Guitar Learning and Feedback Tool

ECE 445 Design Document

Dillon McNulty, Kyle Gibbs, and Oumar Soumare
Group 62
TA: Jonathan Hoff
February 25, 2020

Table of Contents

Introduction

1.1 Objective

Many people have had the desire to learn how to play the guitar at some point in their lives. For some, the large amount of time and effort required to even master the basics can serve as a barrier to entry to the musical world. The guitar is an incredibly popular example of one of these instruments. While there are some popular pieces of technology already in existence to help you get started like Yousician [1], Fret Zealot [2], Fender Play [3], and Jamstik [4], there are several issues with these systems that we believe can be fixed.

Our project looks to address that problem. By synthesizing sheet music into a measure-by-measure breakdown, the user will be able to practice individual measures or sections at a time and receive instant feedback on their performance. This system will greatly accelerate the learning process, as users can receive detailed information on where they are playing well and where they are struggling to perform. By repeatedly practicing difficult sections, they will build the muscle memory and comfort needed to play a guitar well.

1.2 Background

While systems like Yousician [1], Fret Zealot [2], Fender Play [3], and Jamstik [4] have proven to be successful with some people, we believe these products have some faults and would not be as beneficial to the learning process as our proposed system. Fret Zealot [2] is only available on your phone and has the lights atop the fretboard itself, underneath the strings. When the LEDs are situated this way, the user has to crane their neck over the guitar to see where to put their fingers. By placing the LEDs on the side of the fretboard where they are easily visible (see *Figure 1* below), they will learn to play the correct strings by learning to feel where individual strings are in accordance with the lights rather than just placing their fingers wherever it lights up. In Yousician [1] and Fender Play's [3] case, they require a connection to a PC, Tablet, or Smartphone app (including a paid subscription) and record the user's input via a microphone to determine accuracy. Additionally, most information that could be found for Fender Play [3] was locked behind a paywall until a subscription is activated. This is a problem because not only are they expensive applications, but are not as accurate as having a hardlined connection to a computer. By developing a more reliable system with a one-time purchase, we find this to be superior to what Yousician and Fender play can offer.

While Jamstik [4] offers more of a full-package, it is $180 out of the box and is a proprietary section of a guitar with only 7 frets to play on - not an actual guitar whatsoever. Just from the video it looks very uncomfortable as there is nowhere to rest your elbow and cannot be nearly as comfortable as an electric guitar like the method we've designed uses. In addition, a broken string on a Jamstik would require a professional replacement due to the unique design.

The other issue with current systems that arise with both of these platforms is the lack of sectional playback. In all of the systems I described, each song can only be played in its entirety when the user wishes to practice. By having a simple LCD user interface displaying the current measure of the song, the user can focus on the experience of playing the guitar itself rather than trying to react to notes flying across their small phone screen. By allowing variation of tempo and measure length sequences, the user can immediately progress to more difficult versions of the song that they are playing. An approach similar to the game 'Simon' [5] where the lights display the sequences and the user plays them back would be best, as they are developing the habit of playing sequences of notes instead of reacting to lights coming up as quickly as possible.

1.3 High-Level Requirements

I. System is able to detect correct notes played by the user with >90% accuracy and able to compile detection into a feedback metric detailing correct/incorrect notes for each measure and the entire song.
II. Must be able to fetch song data and convert to an LED sequence with <30sec delay to allow the user to play the song.
III. System must allow the user to incrementally increase the difficulty by changing the tempo of the notes to be played (from 40 - 160BPM), and the amount of measures to play in a given sequence.

Design

2.1 Block Diagram

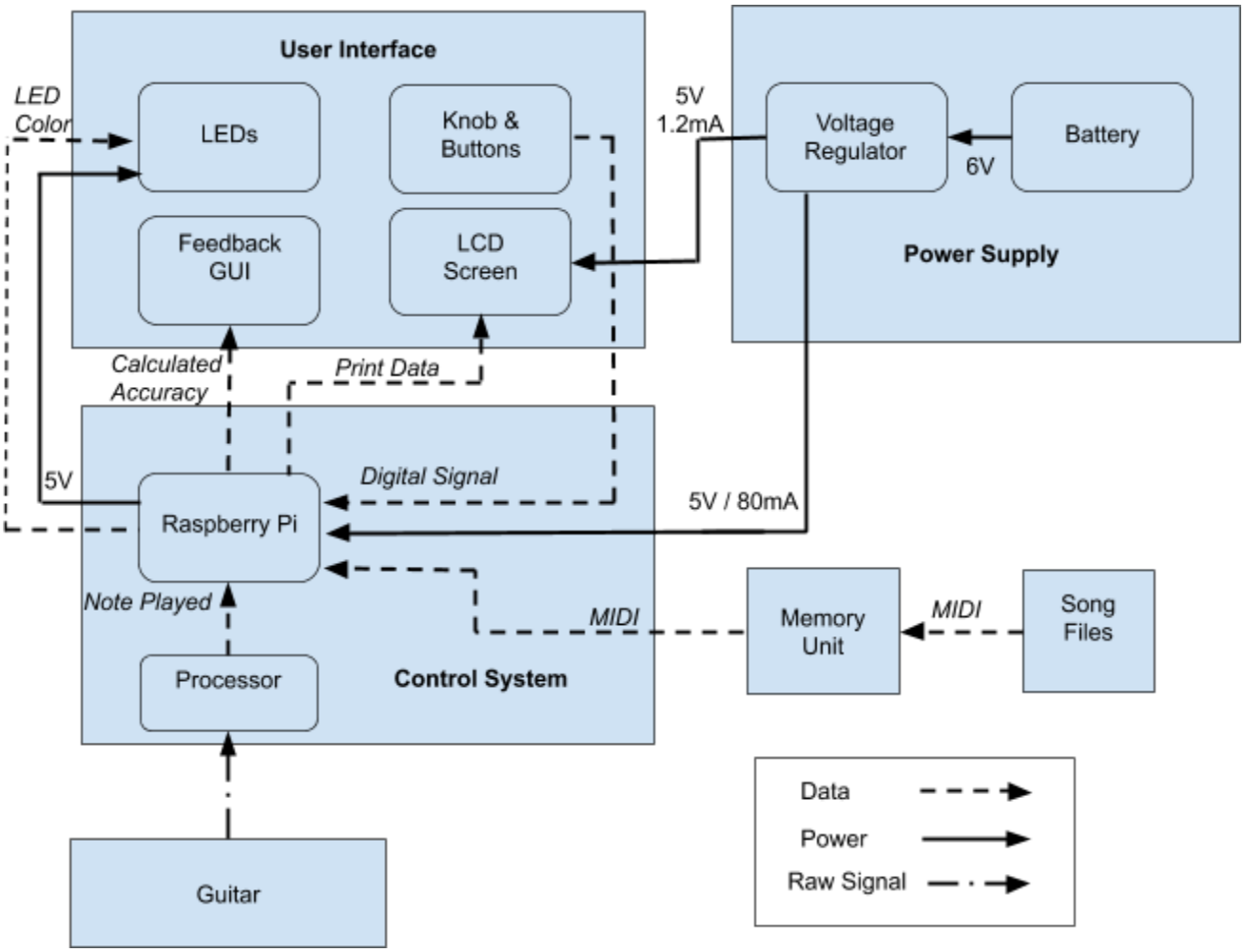

Figure 1: Block diagram for guitar learning and feedback tool. Shows how components will interact with each other and the types of connections between them.

2.2 Physical Design

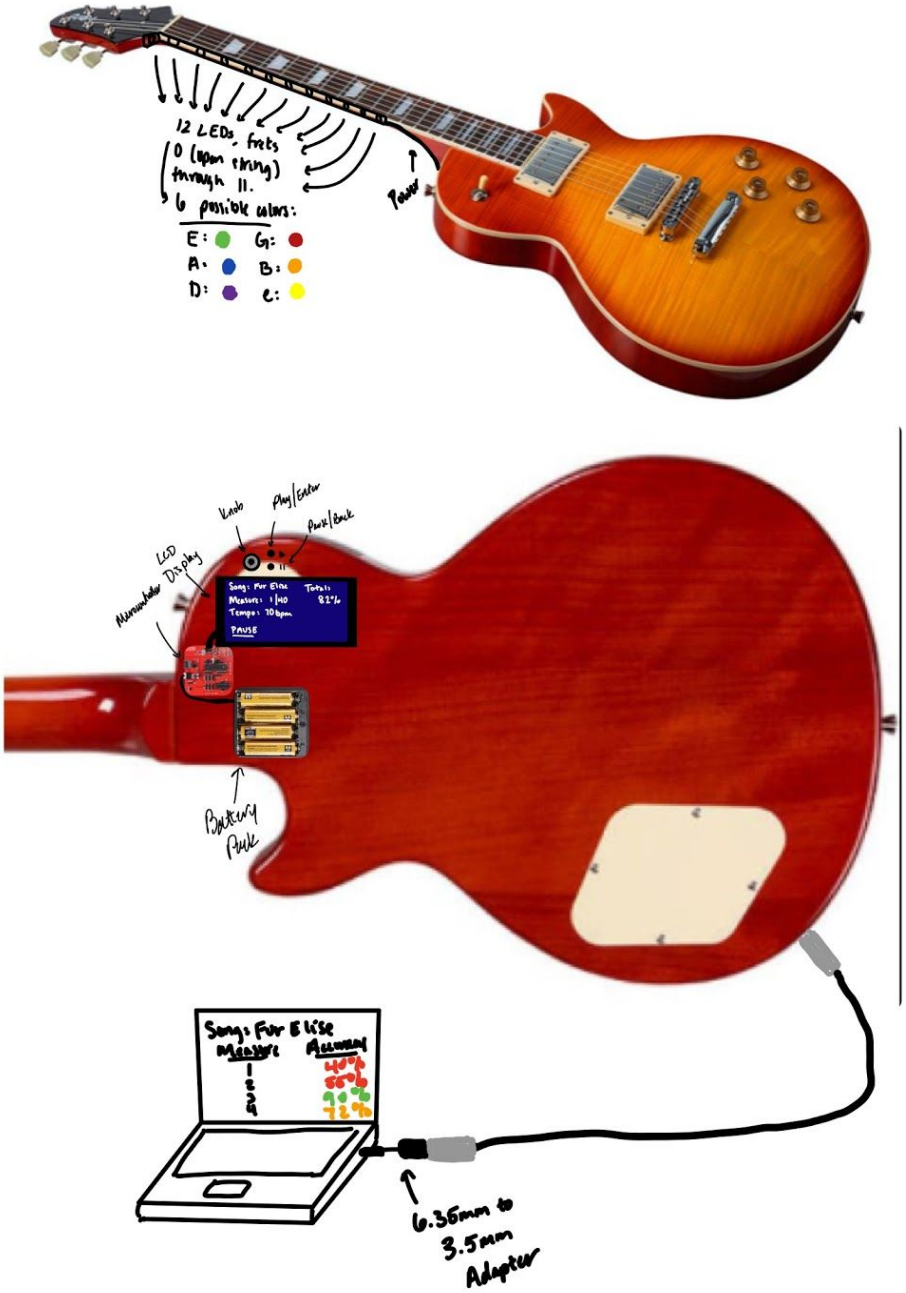

Figure 2: Design explaining the physical appearance of the tool. The 12 LEDs on the side of the fretboard will emit one of six different colors, to indicate the string to be played for that fret.

2.3 Subsystem Descriptions

2.3.1: User Interface

I. The user interface provides the user the ability to select a song, receive instructions on how to play it through color coded LED's, and receive feedback on how well the user played the song.

II. Parts Needed:

A. Knob & Buttons: The knob (**KY-040 Incremental Encoder**) [6] and buttons (**Tactile Button System**) [7] will allow the user to select from a list of songs and the skill level/tempo the user would like to play on. It also provides a method to pause, rewind, or fast-forward a song.

B. LEDs: An LED (**WS2812 RGB**) [8] will be placed along each fret and will be color coded to show which string should be played on that fret.

C. LCD Screen: The LCD screen (**20x4 Character LCD**) [9] will provide the user with a visual queue of the currently selected song, skill level and measure.

D. Feedback GUI: The Feedback GUI will be a Python application displayed on the user's computer that allows them to receive feedback on how well the user played during each measure. This feedback will be provided as a percentage of how the user's measure compares to the original measure.

III. Finite State Machine:

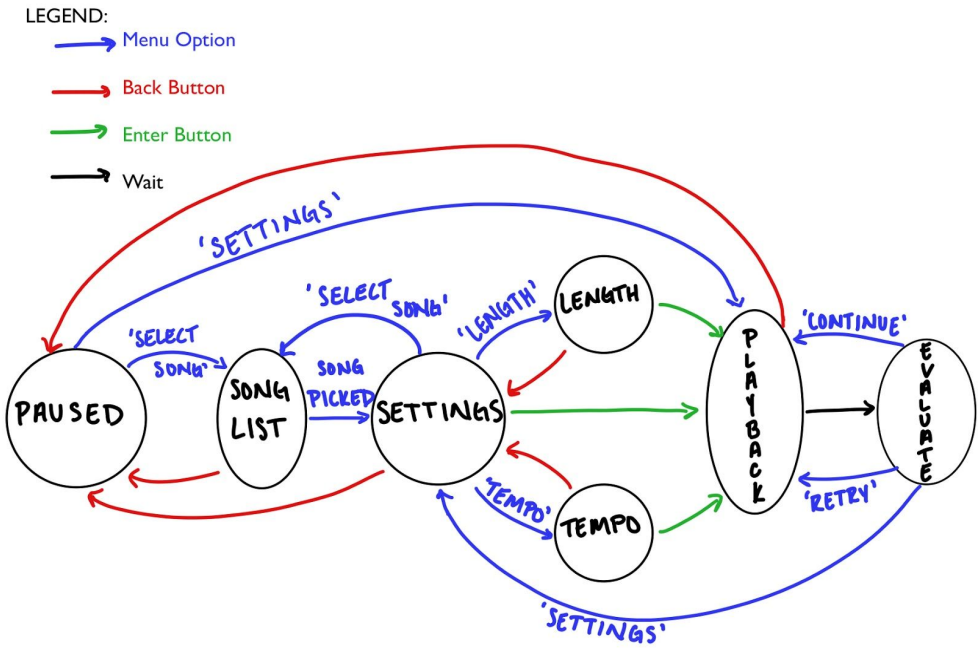

Figure 3: Our user interface will change based on the knob and button inputs.

IV. *State Descriptions:*

A. PAUSED: Nothing is happening. The LCD displays a menu of 'Select Song', 'Playback Settings', or 'Resume Playback' (if a song has been picked)

B. SONG LIST: The LCD screen displays the list of available songs which can be scrolled through and selected for playback.

C. SETTINGS: Once a song has been selected, the screen displays the tempo and measure length. The user can either choose to immediately begin playing, or to adjust each setting. This state can also be reached at any time from selecting 'Settings' in the PAUSED state during playback.

D. TEMPO: The LCD displays the current tempo being played. The user can turn the knob to adjust the tempo up or down (between 40 and 160). The user then needs to press the confirm button for the changes to take effect.

E. LENGTH: This is the same situation as TEMPO, except for adjusting measure sequence length rather than tempo.

F. PLAYBACK: The LEDs are now lighting up according to the song being played. The LCD displays the song name, current measure, and current tempo.

G. COMPARE: After the microcontroller completes playback, it lights up the same sequence again to allow the user to play along. It records the notes played by the user and compares them. After displaying the completion percentage to the user, they can choose to either retry with the same settings, continue the current song, or go back to the settings. (If the song is complete, a performance metric will be generated and the user will immediately be sent back to the settings to increase difficulty and continue).

V. *Proposed Schematic*

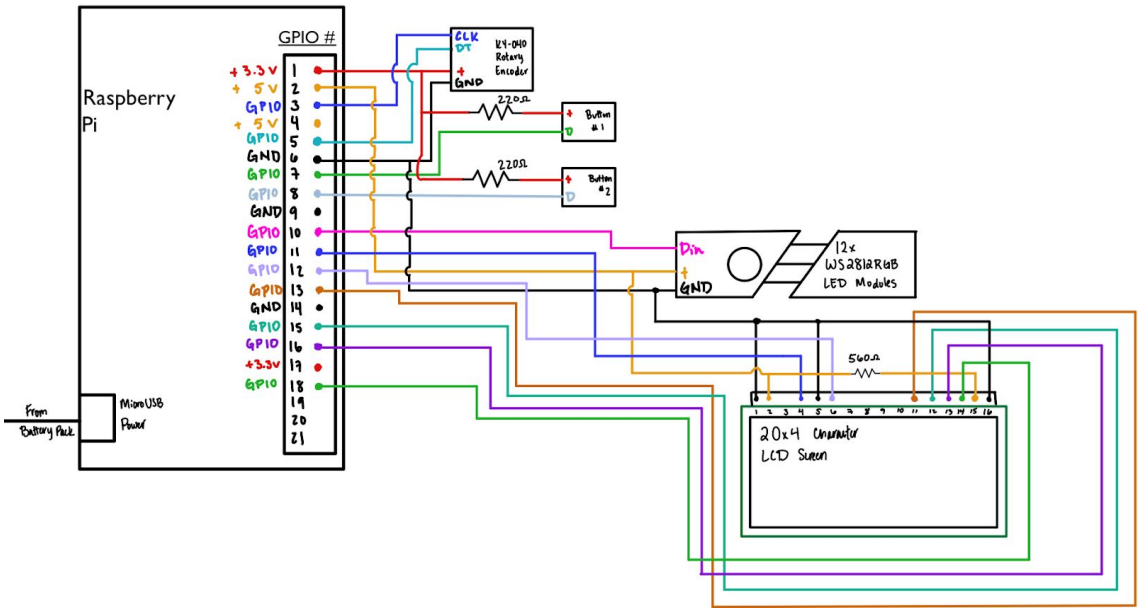

Figure 4: Proposed hardware schematic for the user interface

2.3.2: Power Supply

I. The power supply provides power to the LCD screen and microcontroller. This supply incorporates lithium ion batteries with a voltage regulator for the different component requirements.

II. Parts Needed:

A. Lithium Ion Batteries: A battery pack consisting of 5 Energizer rechargeable AA lithium ion batteries [10] that will be responsible for powering the LCD screen and Raspberry Pi. The pack will be placed on the backside of the guitar out of the way of the user.

B. Voltage Regulator: The LM2575 [11] will regulate the output of the batteries to the specified 5V needed for the raspberry Pi with high efficiency. This will be on the PCB.

2.3.3: Control System

I. The control system will serve as the central point for all communication and data transfer between components. It will convert image files to MIDI format, then MIDI to a comprehensible format for the microcontroller to light LEDs. It will tell the microcontroller when to begin playback, when to stop, and how correct the user's notes were for the duration of the playback. It is the central hub of the entire system, as it builds the handshake between our different components so that they can communicate with each other in a meaningful way.

II. Parts Needed:

A. Microcontroller: After a lot of research concerning different options for microcontrollers, we've decided to go with the Raspberry Pi Zero [12]. Between it's popularity, Python compatibility, and community support, it will make the perfect microcontroller as we will be detecting the guitar's input with a PyAudio library [13]. It is the cheaper version of the Raspberry Pi 3, which will be sufficient in power and size to mount to our PCB. This will allow us to create a synchronized 'brain' between the microcontroller and processor to accurately assess the performance of the user. These inputs will then be translated to output as instructions on the LED's, Feedback on the GUI, and song/measure/tempo status on the LCD screen.

B. Processor: The processor will receive input from the guitar while the user is playing [13], and convert this input into MIDI format to compare with the original song.

C. PNG to MIDI: This will be done with the 'SheetVision' repository that was found on Github [14]. As an immediate concern was the feasibility of this step, it was necessary to use an outside library to make this portion simpler.

D. MIDI to LED: By studying the MIDI data format [15], we can write an algorithm in Python to match certain pieces of data to their respective notes and lengths. Once this is done, we can create a mapping of these notes to their positions on the fretboard and determine which LEDs need to be ignited. The key signature of the original music piece will determine where the user starts to play on the guitar, as one note can be played in multiple places.

E. .25” to 3.5mm Adapter: This will serve as the connection from the guitar to the computer to determine which note is being played by the user [15].

F. Performance Analysis: After receiving the input from the microphone using the PyAudio library [13], further analysis will be required to determine the note that was played and if it was correct or not. A final GUI will be necessary to convey all of this information to the user so that they can assess their performance with a deeper understanding.

III. Overview Flowchart

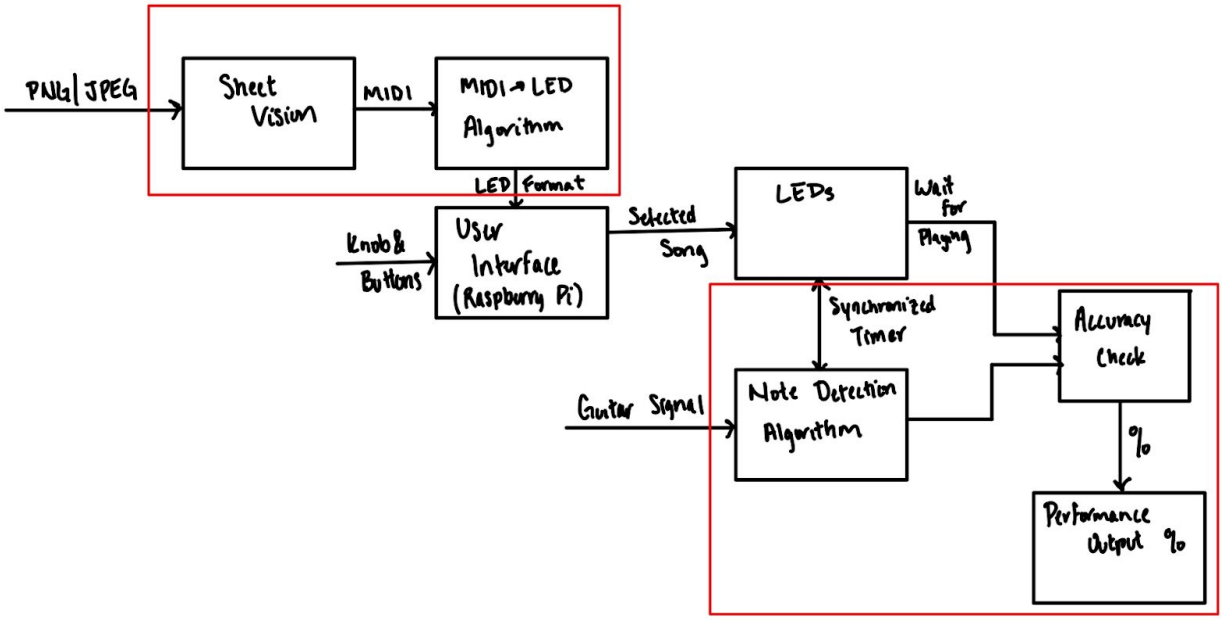

Figure 5: A high-level overview of how the control system will operate as a flowchart. The areas highlighted by the red rectangles are all software components built in Python.

2.4 Requirements & Verifications

2.4.1 High-Level Requirements

Table 1: Verification process needed to ensure proper function of whole project

Requirement	Verification

1. System must provide real time and easily interpreted instructions for playing a song through 12 visual LEDs along the top side of the 12 lowest frets and 6 colors per LED corresponding to each string on the guitar. These LEDs must match >95% of the actual song sheet used.	At all tempos a song will be "played" using the LEDs while one of us will record the fret and string of each note displayed. This will then be compared with the actual song to get an accuracy percentage.
2. System is able to detect correct notes played by the user with >90% accuracy and able to compile detection into a feedback metric detailing correct/incorrect notes for each measure and the entire song.	To verify if notes are being detected correctly, individual notes will be played and the note that it is classified as will be compared to the actual note played.
3. System must allow the user to incrementally increase the difficulty by changing the tempo of the notes to be played, and the amount of measures to play in a given sequence.	While a song is playing, the user can press one of the buttons to pause the song, change the tempo or current measure sequence length for playback with the knobs, and resume playing.

2.4.2 User Interface Verification

Table 2: Verification process to ensure the user interface block is functioning as intended

Requirement	Verification
1. Must be able to pause a song at any time with one of the buttons.	When the pause button is pressed, the LCD screen changes from displaying the current song's information to the menu screen.
2. Tempo of playback can be adjusted	Using the buttons and knobs, the user can either increase or decrease the tempo of playback to adjust difficulty.
3. Length of measure sequence can be adjusted	Using the buttons and knobs, the user can increase or decrease the amount of measures to be played at a time to adjust difficulty.
4. User can choose from a list of songs	While in the paused state, the user can see all song choices on the LCD Screen as the user

	scrolls. In addition, the song selected matches the song played once the user begins playing.

2.4.3 Power Supply Verification

Table 3: Verification process to ensure the power supply block is functioning as intended

Requirement	Verification
1. Lithium ion batteries must be able to supply maximum current 80mA +/- 5mA at 5V +/- 0.25V to the Raspberry Pi.	Connect the batteries to Raspberry Pi power input. Ensure Raspberry Pi functions correctly with an LED sequence running
2. Batteries must be able to support the system at full capacity for >3 hours without recharging.	Run the full system for 3 continuous hours starting with fully charged batteries to see if it still functions after the minimum time has elapsed.

2.4.4 Control System Verification

Table 4: Verification process to ensure the control system block is functioning as intended.

Requirement	Verification
1. Must be able to fetch song data and convert to an LED sequence with < 30sec delay to allow the user to play the song.	Send output signal when song data is being fetched and use an oscilloscope to measure the time it takes for the processing to finish.
2. Ability to convert a sheet music image file of singular notes into a MIDI file with 95% accuracy.	Write a script to convert the MIDI file to their actual note names, and check it against the sheet music.
3. MIDI file can be converted to LED array with 100% accuracy.	We can compare the output of the script from the previous verification to the lights emitting from the fretboard to determine accuracy.
4. A note being played on the guitar can be detected by the processor with >90% accuracy to determine if the user played the correct note.	Strum one note at a time on the guitar, look at the output of the processor's detected note, and compare with the played note.

2.5 Tolerance Analysis

The differences in frequencies from adjacent notes on a string can be very large or very small, depending on what octave we are playing. For the 12 frets we will be using, there is a minimum difference of 4.9Hz (E string fret 1 ⟶ E string fret 2) to 34.92Hz (e string fret 11 ⟶ e string fret 12). Because of this, we will need a **maximum error of 2Hz in either direction** to be able to determine the correct note being played by the guitar. For a minimum frequency of 82.41Hz, this means we can have a valid range if the note detected is from 80.41Hz - 84.41Hz, or a maximum error of:

$$\frac{|84.41 - 82.41|}{82.41} * 100 = 2.43\% \text{ error}$$

from the processor in order to be confident that our processor is getting the right note from the guitar.

Cost and Schedule

3.1 Cost Analysis

Our costs of labor are estimated to be $40/hour, 15 hours/week for three people, over the next nine weeks. This would make our total costs of labor to be:

$$\frac{\$40}{1\ hour} * \frac{15\ hours}{1\ week} * \frac{9\ weeks}{1\ person} * \frac{3\ people}{1\ prototype} = \$16{,}200 \text{ in labor}$$

Table 5: Individual parts and their costs for the prototype of the design.

Part	Cost (prototype)
Raspberry Pi Zero W (x2)	$20
Tactile Buttons- 4	$1
Adafruit NeoPixel Digital RGB LED Strip - Black 30 LED [ADA1460]	$10.98
KY-040 Rotary Decoder Encoder	$1.64

20x4 Parallel Character LCD	$14.88
PCB RONSHIN Memory Card Micro SD Card Class 6 Flash Card Memory Microsd TF/SD Cards for Tablet 128mb C6	$5.13
PCB	$5
Energizer Rechargeable AA and AAA Battery Charger (Recharge Value) with 4 AA NiMH Rechargeable Batteries	$9.99
LM2575 - switching buck converter	$2.22
Total	**$70.84**

Combining the product and labor costs, the total cost for one prototype of our design is estimated to be $16,270.84.

3.2 Schedule

Table 6: Proposed week-by-week schedule and designated tasks for the rest of the semester.

Week	Oumar	Dillon	Kyle
3/2/20	Convert sheet music to MIDI; Store MIDI files on the MicroSD Card	Begin developing finite state machine for user-interface, research SDCard file retrieval on Raspberry Pi	Order all hardware parts. Design Schematics.
3/9/20	Start Developing MIDI to LED Algorithm	Continue developing FSM using console as the LCD screen, and keyboard keys	Test with breadboard and create initial PCB design to get early bird

		to imitate knob/button input	extra credit
3/16/20	Continue developing Algorithm	Ensure complete functionality of FSM prior to loading to Raspberry Pi	Finish prototyping power supply and hardware
3/23/20	Ensure MIDI can be converted to an accurate LED sequence and in <30 sec. Begin working on a timing system for tempo adjustment	Build Python script to record microphone input and connect guitar for the first time to observe signals	Help Dillon with hardware connection from guitar to computer. Help Oumar with LED hardware
3/30/20	Continue working on the timing system	Debug and continue working on note detection until we reach the desired tolerance of < 2% error.	Solder PCB and test 1st PCB. Get 2nd PCB design ordered if needed
4/6/20	Integrate Dillon's user interface and note detection with LED Algorithm and timing system	Work with Oumar to integrate note detection with proper timing.	Solder 2nd PCB and begin testing and integrate with the whole system
4/13/20	Begin working on Performance Metric Output	Debug note detection until we are confidently receiving one specific note per 'tick' from the guitar	Continue testing 2nd PCB and debug any issues
4/20/20 (Mock Demo)	Continue working on Performance Metric Output with Dillon	Help finish Performance Metric Output with Oumar	Continue testing and debugging at critical stress points on the design
4/27/20 (Demo)	Debug as needed; Begin working on Presentation and Final Paper	Debug performance metric and add features as desired until demo	Finish Debugging and assist other team members in finishing the project.
5/4/20 (Present ation)	Turn in all assignments	Turn in all assignments	Turn in all assignments

Ethics and Safety

A safety risk that involves our project concerns the power and hardware of it. Since we are using a mobile battery pack to power the systems aboard the guitar, we must watch out for potential overheating. If the battery pack's heat becomes an issue, we must look into creating a safe comfortable harness to protect the user. Another safety concern is the risk of electrical shock. There will be interaction between the user's hand and the fret where all the LEDs are located. Any exposed or loose wiring between the LED's and control system could lead to electrical shock. This would be a violation of IEEE Code of Ethics #1 and #9 by putting the safety of the user at risk and potentially injuring them [16]. Therefore, we must ensure that the battery and wiring on the system is as secure as possible to mitigate these risks and stay in compliance.

During this project, we plan to convert music sheets into MIDI files that will be stored on the device. This leads to the risk of copyright infringement on music, which is in violation of IEEE Code of Ethics, #7: "to credit properly the contributions of others" [16]. In order to be in compliance with copyright laws, we will need to ensure that we obtain copyrighted music by legal means, or use music that is considered public domain or no longer copyrighted. If this device is later used for commercial use, we will need to conduct further research on the rules for using copyrighted music in this domain.

Another ethical concern that our project faces is ensuring that the device does what was promised to the user. Our device plans to teach the user to play the guitar through LED instructions and feedback. We must ensure that the instructions and feedback adhere to the levels of accuracy stated in our requirements. Providing inaccurate data to the user would be a violation of IEEE Code of Ethics, #3: "to be honest and realistic in stating claims or estimates based on available data" [16]. Therefore, we plan to avoid this through constant testing, debugging, and verification of our device.

References

[1] “How to Play Guitar: Learn Guitar,” *Yousician*. [Online]. Available: https://yousician.com/guitar. [Accessed: 20-Feb-2020].

[2] “Digital Trends,” *Fret Zealot*. [Online]. Available: https://www.fretzealot.com/. [Accessed: 20-Feb-2020].

[3] “Fender Play Online Guitar Lessons - Learn How to Play Guitar,” *Fender Guitars*. [Online]. Available: https://www.fender.com/play. [Accessed: 20-Feb-2020].

[4] “Practice the Guitar & Make Music On-the-Go with This Travel-Sized MIDI Smart Guitar,” *Popular Science Shop*. [Online]. Available: https://shop.popsci.com/sales/jamstik-7-guitar-trainer-righty?gclid=CjwKCAiA7t3yBRADEiwA4GFlI0suONrrWyqCbnK5V8NuvRlZq7ZBXpbM95LC569UtI4aCdAjkfOUJxoCMloQAvD_BwE. [Accessed: 20-Feb-2020].

[5] “Simon (game),” *Wikipedia*, 11-Jan-2020. [Online]. Available: https://en.wikipedia.org/wiki/Simon_(game). [Accessed: 20-Feb-2020].

[6] Banggood.com, “KY-040 Rotary Decoder Encoder Module AVR PIC Geekcreit for Arduino - products that work with official Arduino boards Module Board from Electronics on banggood.com,” *www.banggood.com*. [Online]. Available: https://usa.banggood.com/KY-040-Rotary-Decoder-Encoder-Module-AVR-PIC-p-914010.html?gmcCountry=US¤cy&createTmp=1&utm_source=googleshopping&utm_medium=cpc_bgs&utm_content=frank&utm_campaign=ssc-usg-100-all-0821&ad_id=378825333488&gclid=CjwKCAiA7t3yBRADEiwA4GFlI8B8PyZgKTuLvCr8m1viQiWEZ32fq3CdkW-Tth3kbziiV6e7_E3xbxoCkB4QAvD_BwE&cur_warehouse=CN. [Accessed: 25-Feb-2020].

[7] “Tactile Button Assortment,” *COM*. [Online]. Available: https://www.sparkfun.com/products/10302. [Accessed: 28-Feb-2020].

[8] “WS2812 and WS2812B RGB LED Module,” *WS2812 and WS2812B RGB LED Module - Parallax Inc. - Addressable, Specialty | Online Catalog | DigiKey Electronics*. [Online]. Available: https://www.digikey.com/catalog/en/partgroup/ws2812-and-ws2812b-rgb-led-module/50496?utm_adgroup=Optoelectronics&utm_source=google&utm_medium=cpc&utm_campaign=Dynamic Search&utm_term=&utm_content=Optoelectronics&gclid=CjwKCAiA7t3yBRADEiwA4GFlI9e9u_fSgbWoxGttxqLqWl7mTOey-cZosiAYHI3JlVw6xn5dp6Ur7RoCWMoQAvD_BwE. [Accessed: 28-Feb-2020].

[9] “Smallest LCD Displays,” *Crystalfontz.com*. [Online]. Available: https://www.crystalfontz.com/product/cfah2004atmijt-display-module-20x4-character. [Accessed: 28-Feb-2020].

[10] “Energizer Rechargeable AA Batteries, NiMH, 2000 mAh, Pre-Charged, 4 count (Recharge Universal),” [Online]. Available: https://www.amazon.com/Energizer-Rechargeable-Batteries-Pre-Charged-Universal/. [Accessed: 4-Mar-2020]

[11] “LM2575-5.0WU-TR,” [Online]. Available: https://www.digikey.com/product-detail/en/microchip-technology/. [Accessed: 4-Mar-2020]

[12] “Buy a Raspberry Pi Zero W – Raspberry Pi,” *Buy a Raspberry Pi Zero W – Raspberry Pi*. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-zero-w/. [Accessed: 28-Feb-2020].

[13] V. P. V. Pira, “Reading input sound signal using Python,” *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/35344649/reading-input-sound-signal-using-python. [Accessed: 28-Feb-2020].

[14] Cal-Pratt, “cal-pratt/SheetVision,” *GitHub*, 11-Mar-2018. [Online]. Available: https://github.com/cal-pratt/SheetVision. [Accessed: 20-Feb-2020].

[15] “Standard MIDI-File Format Spec. 1.1, updated,” *Standard MIDI file format, updated*. [Online]. Available: http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html. [Accessed: 22-Feb-2020].

[16] “IEEE Code of Ethics,” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 26-Feb-2020].