

# Interactive Teaching Guitar

First Semester Report  
Fall Semester 2008

by  
Michael Ullmann  
Tim Stansbury  
Joseph Molaskey

Prepared to partially fulfill the requirements for  
ECE401

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 attempts to accomplish this feat by developing a guitar that has the ability to teach a person to play any song they want. This guitar should also be 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 believe 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 which we believe this to be best accomplished is 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 will be using a microcontroller in the body of the guitar. The microcontroller will be the central hub that connects the lights, songs, and additional features. Once we are able to get a simple light set up to display to the user where to put their fingers we will be 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 program works well for the guitar, we believe it would be good to expand this technology to other stringed instruments like a bass.

# Table of Contents

Title	1
Abstract	2
Table of Contents	3
List of Figures and Tables	4
I. Introduction	5
II. Existing Products and Learning Methods	5
III. Hardware	6
A. Microcontroller	7
B. LCD	8
C. LEDs	8
IV. Software	9
A. Support Software	9
a. Tux Guitar	10
b. Power Tab Files	10
c. Power Tab Parser	11
d. Additional Software	12
B. Microcontroller Software	12
a. File Input System	13
b. Button Program	13
c. LED Program	13
d. LCD Program	14
V. Guitar Construction	14
A. Fingerboard	14
B. Neck	15
C. Body	16
D. Finishing Touches	16
VI. Conclusion	17
References	18
Acknowledgements	18
Appendix A - Abbreviations	19
Appendix B - Budget	19

## **List of Figures**

Figure 1	explanation of tablature	6
Figure 2	EmbeddedArm 7260	8
Figure 3	LED data sheet provided by Digikey.	9
Figure 4	Tux Guitar Interface Screenshot	10
Figure 5	Block Diagram of the Microprocessor Software	12
Figure 6	Dremel Routing Jig	15
Figure 7	Neck and Fingerboard schematic drawing	15

## **List of Tables**

Table 1	Budget Spreadsheet	19
---------	--------------------	----

## **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 will use LEDs in the neck that will perform the job of displaying where to place fingers. This guitar will also have an LCD screen so that a user can know what song to play and where in the song he or she is playing. This will allow 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.

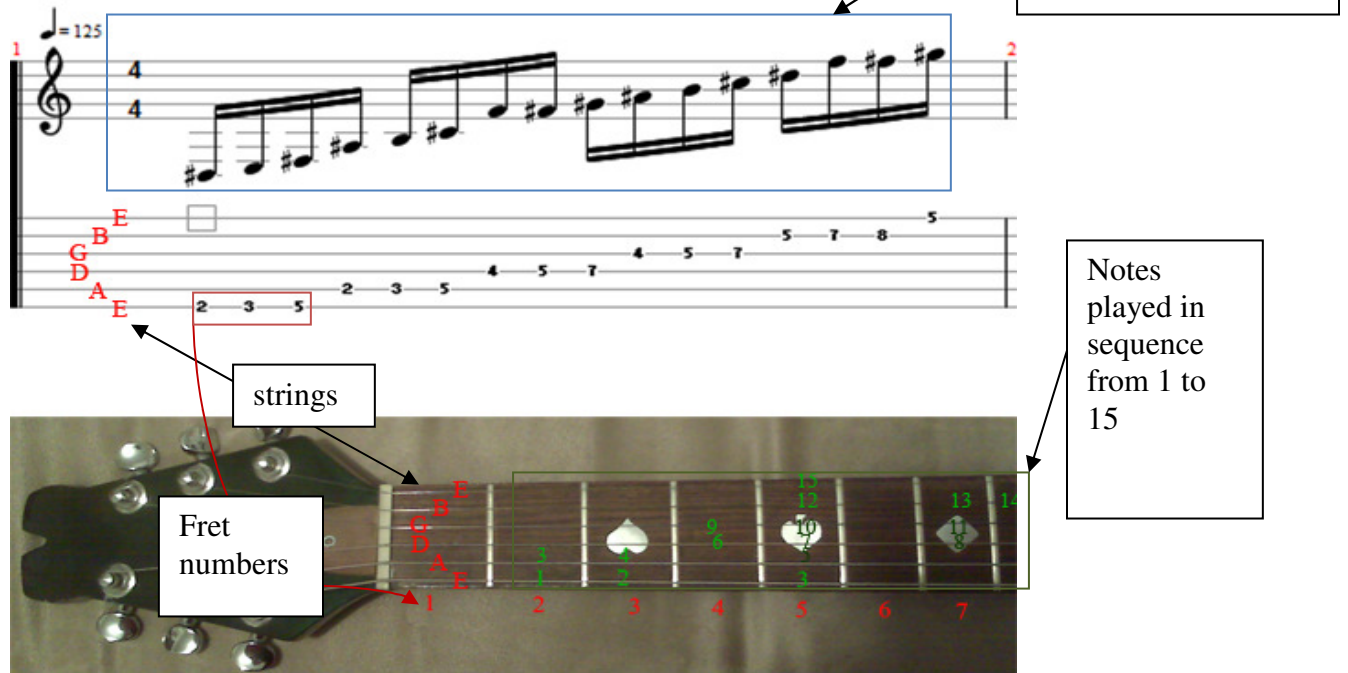
The chapter following the hardware will go into detail about the software that is being developed so that the guitar can make the hardware perform all of the features. There are several programs that have been developed in order to get the music files into a proper format to put onto the guitar. Once 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 is a diagram to show the basic idea of guitar tablature:



**Figure 1-explanation of tablature**

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.

## 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.

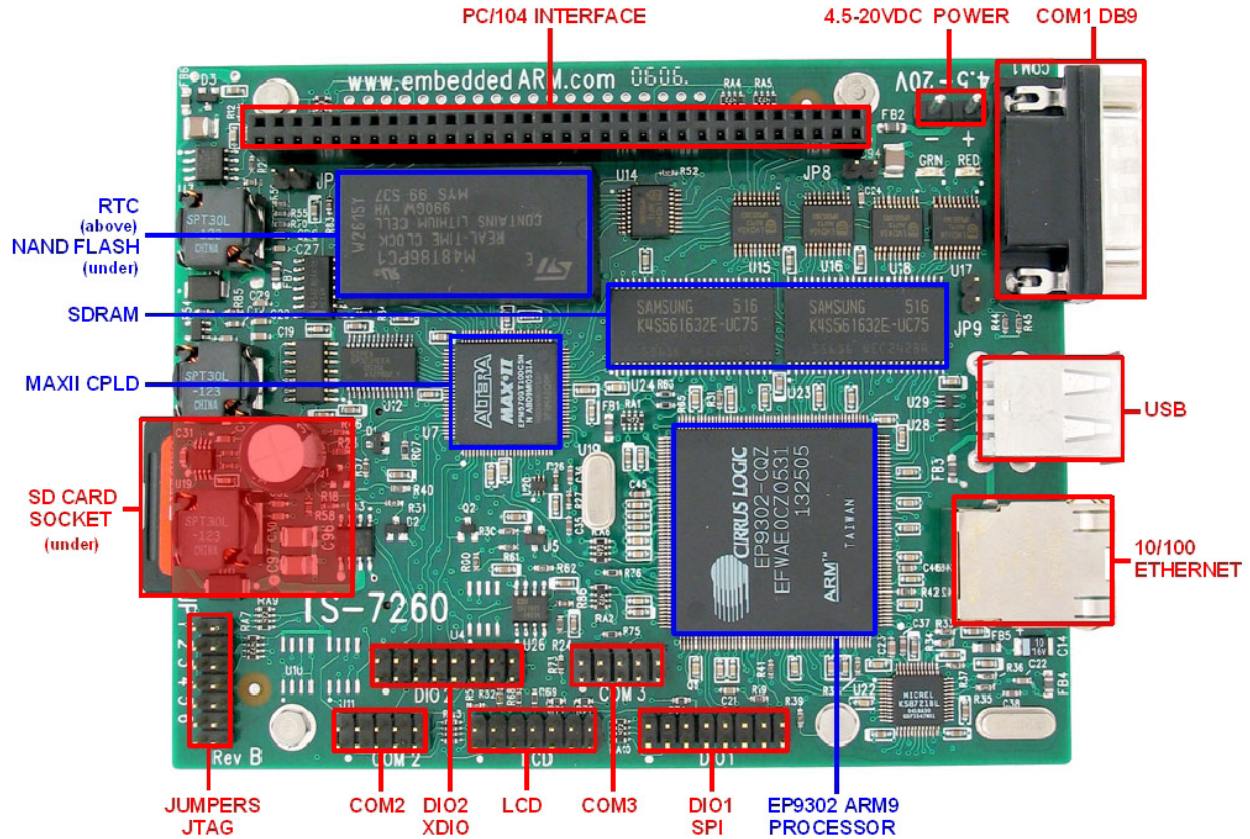


Figure 2 – EmbeddedArm 7260

## 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.

## 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:

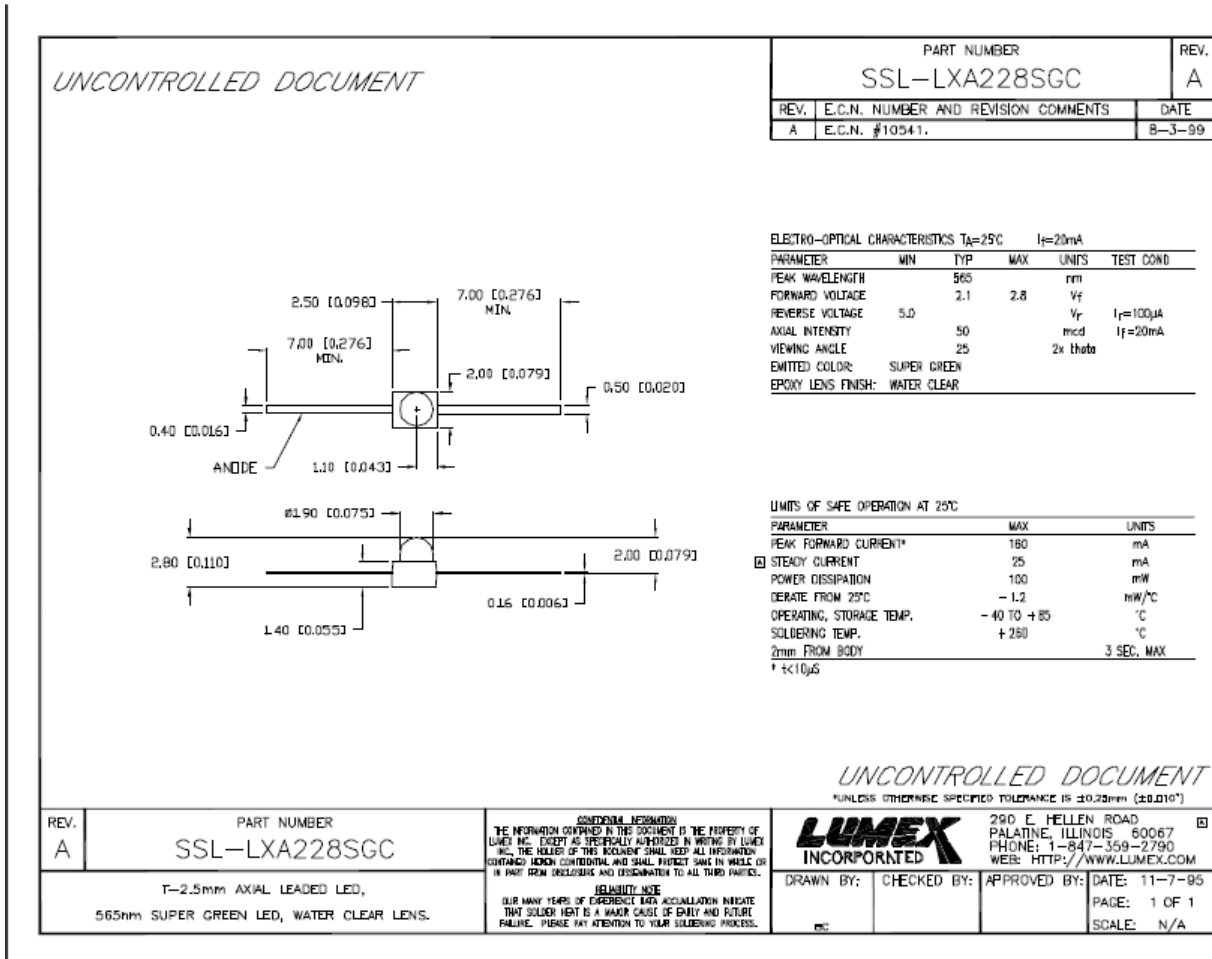


Figure 3 -LED data sheet provided by Digikey.

## Chapter IV – Software

### Support Software

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

## 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.

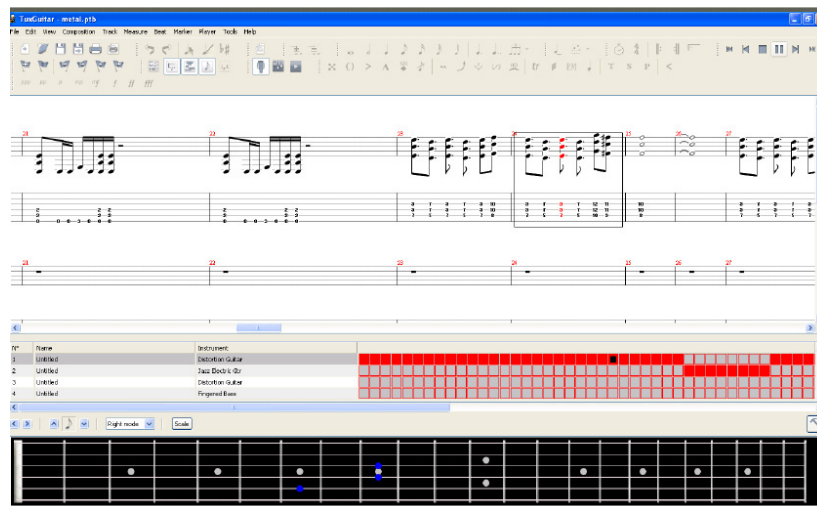


Figure 4 – Tux Guitar Interface Screenshot

## 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. Once 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.

## **Power Tab Parser**

Our power tab parsing program was developed in Java and takes as input a single power tab file and outputs a simplified text file. Due to large amount of information removed from the power tab file, the resulting text file is an average of 1/8 the size original file.

The first step taken by the parser is to separate file into sections and storing them in an ArrayList. Since the file is made up of different instruments, each index in the ArrayList consists of only a single instrument. This was simple since all the information for one instrument is grouped together in one spot followed by the all information for the second instrument and so on. Since only one part can be played at a time on the guitar, only the first instrument is kept while the rest are discarded. The next step involves removing all the information before the first measure of notes in the file. This is necessary because the information from the beginning of the file to the first instrument part of the file appears to be used only by the MIDI component Tux Guitar.

Now that the parser has eliminated all the sections of the file that will have no use on the guitar 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 the 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. 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 parser terminates.

### 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.

### 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.

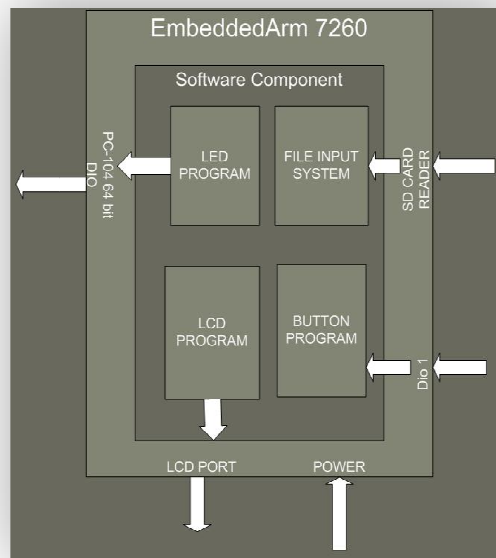


Figure 5 – Block Diagram of Microprocessor Software

## **File Input System**

The file input system is the program that will be reading in the files from the SD card reader that is located on the board. The program will search the SD card and compile a list of music files that are located on the card. This list will be stored in an array and will be the entire path to the file, not just the file name. This array of file paths will be able to be seen and used by the button program, the LCD program, and the LED program.

## **Button Program**

The button program controls the five buttons that will be located on the guitar. These buttons will be attached to the DIO1 on the EmbeddedArm 7260 which means this program will set the DIO1 port to be input only. The five buttons that will be on the guitar are an enter button, a mode button, a stop button, a forward button, and a back button. These buttons will have different features depending on the mode that the guitar is in. The button program will be continually running during every mode so that it can catch when a button is pressed.

The main mode of the program is when there is no song playing and the person is able to search through the list of songs and choose one. When the guitar is in the main mode, the buttons will have the following jobs: the enter button will be used to choose the song that you want and once pressed will begin playing the song that is displayed on the LCD, the forward and back buttons will be used to search through the list of songs that are on the card, the mode button will change the mode to tempo mode, and the stop button will not be used.

The tempo mode is the mode in which the user has the ability to speed up or slow down the speed at which the song is played and the LEDs are lit up. When you are in this mode the enter button will set the tempo to whatever tempo is being displayed on the LCD screen and the program will return to the main mode. The forward and back buttons will change the tempo by a power of two. This means that if the tempo is at the original speed, pressing back will make it play at half the speed, pressing back again will make it play at a quarter of the speed, or pressing forward from the original speed will make it play at double the speed. The stop button does not function in this mode.

The third mode of the program is when the guitar is actually playing a song. In this mode the enter button acts as a play/pause button. If the program is playing and you press enter, the song will pause and if it is paused and you press enter the song will start playing. The stop button will stop the song and return you to the main mode whether the song is playing or is paused. When the song is playing the forward and back buttons will not function, however when the song is paused the forward and back buttons will move the song forward or back a measure at a time. In this mode the mode button will not function when the song is playing or paused.

## **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 signals will be sent to the DIO pins that are located on the PC-104 peripheral, allowing all of the signals to be sent to a continuous address set.

The parsing of the file will occur once the user has selected a song and pressed enter in the main mode. The LED program will read in the file using the files path from the array created in the File Input System. Once the file is read in, the program will go through the file continuously filing a 6 by 25 bit array with the information to light the right frets on the right string. Each row of the array is one string with each value of the row being the information about the frets on the string. As soon as the row is used and the program is on to the next row in the array the previous array will be refilled with the new fret information so that the right frets will once again be lit when the program loops back through the row.

The LED program will continuously loop through the 6 by 25 bit array, where each row is a string and the values in the row are the values being sent to the frets. The values being sent to the frets will be reverse logic since the frets are connected to the cathodes. This means that to light the fifth and seventh fret of the first string, the first row of the array will have zeros at the fifth and seventh position, and ones in all other entries. Only one string will be lit up at a time meaning one row of the array will be sent to the 25 fret pins at one time. The looping through each row of the array will be fast enough that it will look like all strings are being lit simultaneously. When a string is being lit, a one is being sent out of that strings DIO pin and the five other DIO dedicated to the strings will be zero. This process will allow us to light any combination of LEDs using only 31 DIO pins.

## **LCD Program**

The LCD program will be used to control what is being displayed on the LCD at all times. The LCD will display different information depending on which mode the program is in. When the program is in the main mode the program will display two lines of information. The first line will be the name of the song and the second line on the right side of the screen will be the speed at which the song will be played (original, half, etc.). When the program is in the tempo mode, the LCD will still display two lines of information. When in this mode the first line will say TEMPO and the second line will again show the speed that you are currently at on the right side. Finally, when the program is in the playing mode the two lines of information that will be displayed are the name of the song on the first line and on the second line, the measure that the song is on and the speed at which the guitar is being played.

## **Chapter V – Guitar Construction**

### **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.

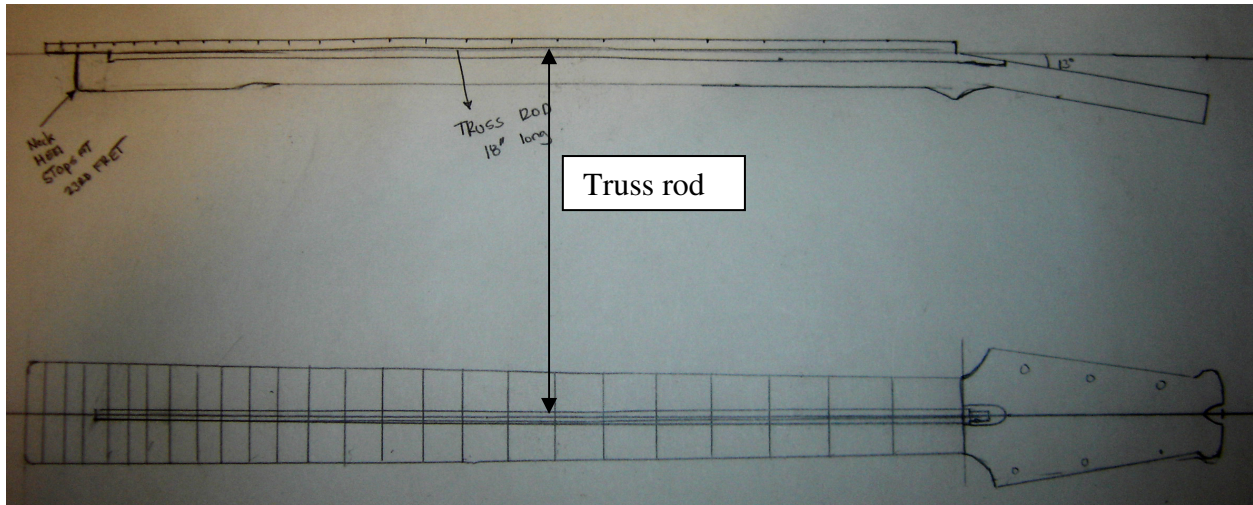
Now that the channeling and wood work is finished on the fingerboard, The LED's need to be installed. Once the LED's are installed they will be cemented in place with epoxy to prevent them from jiggling loose during normal wear of the instrument. The epoxy will also help restore strength to the fingerboard and neck and reduce noise. Once the fingerboard is epoxied it will be ready to glue to the neck.



**Figure 6 – Dremel Routing Jig**

### **Neck**

Then neck will be constructed out of 3 pieces of wood glued together. The reason we will use 3 pieces of wood is for extra strength against natural warping of the wood. After the wood is glued together a shape will be cut from it. A 3/8" deep truss rod slot will then have to be 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**

## **Body**

The body of the guitar will be constructed from mostly softer materials. The softer woods make it easier to achieve clean routes. The Body will first be rough cut with a band saw and then refined later. There will need to be several cavities routed out in order to make room for components. The microcontroller will sit between the bridge and the bottom of the fingerboard underneath the strings. The LCD and buttons will be located on the pick guard underneath the strings. The Battery pack will be on the back of the body near the tail.

## **Finishing Touches**

Once the neck fingerboard is joined together the frets will be installed. Then the complete neck and body will be ready for finish. A stain will be applied in order to change the wood to a green color. After stain is applied, The full instrument will receive 4 coats of nitrocellulose lacquer and buffed once dry. When finish is completed all of the hardware and electronics will be installed.

## Chapter VI – Conclusion

Many things have been accomplished so far but we still have many more things to accomplish next semester. One problem we have to solve before this will be possible however is getting the microcontroller to provide the number of output pins we require. Research has shown that we simply need to add another peripheral device which easily gives us more IO pins than we need. We have also had difficulty meeting timeline requirements set at the beginning of the semester and have had to revise them several times. We feel the progress we are making will begin to increase because we have already learned a lot about how to program the microcontroller to perform the functions we want. This should allow us to stay on schedule much better than we have been able to so far.

The first step next semester is to get a simple simulation running on our microcontroller. This simple simulation program will play a song lighting up LEDs in the correct order and timing to show a user how to play the song. This program will not implement tempo control, song select, or play/pause/stop features. After the simple simulation is working, we will begin adding more functionality to it. First, we will develop the file input program so we can transfer songs to the microcontroller using the SD card interface. Then, we will develop the button program so we can implement the play/pause/stop feature so the user can choose when to play the song. After we have obtained an LCD screen we can develop the program that will display the song information to the screen. Once we have this working, we should be able to add the ability to select the song to be played from a list of the songs stored on the microcontroller. The final function, which we believe to be the most difficult to implement, is the tempo control. This is because we have to develop a successful and efficient method to scale the duration each LED is turned on, based on the speed the user selects. Once all these software components are complete, we will have the fully functional Interactive Guitar Teaching Program.

Based on the purchases we have made this semester, and still need to make, we will be running approximately \$50 dollars over budget. We hope to prevent this from happening by attempting to get the remaining components, the LCD screen and extra peripheral, donated to our project. If these attempts are unsuccessful we will likely look into some form of fundraising project so that we do not have to donate our personal funds to the project.

We would have liked to have made more progress on the project by this time but we are staying optimistic for next semester. Everyone on the team is confident that we will have a complete prototype of our Interactive Guitar ready for presentation at E-Days by the end of next semester.

## 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 Dr. William Eads for sponsoring and guiding this project.

## Appendix A – Abbreviations

DIO – digital input/output  
LED – light-emitting diode  
LCD – liquid crystal display

## Appendix B – Budget

**Initial Budget** \$400

Date	Item	Total
10/2/2008	Embedded Arm TS-7260 Single Board Computer	\$195.00
	Fret LEDs	\$52.80
	Prototype LEDs	\$16.20

**Current Total** \$264.00  
**Remaining Budget** \$136.00

### Future Costs

LCD Screen (expected cost)	\$20.00
PC-104 64-bit DIO Board	\$69.00
Construction Materials (expected cost)	\$150.00

**Projected Total Cost** \$503.00  
**Projected Remaining Budget** (\$103.00)