Interactive Teaching Guitar

The future of interactive music education

Jeremiah Young
Colorado State University
College of Engineering
Fort Collins, United States
jeremiah.young@rams.colostate.edu

Michael Ullmann
Colorado State University
College of Engineering
Fort Collins, United States
mikey013@rams.colostate.edu

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.**

Keywords - Music; Guitar Training; Interactive Intrument

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 they can use multiple learning techniques on that section. 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.

We will begin by discussing the hardware aspect of the project. First we will look into the requirements for each of the guitar's components. Then we will proceed to discuss the final decision made pertaining to the hardware. We will finish up the hardware by discussing future improvements and optimizations of the guitar's hardware.

We will then move to discussing the software that was used on the board. We will begin this section by discussing what the functions and capabilities we needed from the software. Next, discuss each of the programs and functions that were used to receive these capabilities. Finally, we will discuss improvements that should be made to the software form both a features point of view and an optimization point of view.

II. HARDWARE

A. Requirements

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.

2) LEDs -

The LED's must be small since there will be 150 of them positioned under the fret board in the neck of the guitar. Due to the large number of LEDs, each LED must consume very little power as this guitar is battery powered. The final requiremtn is that they must be bright enough to be visible in a well lit environment.

3) LCD Screen -

The LCD screen required for the guitar must be large enough to display the song name, tempo, note value, and an indicator that shows if the song is playing or paused. However, the LCD screen must also not be too large

because it must be positioned in a limited space on the guitar pick guard, located below the strings. Ideally the LCD should have low power consumption, but this is not a requirement.

B. *Implemented Hardware*

1) Microcontroler –

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. 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 is specifically designed to consume low amounts of power, so the batteries will last for a longer time. 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.

2) LEDs –

The LED's that were selected were provided by Lumex and they were through-hole green LEDs. The reason for the through-hole was because it was the most applicable for the neck of the guitar given the limited space.

3) LCD Screen –

The LCD screen that we purchased was a 2x24 Alphanumeric screen with LED backlight. Having two lines of information to display allows us to have the menu title, such as choose song, on the first line , and display the users current option on the second line, such as the song name. Being 24 characters wide is plenty to display the song name or tempo information. The LCD screen was also purchased through embeddedArm which allowed us to just connect the screen to the LCD port on the board and have it work.

C. *Future Optimizations/Improvements*

For the future application of this design it would need to optimized with respect to the board that is being used. The board satisfies our needs but has too many costly features such as USB and Ethernet ports that are not being used. The board also does not contain enough ports needed alone so the extended cost of the PC/104 TS-DIO64 is not necessary. The LEDs are currently illuminated but not as bright as they could be. Some possibilities to resolve this include using a smaller exterior resistor from the output of the board. Another optimization related to power would be to add a charging system to the board and incorporate the use of rechargeable batteries. This would help the cost of constantly purchasing new batteries.

III. SOFTWARE

A. *Required Capabilities*

1) File Input System -

The file input system must be capable of searching through a given directory and compiling a list of the song files that are on available for the guitar to use. The program then must be able to store this list in order for the other functions to use this list.

2) LCD Screen -

The LCD screen needs to be able to display the menu title or question on the first line and the user's current choice on the second line when the user is making the song's selection and then the song's timing selection. Once the song is playing the screen should show the song name on the first line. On the second line it should display if the song is playing or paused, the beats per minute, and the note value a single beat has.

3) LED Finger Placement Display -

The LED finger placement display needs to be able to light any combination of the 150 LEDs for the correct amount of time specified by a given note.

4) File Parser -

The file parser must be able to read in the song files and parse out the information that is needed to set up the timing, send the LCD display all of the correct information, and send the correct information to the LED finger placement function.

5) Button Program –

The button program will be used in several of functions. The button program should allow the user to search through the songs on the guitar and then selct the song. The program should then allow the user to adjust the tempo. Finally, when the song is being played, the buttons should have play/pause and stop capabilities.

6) Additional Features -

The software must also have some additional features that are used for the safety and reliability of the system. The first of these features is that the program must boot directly when power is applied to the board, instead of having to be called by command line in the linux terminal. The program must also be able handle any errors that occur, and restart itself without the user.

B. *Software Implementation*

1) File Input System –

The file input system is the first program that runs when the guitar is powered on. The program searches the root directory of the board and compiles an array of all of the song

files that are in that directory. 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.

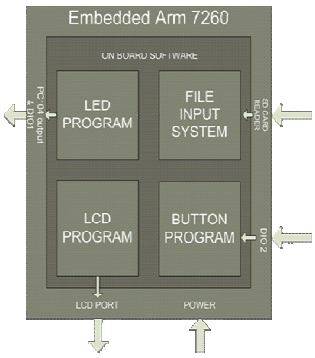

Figure 1 – Block Diagram of Implemented Software

2) LCD Screen –

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 not, 8 to an eighth note, etc.

3) LED Finger Placement–

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 LED program takes in seven arguments, the note type, then the string 1 through string 6 fret values. 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 a 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. 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

4) File Parser –

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. 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.

5) 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.

For the button on the back should only be used by a programmer or developer to change the main program 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 they 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 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.

6) Additional Features

In order to handle the reliability issue, a function that caught segmentation faults was added to the program and was called in each of the other functions. The segmentation fault function would catch any type of segfault that occurred. It would then display a message telling the user there had been a problem and to choose a new song on the LCD screen. Finally the program returned the program to the choose song menu. In order to get the program to start directly on board power up was accomplished by writing a startup script and putting it in the startup directory on the board.

C. Future Improvements/Optimizations

Although the interactive guitar has met all of the original requirements, after developing the guitar there were several features that we believe should be added. The first feature is to enable the use of the SD card on the board instead of having the song files on the guitar itself. This feature would allow the user to easily expand the number of songs the guitar was capable of having and would allow the user to easily change which songs were on the guitar. This feature would also mean that the user would never have to physically connect the guitar to a computer for any reason. The next feature would be to add a headphone jack that had a metronome beat in order to help the user stay at the correct pace with the lights. The third feature would be to allow the user to be able to move forward or back a measure in a song when the song was paused. This would allow the user to immediately go back if he missed a section, or if a part of the song is troubling the user they could skip straight to that section to practice. The last feature would be to allow the user to have more choices for tempo change and even allow them to change the speed in the middle of a song. An optimization that should be made in the code is to turn the LED backlight of the LCD screen off after 5 seconds of inactivity. This optimization would low the power consumption of the guitar and allow for longer battery life.

IV. CONCLUSION

In this paper we have discussed the development of a guitar that should have the ability to help people learn to play faster and easier. We have discussed the hardware and software that was required of the guitar for it to be able to help people, the implementation of this hardware and software in the guitar, and finally the hardware and software optimizations and improvements that still need to be done on this guitar.

This guitar is very functional and has the ability to help people learn guitar as it is now. However, with the hardware and software improved as we have discussed in this paper, this guitar will be able to take the next step forward and become a tool that could be expanded to many other electric string instruments.

ACKNOWLEDGMENT

This project was funded by Colorado State University - Electrical and Computer Engineering Department. Guitar construction provided by Joseph Molaskey. Support software provided by Tim Stansbury.

REFERENCES

[1] TuxGuitar
http://www.tuxguitar.com.ar/tgwiki/doku.php

[2] TS-7260 Manual
http://www.embeddedarm.com/documentation/ts-7260-manual.pdf

[3] TS-7260 Schematic
http://www.embeddedarm.com/documentation/ts-7260-schematic.pdf

[4] Getting Stated with TS-Linux ARM
http://www.embeddedarm.com/documentation/software/arm-tslinux-ts72xx.pdf

[5] Linux for ARM on TS-72XX User's Guide
http://www.embeddedarm.com/documentation/software/arm-linux-ts72xx.pdf

[6] Linux for ARM on TS-7000 Embedded Computers
http://www.embeddedarm.com/software/software-arm-linux.php

[7] Source Code Samples for the TS-7200 Series
ftp://ftp.embeddedarm.com/ts-arm-sbc/ts-7200-linux/samples/

[8] Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification
http://java.sun.com/j2se/1.4.2/docs/api/index.html