ECE 477: Digital Systems Senior Design

Initial Project Proposal

Year: 2016-2017 **Semester:** Spring 2017 **Project Name:** Guitutar
Creation Date: November 3, 2016 **Last Modified:** November 3, 2016

Team Members (#1 is Team Leader):
Member 1: Austin Peterson **Email:** peter174@purdue.edu
Member 2: Brian Rieder **Email:** brieder@purdue.edu
Member 3: Cole Giannotti **Email:** agiannot@purdue.edu
Member 4: Jennifer Isaza **Email:** jisaza@purdue.edu

1.0 Description of Problem:

Guitar is one of the most common instruments people try to pick up and learn, but if you have never played an instrument before, this could be a difficult feat to accomplish. Lessons may be too expensive, or maybe you are taking lessons but want to improve your practicing efficiency to show your teacher that you can move on to harder songs. One of the unique parts about learning guitar compared to other instruments are the chords and strumming combination. Tablature, or tabs, was created to be an alternative to learning how to play a song without learning how to read sheet music. While tabs are easy to find, guitar sheet music is usually not made and strumming patterns aren't connected with the tabs. In order to learn the combination of notes and strumming, users have to listen to a song over and over or find an instructional video on YouTube.

2.0 Proposed Solution:

As the process of learning to play any instrument is difficult, our goal is to simplify this ordeal for someone trying to learn to play the guitar. In order to simplify the learning process, we have chosen to create Guitutar a guitar tutor that is implemented in the neck of an electric guitar. The custom fretboard will help its user learn songs and scales by lighting up fingerings on the correct frets, recognize correct chord and note inputs, and potentially indicate strumming patterns for songs.

The guitar tutor is realized as an array of LEDs embedded into the fretboard that illuminate based on the selection of a preset song on the guitar. This illumination will go note-by-note and progress as the user correctly plays the notes that it has indicated by recognizing these inputs via a pitch recognition system such as a microphone, tensiometer, or pressure sensor within the fret. This serves to progress the user through a learning process similar to a tab and can allow user control through a small interface to move forward and backward within a song in order to practice specific sections with the proper illumination.

Alternative implementations of the illumination system have an overlay on the fretboard in which there are embedded LEDs, however this approach may dampen the sound output from the

guitar and requires external interfacing. Packaging of a custom guitar neck was chosen due to its solid nature and the ability to internalize wiring by routing it through a channel in the neck to support ease of playing and evade the appearance of external wiring.
Pitch recognition can be done in a variety of methods, but is currently planned to be employed by acquiring an analog signal from the guitar and performing signal processing to disseminate fundamental and harmonic pitches from the signal and matching with the expected value. Further detail regarding pitch recognition through a microphone and proof of concept through simulation is shown within Appendix 2.

3.0 ECE 477 Course Requirements Satisfaction

3.1 Expected Microcontroller Responsibilities

In order to realize this design, utilization of at least one microcontroller will be required. Microcontroller duties will include the driving of the LED array that serves as the indicator to the user for which notes to play as well as implementation of pitch recognition through processing of the signals input when the guitar is played. This pitch recognition requires the microcontroller to have the ability to receive inputs from an external interface such as a microphone, tensiometer, or pressure sensor and have the processing power to be able to perform interpretive logic on the signals from this external interface. Both a pressure sensor and a tensiometer yield analog outputs that are easily measurable by most microcontrollers. In the instance of a microphone, the microcontroller must have the processing power to be able to perform Fourier transforms and frequency analysis in order to discern between fundamental and harmonic pitches input through noisy analog signals. Additionally, the ability to interface with flash memory, as most microcontrollers are able to do, in order to store preset songs that the microcontroller will interpret and display to the user.

3.2 Expected Printed Circuit Responsibilities

The printed circuit board for the design will hold the microcontroller and all of its associated connections in order to drive the system. There are three currently considered options for the microcontroller: Beaglebone Black, a Raspberry Pi 3, or TI Tiva C. Each of these microcontrollers have distinct advantages, but all are able to be located on the printed circuit board itself. Due to the speed required in order to do real-time signal processing, the ARM M4 has a significant advantage due to its fast clock speed.
The design requires the implementation of a PCB in order to provide an interface for the microcontroller to the external peripherals attached to the guitar. These peripherals will include a large fret-string associated LED array, a device interface containing a display and controls to interact with the features of the device, an input recognition system such as a microphone, tensiometer, or pressure sensor in order to recognize the pitch being played and analyze for

correctness, flash memory in order to store preset songs for the system to display and interpret, and voltage regulation hardware in order to support all of the above peripherals and interfaces.

4.0 Market Analysis:
Not everyone has the time and money to spend hiring a teacher to help them learn an instrument. In addition, some people would like the assistance of a teacher in addition to learning tablature. This product is directed more to people that would want to learn how to play the guitar with a supplement. A device such as this would allow someone to pick up any guitar and assist them in learning how to play the guitar.

5.0 Competitive Analysis:
There are three ways in order to learn how to play the guitar: hire a teacher and follow his/her instruction, learning and reading tablature, and self-teaching through books and videos. In order to make the learning process simpler and faster, a device that guides a user through a song by lighting up which fret and string to play will allow them to learn the song. Current products on the market include the illuminated fretboards and play through the song, but they do not do a good job in teaching the musician to learn the mechanics of playing the song.

5.1 Preliminary Patent Analysis:

5.1.1 Patent #1: US9218747 [1]
Patent Title: Self-Teaching and Entertainment Guitar Systems
Patent Holder: James Bartos
Publication Date: Jan 20, 2011

This patent describes of a guitar with illuminated holes with fitted LEDs that selectively illuminate showing which string and fret is played according to a song. While our product will utilize a similar feature to indicate which note to play, our project will include pitch recognition so the device knows if you played the note. The device will wait for the user to strum the correct note or chord before it moves on.

5.2 Commercial Product Analysis:

5.2.1 Fretlight Electric Guitar:
The Fretlight Electric Guitar uses a laptop or app to be able to load the chords, songs, and riffs to the guitar. The fretboard then illuminates what fret and string to press. The user would then strum the guitar. Our projects seeks to use a similar method of note and chord indication. In addition, we are utilizing pitch recognition in order for the device to understand if the correct chord or note is played and become more of a teacher rather than showing how it should be played.

5.2.2 Rocksmith:

Rocksmith is a game/software that allows a user to connect an electric guitar to their console or PC to be used as a controller for the game/teaching software. Rocksmith uses pitch recognition in order to determine if the correct note or chord is played, acts as a song teacher by waiting until the correct note is played and also giving a percentage of notes played correctly when played at a certain tempo. Our project will do most of the features that Rocksmith provides without the need of owning a PC or console and software/CD. In addition, Rocksmith requires people to purchase the song and the associated software for Rocksmith to interpret the song.

5.3 Open Source Project Analysis:

5.3.1 Beethoven [2]:

Beethoven is an audio processing library that serves to provide an interface for detection of musical signals. Operation consists of transforming source signals into a processable format and applying a pitch estimation algorithm chosen by the user to discern the fundamental frequency. This project serves the same purpose as our project's pitch analyzing feature by removing harmonics from a noisy signal and returning the fundamental that is to be used for matching the pitch. This project is licensed under the MIT license by Vadym Markov and is available "as is" for usage in any project as long as the license is propagated to that project.

5.3.2 California Polytechnic State University [3]:

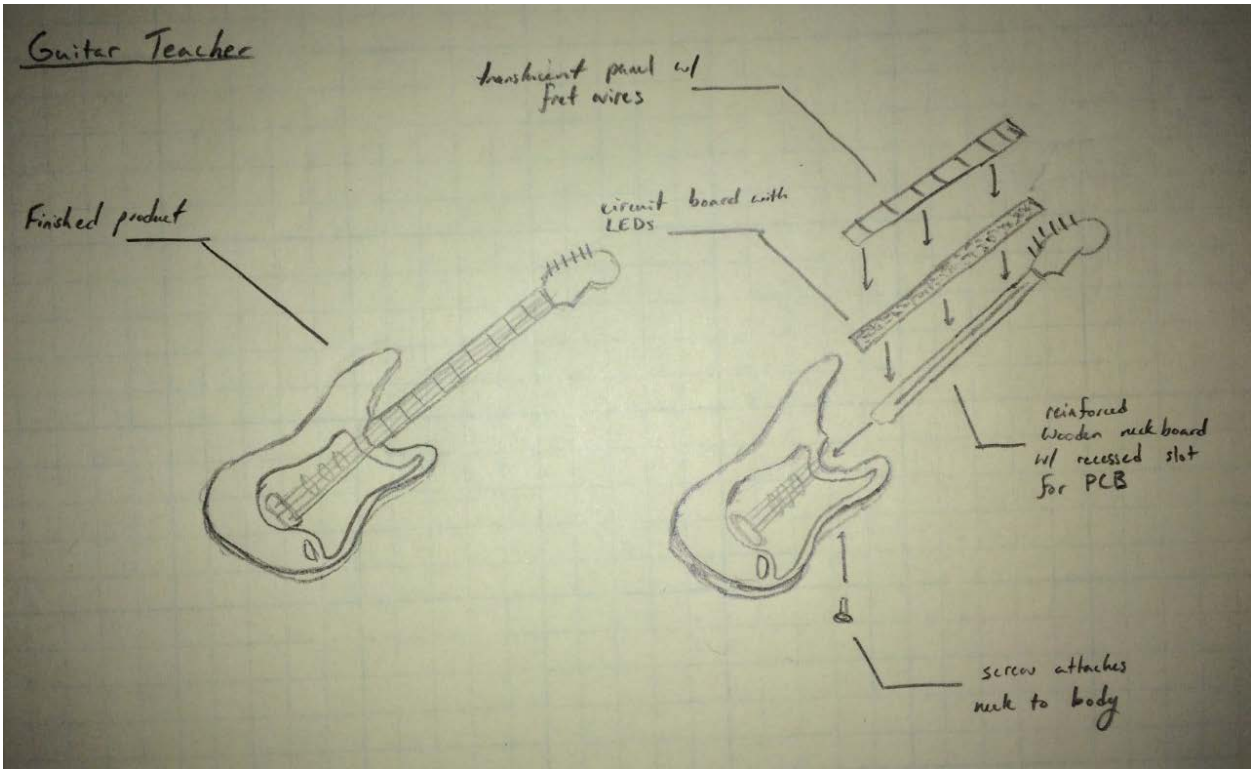
This project is not primarily geared towards ease of teaching guitar playing, but specifically has to deal with the ability to identify the pitch and octave of notes in real-time. While other projects of a similar type have pre-recorded pitches and templates for how specific notes look and sound, this project takes it a step further and automates the process to generate these pitches and templates at run-time for the specific guitar. This project is substantially detailed and uses this information to generate a visualization of the results and output tablature and a fretboard display. Since this project is so specifically detailed, the generation of this tablature is substantially different from our display on the fretboard, but the sections regarding detecting pitches of single notes (2.2.1) and chords (2.2.2) within the project's paper is substantially useful and align closely with our current methods.

6.0 Sources Cited:

[1] J. Bartos, “Self-teaching and entertainment guitar systems,” U.S. Patent 9 218 747, January 20, 2011.
[2] V. Markov. (2015). *Beethoven* [Github Repository]. Available https://github.com/vadymmarkov/Beethoven.
[3] J. Hartquist, “REAL-TIME MUSICAL ANALYSIS OF POLYPHONIC GUITAR AUDIO,” M.S. thesis, Dept. Computer Science, California Polytechnic State University, San Luis Obispo, CA, 2012.

ECE 477: Digital Systems Senior Design

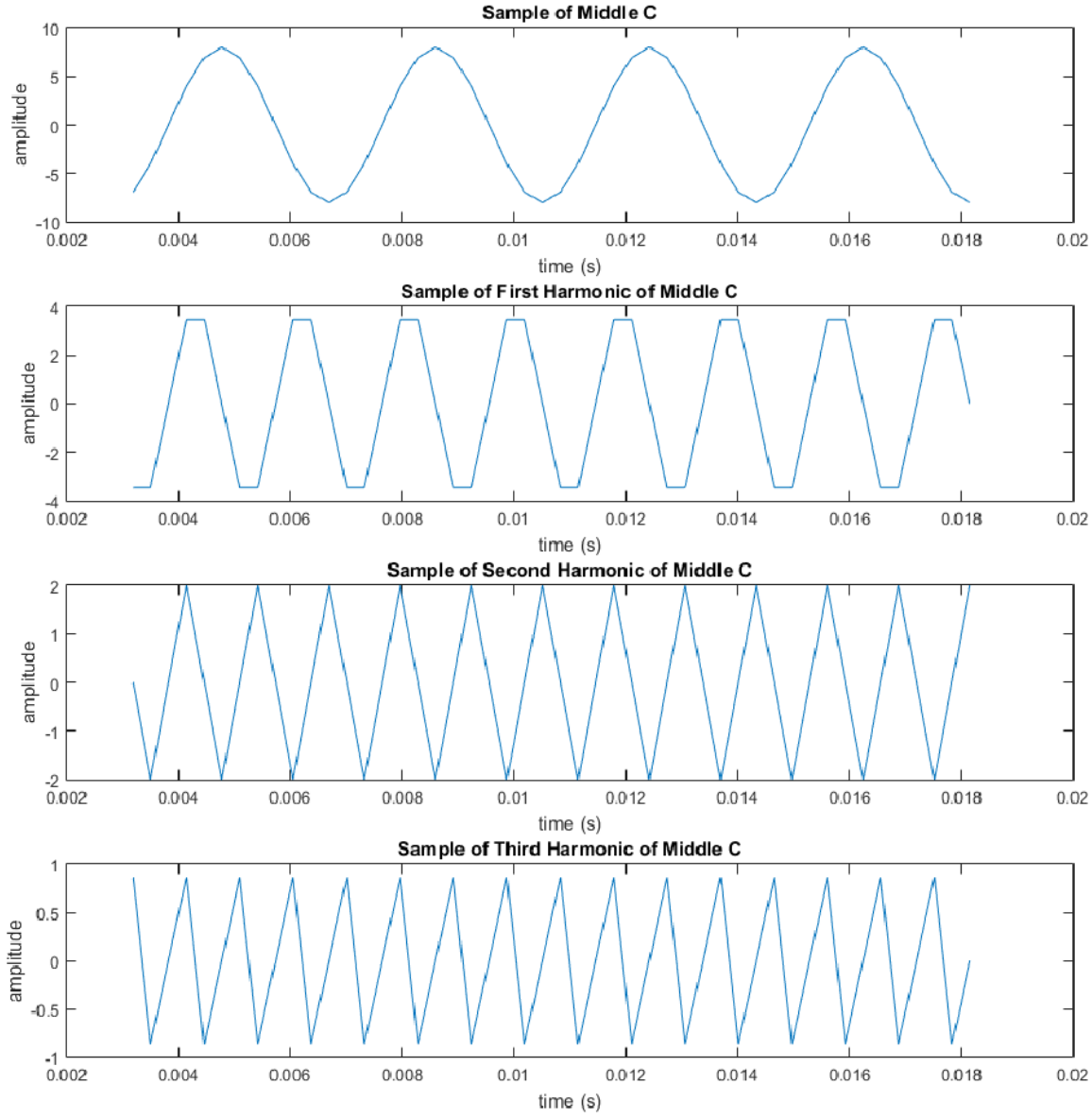
Appendix 1: Concept Sketch



Appendix 2: Simulated Proof of Concept - Microphone Signal Analysis

```
middleC = 261.6; %Hz

fMax = 4*middleC;
fNyquist = 2*fMax;
fSample = 3*fNyquist/2;
t = 0:1/fSample:3;
```

```
h0 = 8*sin(2*pi*1*middleC*t);
h1 = 4*sin(2*pi*2*middleC*t);
h2 = 2*sin(2*pi*3*middleC*t);
h3 = 1*sin(2*pi*4*middleC*t);

[~,l] = min(abs(t-4/middleC));
[~,phase] = min(abs(t-0.75/middleC));
s = 1+phase;
e = l+phase-1;

figure('Position', [100,0,1024,1200]);

subplot(4,1,1)
plot(t(s:e), h0(s:e))
title('Sample of Middle C')
ylabel('amplitude')
xlabel('time (s)')

subplot(4,1,2)
plot(t(s:e), h1(s:e))
title('Sample of First Harmonic of Middle C')
ylabel('amplitude')
xlabel('time (s)')

subplot(4,1,3)
plot(t(s:e), h2(s:e))
title('Sample of Second Harmonic of Middle C')
ylabel('amplitude')
xlabel('time (s)')

subplot(4,1,4)
plot(t(s:e), h3(s:e))
title('Sample of Third Harmonic of Middle C')
ylabel('amplitude')
xlabel('time (s)')
```


Sample of Middle C
amplitude
time (s)
Sample of First Harmonic of Middle C
amplitude
time (s)
Sample of Second Harmonic of Middle C
amplitude
time (s)
Sample of Third Harmonic of Middle C
amplitude
time (s)

```
harmonics = h0 + h1 + h2 + h3;
[dH, omega] = DTFT(harmonics, 512);

figure('Position', [100,0,1024,1200]);

subplot(2,1,1)
plot(t(s:e), harmonics(s:e))
title('Harmonized Middle C')
ylabel('amplitude')
xlabel('time (s)')

subplot(2,1,2)
f = omega * fSample / (2*pi);
plot(f, abs(dH))
title('Frequency Analysis')
ylabel('magnitude')
xlabel('frequency (Hz)')
```