANUFACTURABILITY AND MARKETABILITY	21
VI.1 MARKETABILITY	21
VI.2 MANUFACTURABILITY	21

CHAPTER VII – CONCLUSION	22
REFERENCES	23
ACKNOWLEDGMENTS	23
APPENDIX A – ABBREVIATIONS	24
APPENDIX B – BUDGET	25

List of Figures

Figure 1 - Explanation of tablature	7
Figure 2 - EmbeddedArm TS-7260	9
Figure 3 - LED data sheet provided by Digikey	10
Figure 4 - Tux Guitar screenshot	12
Figure 5 - Block diagram of selected software	14
Figure 6 - Dremel routing jig	18
Figure 7 - Neck and fingerboard schematic drawing.....	19
Figure 8 - Guitar Cavities	20

List of Tables

Table 1 - Budget Spreadsheet.....	25
-----------------------------------	----

Chapter I - INTRODUCTION

This project was developed in order to create a better way to learn how to play songs on a guitar. We believed that there must be a better way to learn how to play guitar than to just simply read sheet music. The method and design that we developed was to have a guitar that would teach you the proper finger positions for the song. This guitar uses LEDs in the neck that will perform the job of displaying where to place fingers. This guitar also has an LCD screen so that a user can know what song to play and where in the song he or she is playing. This allows the user to know which part of the song they get stuck on and if they need more help with that section can use multiple learning techniques on that section. In order to make the guitar more user friendly, we also wanted to allow the user to slow down the song so that he or she could learn at their own speed. The last feature we wanted this guitar to have was the ability to add any song they wanted to the guitar so that the number of songs they could learn was only limited by what they put on the guitar.

In our next chapter we will go farther into the methods and products that are currently available for someone to learn to play guitar. We will discuss how each of these methods attempt to teach guitar, the pros and the cons associated with each method, and how our project attempts to improve on the deficiencies.

The chapters following will then describe the major hardware components that are required to make all the features of this guitar possible. We will go into detail about the about the reasons we chose the microcontroller, LEDs, and LCD we did and what the benefits of each of these hardware are. We will also discuss how all of this hardware interacts with each other to perform all of the features.

The chapter following the hardware will go into detail about the software that was developed so that the guitar can make the hardware perform all of the features. There are several programs that have been developed in order to get the music files into a proper format to put onto the guitar. Once the music is loaded to the guitar there is another whole team of programs that then control the hardware.

The final chapter will describe where all of these features will be placed in the guitar will be constructed and where all of these features will be in the guitar.

Chapter II – Existing Products and Learning Methods

Today there are several existing methods of learning the guitar. The traditional way of learning the guitar is sheet music and then later what is known as guitar tablature. Guitar tablature is much like sheet music except a numbering system is used to indicate the fret fingering. Tablature is much easier to read for a guitarist mainly because there are several positions where the same note occurs on the guitar.

Figure 1 - Explanation of tablature

Sheet music added for the purposes of timing.

strings

Fret numbers

Notes played in sequence from 1 to 15

More contemporarily, software started to be made that will play an electronic version of sheet music and highlight every note as it is played. This advancement added a lot more interactivity to learning guitar. A few examples of this kind of software are: Power Tab, Tux Guitar, and Guitar Pro. Power Tab is the most widely accepted version mainly because it was the first open source software of this type. Power Tab as well as tux guitar are both authoring programs for sheet music as well as players. The fact that people can create songs with these programs means that there are thousands of free songs available for download on the internet.

Currently a guitar exists with a fingerboard lighting system called the FretLight® made by Optek Music systems. This guitar is a USB interfaced unit that runs software from a computer to control the guitar.

Chapter III – Hardware

The following sections detail all of the major hardware components of the project.

III.1 Microcontroller

The microcontroller will be the central hardware component of the guitar, controlling all of the different features that will be on the guitar. Due to the design and features of the guitar, the microcontroller had several important requirements placed on it.

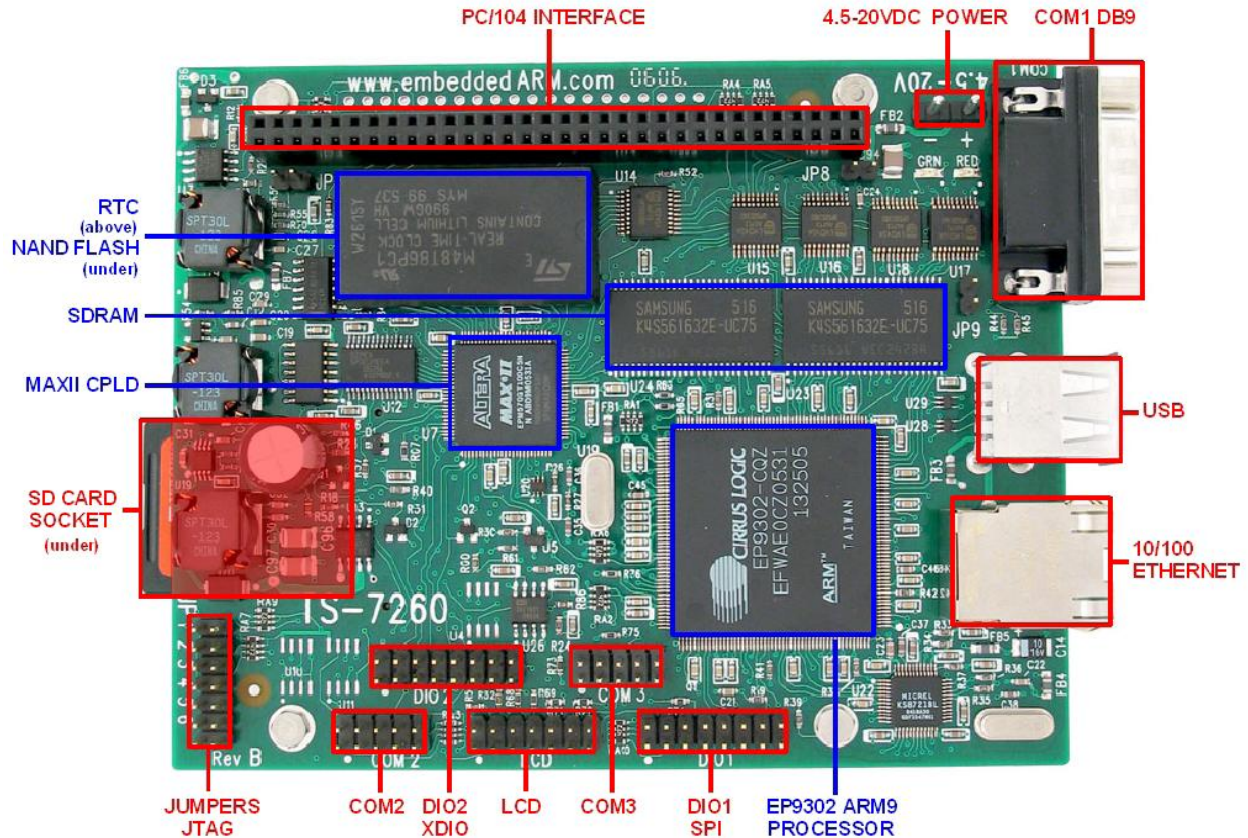
The requirements of the microcontroller ranged from size, to ports, to what type of language environment the board used. The microcontroller had to have a size no larger than 4.5”x6”x1” so that the board would be able to fit in the area allotted between the neck of the guitar and the pickups. The next requirement was that the board must have the ability to expand to at least 36 dedicated DIO pins. Of these 36 DIO pins, 5 of the lines were to be used as input to read the buttons and the other 31 were to be used as output to send signals to the lights in the neck. These DIO pins were to send out a voltage of at least 2.5 volts in order to light the LEDs. The board was also required to have a LCD port for the LCD screen that would display the song title. The final few features of the board that were not required but were highly desired was lower power consumption so that the guitar would not eat through batteries, a friendly software environment so that the coding of the software would be easier, a easy way to add on SD card reader, and low cost.

After much research of the several different types of microcontrollers that were available, we decided that the type of microcontrollers that had the most features and best user environment and documentation was the Arm boards. There were several ARM boards that met two or three of our requirements, but we were very limited in boards that met all of the requirements. The Arm board that met our requirements and was best priced was the EmbeddedArm 7260. The 7260 has the necessary dimensions of 3.8”x4.8”, which is the perfect size to fit in between the neck and the pickups. The 7260 does not have the required number of dedicated DIO pins; however it does have a PC-104 interface. This interface allowed us to buy a peripheral that has 64 dedicated DIO pins. The voltage of the pins on the peripheral can range from 0-40 volts, so we will easily be able to receive the desired 2.5 volts to light the LEDs. Due to this, we will use the 31 of the 64 pins on the PC-104 peripheral to light the LEDs in the neck of the guitar. The DIO for the five buttons will come from the DIO1 port. By keeping all LED output pins on the peripheral and the buttons on the DIO1, we will be able to write the programs that do not have to have pins on different ports overlapping. This fact will greatly improve the ease of code development. The 7260 also has a dedicated LCD port that has the standard 14-pin configuration that is found on most small LCD screens like the one that will be used on this guitar.

The EmbeddedArm 7260 also met all of the desired features. The board is specifically designed to consume low amounts of power, so the batteries will last for a longer time. The voltage to run the board can also range from 4.5 to 20 volts which will allow us to build a battery back that can be much less rigid on always supplying exactly the same voltage. This board also has a built in SD card reader which means that we do not have to deal with finding and interfacing a SD card for the board. The final feature of this board which improved the code development time was

that it is running a small version of Linux and will run C code and executables directly on the board. This feature allows us to develop code in a much higher level language.

Figure 2 - EmbeddedArm TS-7260



III.2 LCD Screen

The LCD screen required for the guitar must be sufficient in order to display song name, tempo, and measure. The LCD does not require a backlight system so we will not use one in order to save cost. The LCD screen must not be too large because it must be positioned in a limited space on the guitar pick guard. The microcontroller also calls for an LCD screen that works with a Hitachi HD44780 LCD controller or equivalent. Ideally the LCD should have low power consumption; however it will be in sleep mode often so this is not as much of a concern.

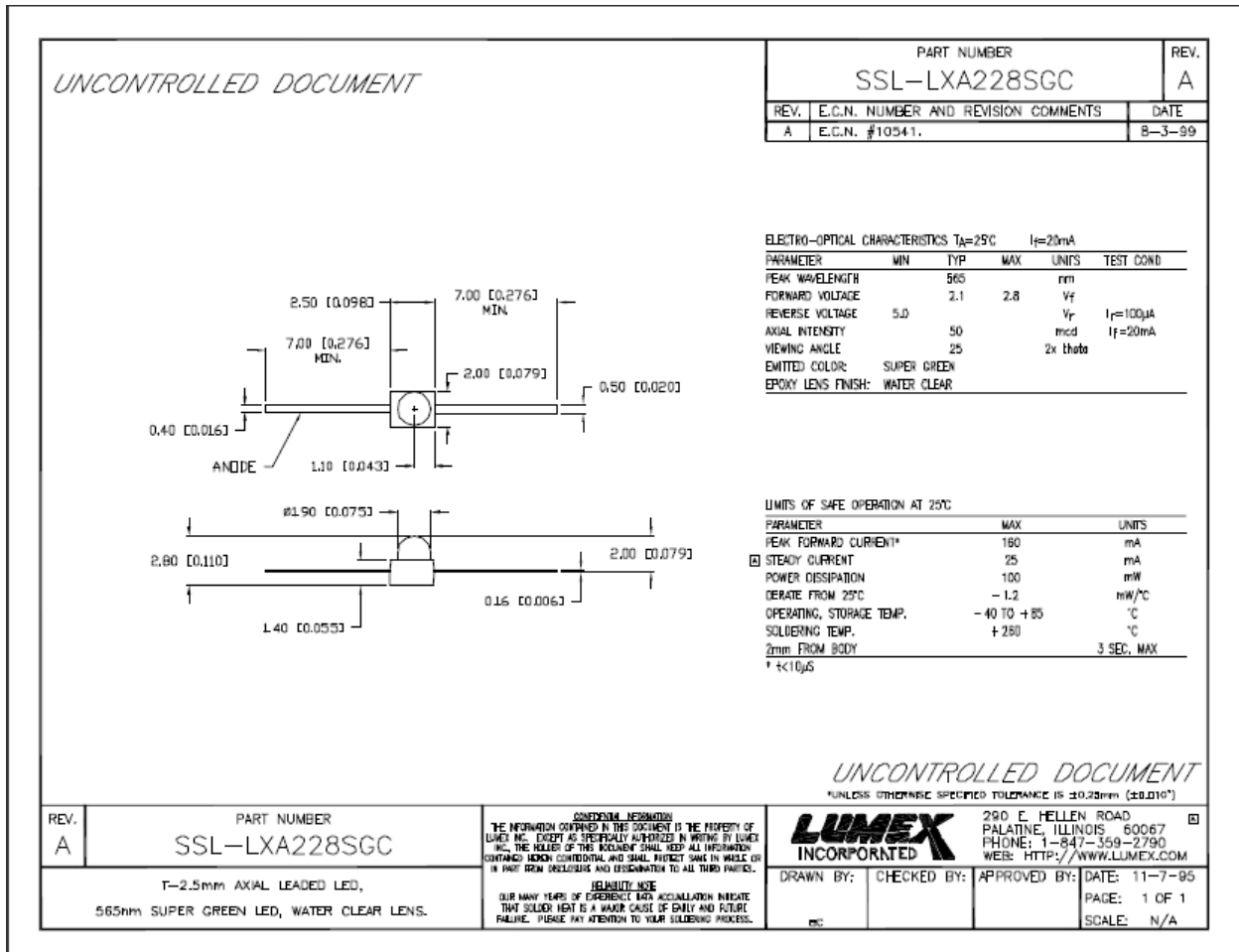
III.3 LEDs

The Interactive guitar will have a total of 150 LED's in the fingerboard. The LED's will be wired in a matrix type configuration in order to reduce the number of required I/O for the microcontroller. The fingerboard lighting system will have a total of 31 inputs. Six of the inputs will be Anode lines and 25 will be Cathode lines.

There were several factors that were considered when choosing the LED's for the neck. The LED's had to be small, consume little power and be bright enough to be visible in a well lit environment. The combination of these three conditions also made cost another primary concern. Eventually we chose LED's available through Digikey and made by LUMEX. The LED's we chose were relatively cheap comparative to others and satisfied our requirements.

All of the specifications are described in the data sheet below:

Figure 3 - LED data sheet provided by Digikey



Chapter IV – Software

IV.1 Support Software

The following sections detail all of the work that has been done on the support software for our project so far.

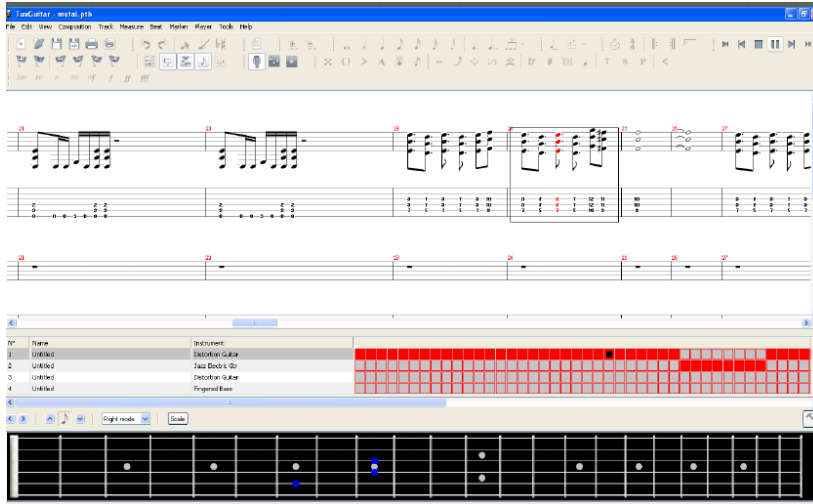
IV.1.1 Tux Guitar

Tux Guitar is a music authoring program that allows users to write their own songs. All aspects of the song can be chosen by the user such as tempo, instruments used in the song, key signature, and length of the song. The program also has the ability to play the song back to the user so he or she can decide if they like the way the music they have written sounds.

One very interesting feature Tux Guitar has is the ability to display a fingerboard at the bottom of the screen when the program is playing a song. This fingerboard is essentially a computer simulation of what our interactive guitar will do. When a note is being played, the fingerboard has a colored dot appear at the fret/string location that should be pressed to play the given note. Researching the Tux Guitar source code and determining how its authors implemented this function may save us a lot of time and work. Modeling their implementation in our code may be possible and may not require much modification.

The Tux Guitar program also has the ability to speed up or slow down playback of a song. Since this is another feature we intend to implement in the guitar, researching this portion of the Tux Guitar software could also be very beneficial to us. Regardless of whether or not we can use some of the Tux Guitar code to add this function to the guitar, it will still be helpful if we can determine the method used in the program to change the playback speed. If we can simulate the technique used it will save us time because we will not need to develop our own method for slowing the song down.

Figure 4 - Tux Guitar screenshot



IV.1.2 Power Tab Files

The digital sheet music files used by Tux Guitar are called power tab files. These music files are free for download through many websites on the internet. This makes using the power tab files ideal because they give us access to almost any song we could want. However, there were a couple issues with using the power tab files directly.

First, power tab files have their own unique format. When opened through a text editor such as Word Pad the files appear to be made up of various symbols. After studying existing power tab files and creating our own, we determined that there are no apparent patterns of symbols to represent things like notes. So making a table of patterns and using it as a reference so we could understand each power tab file was not a possibility. After researching Tux Guitar we learned that we could use it to convert the power tab files to XML files. This solved the first problem we were having because when an XML file is opened in a text editor it appears as English words and numbers. We could then clearly see how all the different instruments, measures, and notes in the song were divided up. One this was accomplished we ran into our second major problem with the power tab files.

Because of all the functionality of the music authoring programs, the power tab files store a vast amount of information. This information includes things like the instruments used in the songs, instrument tuning information, MIDI playback information, and note names. The files also have information for all the instruments that are to be played as part of the song. Since much of this information will have no use on the guitar we decided to create a program to parse the files and remove all the unnecessary information.

IV.1.3 Power Tab Converter

Our power tab parsing program was developed in Java. When executed, it asks for the name of the XML converted power tab file as well as a part number so the user can select between the

different parts and instruments used in the song. The converter outputs a simplified .txt file that is, on average, 1/8 the size of the original file due to large amount of information removed.

The converter begins by separating the file into sections and storing them in an ArrayList. Since the file contains different instruments, each index in the ArrayList consists of only a single instrument. This was trivial because all the information for one instrument is grouped together in one section of the file followed by the all information for the second instrument and so on. The part number the user entered when the program was first executed is then used to determine which instrument is used in the rest of the converter. The next step involves removing all the information in the selected part before the first measure of notes in the file. This information appears to be used only by the MIDI component of Tux Guitar

Now that the converter has eliminated all the unnecessary sections of the file before the actual note information of the selected part, the next step taken is to break up the part into smaller sections and store them in a new ArrayList. This time, each index in the ArrayList represents an individual measure of the song and stores an ArrayList consisting of the notes in the measure. This can be visualized as a matrix where the first ArrayList represents the rows, which are the measures, and all the ArrayLists it stores represent the columns, which are the notes. This step also makes sure that information inside the measures that will not be needed is not added to the ArrayList. Once this ArrayList of ArrayLists is created, the final step is to write the new text file.

The first information written to the new file is the title of the song. Some songs have tempo changes at various points in the song. To simplify our on-board program, the converter only writes the tempo the song starts with to the file. So, all songs that are intended to have multiple tempos throughout the song will only have one. Some notes can be what are called dotted notes. In sheet music this is represented as a small dot next to the note and adds duration of half of the notes current value onto the note. There can even be multiple dotted notes. However, simplify our on-board program even more, the converter will only write one dot per note to the file. So, if a note had multiple dots it will be written to the file as a note with a single dot. The file is written using two for loops with one nested in the other. The outside loop corresponds to the measures of the song, or the rows of the matrix. For every step this for loop takes, it writes the measure number to the file. The inside for loop corresponds to the columns of the matrix and writes all the notes to the file. Once the for loops exit the creation of the simple file is complete and the converter terminates.

A special Note object was created to aid in writing the note information to the file. The Note object has a note type value which represents whether the note is an eighth, quarter, half, whole, etc., as well as values for all six strings. To write a note that is not part of a chord to the file the converter sets all these values in a note object then uses the object to write the information to the file, and then resets the object returning all its values to a default 0 state. If the note is part of a chord, the converter will add all the chord information to the note object before it is written to the file and reset.

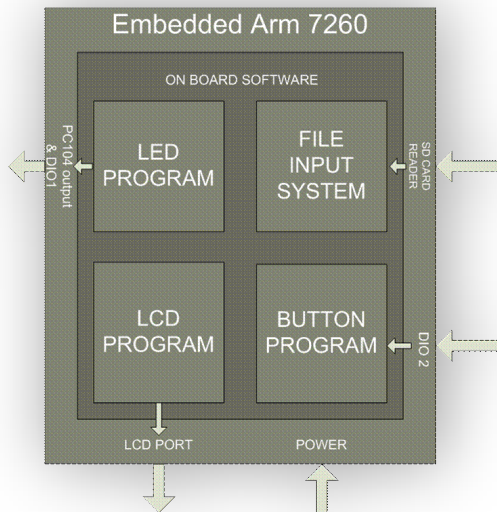
IV.1.4 Additional Software

Once we received the microcontroller we were able to begin reading the documents and learning about how to program the board. The information we found explained that since we are using computers with Windows we needed Cygwin and a crosstool compiler for C. Cygwin allows us to run a Linux style terminal on our Windows desktop. Once this was installed we had to install the crosstool compiler, Cygwin Crosstool gcc-3.3.4-glibc-2.3.2. This C compiler converts the source code files into executable files specifically designed to run on the ARM – Linux distribution installed on the microcontroller. Once the executable has been generated the application HyperTerminal is used to transfer the file from the computer to the microcontroller.

IV.2 Microcontroller Software

The following sections detail all of the work that has been done on the microcontroller software for our project so far. The microcontroller software is all of the programs that will be loaded on to the Embedded Arm 7260 board. This software has been broken down into four main components: the file input system, the button program, the LED program, and the LCD program.

Figure 5 - Block diagram of selected software



IV.2.1 File Input System

The file input system is the first program that runs when the guitar is powered on. The program searches the SD card that is located on the board and compiles an array of all of the song files that are on the card. This list, which contains the entire path of the song file, is then put into an array and returned to the main program in order for the three other programs to use this list.

IV.2.2 Button Program

The button program controls the four buttons that are located on the front of the guitar and the one button located on the back of the guitar. These buttons are attached to the DIO2 port on the EmbeddedArm 7260 which has been set to be input only on program start-up. These four buttons on the front and button on the back all have different features depending on which section of the main program you are currently in. The button presses are all caught using IF statements at the beginning of each section that is currently running.

The sections can be broken into two main sections, selection of a song and playing of a song. The selection of the song is the main section that the main program starts up to. This section can be further broken down into the selecting of the song and the selecting of the song's tempo. The main section and section that the main program starts up into is the song selection part of this section. In this section, the button program enables four of the buttons, three of the front buttons and the button on the back. When choosing a song, the up, down, and enter (right) button are enabled. The user searches through the list of songs for the one they want to play using the up and down buttons. These buttons do not loop through the list so once the bottom of the list is reached, if you continue to press the down button it will stay on that last song; this is the same for the top song of the list and the up button. Once the song you want is showing, the user selects the song by pressing enter which is the right button.

For the button on the back to work, the main program must be in this section, connected to a computer with the DB9 null modem cable, and the computer must be connected to the board with a terminal emulator like HyperTerminal. When all this is done, if the button on the back is pressed, a prompt will appear on the terminal window asking the user if they would like to end the program running on the guitar (y/n). If the user says yes, the program exits and the board loads the Linux terminal window. This should only be done by a programmer or developer to change the main program, song files if the SD card is having problems, or the start-up procedure of the board.

When the main program is in the tempo selection section, the buttons have basically the same functionality as they did in the song selection; however the back button is no longer enabled. As you press the down button, the percentage of normal that the song will be played at decreases (75%, 50%, etc.) until the lowest percentage has been reached and then stays there no matter how many times down is pressed. The max speed that the guitar plays at is 100%, so this percentage will not increase no matter how many times the up button is pressed. Once you have decided on the speed that you would like the song played at, you press the enter (right) button and the song begins to play.

The playing of the song is the final section of the main program that the button program controls. When in this section only the play/pause (right) and stop (left) button are enabled. When the song is playing, the user can either press stop which will return the program to the song selection section of the main program or pause which will pause the song and change the LCD screen to say paused. When the program is paused, the only button that will work is the play (right) button. When the play button is pressed, the song starts back up in the spot it was paused at after a slight delay to allow the user to get set to play.

IV.2.3 LED Program

The LED program is used to control the lighting of the LEDs in the neck of the guitar. This program has two main jobs, to parse the music files and to send the correct signals to the LEDs. The ports used to light the LEDs are located on the DIO1 and the output pins located on the DIO64 PC-104 peripheral.

The parsing of the file will occur once the song has been selected by the user. The first line of the file is read in with the first argument being the song name, the second being the note value of one beat (4 for quarter, 8 for eighth, etc), and the third value being the beat per minute. Once this line is read in, the program jumps to the section of the program where the user selects what speed to play the song at. After the user has made this selection, the adjusted beats per minute is returned to the LED function. Once the LED program has the adjusted beats per minute, it calls a function that sets the count values of the loop very every type of note. These count values are used in the section of the LED program that is responsible for lighting the LEDs. Once all of these values have been set, the program begins to read in the file one line at a time. Each line has the format: note type, string1, string2, string3, string4, string5, string6. The note type is a double that corresponds to a quarter note (4), a half note (2), and even dotted notes, i.e. a dotted quarter note (4.5). The next 6 values are a decimal representation of a 25-bit binary string, where each bit corresponds to a fret and a 1 corresponds to that fret being lit. These seven values are then passed to the LED lighting function. When this function ends, the next line of the file is read in and passed to the LED lighting function. This continues until every line of the song file has been read.

The LED lighting section of the LED program takes in the seven arguments discussed above. It then uses the first argument, the note value, to set the count for the for loop using an if statement to set the count to the correct count value that had been set in the previous section of the program. Once the count has been set, the function enters a for loop, looping through for count number of times. In this for loop a value of 1 is sent down the first string, which corresponds to pin 1 on DIO1 and the second argument which corresponds to the values of the frets on string 1 is sent to the output pins of the PC-104. All the DIO1 pins are then sent to 0, as well as the PC-104 pins. Then a value of 1 is sent to pin 2 of the DIO1, which is the second string and the third argument corresponding to the fret values for the second string is sent to the PC-104 output pins. Both the PC-104 and DIO1 pins are then again set to 0. This process continues for all of the strings on the guitar. Once all the strings have been lit once, the loop has gone through one cycle. This cycle is repeated count number of times. This loop of lighting only one string at a time is due to the matrix style wiring scheme that was used on the LEDs. The string switch going on and off, and the entire loop is able to cycle quick enough that it appears that every string and fret is being lit at the same time. Therefore, this process of lighting the LEDs allows us to light any combination of the LEDs that is desired, from a single LED on the neck to all 150 LEDs.

IV.2.4 LCD Program

The LCD program contains two main components; the initialization of the screen and the displaying of the characters to the screen. The main program calls the initialization portion of

the program at the very start of the main program when the guitar is turned on. The second part of the program that controls the screens display is called throughout the main program and other four program components in order to convey information to the user.

The LCD screen first lets the user know that the guitar is started and ready to be used by displaying “Welcome to Interactive Guitar”. This display stays on for a few seconds while the guitar finishes warming up, at this point the screen changes to ask the user to choose a song. The user will then use the buttons to scroll through the songs. At this stage, “Choose a song to play” is on the top line and the current song name is on the bottom of the screen. Once the user has selected the song, the screen then prompts the user to select which speed they would like the song to be played at. At this stage, the first line displays “Choose song speed” and then what the normal beats per minute of the song are, i.e. the song at 100%. On the second line is the percentage that the user would like the song to be played at; this percentage increases or decreases as the user scrolls through the buttons. Once the user has gone through the whole process of selecting the song and its tempo, the screen displays the song’s name on the first line and on the second line displays whether the song is playing or paused, the beats per minute (BPM) the song is playing at, and the note value (NV) of a single beat. In the note value section a 4 corresponds to a quarter note, 8 to an eighth note, etc.

Chapter V – Guitar Construction

V.1 Fingerboard

The Fingerboard has proven to be the most difficult part of the construction process. The LED's have to illuminate from the back side of the fingerboard which required drilling 150 holes. The back side of the fingerboard had to have channels cut in order to route the wiring. Initially there was trouble deciding on a method for channeling the back side of the fingerboard. Eventually the channeling was achieved with a jig created from some scrap wood and a dremel tool.

Once the channeling and wood work was finished on the fingerboard, The LED's were super glued in place and soldered accordingly. Once the LED's were installed they were cemented in place with epoxy to prevent them from jiggling loose during normal wear of the instrument. The epoxy helps restore strength to the fingerboard and neck and reduce noise. Once the fingerboard is epoxied it will be ready to glue to the neck.

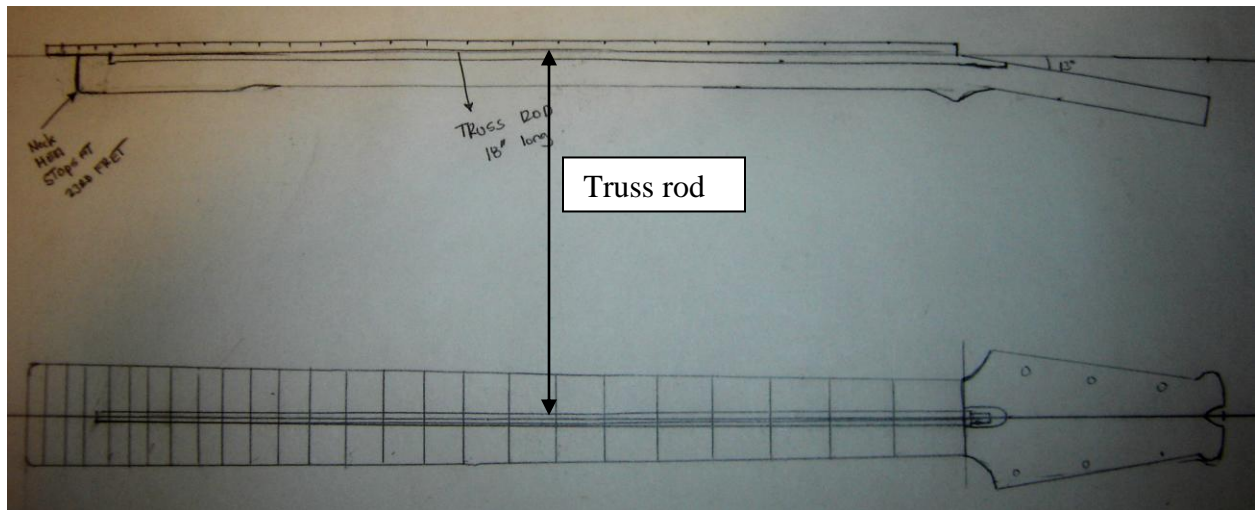
Figure 6 - Dremel routing jig



V.2 Neck

Then neck was constructed out of 3 pieces of wood glued together. The reason we used 3 pieces of wood is for extra strength against natural warping of the wood. After the wood was glued together a shape was cut from it. A 3/8" deep truss rod slot was routed down the center of the neck. The truss rod is an adjustable steel reinforcement placed in the neck to adjust forward bow and backward bow.

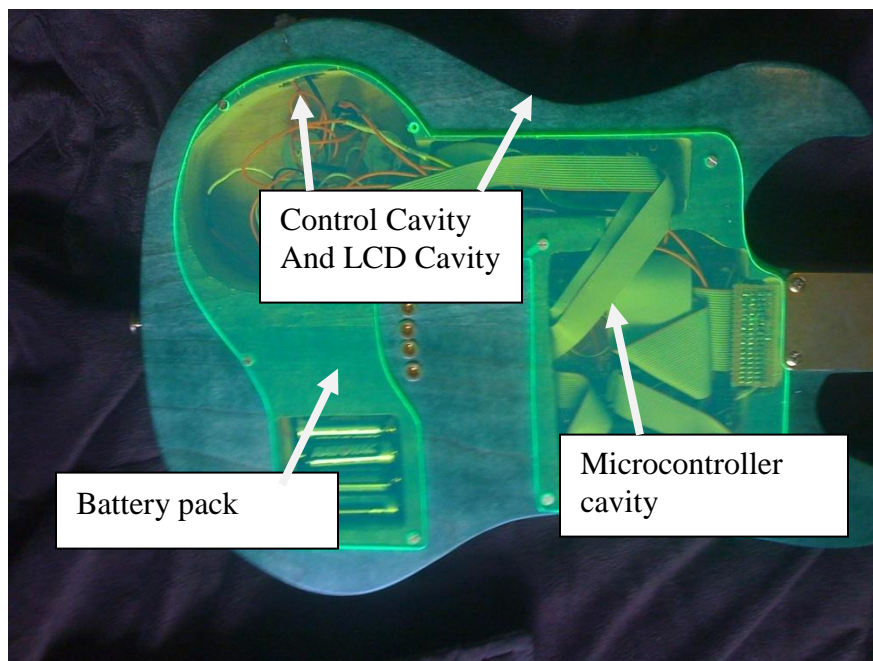
Figure 7 - Neck and fingerboard schematic drawing



V.3 Body

The body of the guitar was constructed from Alder and Poplar. These woods were chosen because of both their low cost and softness. The softer woods make it easier to achieve clean routes. The Body was rough cut with a band saw and then refined to a smoother shape. There were several cavities routed out in order to make room for components. The microcontroller sits between the bridge and the bottom of the fingerboard underneath the strings and is installed on the back side of the guitar. The LCD is below the strings and the microcontroller. The battery pack is above the bridge and the knobs. A neck pocket was also created in order to install the neck at the correct height above the surface of the guitar.

Figure 8 - Guitar Cavities



V.4 Finishing Touches

Once the neck fingerboard was joined together the frets were installed. Then the complete neck and body were ready for finish. A stain was applied in order to change the wood to a green color. After stain was applied, the full instrument received 6 coats of nitrocellulose lacquer and buffed once dry. When finish was completed all of the hardware and electronics were installed.

Chapter VI – Manufacturability and Marketability

VI.1 Marketability

The Interactive guitar is a very marketable product. The Optek Fretlight guitar is a similar item that has proven marketability. There are several problems with the Fretlight guitar that we see limits the marketability of the product. One limitation is the fact that the Optek guitar has to be plugged in to a computer in order to use it. The Interactive Guitar is an improvement over this because it is a completely stand-alone unit once the songs are loaded. Optek also charges their customers for the sheet music that can be used with the guitar. The Interactive Guitar operates on an entirely open source platform that requires no additional cost. The additional cost of software makes the Optek Fretlight far less attractive to a consumer. The Interactive Guitar should be more marketable than the Optek Fretlight guitar due to the obvious advantages.

VI.2 Manufacturability

In order to market the Interactive Guitar we would have to design a low cost manufacturing scheme. The LED system that was created is far too complicated to manufacture quickly and costly. The LED's chosen were also far too expensive. In addition to the cost of the lighting system, the microcontroller would have to be much cheaper. This would require designing a microcontroller specific to the product and removing all unnecessary components. We believe that if we could drive the unit cost to around five hundred dollars it would be a very marketable system.

Chapter VII – Conclusion

As of May 3, 2009 the Interactive Guitar has been completed. All of the feasible design specifications we started the project with have been successfully met. After coming up with the idea for the guitar we researched all the components we would need, learned how to use them, implemented the various programs we needed and developed a finished product that could be potentially be marketed and sold.

Due to this, we recommend for this project to not be continued as a senior design project next year. Our reasons for this project not being continued are that this project was a student developed and designed project in which we found a professor to sponsor our idea, and this project has been completed in its first year. Due to both of these factors, this project does not make sense to continue next year.

REFERENCES

TuxGuitar

<http://www.tuxguitar.com.ar/tgwiki/doku.php>

TS-7260 Manual

<http://www.embeddedarm.com/documentation/ts-7260-manual.pdf>

TS-7260 Schematic

<http://www.embeddedarm.com/documentation/ts-7260-schematic.pdf>

Getting Stated with TS-Linux ARM

<http://www.embeddedarm.com/documentation/software/arm-tslinux-ts72xx.pdf>

Linux for ARM on TS-72XX User's Guide

<http://www.embeddedarm.com/documentation/software/arm-linux-ts72xx.pdf>

Linux for ARM on TS-7000 Embedded Computers

<http://www.embeddedarm.com/software/software-arm-linux.php>

Source Code Samples for the TS-7200 Series

<ftp://ftp.embeddedarm.com/ts-arm-sbc/ts-7200-linux/samples/>

Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification

<http://java.sun.com/j2se/1.4.2/docs/api/index.html>

Data sheet for LED's

<http://media.digikey.com/pdf/Data%20Sheets/Lumex%20PDFs/SSL-LXA228SGC.pdf>

ACKNOWLEDGMENTS

We would like to thank the following people for their guidance and support on this project:

- Dr. William Eads
- Tom Aurand

Appendix A – Abbreviations

DIO – digital input/output

LED – light-emitting diode

LCD – liquid crystal display

Appendix B – Budget

Initial Budget	\$400
Additional CSU Funds	\$264

Item	Total
Embedded Arm TS-7260 Single Board Computer	\$195.00
Fret LEDs	\$52.80
Prototype LEDs	\$16.20
16-Pin ribbon cable	\$3.66
14-Pin ribbon cable	\$3.32
LEDs for neck prototype (170)	\$17.82
64 Digital I/O – PC/104 Daughter Board	\$74.00
2X24 LCD screen with LED backlighting and cabling	\$48.00
DB-9 F-F connector	\$10.66
32-Pin ribbon cable	\$5.99
Guitar tuning heads	\$58.99
Guitar bridge	\$23.42
Guitar pickup	\$29.00
Body woods	\$15.00
Capacitors	\$1.49
4 AA battery holder	\$1.79
PC board	\$1.99
Sandpaper	\$4.49
Spray adhesive	\$8.49
Guitar knobs, Jack Plate, Screws	\$7.70
Colortone stain bright green	\$17.40
Switchcraft mono output jack	\$3.64
500k potentiometers	\$5.46
Gold neck plate with screws	\$11.75
Gold ferrules set	\$8.78
Sook Control Port	\$13.65

Total	\$640.49
Remaining Budget	\$23.51