Autonomous String Tuning

A Major Qualifying Project

Submitted to the Faculty of

Worcester Polytechnic Institute

In partial fulfillment of the requirements for the

Degree of Bachelor Science in

Computer Science	Electrical and Computer Engineering
Garrett Smith	Fiona Doyle
Victoria Thornton	Ryan Hennigan
	Vien Phuong T. Le

Project Advisors
Professor Scott Barton
Professor Yarkin Doroz

Abstract

The primary goals of this project were to create a system capable of autonomously tuning five strings accurately and quickly, within 100-250 milliseconds, dependent on the musical interval traversed, and with an accuracy of six cents - the musical interval barely noticeable to the human ear. A load cell with a guitar string through it is utilized as the primary determinant of frequency, and motors coupled to tuning pegs change the tension of these strings. This operation is performed by the load cell referencing a lookup table that details what encoder position the motor must reach in order for the string to achieve a certain tension which corresponds to the target frequency of the tuning operation. A Fast Fourier Transform is performed on readings from a photoresistor system to determine the true string frequency, solenoids actuate the strings, and the device is controlled from a local server built on Node.js.

Acknowledgements

We would like to acknowledge Professors Scott Barton & Yarkin Doroz for their guidance throughout this process. Furthermore, thank you to all WPI staff members who strived to maintain a semblance of normalcy on campus despite the pandemic - this project would not have gotten as far as it did without those combined efforts.

Thank you to all those who work in the Washburn machine shops and the Foisie Innovation Studio who assisted in the fabrication of the parts used. Finally, thank you to Mr. Appleyard for assisting in the procurement of parts.

Table of Contents

Table of Figures

Table of Tables

Table of Equations

1) Introduction

The purpose of this project is to create a novel musical instrument equipped with an electromechanical system capable of operating autonomously to accurately and quickly tune the instrument's five strings. There are many applications that this device could suit. For example, an instrument user would not be required to tune their instrument manually, which is a process that requires both time and proper skill to perform correctly. Alternative tunings could be transitioned to readily, and the instrument would never go out of tune while in use.

Not only could the creation of this instrument further the interactions that an individual has with their instrument, more musical experimentation through autonomous functionality can be explored. Chords could be formed on this string instrument with no human interference required, which may be useful to many musicians who are incapable of utilizing the frets of a guitar to transition between notes. Furthermore, since the strings' tuning can be changed mid-performance via a user interface, new effects such as sliding, detuning, and pitch modulation may be implemented. With these capabilities in mind, the incorporation of solenoids to strike the strings of the instrument at programmable intervals could allow the instrument to autonomously perform whole, complex songs that would typically require the skill of a professional musician. The concept of autonomous string tuning may enable the creation of new types of music, and the invention of other, more complex novel musical instruments.

2) Background

2.1) Physical Principles of String Instruments

Music and mathematics share a very rare characteristic - they are both universal. Every culture has some expression of both concepts, and has created ways of explaining them. As both fields evolved, the intersection between science and music became far more apparent. Because of these scientific advancements, it is possible to explain why musical instruments create the sounds they do via mathematical equations, and the properties of said instruments can be manipulated to create new sounds under the guidance of these mathematical principles. The way in which a string instrument functions can be explained using concepts of wavelengths usually taught in physics classes.

The frequency of a note is the number of oscillations per second the string of an instrument will undergo in order to produce the target tone. The period of a note is the time it takes to complete one of these oscillations. In order to trigger these frequencies, a user may create motion in a string with fixed endpoints by plucking, hammering, or hitting said string. This motion causes excitation in its medium, which is expressed in a shape called a waveform. This waveform creates vibrations that move at a certain velocity, which can be found using Equation 1. This velocity can then be used to determine the fundamental frequency of the string utilizing Equation 2. When a string is oscillating at its fundamental frequency, it can also oscillate at integers of this frequency, which are referred to as harmonics. These harmonic frequencies can be found utilizing Equation 3. The complex signal of a stringed instrument is the result of these many frequencies at which the strings vibrate. These harmonics may provide an additional challenge to detecting the fundamental frequency of a given string, as the sensor may pick up these frequencies, as well as the fundamental, and have to differentiate between them. More details as to how this may be accomplished can be found in section 2.4) Fundamental Frequency Determination & Tuning Algorithms.

$$v_{wave} = \sqrt{\frac{T}{\frac{m}{L}}} \tag{1}$$

where T = string tension, m = string mass, and L = string length

$$f_1 = \frac{v_{wave}}{2L} \tag{2}$$

$$f_n = n * f_1 \tag{3}$$

[1]

String tension can be related mathematically to frequency. The relationship is shown through Equation 4, which can be used to calculate the tension of a string using the scale length of the instrument, the unit weight of the string, and the target frequency.

$$T = \frac{UW \times (2 \times L \times F)^2}{386.4} \qquad (4)$$

where T = string tension, UW = unit weight, L = scale length and F = frequency

[23]

Two common methods are utilized to allow string instruments to be capable of producing a large range of pitches. Some have large quantities of strings, such as a piano or harpsichord. These instruments' strings do not change in length - in order for a standard Yamaha piano to be capable of producing 88 notes, for example, the instrument houses 88 strings in its body. These strings create sound when the user presses a key that causes a small hammer to strike the string, producing the frequencies discussed above. Ideally, string tension, mass, and length will not change in this configuration, each string will only produce its assigned frequency. [2]

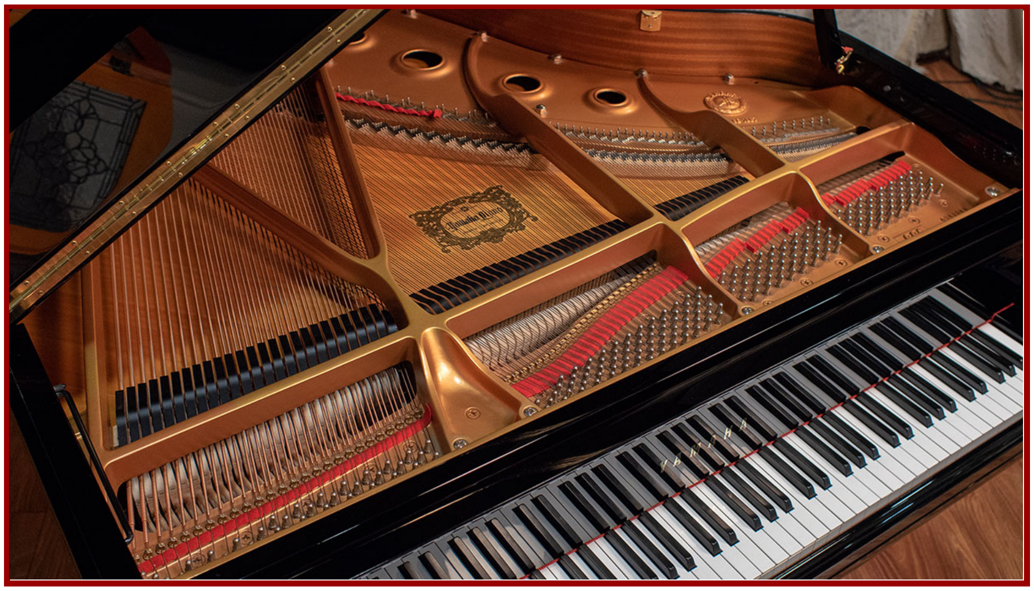


Figure 1: The Strings, Keys, & Hammers of a Yamaha Grand Piano [3]

Another method of creating range in string instruments is to allow the user to manipulate the vibrating length of the string. This is typically performed by the user placing a finger or other object that restricts the moveable length of the string. By performing this action, both the velocity and frequency of the string are altered, allowing for new notes to be reached. Some of these length changes have been standardized on instruments such as a guitar - the neck of the instrument contains protruding bars, called frets. These bars are set at certain intervals along the instrument in such a way that certain musical intervals are achieved when a user restricts the length of the string where the fret indicates. Please see Figure 2 for a detailed diagram of the intervals found on the neck of a guitar.

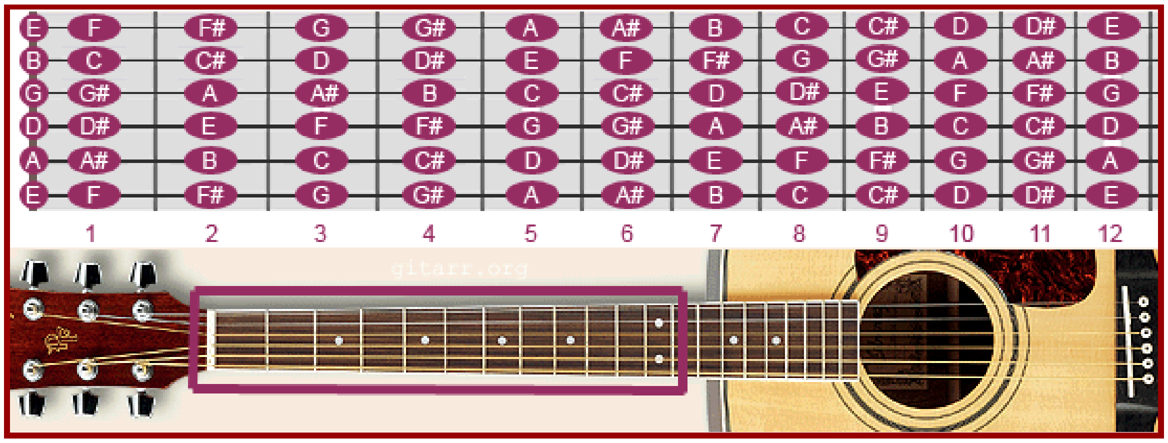

E F F# G G# A A# B C C# D D# E
B C C# D D# E F F# G G# A A# B
G G# A A# B C C# D D# E F F# G
D D# E F F# G G# A A# B C C# D
A A# B C C# D D# E F F# G G# A
E F F# G G# A A# B C C# D D# E
1 2 3 4 5 6 7 8 9 10 11 12
gitarr.org

Figure 2: Intervals on the Neck of a Guitar [4]

Occasionally, the guitar's user will want to shift all of the notes they are playing down by a certain number of frets to achieve an alternate tuning without manually tuning the instrument. In this case, a tool called a capo is utilized - this spring-loaded clamp compresses the strings at the artist's desired length, allowing the guitar's notes to rest about a new scale. Please see Figure 3 to see a capo in use.

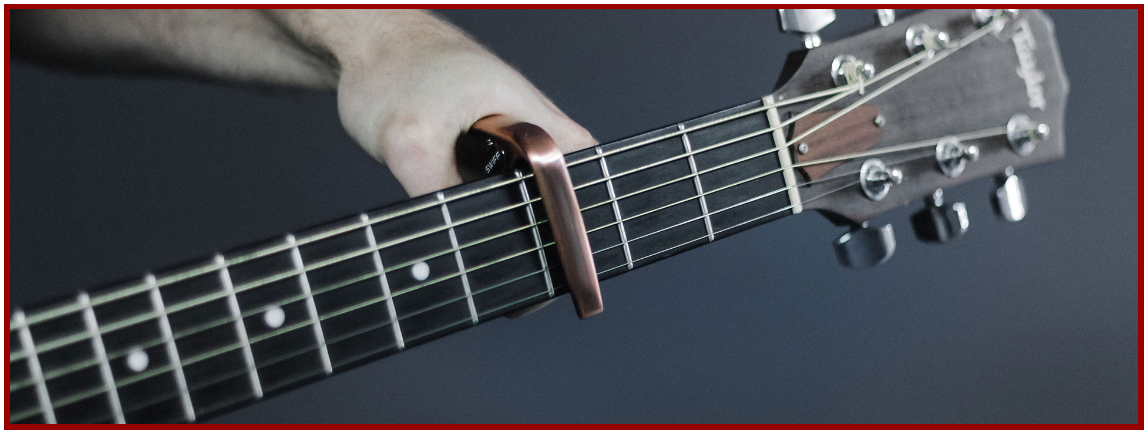


Figure 3: A Capo in Use [5]

It is possible to change the frequency of a string by changing the tension applied to it. The majority of stringed instruments utilize some form of tuning mechanism to maintain the tension of a string. Many stringed instruments utilize tuning machines to allow the user to easily change the pitch of their strings. This action is performed for a variety of reasons, from creating new sounds by changing the key of the instrument, to releasing all of the tension a musical instrument's neck is under before flying in an airplane in order to prevent it from snapping.

Tuning machines function by wrapping the excess length of an instrument string around a cylindrical piece of metal attached to a gear, referred to as a pinion gear. This gear slots together with a worm gear, a mechanical component which translates the rotational motion generated by the user from turning the tab adjacent to it. Please see Figure 4 for a photo of a tuning machine - the pinion gear is the large gold gear, and the worm gear is the silver component. For the purposes of this project, the tuning machine gear reduction ratio is 15:1, where Z_A, or the worm gear's ratio value is 1, and Z_B, the pinion gear is equal to 15.

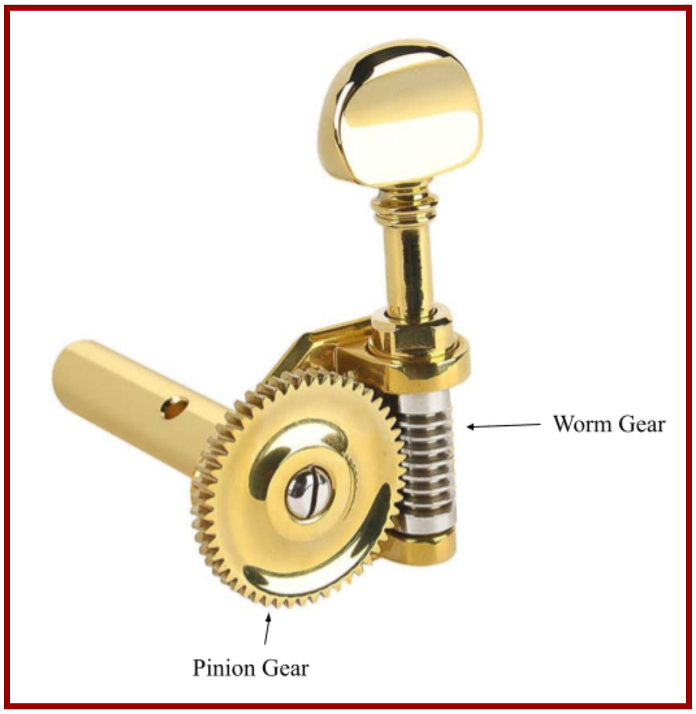


Figure 4: An Example of a Tuning Machine [6]

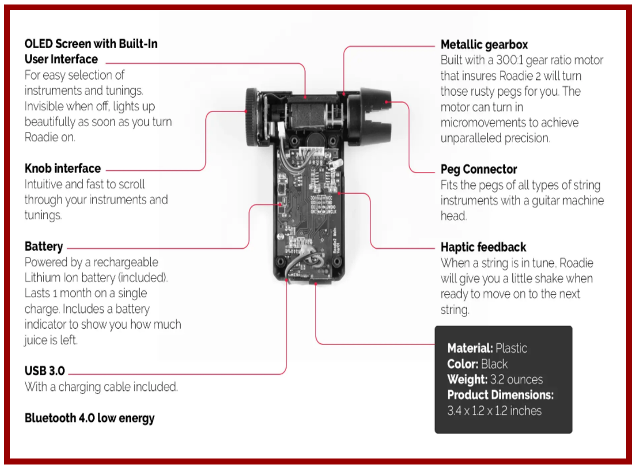
While tuning machines have mechanically simplified the tuning process, achieving the correct tuning for an instrument can still be challenging. Each string of a guitar is tuned individually, and the user must ensure that the string tension is set correctly for the string to achieve its target frequency. Furthermore, tuning machines are imperfect at maintaining their tensions, and are occasionally susceptible to detensioning their strings when jostled during transportation or mid-performance, so the instrument must be retuned at regular intervals. Strings often go out of tune due to the effects of temperature & humidity on the string's physical properties as well [7]. Unless the musician is particularly practiced and has memorized the tones they are attempting to reach, an external tuner typically must be used. It is rare for these tuning machines to be utilized mid-performance - tuning machines require attention and practice to manipulate quickly and efficiently, and would require the musician to pause their song in order to perform the action. Individuals have attempted to create a solution to this problem by inventing autonomous string tuners that are capable of detecting the frequency of a string.

2.2) Current Autonomous Tuners on Market

The majority of commercial autonomous string tuners that can be found in stores today are marketed specifically for guitars. These tuners fall into two common categories - they either attach directly to the head of the instrument, or are handheld devices that the user positions around one tuner mechanism at a time. These two systems each have their benefits and their drawbacks - while the former requires the user to modify their guitar, the latter is not as efficient.

2.2.1) The Roadie 2 Automatic Guitar Tuner

An example of a handheld autonomous tuner is the Roadie 2 Automatic Guitar Tuner. With a retail value of $129, this handheld device allows the user to choose between over 40 preset alternate tunings, with the option to add custom tunings as well. It can be used on most handheld string instruments, and operates as a standalone device. The system utilizes a vibration sensor rather than a microphone, so that tuning can be accomplished in environments with excess ambient noise. In order to tune an instrument, the user must hold the device over the desired tuner machine and pluck the string being tuned. The device autonomously tunes the strings in this fashion, one by one. [8] Figure 5 displayed the Roadie 2 and a few of the features listed on its website.



OLED Screen with Built-In User Interface
For easy selection of instruments and tunings. Invisible when off, lights up beautifully as soon as you turn Roadie on.

Knob interface
Intuitive and fast to scroll through your instruments and tunings.

Battery
Powered by a rechargeable Lithium Ion battery (included). Lasts 1 month on a single charge. Includes a battery indicator to show you how much juice is left.

USB 3.0
With a charging cable included.

Bluetooth 4.0 low energy

Metallic gearbox
Built with a 300:1 gear ratio motor that insures Roadie 2 will turn those rusty pegs for you. The motor can turn in micromovements to achieve unparalleled precision.

Peg Connector
Fits the pegs of all types of string instruments with a guitar machine head.

Haptic feedback
When a string is in tune, Roadie will give you a little shake when ready to move on to the next string.

Material: Plastic
Color: Black
Weight: 3.2 ounces
Product Dimensions: 3.4 x 1.2 x 1.2 inches

Figure 5: Information Regarding the Functionality of the Roadie 2 [8]

There are many aspects of the Roadie 2 that a potential consumer may find attractive when purchasing an autonomous tuner. The portability of this system is an asset, especially if a user intends to tune multiple instruments with their device. This would not be achievable with the systems that are attached to the guitar itself. Furthermore, consumers may find it appealing that no modifications need to be made to their instrument in order for the device to function. However, it is important to consider that this device can only tune strings one by one - this may not be acceptable depending on the purpose behind why one uses their tuner. For example, if a guitarist is tuning their instrument in a rehearsal they may not prioritize efficiency in their tuner, and can utilize this device. However, if the guitarist is in the midst of a performance and their instrument falls out of tune, a more rapid solution is required.

2.2.2) TronicalTune PLUS

If more speed is required, the user may consider the second common autonomous tuner design - a model mounted onto the guitar. An example of an instrument mounted self tuning system is the TronicalTune PLUS system. In order to install this mechanism, the user must remove the original tuning pegs for their guitar and install TronicalTune's motorized machine heads. The rest of the system is then attached to the back of the guitar. Once installed, the guitar can be tuned to any of the 24 preset or 12 customizable tunings in five seconds. The device can either tune one string at a time or all six strings at once, depending on which mode the user selects. Additionally, the TronicalTune PLUS can detect if there is a capo on the instrument between the 1st and 6th fret. Depending on the style of tuner one chooses to purchase, the TronicalTune PLUS costs between $239 and $293 [9]. While there are other models of instrument mounted self tuning systems such as the Gibson Min-ETune, it is important to note that Gibson's systems are simply the TronicalTune system that was licensed to Gibson for use. In 2015, Gibson began to sell their guitars with tuners preinstalled, but many users found them to be 'over-complicated' - some paid to have the system removed from their instruments [10]. In 2018 the inclusion of these systems became a rarity on Gibson guitars [10]. Figure 6 displays this tuning system affixed to the back of the instrument.

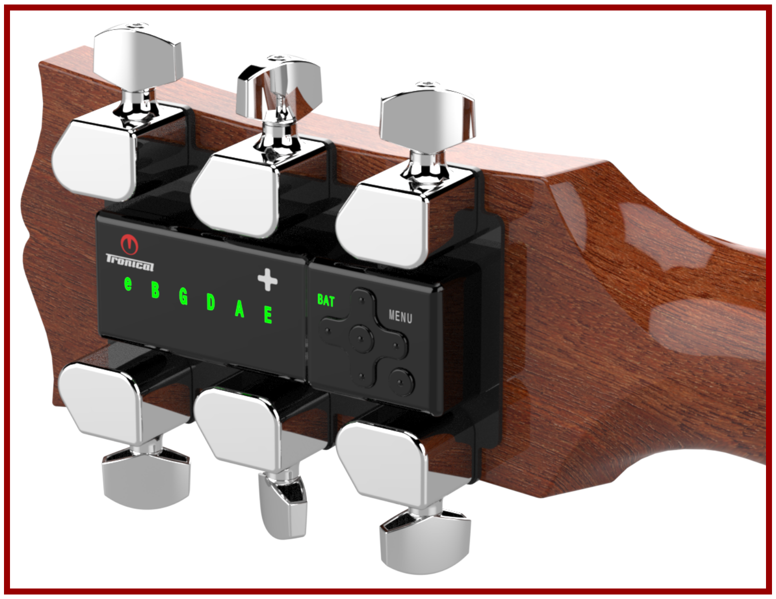


Figure 6: The Gibson TronicalTune Plus System [9]

This model of tuner is certainly attractive if one does not have ample time to tune their instrument - for example, this tuner would be put to excellent use when switching tunings mid-concert. The user does not have to hold the tuner in order for it to function, as well. However, this model can not be used across different types of string instruments, and can only be practically used on one instrument at a time. As the design of guitar heads vary so widely across the market, TronicalTune has created a variety of potential tuners that one can order, so that the sizing of the device will fit. If the user were to order a type of TronicalTune system that was not designed for their guitar, it would be unusable. If a performer were to want to use a single guitar for a performance and wanted it tuned as efficiently as possible, this device would certainly be one to consider. This tuner is also incapable of dynamic tuning, or modifying string pitch mid performance in order to create new musical effects. If one is seeking this level of interaction from a tuning system, they would not find it with the TronicalTune Plus.

2.3) Current Autonomous Tuning Instruments

2.3.1) Electro-Thermal Tuning

Quite a few autonomously tuning instruments have been patented in the United States which utilize a wide variety of methods to achieve their results. Some of these methods are clearly designed with the intent for a user to never touch the strings during operation. An example of such a patent is held by Donald A. Gilmore for the "Apparatus and Method for Self-Tuning a Piano" [11]. Metal piano strings are passed in between a red LED laser and a phototransistor, a device that changes the magnitude of an electric current passed through it when exposed to light. The number of times the current fluctuated is counted in order to determine how many times the string passed between the laser beam and the phototransistor, and by utilizing a high speed oscillator with a clock frequency of 100MHz, Gilmore was able to produce a “crisp square wave at the fundamental frequency of the string’s vibration” [12]. If the string frequency needs to be changed, electricity is passed through the string in order to elongate the piano wire utilizing the principles of thermal expansion. The current of this electricity must be closely monitored to ensure that nothing dangerous occurs. Furthermore, if a string on the instrument must be replaced, it must be made with a metal with relatively low resistance - if the resistance were too high, the string could heat to a dangerous level. An interesting modification had to be made to the lowest strings of the piano in order for this method to work. While the heart of the strings is made from the same steel as the rest of the piano’s wires, these lower strings are traditionally wound in copper, which would have rendered the tuning method ineffective due to the lower resistivity of copper in comparison to the other steel piano strings. In order to circumvent this issue, the string was wrapped in magnet wire as is utilized in the creation of solenoids. Its lacquer-coated, insulated surface prevented the wrapping from interfering with tuning in the way that traditional copper would have.

One drawback to this tuning method is that the instrument cannot be played while it is being tuned - the user has to set their piano to a tuning mode, press a pedal, and keep their hands off of the keys until they can no longer hear electrical current passing through the strings of the instrument. Furthermore, it would be ill-advised to attempt to transfer this tuning method to an instrument that offered less protection to the user. It would be, for example, far easier for a bass player to electrocute themselves on their tuning, unhoused strings. The strings offer a further challenge to this tuning method; they must be made of a conductive material, steel per the patent's recommendation. Many instruments do not utilize metal strings - harps, for example, typically use linen or gut strings, which would render this method non transferable.

2.3.2) Milwaukee Servoelectric Guitar

An example of a fully autonomous musical instrument is the Milwaukee Servoelectric Guitar, constructed by inventor Keith Baxter. While Baxter has designed many iterations of his electric guitar, his first model found the frequency of the three 0.017 nickel plated steel strings of the instrument via tension measurements read from two potentiometers [13]. Each tension measurement corresponded to a certain frequency, and these relationships were stored in the software of the device in a lookup table. A user could press a button on the side of the instrument that related to a chord, and three servo motors would change the tension of each string in order to create said chord. [14] The most recent iteration of the instrument can tune its three identical strings from D at 146.8Hz, to C# at 277.2Hz, providing just below an octave of range.

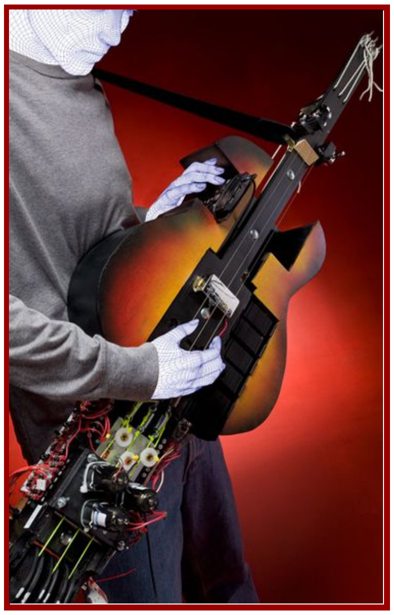


Figure 7: Milwaukee Servoelectric Guitar [13]

While this self-creating chord method of autonomous instrument is effective, it does have one clear flaw. The tension values of a string can be mapped to the frequency of a string, as was displayed by Equation 1. However, since the bodies of musical instruments naturally warp and change over time, other factors in that equation can change - namely, the string length. Because of this, over time the initially measured relationship between the two factors that had been based on a supposedly 'unchanging' variable will become inaccurate. This could be prevented by incorporating a second method of checking the frequency that double checks the accuracy of the first.

2.3.3) Cyther V3

A final example of an autonomously tuning string instrument is *Cyther V3*, a mechatronic string instrument which is capable of self tuning [15]. In a similar manner to the Milwaukee Guitar, *Cyther V3* utilizes tension sensing mapped to frequency values in order to accurately tune the ten strings of the instruments. However, *Cyther V3* is capable of validating its own table by utilizing an additional method of tension sensing. While tension sensing remains the device's primary tuning mechanism, a photoresistor set across from a laser diode is utilized to determine the frequency of oscillation of the string. By counting the number of times the string passes between the photoresistor and the laser diode, a frequency value is determined, and the system double checks itself. In order to tune the strings properly, the relationship between the frequency and the potentiometer utilized to sense the tension value of the string was analyzed. This relationship is illustrated in a potentiometer frequency relationship curve, which was initially determined by utilizing equations and testing the first iteration of the instrument. Then, as the instrument continually checks its own frequency values, it performs calculations to add new data points to this curve, and new best fit lines are calculated, ensuring the instrument always understands the relationship between these two values, even as they change over time.

While *Cyther V3*'s tuning system is admirable, it unfortunately suffers from a few drawbacks. As the system is quite complex, it requires additional time for the system to fully operate. Therefore, a listener can detect that the instrument is out of tune faster than the machine can correct itself. Furthermore, due to the complexities of detecting the frequency of a string, the final pitch of many notes were more than 3.6Hz away from their goals.(Prihar, '17) It is important to note that, while the other systems discussed did not report their drawbacks in the same way that *Cyther V3* presented itself, it should not be assumed that the other systems are perfectly accurate. This resource was a final project from a major capstone, therefore presenting the less than desirable results are a requirement that the other sources may not have been required to meet.

2.3.4) STARI

Created by individuals from the University of Victoria, STARI was the result of a project that was both attempting to standardize the methods utilized in autonomous tuning as the field developed, and to build upon previous work one of the authors had completed on the project *AMI - Automatic Monochord Instrument* [16]. The system is both human and autonomously playable with the use of a pick attached to a servo motor. Furthermore, it is capable of tuning a single string via an unspecified transducer component as a frequency detector, which works with a stepper motor attached to a tuning machine machine. The frequency analysis system on this device modeled to function with a monochord. Please see Appendix A for further explanation of monochords.

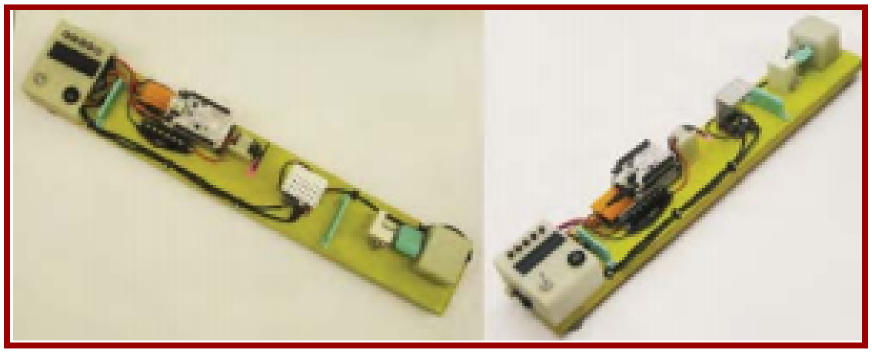


Figure 8: STARI [16]

The software for the signal processing component of the device was written in the language Pure Data and stored on a microcontroller referred to as a Beaglebone, which is capable of housing all software necessary for the operation of this part of the system. Other operations were controlled via an Arduino UNO, eliminating the need for an external computer. This is one of the primary improvements this self-tuning system offers when compared to AMI, as the latter system required the use of a host PC. The team was still able to utilize the software libraries they had hoped to by implementing this change, as both Beaglebone and Puredata are Linux compatible, rendering their previous work translatable.

The primary benefit that the system above offers is that it is entirely self contained - there is no need for a host PC in order to operate the system. However, there are a number of challenges that this design does not address, especially if one wanted to utilize this work as a basis for creating a multi-stringed instrument. The method of frequency detection may cause challenges when attempting to use an electromagnetic pickup over multiple strings - each string would need its own pickup, and each pickup would either need adequate space from adjacent strings or additional filters in order to ensure that there is no interference from its neighbors. Furthermore, the system is tuned via a singular input - it would be wise to have a secondary tuning method double check that the frequency is correct in order to minimize error. Finally, as more strings are added and more frequency analysis must take place simultaneously, the software may become more complex and the memory of the system's boards may not be large enough to accommodate all that the user requires of it.

2.3.5) Swivel 2

Swivel 2 is a six stringed mechatronic slide guitar, and is the second iteration of an instrument of its type [17]. The primary goals of this system were to be less costly and cumbersome than its predecessor, and to allow for more nuanced musical effects to be performed, such as a portamento, or pitch bend. The unique design of the instrument houses it's six strings in a vertical stack, as can be seen in Figure 9. Each string is discrete, and can be played either in conjunction with the other strings or independently. Each string is equipped with a custom PCB, wired together via a communication bus that utilizes MIDI signals to communicate with a host computer. The system is only autonomously playable, with a single pick attached to a servo motor plucking the string, and a second servo motor providing a damping effect when necessary. In an effort to replicate the string fretting one would perform while playing a slide guitar, an aluminum rod is moved along the string's length via a pitch shifter and a clamper servo. This operation is what changes the pitch of the string - there is no manipulation of the tension of the string, but a change in the vibrating length which is dependent on where the aforementioned aluminum rod is placed. The frequency of these strings is detected via individual electromagnetic pickups.

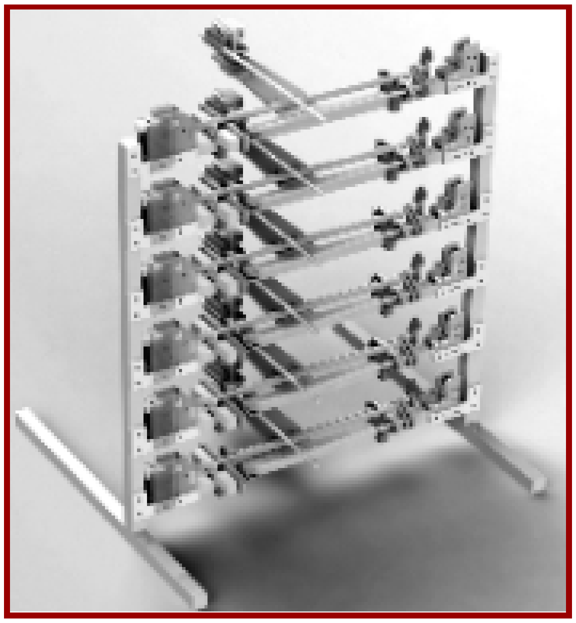


Figure 9: Swivel 2 [16]

This instrument suffers from the same drawback as STARI in that an electromagnetic pickup is the only frequency detection method, and having a backup method would be preferable to ensuring proper operation of the system. Furthermore, the instrument is limited in the sense that it does not offer a method of utilization for a human player. If the user is not seeking this feature, however, this instrument provides a great deal of versatility in its functionality, and could be utilized for a variety of projects.

While this system is not technically considered to be an autonomous tuner (the act of playing a slide guitar is not considered to be a string tuning operation), the principles involved in its function could be implemented in order to create a self-tuning instrument. For example, the vibrating length of the string could still be changed via the methods used above. However, this method of manipulating string length requires use of the vast majority of the length of the string - depending on the goals of the instrument being constructed, this may not be ideal. Furthermore, the stacked orientation of the individual strings is not conducive to human playability. If the strings were laid on their sides, a new sliding mechanism would need to be designed that is more compact so as not to impact adjacent strings. Furthermore, one may encounter interference from the other strings over the pickups if brought closer together.

2.3.6) Automatic Pitch Processing for Electric Stringed Instruments

While it is possible to create pitch changes in a stringed instrument by modifying the mechanical components of the system, there are other methods that can be implemented. One example of a pitch altering system that does not involve the physical manipulation of the system at all can be found in the patent “Automatic Pitch Processing for Electric Stringed Instruments”, held by

Alexander J. Stevenson [18]. This system electronically processes a signal from an instrument and manipulates it utilizing pitch parameters from a memory table. This modified, ‘corrected’ output is then what the instrument produces. This method can also be utilized to “morph” [18] the instrument’s sound into another by manipulating the intervals of sound the instrument produces. The system is also described as a “virtual capo” [18] - instead of having to place a capo on their instrument as described in Section 2.1, a musician simply informs the system that it should tune the guitar’s output differently.

There are many benefits to a system such as the one described above. By causing all pitch modulation to occur via pitch processing and not by mechanical changes to the instrument, many potential points for error are removed. However, by removing the mechanical component from pitch alteration, one is essentially playing a synthesizer whose notes are triggered when the chords of the guitar are strummed - if any note can be produced by the string because of the pitch processing and modulation implemented by this invention, it removes some of the importance of the physical characteristics of the system. However, many of the applications of this system that were explained in the patent explain how the new, modulated sounds are typically made based upon the frequency of the reference string, so the physical characteristics do have some impact on the sounds. If a user would prefer that the mechanical properties of their instrument were utilized more thoroughly while playing, this tuning system may not be for them. Please see Table 1 for a comparison of the current autonomous string tuners on the market, and for a succinct description of their tuning methodology and capabilities.

Table 1: Comparison of Current Autonomously Tuning Instruments

Section Name	Autonomously Playable	Human Playable	Autonomous Tuning	Simultaneous playing and tuning	Mechanical component	Tuning Method
Electro-Thermal Tuning	No	Yes	Yes	No	Yes	Manipulating string properties by heating strings via electrical current, frequency sensing via phototransistor
Milwaukee Servoelectric Guitar	No	Yes	Yes	Yes	Yes	Tension sensing, change in string tension via servo motor

Cyther V3	Yes	Yes	Yes	Yes	Yes	Tension & frequency sensing, change in string tension via gearmotor
STARI	Yes	Yes	Yes	Yes	Yes	Frequency sensing via EMF pickup, tension manipulation via tuning machine and stepper motor
Swivel V2	Yes	No	Yes	Yes	Yes	Changes vibrating length of the string
Automatic Pitch Processing for Electric Stringed Instruments	No	Yes	Yes	Yes	No	Utilizes software to digitally modify instrument signal to produce alternate pitches

2.4) Fundamental Frequency Determination & Tuning Algorithms

Machine assistance is required to find fundamental frequency, regardless of the method pursued. If string tension is measured, then a microcontroller can calculate fundamental frequency using the outputs of tension sensors, as well as the given string lengths and mass. Alternatively, a microcontroller could store the frequency values that correspond to different tension measurements in a lookup table that can be referenced while tuning. For sensors that output a waveform that must be analyzed to determine fundamental frequency, such as the PSD, hall effect sensor, piezoelectric sensor, magnetic coil, and photoresistor, signal processing can be performed either completely on a microcontroller, or assisted by an intermediary circuit. According to the Nyquist theorem, the sampling frequency must be twice the frequency of the string frequency for an accurate measurement [26].

One method of determining frequency with an intermediate circuit is the utilization of a digital frequency meter. In order to function, the meter requires a pulse converter to change the continuous waveform inputted from the sensor chosen into a square waveform. This signal increments a counter that is divided by clock time when the measurement interval lapses [19]. A digital frequency meter can be purchased, or constructed from electrical components. One method that can be utilized to convert the sinusoidal waveform into a square wave is by using a Schmitt Trigger. A Schmitt Trigger outputs a digital high when a certain voltage +K is reached, and low when a voltage -K is reached [20].

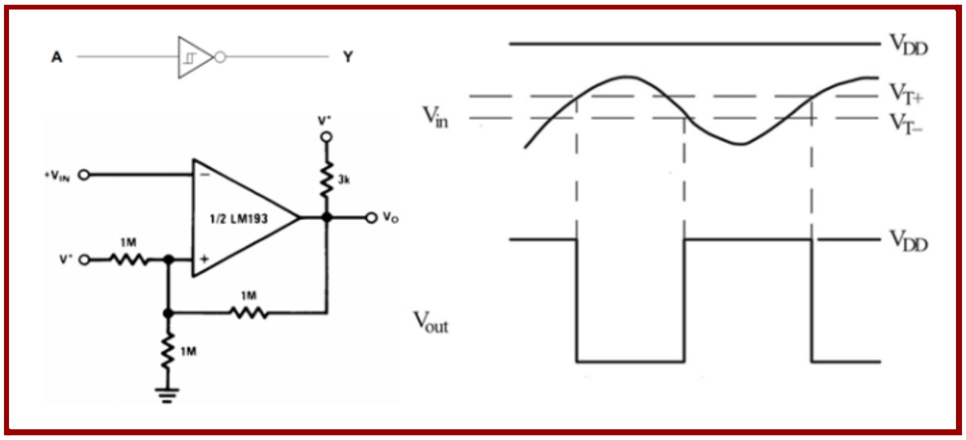


Figure 10: Schmitt Trigger Schematic & Effect [21]

There are also asymmetric Schmitt Triggers in the case that the wave is not perfectly sinusoidal. This digital output can be sent to a synchronous counter that outputs to a microcontroller [22]. The microcontroller can begin a periodic timer interrupt and read and reset the counters. The read counters will be divided by time elapsed to determine individual string frequencies.

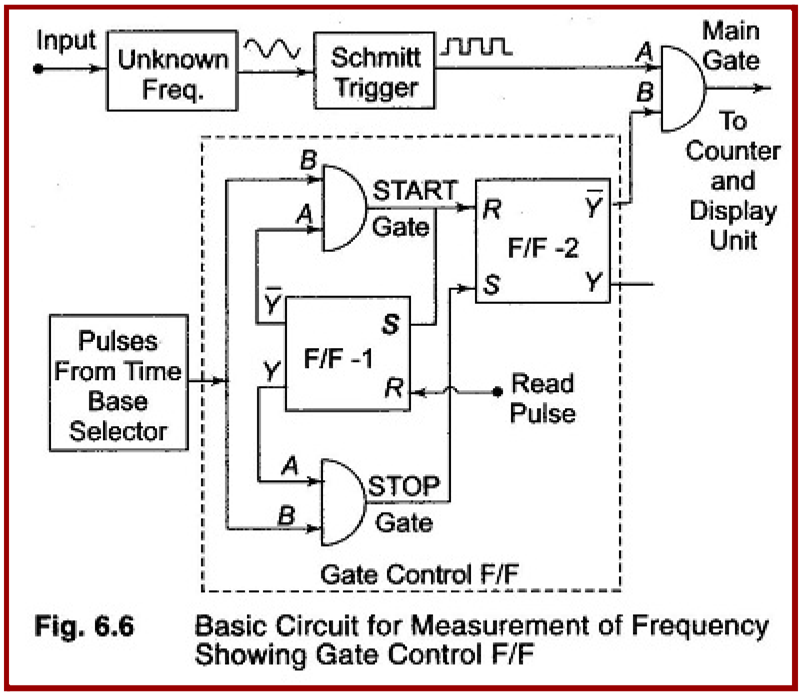

Input
Unknown Freq.
Schmitt Trigger
A
B
Main Gate
To Counter and Display Unit
B
A
START Gate
R
Y̅
F/F -2
S
Y
Y̅
S
F/F -1
Y
R
Pulses From Time Base Selector
Read Pulse
A
B
STOP Gate
Gate Control F/F
Fig. 6.6
Basic Circuit for Measurement of Frequency Showing Gate Control F/F

Figure 11: Digital Frequency Counter [23]

This method will not work, however, if the signal outputted from the sensor does not have a clearly defined voltage range - it is impossible to build a Schmitt Trigger without knowing the target voltages of K+ and K-.

Another method of determining frequency involves utilizing fundamental frequency detection algorithms, which are driven by Fourier Transforms. A Fourier Transform is used to convert a time domain function into a frequency domain function. It determines what frequencies are present in a signal and in what proportions, placing them in discrete bins from 0 to the sample frequency. This allows for analysis of what frequencies occur in a signal by observing peaks in the frequency domain. The lowest frequency that occurs often within the signal is the fundamental frequency.

A Discrete Fourier Transform (DFT) is a discrete algorithm that applies the continuous Fourier Transform mentioned prior. A Fast Fourier Transform (FFT) is any efficient algorithm that applies a Discrete Fourier Transform. The difference between a DFT and FFT is in terms of time complexity and can be described using big-O notation. Discrete Fourier Transforms have a complexity of O(N*N) time, which means that the time it takes to process the information passed through the algorithm can be graphically represented by Equation 5, where n is the number of data points passed through the algorithm.[24]

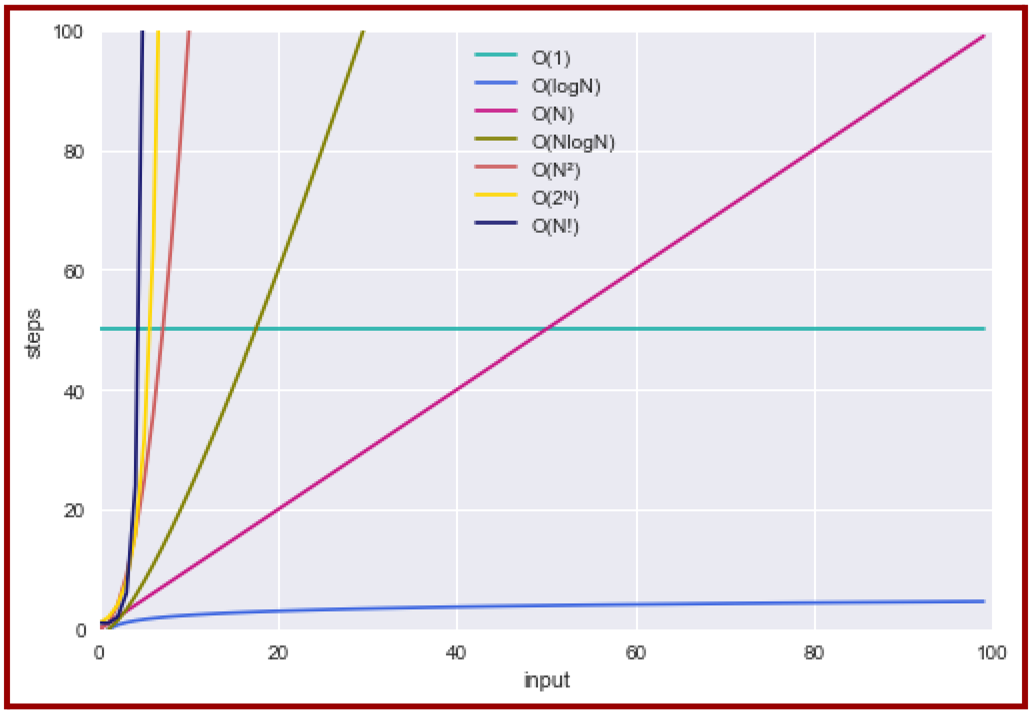


Figure 12: Big-O Notation Graph [74]

Therefore, the more data that the algorithm has to process, the longer it will take. This makes the Fast Fourier Transform (FFT) preferable as it is represented by O(N*log(N)) time, meaning the time it takes to process information with this algorithm can be represented by the graph of Equation 6. As can be seen by the figure below, it will take far less time to perform a Fast Fourier Transform. In practice, this method requires a sampling interval of 1-2 seconds.

$$f(x) = n^2 \qquad (5)$$
$$f(x) = n * log(n) \qquad (6)$$

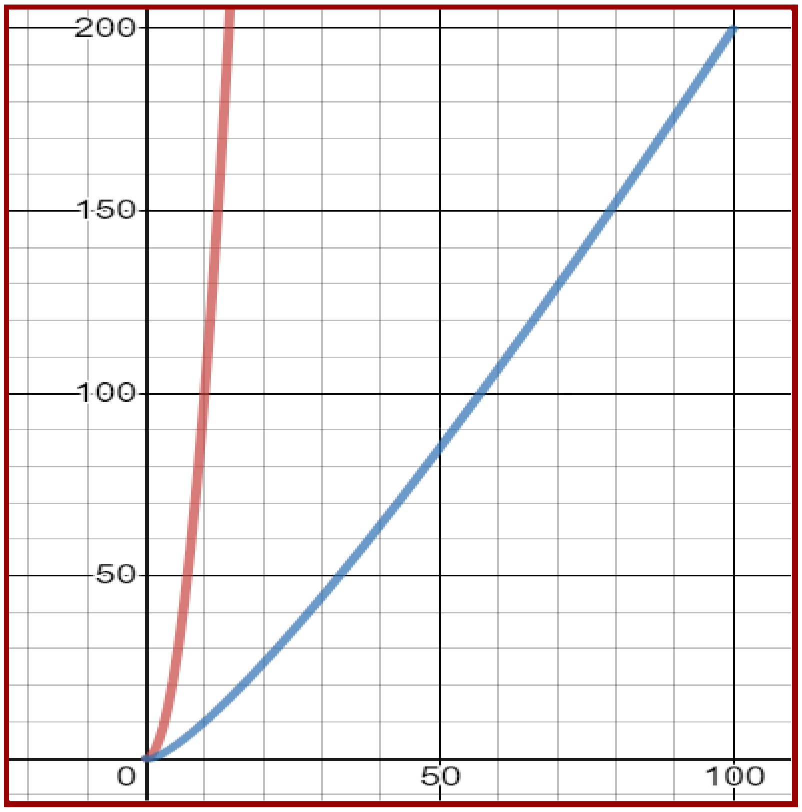


Figure 13: $f_1(x)=x^2$ (red) and $f_2(x) = x*\log(x)$ (blue) $\{0<x<100\}$
X axis - data passed through Y axis - time for transform to occur

A second form of transform is known as the Wavelet Transform. A Wavelet Transform works by decomposing a function into a set of wavelets, or wave-like oscillations that are localized in time. This makes determining where certain oscillations occurred easier. The major difference between this method and those previously presented is that the Wavelet Transform is less accurate when determining frequency, but more accurate in determining when that frequency occurred [27].

Typical autocorrelation (ACF) works by correlating a signal with a delayed copy of itself. It determines the level of correlation based on the time delay between these two signals, which can determine the main frequency present in a signal [75]. A new method, called Bitstream Autocorrelation [28], claims to be faster than the existing autocorrelation method. Bitstream Autocorrelation works by applying ACF (popular time-domain autocorrelation) which runs in O(n^2) time, activated when zero crossings of the input sine wave are detected. An error of less than 0.1 cents was reported when bitstream autocorrelation was tested with an artificially generated 261 Hz frequency [28]. A C++ library for this is available. This method is also effective in reducing noise and interference by other harmonics.

Another method studied was frequency estimation using circular statistics. An FFT method is used to determine the frequencies in the signal, which is then operated on using this circular statistics method. A vector is computed for each peak frequency using a peak amplitude and Equations 7 and 8, where c is a cent value. The sum of all cent vectors is computed and divided by the number of values N to get a mean vector of circular quantities. The mean vector is calculated and the system's frequency estimated by Equation 7 [76]. The algorithm did not have an audible estimation error when researched along with NLS and is more computationally simple for an electronic device. Please see Section 5 for further information as to how fundamental frequency detection ultimately played a role in the completion of this project. Below we compare frequency detection methods based on their advantages, disadvantages and time complexity in Table 2. To summarize Table 2, it was found that FFTs are the most time-costly algorithms, but are the most accurate at determining higher frequencies. FFTs have two major problems, they may determine harmonics of the fundamental frequency and they are less accurate at lower frequencies. Wavelet transforms are less costly in terms of time, but are more suited for finding when a particular frequency occurs than what frequency is most prominent in the signal. Autocorrelation also has a low time-cost, but is very susceptible to error when the signal is not uniform throughout the sample. If the signal changes or noise is introduced, it greatly affects accuracy. Simply polling a digital frequency meter is the least CPU-intensive strategy, but it adds electrical complexity and its accuracy is based on the accuracy of the circuit in determining frequency - which is a field of research in itself.

$$u = \frac{2\pi}{100} * c \tag{7}$$

Where u = main vector, c = cent value

$$F = \frac{1}{2\pi arg(u)} \tag{8}$$

Table 2: Comparison of Frequency Detection Methods			
Method	Complexity	Pros	Cons

Fast Fourier Transform	O(Nlog(N))	Shows all frequency components of signal.	Cannot compute fundamental frequency if it isn't present in the signal, tends to determine harmonics, less accurate at low frequencies.
Wavelet Transform	O(N)	Signals are sparse and therefore easier to filter out.	Better for determining when a signal occurs than what frequency the signal is.
Autocorrelation	O(N2)	Can find fundamental frequency as long as peaks are harmonically related.	Inaccurate if not perfectly repetitive signal.
Digital Frequency Meter	O(1)	Simple to implement in software, can be accomplished with an electric circuit.	Requires a constant zero to be accurate, the signal should not shift.

Through the research that was conducted, it was identified that there is a lack of autonomous string instruments that allow for both chord selection and selection of the notes of individual strings via a beginner-friendly user interface. While there are autonomous string tuning instruments that are capable of self-tuning, there is not a system that is capable of performing these operations for both the creation of chords and the selection of individual notes while also allowing the user to play the instrument manually or autonomously. By allowing for both forms of musical experimentation, string instrument performance becomes a more accessible form of self expression for more individuals.

3) Design Requirements

The goal of this project is to construct an autonomous string instrument that is capable of self-tuning, modification of said tuning mid performance, and creation of chords similar to those commonly played on guitars. The design requirements that will be adhered to in the pursuit of this goal have been refined based upon the commentary that was received during the Preliminary Design Review presentation.

Pitch Range

The instrument will cover a three octave range. A standard six string guitar can cover nearly four octaves in range [29]. However, as this project intends to utilize fewer than six strings, a three octave range has been deemed sufficient. Furthermore, the Milwaukee Servoelectric Guitar can only cover two octaves, so this design will be an improvement over the current state of development in this field. This will be achieved by utilizing a variety of strings which, between them, will be able to cover this range adequately

Dynamic Range

To emulate typical chamber music [30] [31] as well as remaining in a low risk threshold of noise [32], the device will have a range of 60 dB, starting at a 30 dB and extending to 90dB. This dynamic range accommodates the volumes at which it is safe to listen to a noise without appropriate protection for an extended period of time. [33]. Assuming the string has the same tension as a guitar, the force of a human finger should be enough to span this dynamic range.

Tuning Speed

The device will be able to adjust up or down a semitone within 100 milliseconds, as this would allow for the production of complex, quick pitch changes that will lend themselves to intricate musical production. The human brain is only able to detect changes in pitch if they occur in more than the time frame specified above [72]. While the TronicalTune device tunes a guitar within five seconds, this instrument should be capable of changing between tunings during performance for musical experimentation, and must operate at a more rapid speed. This project will reference the previous efforts of *Cyther V3*, which set 100 ms as an objective and was unable to reach it, achieving a top speed of 166.089 ms/semitone. The documentation of the previous project recommends acquiring more powerful motors and paying special attention to the friction generated by the tuning system as potential sources of improvement [15].

The tuning speed of the instrument can alternatively be viewed as the interval of time it would take to change tuning between every two consecutive notes at maximum speed. Tuning intervals larger than a semitone will, naturally, take longer to reach. Please see Section 6) System Operation and Design for further explanation of how this requirement was implemented.

Input Capabilities

The instrument should be able to accept MIDI-based instructions from a user interface, which will cause the instrument to tune wholly to a certain chord, or to tune individual strings to individual notes, depending on what the user would prefer. Another requirement for the input is that the time it takes for the instrument to receive instructions from the user interface does not significantly increase the tuning time of the instrument. This operation should fit within the 200ms per tuning operation goal outlined previously.

Latency

It should not take the device a long time to receive commands from an external source. This can be changed with baud rate and communication protocols. 3ms will be used for the latency requirement as it is insignificant enough not to affect the operation of the device. All software should be non-blocking. The time it takes to initialize will be subjectively defined as an interval that does not annoy the user as taking too long, which we will set as ten seconds. The time it takes to loop through the program once should be 10 ms. This is a subjective requirement based on allowing the control system to be called enough to smoothly tune the system.

Tuning Accuracy

The minimum frequency difference the human ear can perceive is 5-6 cents [34] and notes sound out of tune when the tone played is at least 12 cents removed from the target tone [35]. The device should maintain, at most, a six cent difference between the actual and goal frequency. This should be maintained when other strings are playing, and when the instrument is interacting with a player.

Weight/Dimension

This device should be portable, and should be able to fit through doorways, which have a standard width of three feet. The weight requirement has been set based upon the weight of other stringed instruments not held by the player; a zither typically weighs twenty pounds, and a double bass weighs from twenty to twenty-five pounds. Cellos weigh only five to seven pounds, which may increase to twenty-five pounds including the case and accessories, and are generally considered to be portable instruments. Therefore, the instrument constructed for this project will weigh approximately twenty-five pounds including its case and accessories.

Target Audience

This product will be designed for individuals who would like to learn to play a string instrument, but do not have the ability to both create chords with the frets of an instrument and strum simultaneously. This could also benefit string instrument players who have lost the ability to perform the previously listed actions, whether through infliction of a disability or other means. Furthermore, a musical artist could program the instrument to autonomously perform a song without human intervention with the incorporation of a web-based control system.

4) Overall System Design

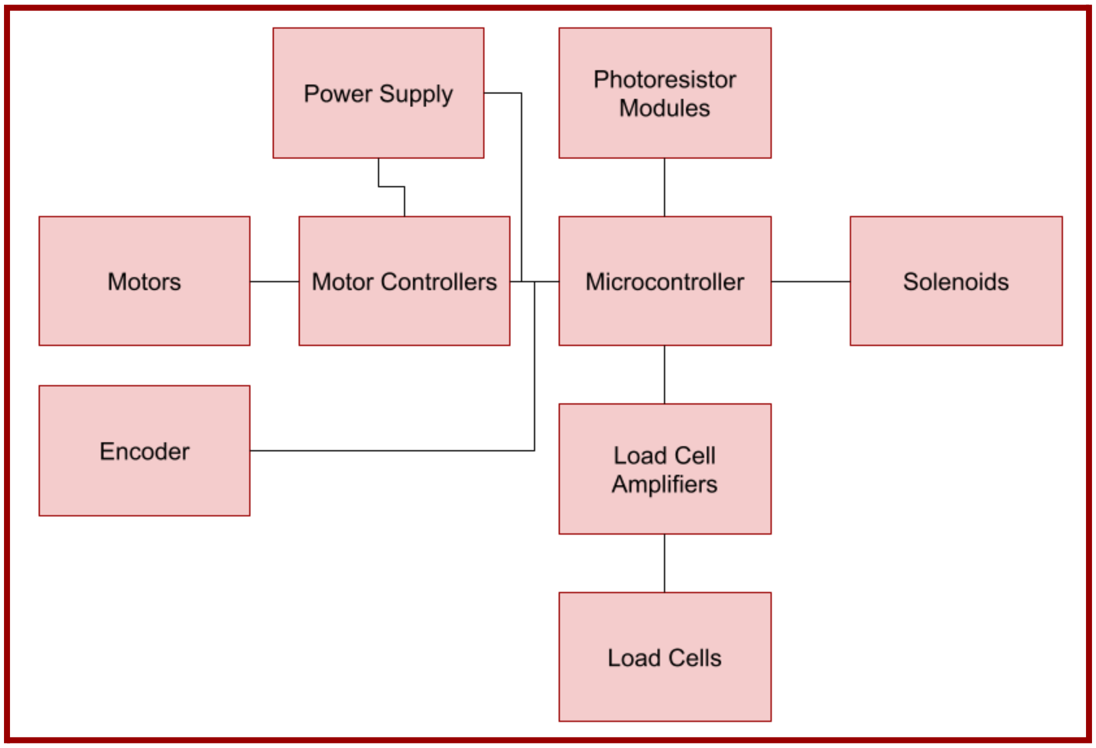


Figure 14: Overall System Block Diagram

The instrument constructed for the fulfilment of the above design requirements will consist of five strings, each of which capable of spanning an octave. These strings will be capable of changing and maintaining their tunings autonomously via the utilization of three sensors per string - a load cell, an encoder, and a photoresistor - in conjunction with motors to adjust the tension of the strings. The load cell is utilized as the primary method for determining the tuning of its string via tension, the encoder functions in cooperation with the load cell in the string tuning process with motor shaft positioning feedback, while the photoresistor is used to double-check the tuning of its string via counting gate interrupts in order to calculate the frequency of the string. The motor is attached to a tuning machine, which will be utilized to change the string's tension as instructed by the microcontroller, an Arduino Due, based upon sensor input. An electromagnetic pickup will be used for the instrument's audio output. The pickup will not require external power, and will be kept independent of the rest of the system.

4.1) Operation Process

When the system is turned on, the sensors will calibrate first. The load cell will tune the string to what it believes is the correct tension measurement for the target frequency. This target frequency is the frequency of the lowest pitch the string supports by default. A lookup table is generated based on Equation 4, a constant, and an array of frequencies the string should support.

The program then figures out what tension value should exist for each of these frequencies, and adds that to the lookup table. The program table also creates an additional lookup table for determining the encoder position from a given pitch. When first created, this encoder position is estimated based on the difference between current and determined load cell values multiplied by a constant. For any pitch, there are two entries in separate lookup tables that match that pitch to both a tension value and an encoder value. The microcontroller will reference this lookup table when receiving a tension measurement which is converted from an analog value read from the load cell amplifier. Next, the motor's position will be calibrated to the encoder position saved in the map for the respective pitch it is tuning to. Then, the user will pluck the string, allowing the photoresistor to double check that the target frequency has been reached. If it has, the instrument is ready for use. If it has not, the system will adjust the tuning of the string until the photoresistor determines the target frequency has been reached. The tension will then be recorded, and the lookup table which is being referenced to determine the required tension to reach each frequency will be updated to reflect this new value.

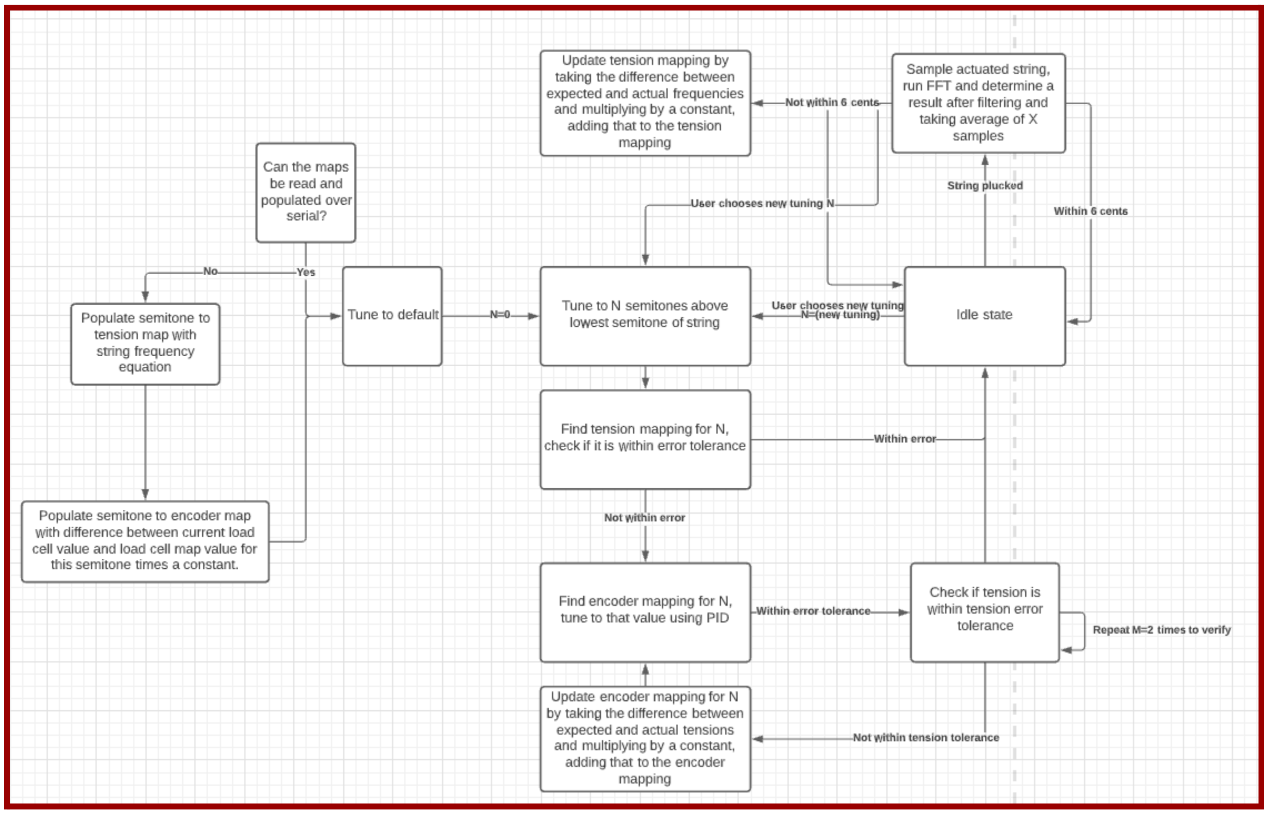

Update tension mapping by taking the difference between expected and actual frequencies and multiplying by a constant, adding that to the tension mapping
Sample actuated string, run FFT and determine a result after filtering and taking average of X samples
Not within 6 cents
String plucked
Within 6 cents
Can the maps be read and populated over serial?
User chooses new tuning N
No
Yes
Populate semitone to tension map with string frequency equation
Tune to default
N=0
Tune to N semitones above lowest semitone of string
User chooses new tuning N=(new tuning)
Idle state
Find tension mapping for N, check if it is within error tolerance
Within error
Not within error
Populate semitone to encoder map with difference between current load cell value and load cell map value for this semitone times a constant.
Find encoder mapping for N, tune to that value using PID
Within error tolerance
Check if tension is within tension error tolerance
Repeat M=2 times to verify
Update encoder mapping for N by taking the difference between expected and actual tensions and multiplying by a constant, adding that to the encoder mapping
Not within tension tolerance

Figure 15: Operation Process - Specific

While the system is in use, the user will choose which chord they want the system to tune itself to via a user interface. The tuning of the strings will be primarily changed and maintained via the load cell, and the photoresistor will operate as a secondary sensor to ensure the accuracy of the load cell. The user can choose a new chord at any time, allowing them to play the instrument as one traditionally would - when the user would typically change the chord by moving their fingers to a different fret of the instrument to change the string's vibrational length, they simply need to press a button, and the system will change the tuning for them. Please see the block diagram below for further information as to how the tuning system's modules are connected to one another.

An additional component to the final instrument design is the solenoid array. The solenoids will strike the strings of the instrument when instructed to. These components can be utilized to initialize the photoresistor's frequency sensing, or can be used to play music autonomously.

4.2) Component Consideration & Procurement

The following sections detail how the sensors and motors utilized in the final prototype were selected, and discusses which components were considered and discarded throughout the design process. There will also be an overview of how the number of strings on the instrument was determined, a calculation of the device's power consumption, and an explanation of the motor control mechanisms utilized. Below is an image of the electrical architecture of the system, which displays how each selected component will communicate with the rest of the system.

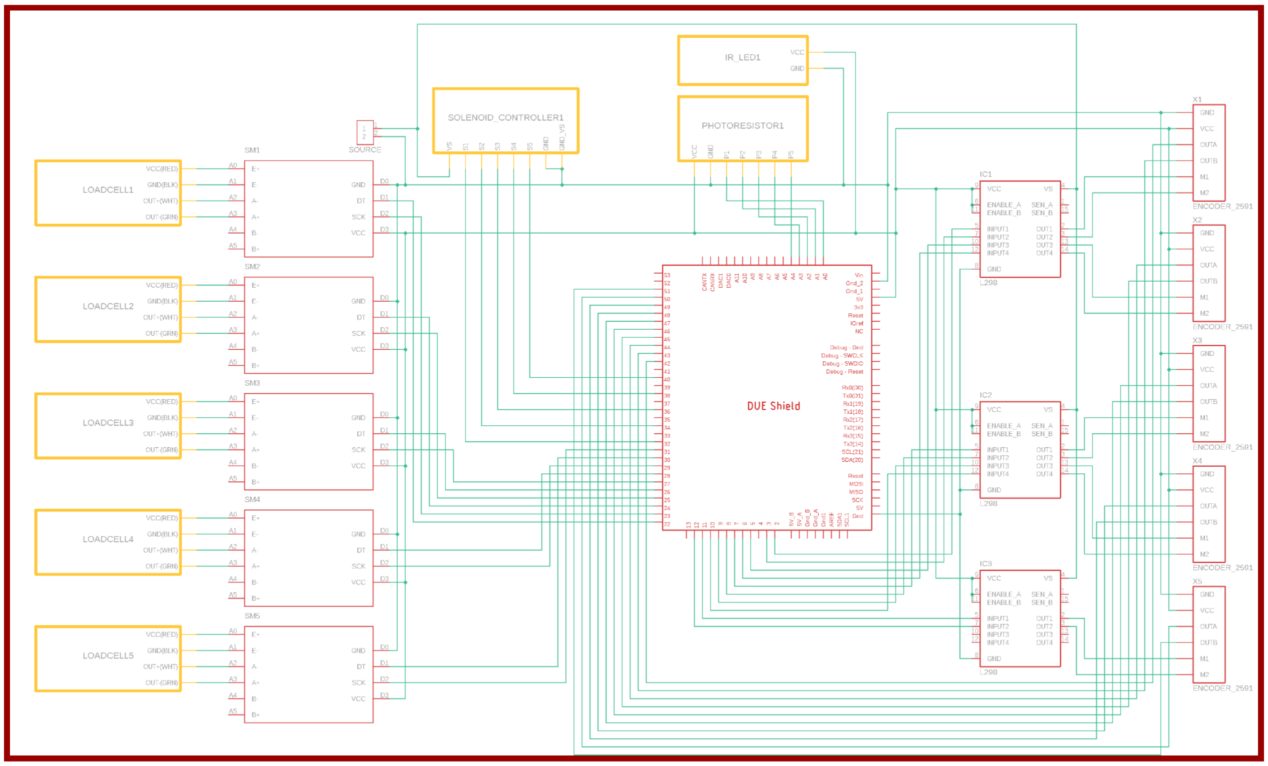


Figure 16: Electrical Architecture

4.2.1) Frequency Sensors Considered

4.2.1.1) Hall Effect Sensor

Hall Effect is a phenomenon in which a voltage difference is created in an electrical conductor when a magnetic field is applied perpendicularly to its current flow in a thin film. Hall Effect sensors are tools utilized to detect this change in the magnetic field of the environment, as shown in Figure 17 below. The sensor could be used to obtain a signal that corresponds to the changing position of a vibrating ferromagnetic string. The changing position of the string is also a representation of the sound output. Therefore, the signal could be processed to find the fundamental frequency of the vibrating string.

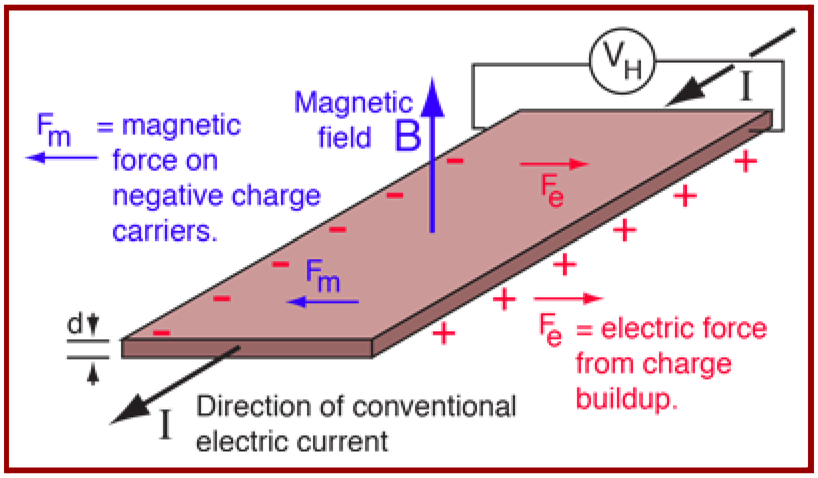

F_m = magnetic force on negative charge carriers.
Magnetic field B
V_H
I
F_e
F_m
d
F_e = electric force from charge buildup.
I
Direction of conventional electric current

Figure 17: Hall Effect Diagram [41]

Hall Effect sensors were considered for sensing the string vibration due to its non-intrusive measurement method. Due to the devices' significantly low resistance, the received input value's accuracy is not affected by the sensor [42]. Furthermore, the sensor offers low impedance, which reduces its noise input. Hall Effect sensors are typically water, vibration, and dust resistant.

The disadvantages to this system, however, ultimately caused it to be eliminated as a potential sensing method. There are concerns regarding the accuracy of these sensors - interference may occur from external magnetic fields potentially caused by batteries, motors, or nearby magnets in the system, impacting the sensors ability to accurately represent the magnetic field of the string within the detection distance of 10cm [43]. This could be solved by utilizing shielding and increasing the distance of unwanted magnetic fields from the device [44]. However, each sensor's proximity to the next string's sensor could cause significant problems that would be difficult to overcome with shielding, as the strings will be positioned closer than

10cm together. Furthermore, the output signal is relatively small - at around 10-20 mA, an amplifier will be needed for its signal to be discernible.

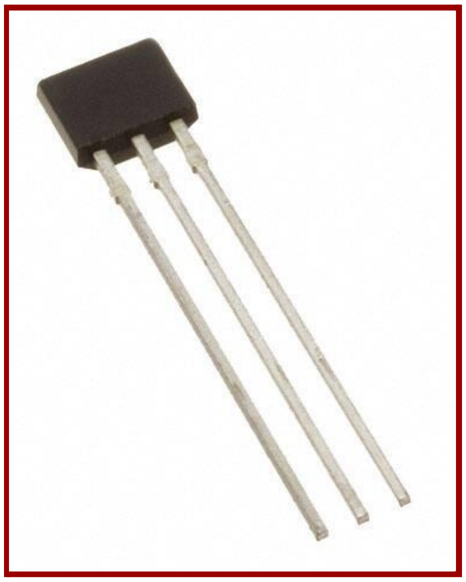


Figure 18: An Example of a Hall Effect Sensor [45]

4.2.1.2) Electromagnetic Coils

Electromagnetic coils are composed of wire wound repeatedly in a circle. Their purpose is to either produce a magnetic field by applying current to the wire, or to induce a signal due to changes in the magnetic field at the center. Coils are the most popular type of guitar pickup used in electric guitars, which consist of a coil utilized in conjunction with permanent magnets affixed under each string. As the ferromagnetic strings vibrate when played, the flux in the permanent magnets changes with their vibration. The coil then induces the signal due to the changing magnetic field. Much like the hall effect sensor, the coil would produce a signal that corresponds to the sound output of the vibrating string. This signal could be processed to find the string's fundamental frequency.

Initial testing with the coil exposed major drawbacks in their application for this project. Since the signal of each string must be processed individually, a separate pickup would be required for each string. However, individual pickups tend to acquire a great amount of interference from nearby strings, as well as 60Hz noise from the environment. These problems could be overcome with increased signal processing. This may limit the ability of the instrument to reach the rapid tuning goals.

A pickup will still be utilized for the audio output. The instrument's body will not be constructed with resonance in mind, so it will be necessary to electrically amplify the signal. A ¼ inch jack will be connected to the instrument, so a standard guitar amplifier can be utilized. These two components will operate independently from the rest of the system.

Figure 19: An Example of a Guitar Pickup [46]

4.2.1.3) Photoresistor

A photoresistor is a semiconductor that operates via current modification of a circuit when exposed to light, and typically functions within a bandwidth of 250kHz. The current of the circuit will increase by the gain of the photoresistor chosen when exposed to light. The device can operate utilizing a broad spectrum of light - from ultraviolet to almost infrared.

These devices work very well for the application of autonomous tuning when a string is positioned between the photoresistor and a light source - for the purposes of this project, a 650nm laser diode will be utilized. When the string is at rest, the laser's light cannot pass to the photoresistor, minimizing its exposure. After the string is plucked for the first time, the photoresistor will change the current of the circuit, allowing the microcontroller to determine the period of the string by detecting the amount of time it takes for the string to complete one oscillation. This information can be utilized to calculate the frequency of the string.

As with any light sensor, ambient light may affect the values being received by the microcontroller for detection purposes. It is important to construct a housing for the photoresistor to decrease the chance of sensor interference from external light. An additional challenge will come from ensuring that the light from the adjacent strings do not impede the ability to accurately read the frequency of the string being tuned. Ensuring the beam of light coming from the laser diode is thin and focused resolves this problem. Figure 21 below details the ways in which the electrical connections will be placed on the prototype board, and Figure 22 represents how the laser diode board will be configured. The spacing of these circuits must be constructed carefully in order to ensure that the laser diodes are positioned directly across from the photoresistor.

Figure 20: The Reactive Head of a Photoresistor [47]

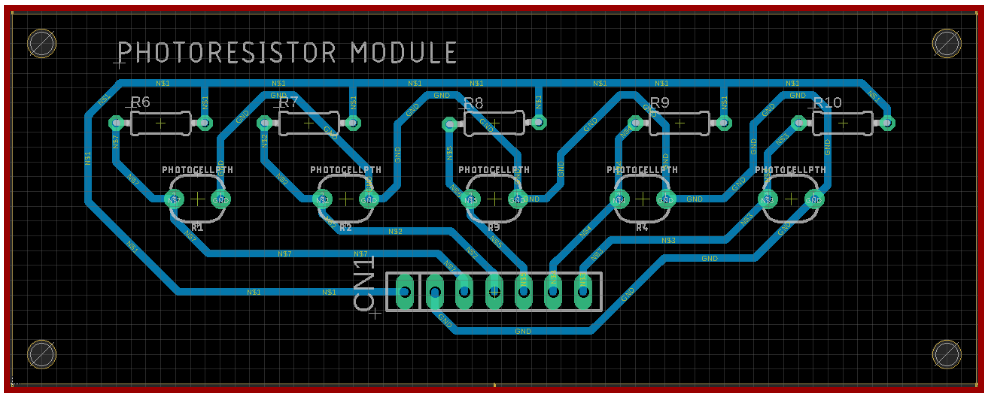


Figure 21: Photoresistor Module Schematic

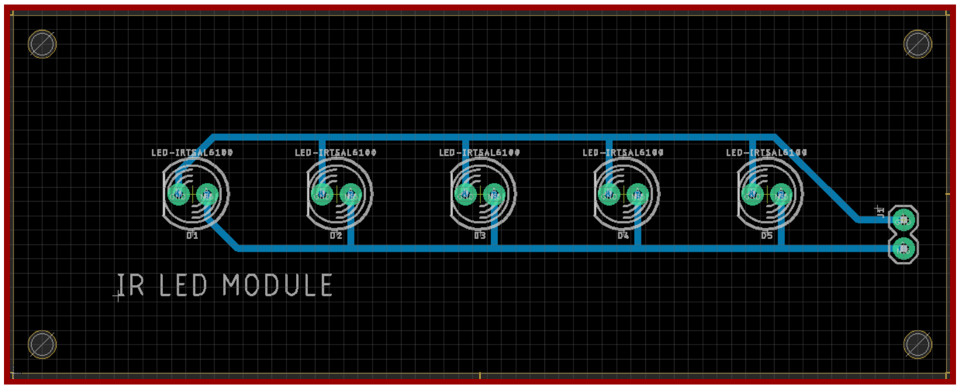


Figure 22: Laser LED Module

4.2.1.4) Position Sensitive Detector

A position sensitive detector, or PSD, is a sensor that measures the position of a spot of light. This can occur in one dimension or two, depending on the construction of the device. There are two common classes of sensors currently on the market - isotropic sensors, and discrete sensors.

The isotropic sensors utilize a diode with heavily doped n-type and p-type regions that surround an undoped I-semiconductor region - these are referred to as PIN diodes [48]. Figure 23 displays the makeup of these types of diodes. When electricity is passed through the gate of the transistor, labeled “G” in the figure below, electricity can be conducted between the n or p regions of the component, depending on whether you are considering the NMOS or PMOS circuit. Without the allowance from the gate, it becomes nearly impossible for electrons to bridge the gap between the source and drain regions of the transistor, preventing the flow of electricity.

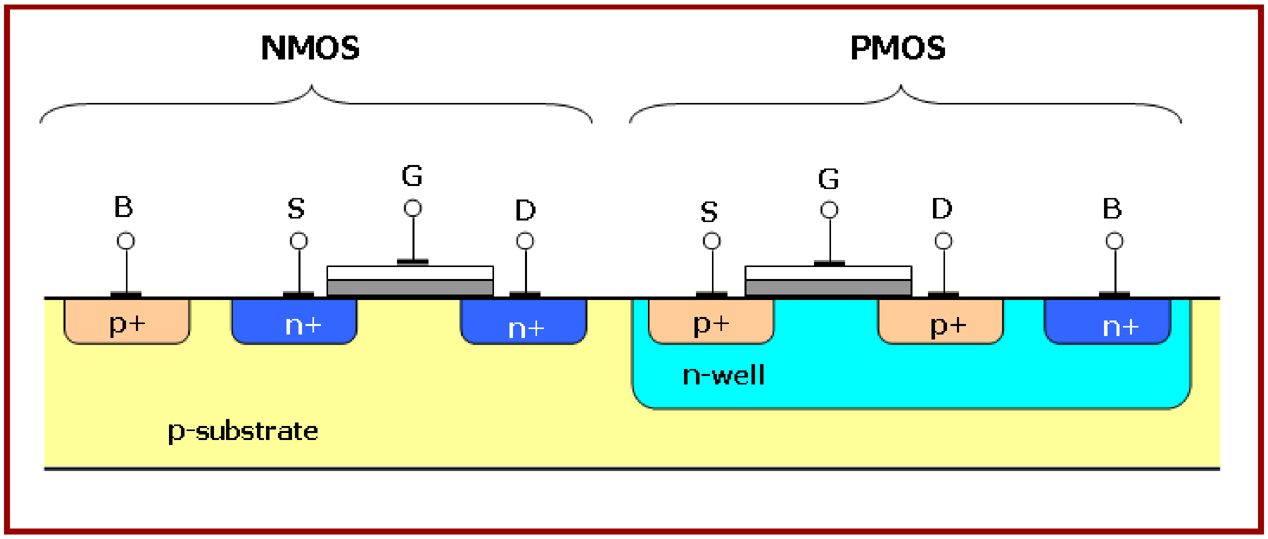


Figure 23: A Diagram of the Makeup NMOS and PMOS Transistors [50]

A photoresistor works in a similar fashion - when light hits the surface of the device, valence electrons become excited and gain enough energy to cross a gap between its internal components, allowing electricity to flow. This means that when exposed to a spot of light, the resistance in the device changes, allowing a microcontroller to understand the position of the spot of light. The second class of sensor is the discrete sensor. Typically CMOS type devices, these sensors operate by processing the exposure of pixels on the device, and then computing the location of the light pinpoint by measuring the distribution of light across said pixels [49].

Figure 24 displays a simplified image of the grid that is found inside the PSD. When light shines on a part of the sensor, four current values are read - these are labeled as I_1 through I_4 in Figure 25. These values are found by thinking of the grid above as axes on a graph - when one of the squares experiences a change in voltage, the current coordinates given to the microcontroller will provide a precise location with some computational processing of the data. This allows for the position of the light point to be determined via software analysis of the signals - the current values are used to determine where the light point rests, as the closer the light is to a certain point, the larger the current change will be.

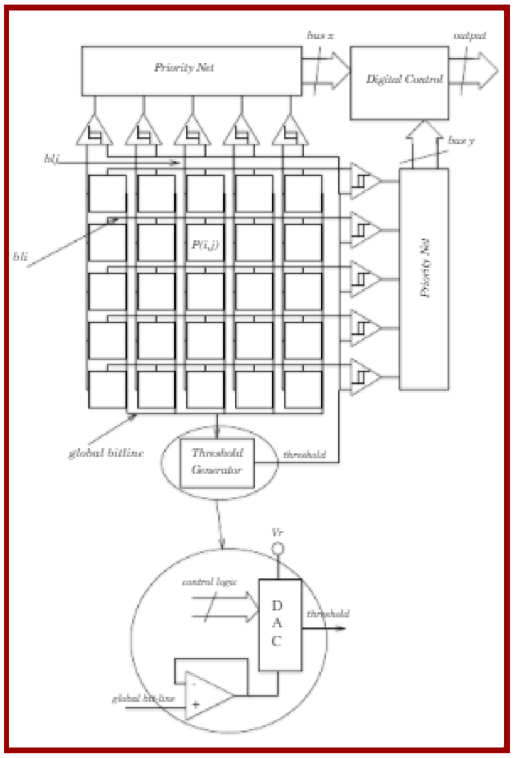


Figure 24: PSD Schematic Block [51]

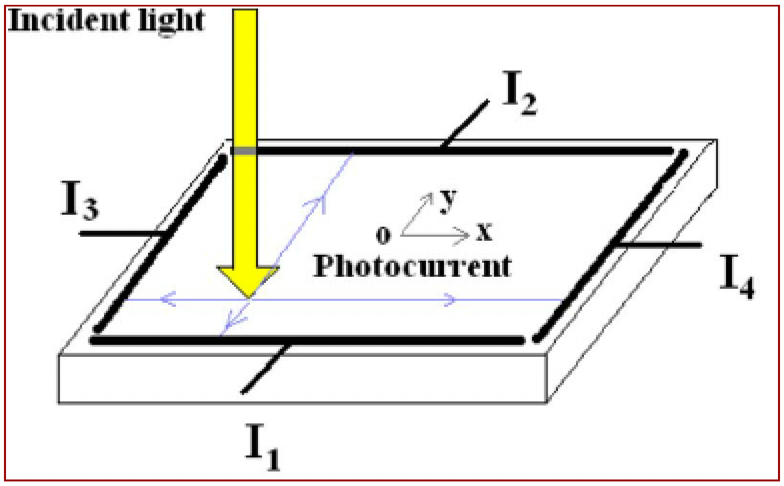


Figure 25: A Position Sensitive Detector’s Two Dimensional Sensing Capabilities Explained [52]

Either of these sensors could potentially be utilized as a solution to autonomous tuning. A mechanism could be constructed that shines a light directly over the strings being tuned, and when the string is plucked, the light will be exposed to the sensor as the string oscillates. These periods of light can be used to calculate the period of the wavelength of the string, which can in turn be used to calculate the frequency. As can be seen in Figure 26 below, the string will block the light from the laser while at rest, as it will be positioned between the two electronic components. The speed of these sensors would not impede the accuracy of the device being constructed - the isotropic sensors can measure up to 100kHz, and the CMOS sensors can operate at 250kHz. However, it may be difficult to accurately pinpoint the movement of the string, since these sensors specialize in the movement of light, not the movement of dark spots - as the sensor sends back information as to the light spot's two dimensional location, it may be challenging to computationally sift through the extraneous information to determine the frequency of movement. This challenge could be overcome by only calculating the X component of movement and not the Y component - these values are found by utilizing current outputs from the sensor. These devices are able to pinpoint the movement of a spot of light very accurately in two dimensions, which may be unnecessarily sophisticated for the purposes of the project - only detection of the light in one dimension is necessary. While it would have been possible to construct a functional device with these sensors, the photoresistors are adequately sophisticated for the purposes of this project, and the use of a PSD would have been unnecessarily complex.

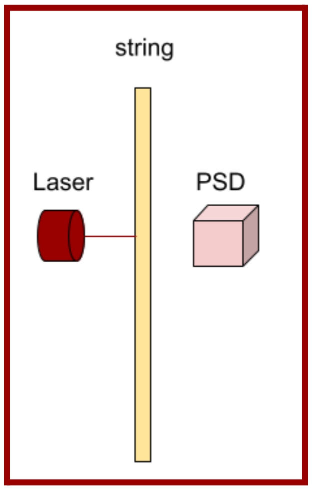


Figure 26: PSD String Diagram

4.2.1.5) Piezoelectric Sensor

A piezoelectric sensor utilizes the piezoelectric effect to measure the electrical potential caused by mechanical displacement. By deforming a quartz crystal, the sensor outputs the force required to create that change. Piezoelectric sensors can measure a variety of inputs, such as pressure, vibration, and acceleration. [53].

The piezoelectric sensor can be implemented to sense the frequency of each respective string by positioning the string on a semi-dome spacer, which is placed directly on top of the sensor. Due to the ability to sense the pressure applied to the sensor's surface, the piezoelectric element can measure the vibration of the string, and convert it into an electrical signal to be processed by the microcontroller. However, due to the desired close proximity of the strings for this project, each piezoelectric sensor will pick up on the signals from the surrounding strings, thus rendering signals more difficult to process.

The piezoelectric sensor does have potential to be implemented as a frequency detection method; the component is compact, has a strong frequency response of 20 Hz to 10 kHz, and is relatively inexpensive. However, the sensitivity of the device can be impacted by temperature variation, and may damage the circuit by overloading the voltage of the device [53][54]. It was noted by Nicolas Lynch-Aird and Jim Woodhouse that the capacitance of the piezoelectric transducers they utilized to autonomously tune the strings of a harp fluctuated considerably with environmental temperature changes [55]. In order to avoid this potential complication, this sensing method was eliminated in favor of the more stable photoresistor method.

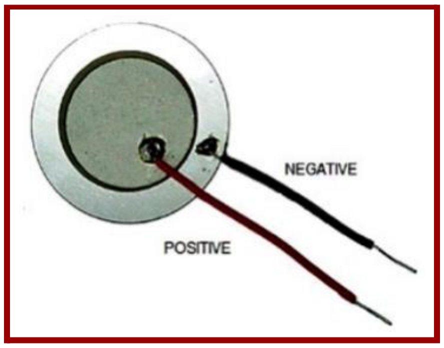


Figure 27: An Example of a Piezoelectric Sensor [56]

4.2.2) Tension Sensors Considered

4.2.2.1) Load Cell

The data from the load cell module can be used to tune an instrument string without requiring string vibration. The load cell is constructed from strain gauges attached to a metal bar. These strain gauges measure a resistance change that occurs when the load cell bends from a load, such as tension. This measured resistance changes in proportion to the load. However, this change is very small, and a circuit such as a Wheatstone Bridge can be utilized to accurately measure it. Using a Wheatstone Bridge can also lessen the effects of environmental factors, such as temperature fluctuation, on the sensor's strain measurements [38]. A schematic for the Wheatstone Bridge can be found in Figure 28 below. The Wheatstone Bridge configuration consists of four resistors in a square with a voltage meter to bridge two corners of the square and power and ground connected to the other corners[82]. In the case of the load cell, the resistors in the configuration are the strain gauges attached to the load cell. The four wires that extend out from the Wheatstone Bridge circuit of the load cell are the Excitation +, Excitation -, the Output -, and Output + wires which are labeled in Figure 28 below. The signal from the load cell, amplified from the load cell amplifier, is passed to input ports on the microcontroller for processing. For the purposes of this project, the load cell can be used to measure the tension of the string, which will change when the string is tightened or loosened by the motors.

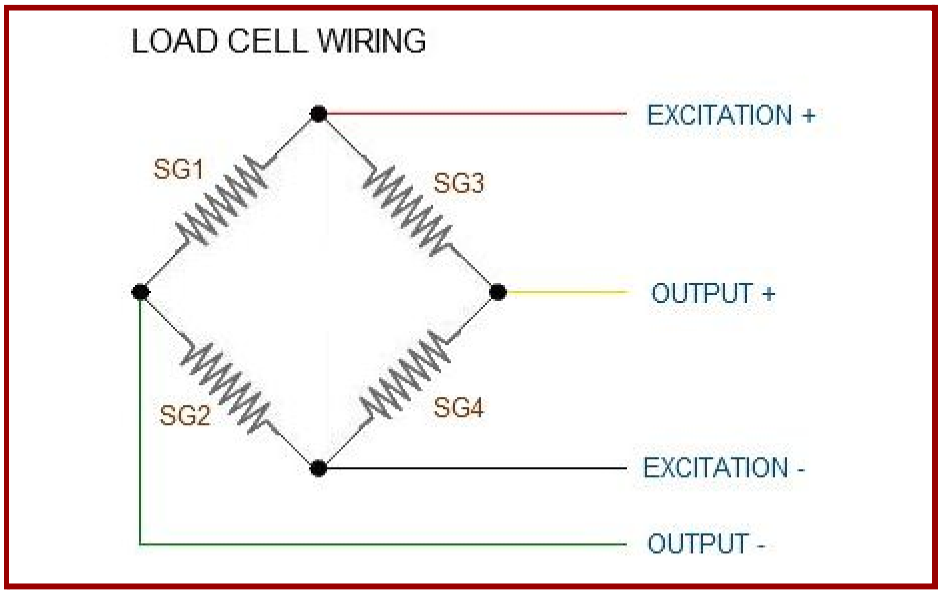


Figure 28: Schematic for Wheatstone Bridge Configuration [39]

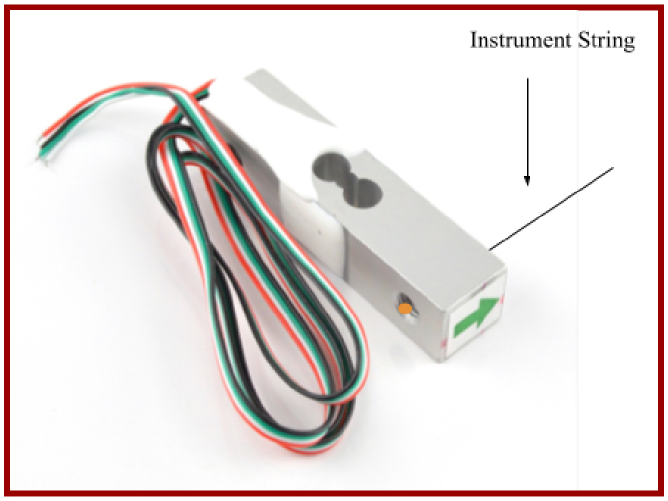


Figure 29: Load Cell [40]

To implement the load cell into the project, every string of the instrument will have a load cell attached at one end in order to capture all necessary measurements. The load cell will be securely mounted in a way that allows it to bend according to the changes in the tension of the string it is attached to. The string will be run through the hole in the load cell underneath the green arrow shown in Figure 29, and will be stung in such a way that force is applied in the direction that the arrow is pointing. For each string, the current tension measurement from the load cell will be used to inform the response of the rest of the electromechanical system to reach the target frequency. The tension measurement is computed by a library that receives serial data from the load cell amplifier, and converts that to an analog value calibrated to pounds of tension. This is calibrated by zeroing the sensor and placing a 1lb weight on the load cell, then dividing this by the tension determined as previously described.

Testing was performed for the load cell using a test rig with a single string, an HX711 load cell amplifier, a load cell with a rating of 50kg, and a tuning machine. The schematic for the amplifier can be found in Figure 17 above. A picture of the test rig used can be found in Appendix H. The testing process included creating a lookup table that maps string tensions to frequencies. To create the lookup table the string was manually adjusted to specific tensions read by the load cell and the frequency was recorded using a tuner. Next, the accuracy of using the tensions from the lookup table to tune to a specific frequency was recorded. Table 3 below shows some of the data from testing the load cell. The first two columns represent the tension and frequency values saved in the lookup table. The third column shows the actual frequency recorded by the tuner after tightening the string to the tension value in the lookup table. The final column shows the difference in cents between the expected frequency from the lookup table and the actual frequency recorded from the tuner after tuning using the tension value.

Table 3: Load Cell Test Data

Lookup Tension (lbs)	Lookup Frequency (Hz)	Actual Frequency from Tuner (Hz)	Cent Difference
14.2	65.4	65.2	-0.2 cents
17.9	73.4	72.7	-0.7 cents
22.7	82.4	81.7	-0.7 cents
25.5	87.4	86.2	-1.2 cents
32.5	98.0	97.5	-0.5 cents

The frequencies that resulted from this test were within the accuracy range of 6 cents specified in the project goals. The tests also verified the repeatability of the functionality of the load cell when this accuracy was maintained in subsequent tests. Due to the testing results, the load cell was considered for implementation in the project design. To maintain tuning accuracy, the load cell can be used in combination with another frequency-detecting sensor, so that the lookup table may be updated if the tension does not match the target frequency. Since the lookup table values were experimentally determined, and the string length can change due to the warping of the instrument body resulting in a change in these values, a second method of determined frequency can be utilized to ensure that these potential error sources are resolved.

Sensors play an invaluable role in the design of an autonomous tuner. The purpose of the sensor module is to detect the frequency of a string's vibration, so that it may be compared to the target frequency and tuning may be changed if necessary. When selecting a sensing method, it is important to consider the physical properties of the sensor, such as size, environmental resistance, and electrical requirements. Furthermore, one must consider the type of signal that will be outputted from the sensor, as this will affect the rest of the tuning system. Six sensors were considered for this purpose, as are outlined below. These sensors detect frequencies in different methods - some will observe the tension, others will check for movement, and still more seek an impact on its magnetic field.

4.2.3) Comparison of Tension & Frequency Sensors

Below is a table that compares the advantages and detriments of each sensor discussed above.

Table 4: Sensor Comparison

Sensor Type	Pros	Cons	Sensing Method
Hall Effect Sensor	● Low impedance, Low noise ● Non-intrusive ● Resistant to vibration, dust, and water ● Bandwidth of 20 Hz - 25 kHz	● Low output, requires amplifier ● External magnetic field interference could be crippling	Observes changes in a magnetic field
Strain Gauges	● Senses tension accurately ● Testing proved it	● Load cells can be bulky in size ● Must ensure the material it	Measures the change in tension

	to be effective ● No signal processing - faster frequency determination	is attached to is strong enough to sustain the force applied to this sensor	
Electromagnetic Coils	● Senses string signal accurately	● Low output, requires amplifier ● External magnetic field interference ● Pre-manufactured pickups are expensive ● Individual pickups interfere with each other	Measures the change in a magnetic field
Position Sensitive Detector	● Senses spots of light very accurately - accurate frequency measurements ● Very quick response time	● More sophisticated than is warranted for the purposes of this project ● Must be shielded from for ambient light ● More complex software implementation	Counts the number of times the string passes between it and a beam of light
Photoresistor	● Small in size ● Quick enough response time for the purposes of this project ● Noninvasive configuration	● Must be shielded from ambient light	Counts the number of times the string passes between it and a beam of light
Piezoelectric sensor	● Good frequency response, 20 Hz - ● Small in size ● Flexible shape	● Small output, requires amplifier ● Measurement variation due to temperature ● Very susceptible to environmental impact	Measures electrical potential generated by mechanic displacement

Upon the completion of this research, it was decided upon that the photoresistor and the load cell would be the sensing methods pursued - the load cell will be utilized as the primary detection method, and the photoresistor will be used to ensure the accuracy of the first method. The load cell was the most consistent sensing method throughout the testing process, and tension sensing should not be impacted by environmental factors. Furthermore, the photoresistor was the second most promising sensor after testing and evaluation was completed as it is the least susceptible to environmental interference, especially if a case is constructed to prevent the effects of ambient light on the circuit.

4.2.4) Motors

The motor module's aim is to control and adjust the tuning peg to ensure that the string is tuned within the 6 cents accuracy as specified in the design requirements. It also must be capable of moving to its final position rapidly in order to adhere to the goal of achieving tuning within 100ms. The motor utilizes an encoder to control its angular positioning, which receives adjustment input from the microcontroller's signal processing and PID control, which will be discussed further in Section 6.1. After having its position initialized, the motor will receive an analog signal from the input pins of the microcontroller, which is processed via the encoder for angular control. Additionally, a motor driver is needed to activate the motor every time position adjustment is needed.

The motor in this project is implemented to be attached to a tuning machine, which is constructed via a worm gear. The worm gear should be capable of maintaining the average tension of a guitar string, which sits between 13.5-17.9 lbs-in [57]. Furthermore, the string will not need to be tuned to a higher tension than 40-45 lbs-in, or 640-720 oz-in [58]. Based on the knowledge that the efficiency of the tuning machine is roughly η=0.9 [59], the minimum and maximum torque required can be found utilizing Equations 9 and 10 - which are the equations for calculating worm gear transmission maximum and minimum torque, with: T_a is the torque needed for the motor shaft, T_b is the torque needed to hold the string, Z_B is the number of teeth on the worm drive, and Z_A is the number of teeth on the worm.

$$T_a = \frac{T_b}{(\eta \frac{Z_B}{Z_A})} = 16 - 21.21oz - in \tag{9}$$

where T_B= [13.5-17.9lbs-in], Z_B = [15], Z_A = [1].

$$T_a = \frac{T_b}{(\eta \frac{Z_B}{Z_A})} = 47.41 - 53.33oz - in \tag{10}$$

where T_B= [40-45lbs-in], Z_B = [15], Z_A = [1].

The motor module will be mounted on the head and neck of the instrument using a mounting bracket, with its shaft attached to the tuning peg. As per the design requirement, the module would need a set of 5 motors controlling 5 tuning machines. Anytime there is a change in analog input from the microcontroller, the respective motor will be activated. This will cause the tuning peg to turn, either tightening or loosening the string depending on the needs of the system. The motor will send its angular position to the microcontroller for position monitoring - this information will also be stored for faster adjustment in the future. Below is an examination of the motors considered for use.

4.2.4.1) DC Motor

One type of motor that was considered for the project is a DC motor. The two common types are brushed and brushless DC motors. Brushless motors would be more effective for this project because they require less maintenance, are more efficient, and produce less noise when compared to brushed motors. An advantage of the brushless DC motor is that high torque can be achieved at low speeds. Some disadvantages of the brushless DC motor include that it is more expensive than the brushed alternative, and while it is less noisy than their brushed counterparts, they can still be electrically and mechanically noisy. The position of a DC motor can be controlled with the use of a rotary encoder, which functions by measuring angular position and translating it into an analog or digital signal. A rotary encoder can also allow for determining measurements other than position as well; from the angular position reading of the encoder, the direction of and speed at which the motor shaft is turning can further be determined.

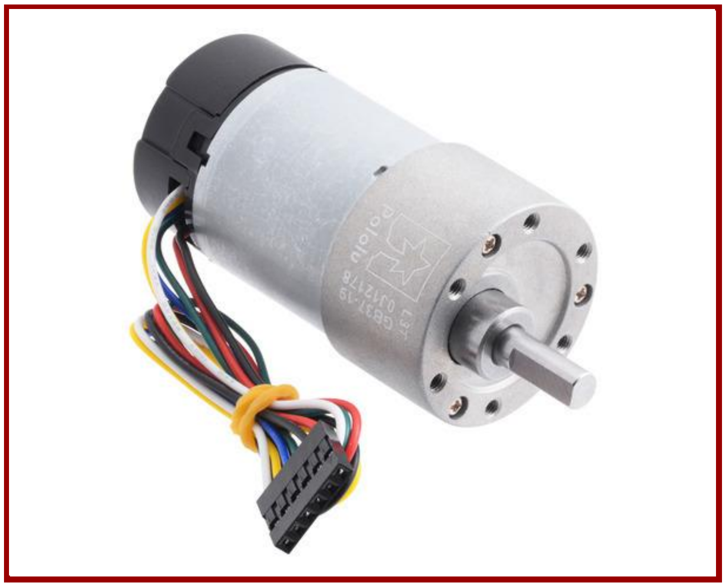


Figure 30: Motor with Encoder [60]

4.2.4.2) Linear Actuator

Another option for adjusting the string tension is a linear actuator. This device converts the rotational motion of a motor into linear motion. The direction of the motor movement affects whether the shaft of the linear actuator expands or contracts. Some advantages of the linear actuator include reliability from having no-wear components, and the ability to achieve high-speed acceleration, accuracy, and repeatability [61]. There are three types of linear actuators: pneumatic, hydraulic, and electric. To adjust the shaft of the actuator, the electric option uses a motor, gears, and a lead screw, the pneumatic uses pressurized air, and the hydraulic uses an incompressible liquid. Compared to the electric linear actuator, the pneumatic and hydraulic options require more external components. For the pneumatic, a compressor and tubing is required and for the hydraulic, a pump and fluid reservoir are required. Pneumatic and hydraulic linear actuators also require more maintenance over time compared to the electric because they can leak. The electric linear actuator was determined to be the best suited for this project because compared to the other options it requires less external components which more closely fits with the portability design requirement for the instrument, it has minimal noise levels, and there is little to no maintenance required over its lifetime. Some versions of electric linear actuators also have potentiometers or optical sensors attached to provide positioning and motion control capabilities [62]. The disadvantage of the electric linear actuator in comparison to other varieties is that they are relatively more expensive. Another disadvantage is that the driving mechanism of the linear actuator option limits the possible speed that can be obtained. A common top speed for a linear actuator is 180 to 200 mm/s [81]. Therefore, since the linear actuator option has a lower limit for the possible speed that can be obtained and the cost of electric linear actuators that provide positioning is more expensive compared to the motor options, it was determined that the other motor options would lend more versatility to this project.

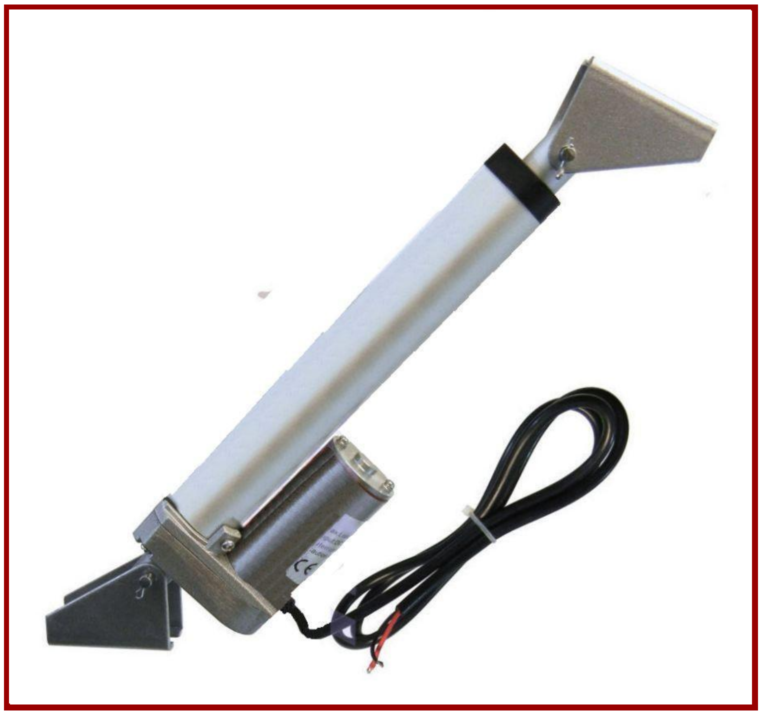


Figure 31: A Linear Actuator [63]

4.2.4.3) Servo Motor

A servo is a rotary actuator which is capable of controlling the speed and the angular position of the motor. Servo motors control angular position by utilizing a built-in position feedback sensor. There are two types of servo motors that may be considered for use: the hobby servo and the continuous rotation servo.

The hobby servo is a common servo used for controlling position and speed. It utilizes an encoder to assist in controlling its exact angular position. This servo can only spin in a range from 180^{O} to 270^{O}, depending on the motor's specification. The continuous rotation servo is a hobby servo whose encoder is disabled , which provides a continuous rotation of 360^{O}. This motor configuration, however, can only be used for speed control, since its angular positioning control has been disabled. One advantage of the servo is that it offers high torque at high speed. This would lend itself well to the goal of achieving accurate tuning at high speeds. Additional advantages are that this motor operates with 80-90% power efficiency, and it is impervious to environmental impact such as vibration.

However, there are several drawbacks that should be considered. Due to having more components, servo motors require frequent maintenance and calibration to function correctly. An additional problem a servo can cause is their noise, also known as chattering. This noise, which occurs when the servo holds position or moves, can cause distraction for users and observers of the instrument. The last disadvantage of the servo in the scope of this project is its angle control restriction. The hobby servo has a maximum angle ranges from 180^O to 270^o. This would restrict the project since it only offers a limited range of angle rotation, while the tuning process may occasionally require a greater range of rotational motion. The continuous rotation servo would meet this requirement, as it can turn more than 360^o. However, since this type of servo has its encoder disabled, it is difficult for this servo to control and remember its angular positioning for later use.

Figure 32: High Power Servo Example [38]

4.2.4.4) Stepper Motor

Another variety of motor considered for this application is the stepper motor. This is a variation on a DC motor that moves in discrete steps by using multiple coils, called phases. By energizing the phases in sequence, the motor is pulled to rotate one step at a time. Controlling the steps of a stepper motor could be implemented in three ways: full steps, half steps, and microsteps. There are typically four full steps in the stepper motor, whose step counts are located on either 0°, 90°, 180°, and 270° (one-phase on stepping), or 45°, 135°, 225°, and 315° (two-phases on stepping). Stepper motors can also offer half steps every 45° turn, creating 8 half steps in one revolution [64]. Additionally, these motors can also create microstepping, which range from 16 up to even thousands of points per revolution, to increase motor resolution as well as decrease motor resonance at lower speed [65]. This high resolution can be achieved through a PWM voltage signal to modulate the current to the motor windings. A stepper motor has 3 main components:

the stator, the rotor, and the coils. By varying the mechanism and structure of these three components, various types of stepper motors are created. There are three main types of stepper motors: permanent magnet (PM) motor, variable reluctance (VR) motor, and hybrid synchronous motor. The PM motor is controlled based on the polarity of the permanent magnets inside the motor. The VR motor is controlled by using the minimal reluctance principle. In other words, this type of motor moves by having the iron rotor rotate to the nearest stator magnet pole. Finally, the hybrid synchronous motor utilizes both features of the PM motor and the VR motor to optimize the power in its small package.

One advantage of utilizing a stepper motor is its ability to move precisely by moving in replicable steps. This is advantageous for this project as it can store positioning data for later use. Additionally, this type of motor possesses precise speed control, which could both ensure the efficiency of the tuning process, and reduce the possibility of accidentally snapping a string by rotating too quickly. Furthermore, stepper motors can spin continuously, which would be beneficial since adjusting the string to the desired frequency may require the tuning peg to be rotated multiple times if the instrument is drastically out of tune.

However, this component suffers from drawbacks as well. In addition, it offers less torque at high speeds, causing the device's motion to be less precise when running at high speed. Unlike servo motors, a stepper motor does not offer positional feedback, and is prone to resonances caused by overshooting its destination when moving big steps such as whole steps or half steps. Moreover, stepper motors are susceptible to losing steps, which introduces cumulative step counting error. Such a phenomenon occurs due to causes such as an insufficient power supply, an excessively large load inertia, or the speed difference between the rotor and the stator [66].

Figure 33: Stepper Motor Example [67]

4.2.4.5) Motor Comparison

Please see a comparison between the benefits and detriments of the four motors below.

Table 5: Motor Comparison Table

Pros	Cons
DC Motor (brushless)	
● 85-95% efficient ● Little to no maintenance needed	● Electrically and mechanically noisy
Electric Linear Actuator	
● Minimal noise level ● Little to no maintenance needed ● Includes potentiometer or optical sensor for position control	● Low speed
Stepper Motor	
● Repeating steps possible, good for positioning ● Precise speed control ● Can spin continuously. could spin tuning peg more than one spin	● Consumes the most power when doing no work ● Low accuracy at high speed ● No positional feedback ● Prone to resonance issue
Servoelectric Motor	
● More accuracy high speed ● 80-90% power efficiency ● Regular hobby servo: have positional feedback, can save angular position into the system ● Continuous rotation servo: unrestricted rotation, can spin the tuning peg more than one rotation ● Does not suffer from vibration	● Require frequent maintenance and calibration ● Regular hobby servo: angular range 180º-270º, restricts spin for adjusting the tuning peg ● Servo noise

From the pros and cons comparison graph above as well as the torque requirement analysis at the beginning of the motor selection, the type of motor considered to be chosen is the DC motor due to its efficiency, its low maintenance, as well as its economical budget.

Testing occurred in order to determine which motor would be appropriate for final implementation Experimentation of the 172:1 DC gear motor determined that it takes an average of half a revolution to tune up a semitone with middle range notes. Therefore, the motor must be capable of operating at least 300RPM in order to achieve the design requirement for tuning speed of 100ms using Equation 11 below. Furthermore, please see Section 5) Testing Methods for further explanation of how these values were derived.

$$RPM = \frac{r}{t} \times 60 = 300RPM \tag{11}$$

Where r = number of motor revolutions in t second = [0.5 revolution]

t = time taken to cover r revolutions = [100ms]

From required torque range calculation in Equation 9 and Equation 10, along with the 300 RPM speed requirement, the procured 19:1 12V DC motor is capable of operating at 530RPM with no load, and can produce 120 oz-in of stall torque. This motor can produce a torque of 52.77oz-in (38kg-mm) at a speed of 300RPM, which is displayed in Figure 34. This torque at 300RPM fulfills the aforementioned motor torque requirement in Section 4.2.4). More information on torque-speed performance can be consulted in Figure 34 below. The 19:1 motor was procured so that the maximum torque of the string would be unreachable, which would prevent the motor from breaking the strings by applying too much force.

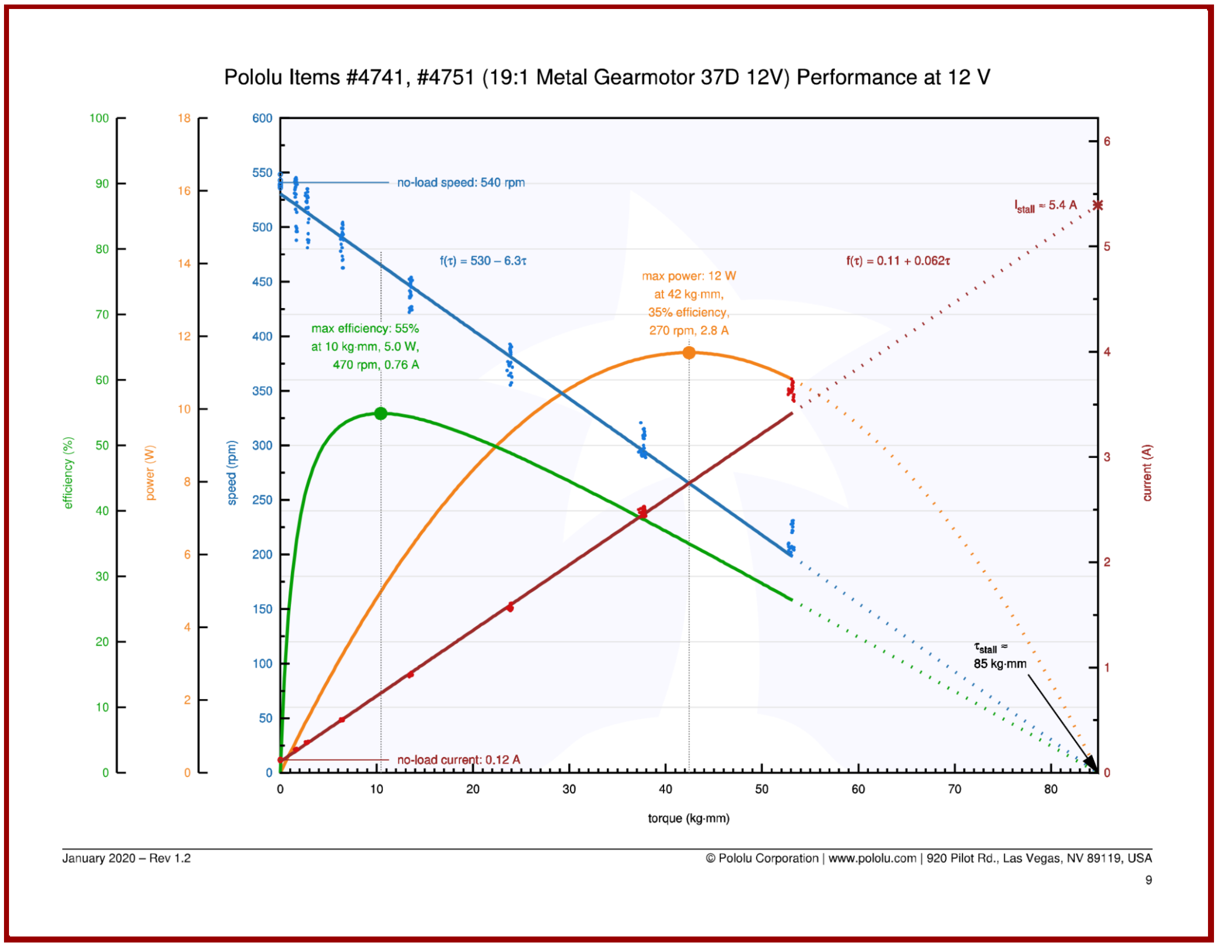


Figure 34: Torque-speed performance of the procured motor [80]

4.2.5) Motor Control

The motor is controlled through the motor driver L298N and the encoder, which is axially integrated into the DC motor. This driver is chosen because of its ability to handle high voltage up to 46V and high current up to 2A, more than enough to support the procured motor, whose operating power is 12V and 0.76A [83]. This driver is also chosen due to its widely available documents and implementations with DC motors. To control the positioning of the motor, the project utilizes the PID algorithm to ensure its accuracy in tuning the tuning machine. The encoder is to record the position of the motor shaft, and has its data sent to the microprocessor, which compares the data to the targeted setpoint and calculates the needed power. This power signal is sent to the motor driver L298N through the IN pins, which is sent to the motor through the OUT terminals on the driver module. Please see Section 6.1.4) Custom PID Development for further discussion of the motor control techniques utilized.

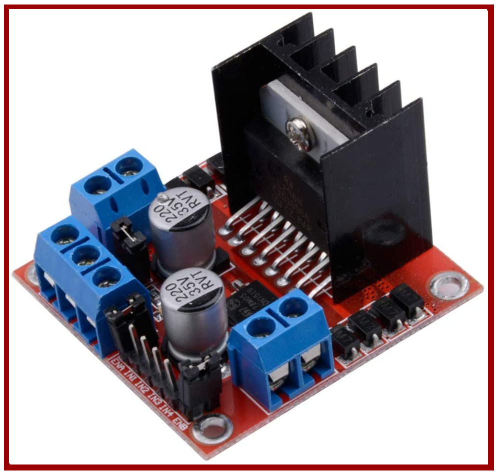


Figure 35: L298N Motor Controller Module [69]

4.2.6) Microcontroller

The primary factor that influenced the selection of the microcontroller utilized was the input/output (I/O) communication requirements of the project, given the large number of sensors and motors that need to be driven. As can be seen in the electrical architecture in Figure 16, the sensor block, motors, and solenoid actuation for each string require 5 digital pins and 2 analog pins. Please see Figure 36 below for an image of how the pins will be connected. Another component of the selection process was that the microcontroller needed to have a large amount of memory that would not be erased when the system shut down, as it needed to be able to store information regarding the position of the motors in order for calibration to be achieved successfully.

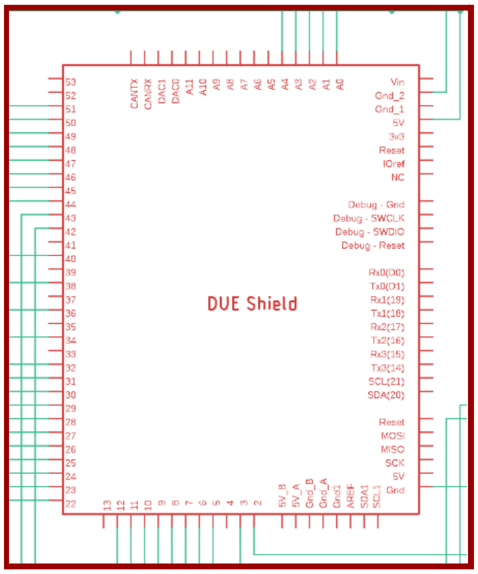


Figure 36: Arduino Due Microcontroller Pin Map

An Arduino Due was utilized as the microcontroller for this system. The board has 54 digital pins and 12 analog input pins. It was determined that this project would incorporate five strings, as the group sought to avoid utilizing every analog pin the Arduino Due has in order to plan for the event that a string system would need to utilize one of the two remaining pins, and it would be disadvantageous to not be able to do so. Another factor that led to the selection of this microcontroller is its memory capabilities., and with 512KB of flash memory and 96KB of SRAM, the board will be capable of holding the code necessary for the system along with this calibration data.

The board does not have the memory capacity to store all of the system's software, nor would it have been capable of performing fundamental frequency analysis. Due to these factors, a host computer was utilized to assist with these tasks. Furthermore, a more advanced user interface could be constructed on a host computer rather than constructing an external module that the Due would not have had the memory to store regardless.

4.2.7) Power Consumption

In order to determine which power system to purchase, the power consumption of the systems attached to each string - solenoids, sensors, and motors - was calculated utilizing the table below. The maximum power consumption possible for each device was used, so that one could be assured the device would function at worst-case power draw. The values in the table were either determined via experimentation or located on the data sheets of the products used. The power consumption for these systems was found to be 61.68W. The power consumption of the Arduino Due is found to be 1.2W at 12V [71], making the total power consumption to be 62.88W. Therefore, a 12 Volt, 8 Ampere, 96 Watt power adapter was acquired to supply power to the system.

Table 6: Power Consumption of String Systems

Item	Voltage (V)	Current(A)	Units	Total Power Consumption (W)
Solenoids	12	0.25	5	15
Motors	12	0.76	5	45.6
Lasers	3.3	0.0015	5	0.02475
Photoresistor	5	0.0005	5	0.0125

Load Cell	5	0.0044	5	0.11
Load Cell Amplifier	5	0.0015	5	0.0375
L298N	5	0.036	5	0.9
Arduino DUE	12	0.1	1	1.2
			Total Power Consumption	62.88

4.3) System Operation Process Design

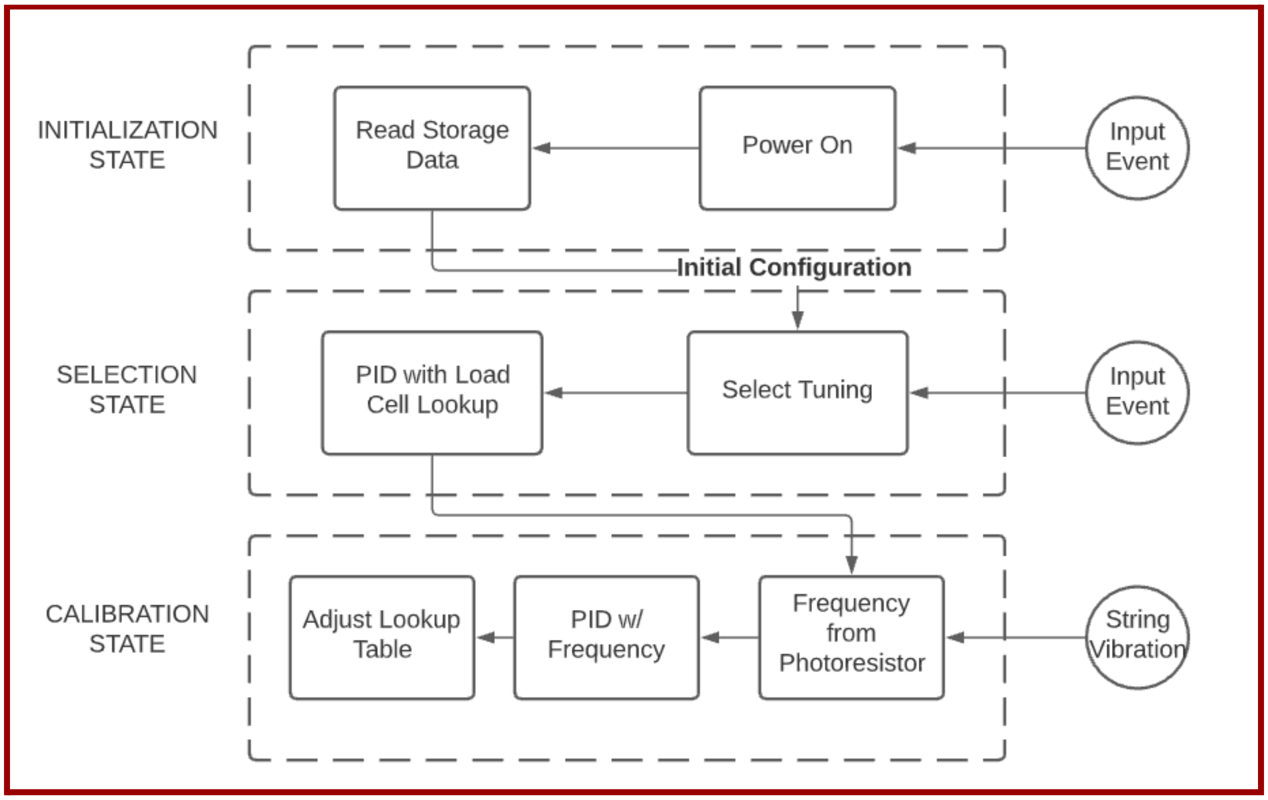


Figure 37: Operation Process

The system is state and event based. The system begins in an initialization state when triggered by an input event such as an 'on' button press. The motor encoder value, as well as the tension and encoder mappings, are read from storage through communication using ASCII commands. The selection state is entered whenever the user interacts with the interface to change string tuning. Using the load cell value as a source of error, the device then tunes individual strings to this set frequency with a PID-type controller. Since the load cell can be polled at a rate no greater than 10 Hz, this is not responsive enough to be used with PID control that updates at a rate of up to 200 Hz (this was expected and then determined in practice). Therefore, an encoder value is expected for a specific tension value, which the device tunes to using PID. If at this encoder value the tension value is not within the threshold, the encoder value once again adjusts. The device then transitions to a calibration state. In this state, whenever a string is played either by the user or solenoid, the photoresistor determines the actual string frequency and uses this to change the strain gauge and frequency mapping table if necessary. A signal is then sent to trigger a change in tension using the motor and PID controller.

4.3.1) Initialization of the Device

The device is initiated with a 'start' command sent by a Node.js application the device is communicating with. Storage is read by sending an ASCII command and an index mapping to the application, and a value is then returned. If no value is returned, the device creates a map and sets all encoders to zero according to its internally stored constants.

4.3.2) Load Cell Tuning Process

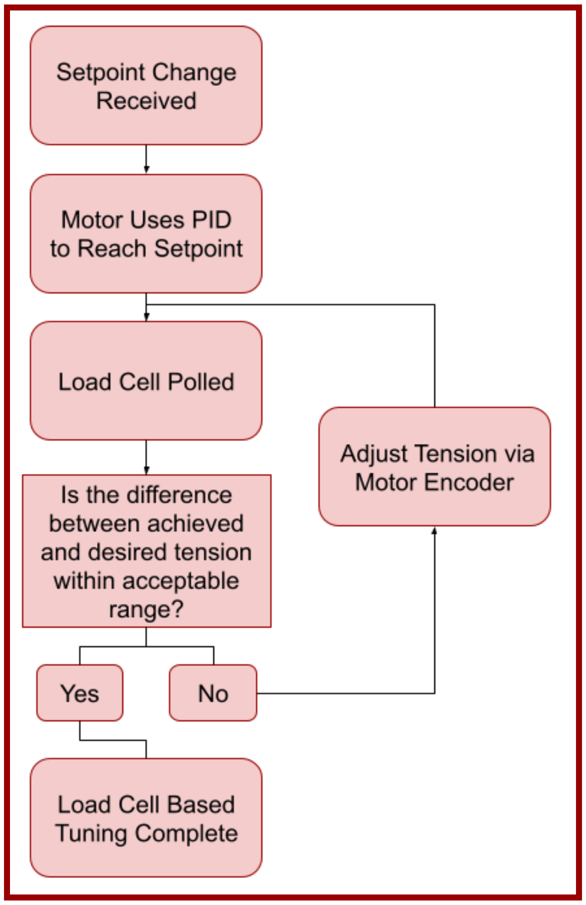

Setpoint Change Received
Motor Uses PID to Reach Setpoint
Load Cell Polled
Adjust Tension via Motor Encoder
Is the difference between achieved and desired tension within acceptable range?
Yes
No
Load Cell Based Tuning Complete

Figure 38: Load Cell Tuning Process Flowchart

To tune the device, a string module object is sent to a setpoint, which is the lowest pitch within the determined string range. This setpoint is received when an ASCII command is received. If the setpoint changes, the algorithm determines the current load cell value and the encoder value derived from it. Then it resets a number of runtime variables. During the control loop, the motor uses PID to reach the setpoint, if the difference between expected and actual encoder value is below a threshold, it polls the load cell which is checked to be within the accepted threshold of the desired tension value. If so, it increments successes and counts a certain number of successes occur in a row, it returns true and shuts down the motors. This number of successes is determined by understanding the potential sacrifices made for speed in favor of accuracy - if the number of successes is reduced the tuning operation time is decreased, but if the number is increased the accuracy increased. For the purposes of the prototype, two successes were required. If the value received from the load cell is not within error, it adjusts the desired encoder value by a factor proportionate to the difference between expected and actual tension.

4.3.3) Photoresistor Tuning Process

The photoresistor captures samples at a rate of 15000 samples per second, and is loaded into a two dimensional array. This rate was chosen first by taking in consideration Nyquist's theorem, that operating frequency can only be accurately determined if samples are taken at the rate of at least twice this frequency. The highest frequency expected is 392 Hz, so at least a 784 Hz sampling rate is required. In practice, higher rates are often used. The 15 kHz sampling was chosen as higher sampling rates are preferable, and it is sufficiently greater than the theoretical minimum. Once 512 samples are acquired, the device determines the frequency of all strings using FFT with a Blackman-Harris windowing function. Additional parameters are built into the library itself but the library is modifiable. These built-in parameters dictate that the window size is 512, the same as the sampling size, and the maximum number of bins is the same count as the given samples. The FFT calculation is being made on the microcontroller using Enrique Condes' FFT library [77]. The algorithm first checks that the data being received from the photoresistor is within some percentage of the expected frequency that filters out unrealistic data. This percentage is arbitrarily set at 8%, as it was determined empirically that most predictions are either very far from expected or within some realistic margin that can't be filtered out without potentially losing valid data. Then the algorithm checks if the load cell is within the permitted margin of error - meaning that the system is properly tuned. If so, the tension value in the map is changed by first determining the difference between FFT-measured frequency and desired frequency. This difference is averaged over three samples, it is then multiplied by a weight and added to the tension map. This weight was chosen by experimentally determining a value that both makes the map resilient to constant change, and able to stabilize on a correct value.

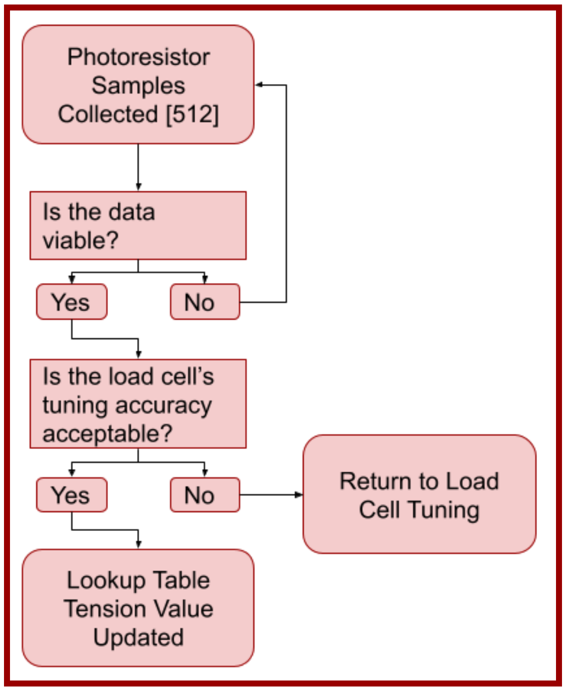

Photoresistor Samples Collected [512]
Is the data viable?
Yes
No
Is the load cell's tuning accuracy acceptable?
Yes
No
Return to Load Cell Tuning
Lookup Table Tension Value Updated

Figure 39: Photoresistor Tuning Process Flowchart

4.3.4) Custom PID Development

The device uses a custom-made PID - the use of a library was initially attempted, but it was based on using a constant sample interval, which is not realistic when the number of operations changes depending on how many strings are being simultaneously tuned. The I and D term contribute power proportionate to the time elapsed between iterations of the control loop. The I term is unused until the error is below a certain error threshold in order to make the operation more stable (Appendix F Example 4).

4.4) User Interface Design

In order to meet the project goals for input capabilities, a user interface was constructed. The requirements of the user interface were to allow the user to tune individual strings to individual notes and tune the instrument to a chord. The final user interface for the project has two different modes that meet these requirements. Custom mode allows the user to choose individual notes for individual strings and the chords mode allows the user to select a chord type and a chord root. The screen for custom mode is shown in Figure 40 and the screen for chords mode is shown in Figure 41 below. There are also diagrams that label the elements of the user interface that the user can interact with in Figures 42 and 43. In custom mode, the user can select notes corresponding to the notes listed on the slider for each string. The note ranges for each of the strings were determined through string testing detailed in Section 7.1.1. Once the user has chosen the notes for each string, the select tuning button can be pressed to have the instrument adjust to that tuning. For the chords mode, once the user has selected the chord type and chord root the select tuning button can be pressed to have the instrument adjust to that chord. The solenoids can be actuated from the user interface through keypresses. Pressing keys 1 through 5 will actuate the solenoid corresponding to the string number of the key and pressing the key 6 will actuate the solenoids for all of the strings at once. The user interface was developed using HTML, CSS, and JavaScript. The full code for the user interface can be found in the zip file included with the project. The user interface sends the notes to the Arduino Due through serial communication with a web socket and the Node.js runtime environment was used to handle the communication.

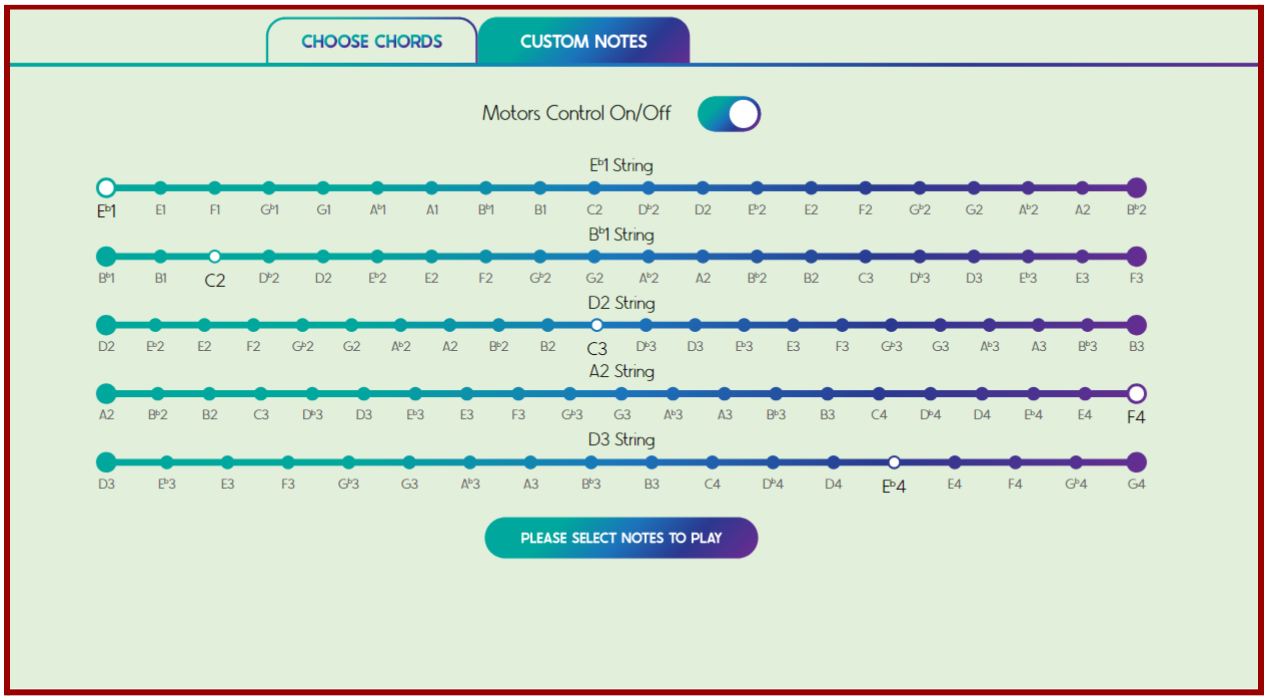

CHOOSE CHORDS
CUSTOM NOTES
Motors Control On/Off
E♭1 String
E♭1 E1 F1 G♭1 G1 A♭1 A1 B♭1 B1 C2 D♭2 D2 E♭2 E2 F2 G♭2 G2 A♭2 A2 B♭2
B♭1 String
B♭1 B1 C2 D♭2 D2 E♭2 E2 F2 G♭2 G2 A♭2 A2 B♭2 B2 C3 D♭3 D3 E♭3 E3 F3
D2 String
D2 E♭2 E2 F2 G♭2 G2 A♭2 A2 B♭2 B2 C3 D♭3 D3 E♭3 E3 F3 G♭3 G3 A♭3 A3 B♭3 B3
A2 String
A2 B♭2 B2 C3 D♭3 D3 E♭3 E3 F3 G♭3 G3 A♭3 A3 B♭3 B3 C4 D♭4 D4 E♭4 E4 F4
D3 String
D3 E♭3 E3 F3 G♭3 G3 A♭3 A3 B♭3 B3 C4 D♭4 D4 E♭4 E4 F4 G♭4 G4
PLEASE SELECT NOTES TO PLAY

Figure 40: User Interface Custom Mode Screen

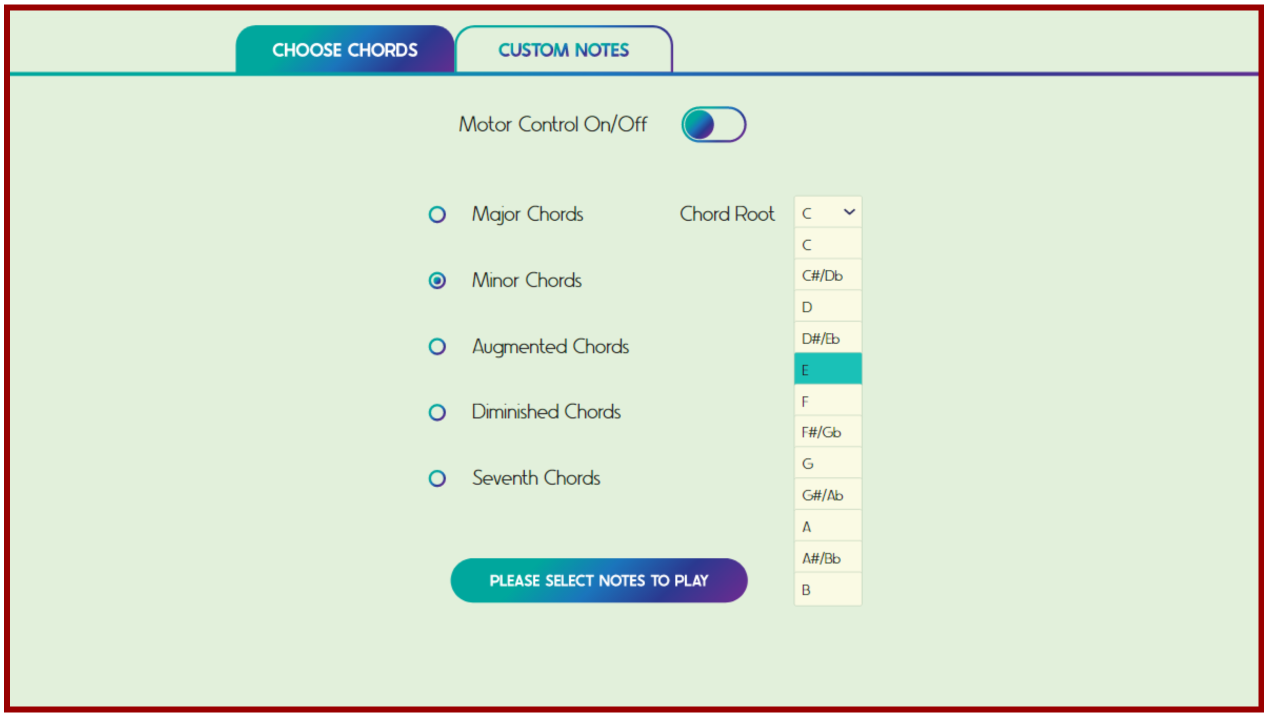


Figure 41: User Interface Chord Mode Screen

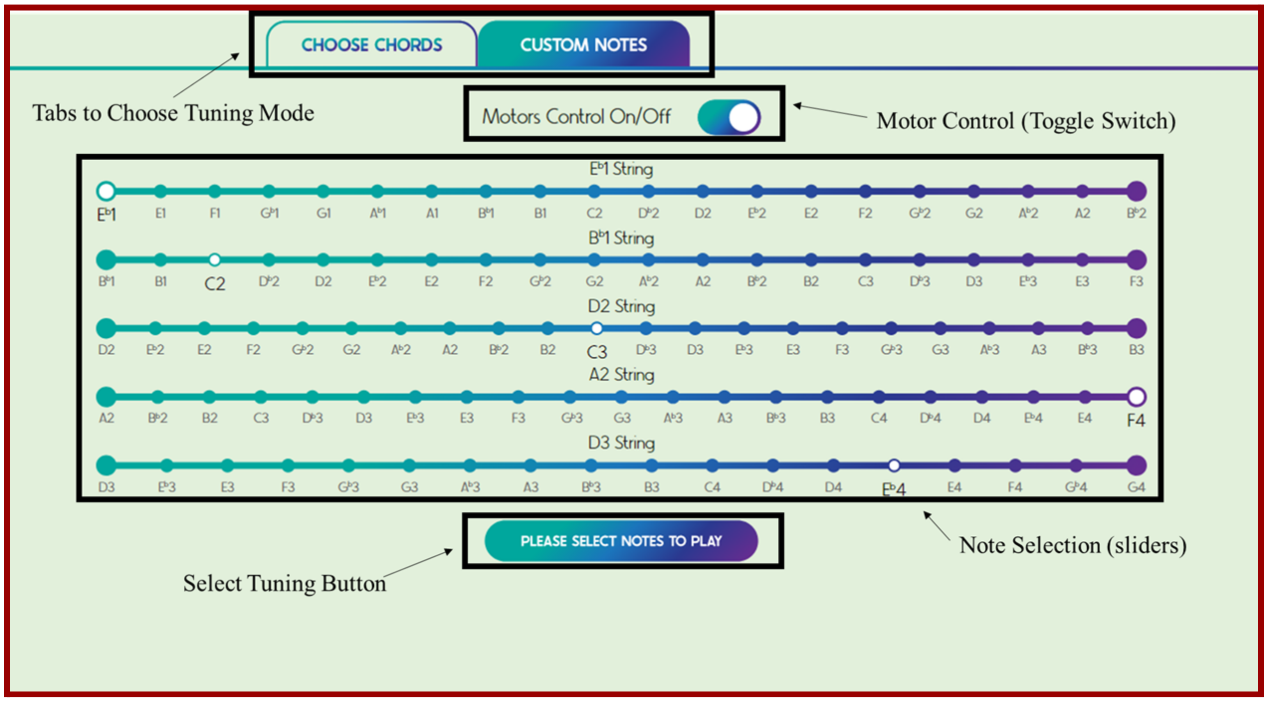


Figure 42: User Interface Custom Mode Screen (Elements Labeled)

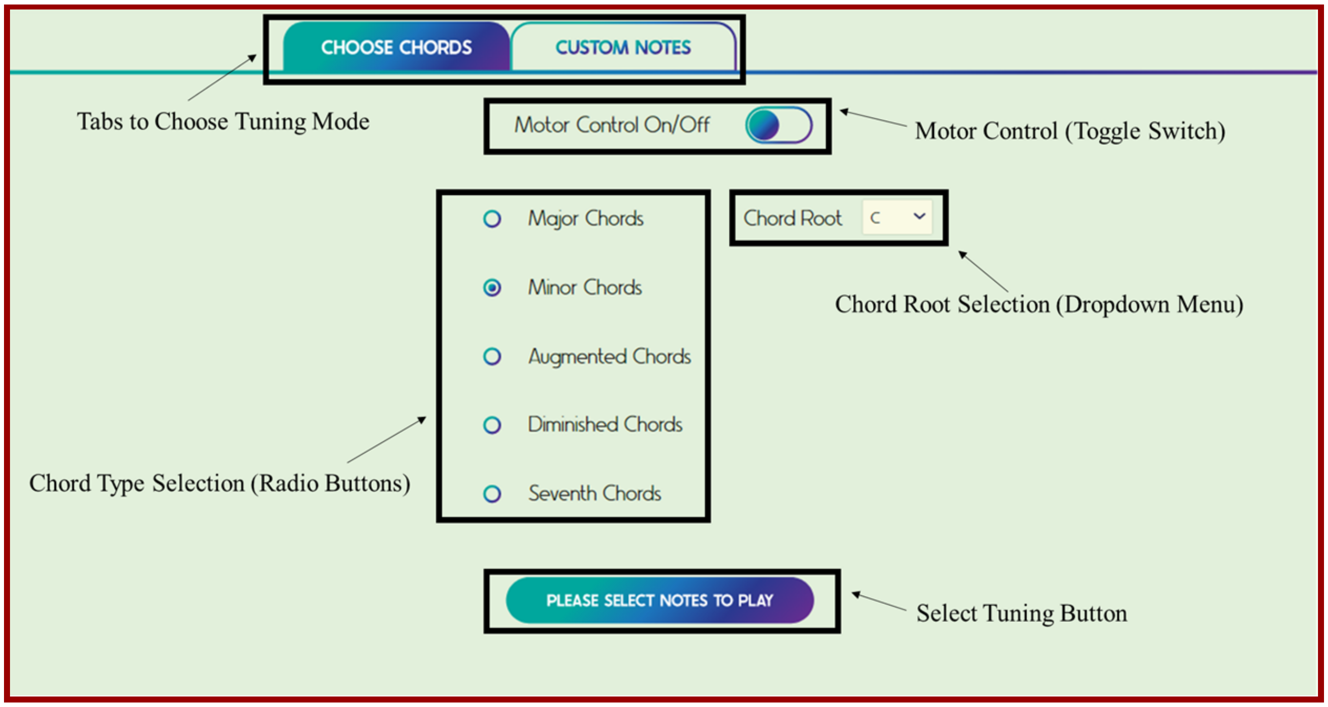

CHOOSE CHORDS
CUSTOM NOTES
Tabs to Choose Tuning Mode
Motor Control On/Off
Motor Control (Toggle Switch)
Major Chords
Minor Chords
Augmented Chords
Diminished Chords
Seventh Chords
Chord Root
C
Chord Root Selection (Dropdown Menu)
Chord Type Selection (Radio Buttons)
PLEASE SELECT NOTES TO PLAY
Select Tuning Button

Figure 43: User Interface Chord Mode Screen (Elements Labeled)

4.5) Instrument Body Design

Below is an initial CAD model of the autonomous string instruments. Given that the wooden board that comprises the instrument body is 3 inches wide, and is supported by two side planks each measuring 0.75 inches wide, the instrument's total width is 4.5 inches. At a total length of 35.5 inches, the strings stretch over 25.5 inches of the instrument's total length, with the motor module taking up the remaining 10 inches.

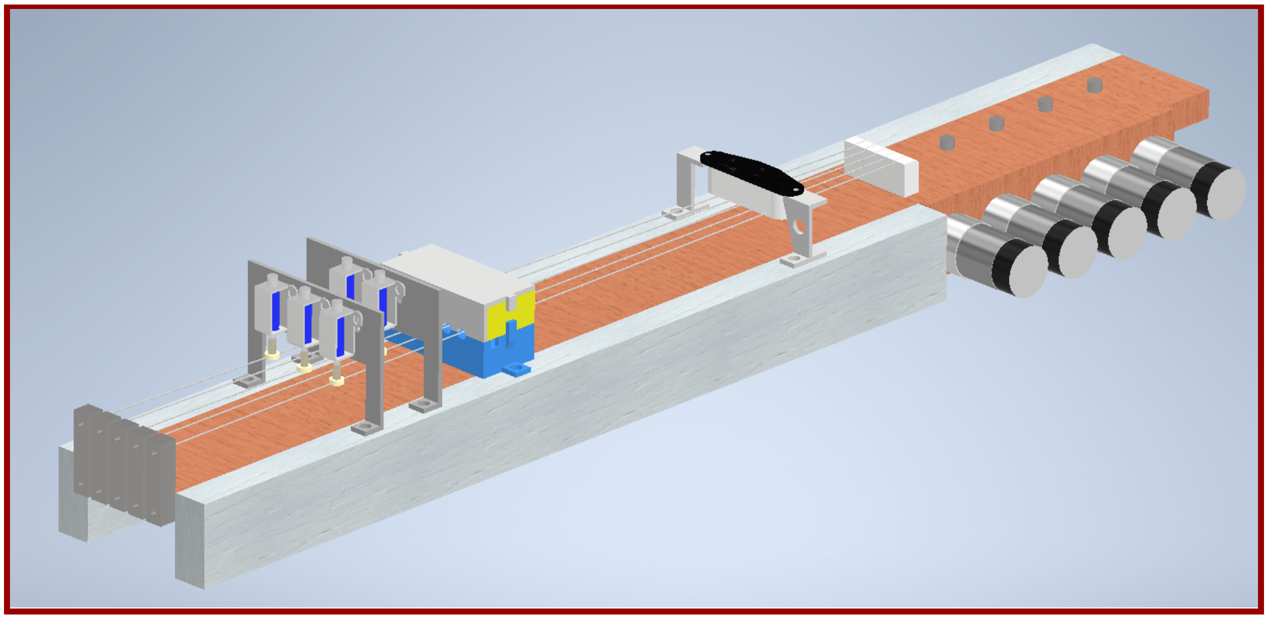


Figure 44: CAD File of the Instrument Body

The sensors determined for use in the previous section were then integrated onto the instrument. The load cell takes the tension measurements of the string, and is mounted at the instrument's base. In order to leave enough room for each load cell to function uninhibited, the center of each sensor was placed 0.625 inches apart.

The photoresistor and laser module, which is used to determine the frequency of the string, is enclosed in housing that blocks as much ambient light as possible so as to prevent its interference with the functionality of the instrument. Furthermore, the laser beam must be focused and aimed properly - if not, the beam of light may be too wide and bleed over the sides of the string, causing the photoresistor to detect movement when none is occurring. This is avoided by the construction of the photoresistor case; each laser is held in place, and its light is focused via a 3D printed component placed in between the lasers and the string. At the point where the module is placed, the strings are 0.5 inches apart.

The strings then run underneath a guitar pickup. This module's sensor is an electromagnetic pickup, whose functionality can be found in Section 5.1.3) Electromagnetic Coils. The strings are then held in place by a 3D printed part at the neck of the instrument, referred to as a nut. At this point, the strings are 0.34 inches apart from each other, and pass towards the motor module.

Each string is tuned via a tuning machine attached to a motor. This attachment is facilitated via a coupler. One end is constructed to hold onto the knob of the tuning machine, which has a diameter of 4 millimeters, and the other to hold the 6 millimeters in diameter motor shaft. Please see Appendix C: Instrument Concept & Final Design Images for further photos of the instrument, and comparisons of the CAD files of parts and their physical counterparts.

With all of the components mounted, electrical connections are established as outlined in Section 4) Overall System Design. To make the process more convenient, the components are divided into separate electrical modules. Please see Figure 16 for a more detailed electrical schematic.

4.6) Electrical Architecture Design

In order for the instrument to be capable of playing itself, solenoids will be affixed above each string. These devices operate by energizing a coil of copper wire, inciting movement from a metal armature that is pushed and pulled with the electromagnetic field generated by the wire. For the purposes of this project, the device will use its armature to strike the strings, either to autonomously play the instrument or to initiate the photoresistor's tuning capabilities. The Adafruit Small Push-Pull Solenoid will be implemented on the final prototype. Below is a schematic of how the solenoid can be controlled using digital I/O from a microcontroller.

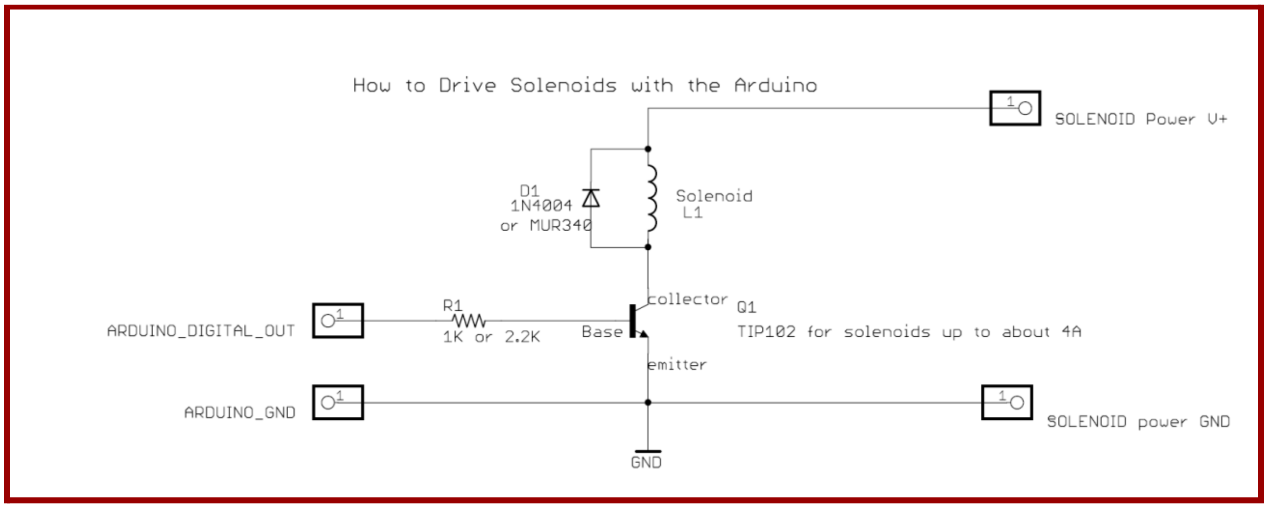


Figure 45: Solenoid Control [70]

The photoresistor module consists of 5 parallel voltage dividers, which are photoresistors serially connected to the 4.3kOhm resistors. These voltage divider outputs are connected to analog inputs A0-A4. The laser diode module consists of 5 parallel laser LEDs. These LEDs do not provide any input or output values, and consequently are not connected to any I/O ports on the DUE. Each of the photoresistors and LEDs are spaced 0.5 inches apart from each other to align with the string's spacing.

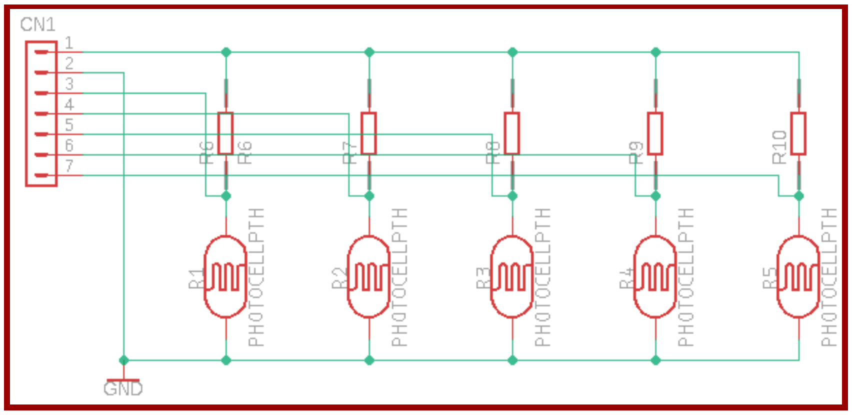


Figure 46: Photoresistor Module Schematics

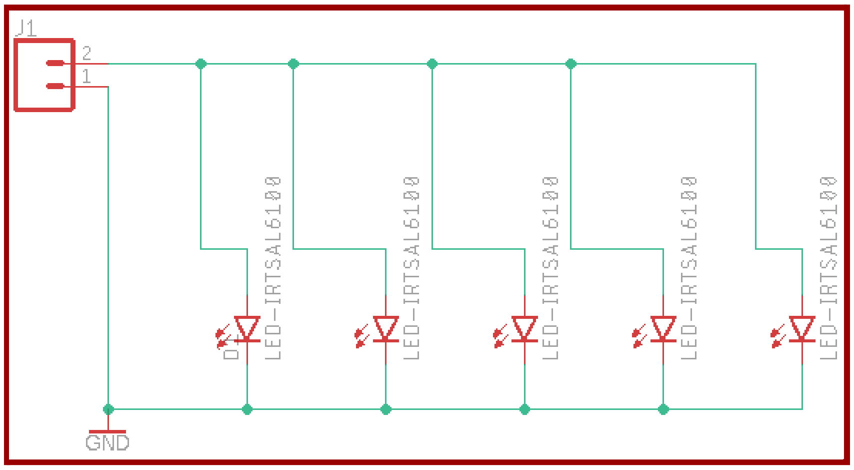


Figure 47: Laser LED Module Schematics

The load cells are connected to each individual load cell amplifier, with the red, black, green, and white wires soldered into the RED, BLK GRN,WHT wires of the sensor, respectively. The amplifiers' DAT and CLK pins are wired to DUE digital pins 22-31 in pairs.

The solenoid module consists of 5 solenoids (L1 to L5) wired in parallel to each 1N4004 diode. Every solenoid has one end wired to the power source, while the other end is wired to the TIP102 transistor's Collector. These transistors each have their Emitter grounded and their Base connected to a 1kOhm resistor before being wired to pins 32, 34, 26, 38, 40 on the DUE.

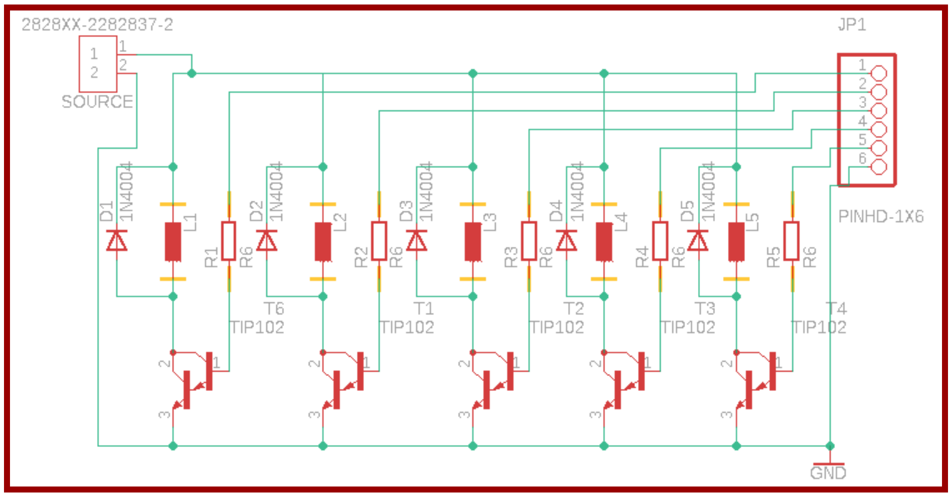


Figure 50: Solenoid Controller Module

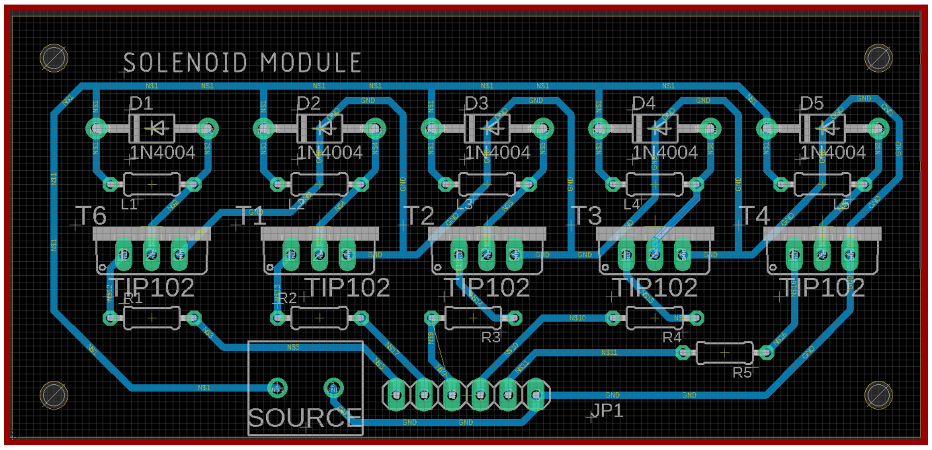


Figure 51: Solenoid Control Module PCB Model

The DC motors' red and black wires are connected to the OUT terminals on the L298N motor controller pair by pair. With this chosen motor controller's capability of operating 2 motors in parallel, 3 controllers must be utilized for 5 motors. Motor 1 and 2 are operated by the first L298N controller via ports OUT1, OUT2, and OUT3, OUT4. Motor 3 and 4 are operated by the second controller through ports OUT1, OUT2, and OUT3, OUT4. Motor 5 is operated through ports OUT1 and OUT2 on the third controller. The IN1, IN2, IN3, IN4 ports from the first 2 controllers, and IN1 and IN2 on the last controller are bussed to PWM pins 2, 3, and 5-12 on the DUE. The function of these pins is to send a PWM signal from the Arduino to the L298N module to control the DC motor's direction. Lastly, each motor's encoder 1 and 2, which are the motor's white and yellow wires, are connected in pairs to the DUE's pins 42-51, in order to output the position of the encoder to the microcontroller.

To power the system, the Arduino DUE, motor controller, and solenoid controller modules are wired to a PCB power shield, which is connected to the 12V-96W power adapter.

The laser diode and photoresistor modules derive their power from the 3.3V pin on the Arduino board, while the load cell amplifiers' VCC and VDD pins take up the Arduino's 5V pin. Additionally, the motor controllers and the DC motor's encoder also use the 5V pin to power their logic circuits. All of these modules share a common ground pin GND on the Arduino.

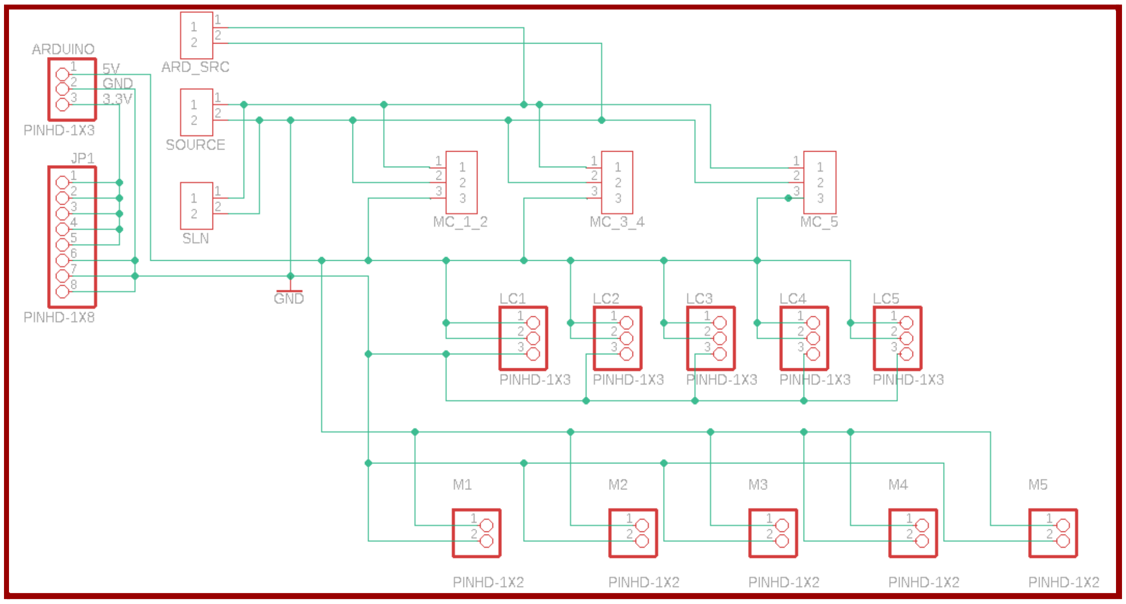


Figure 52: Power Shield Schematics

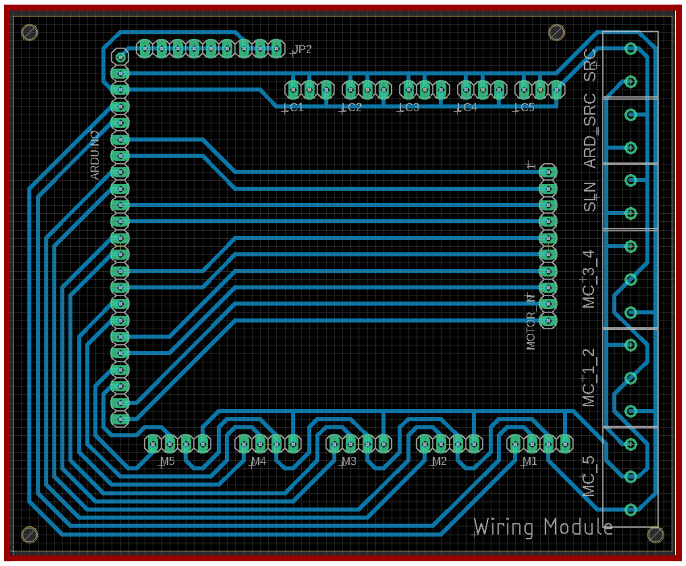


Figure 53: Power Connection Module

5) System Construction & Evaluation

This section displays both the results of the construction of the system, and the results of testing performed in order to evaluate the effectiveness of the instrument with respect to design goals.

5.1) System Construction Results

5.1.1) Physical System Results

Below are images of the final constructed instrument body. The dimensions match those of the CAD models described above. It is 4.5 inches wide by 35.5 inches, with the strings vibrating across 25.5 inches of the body. The body of the instrument is held together by wood screws.

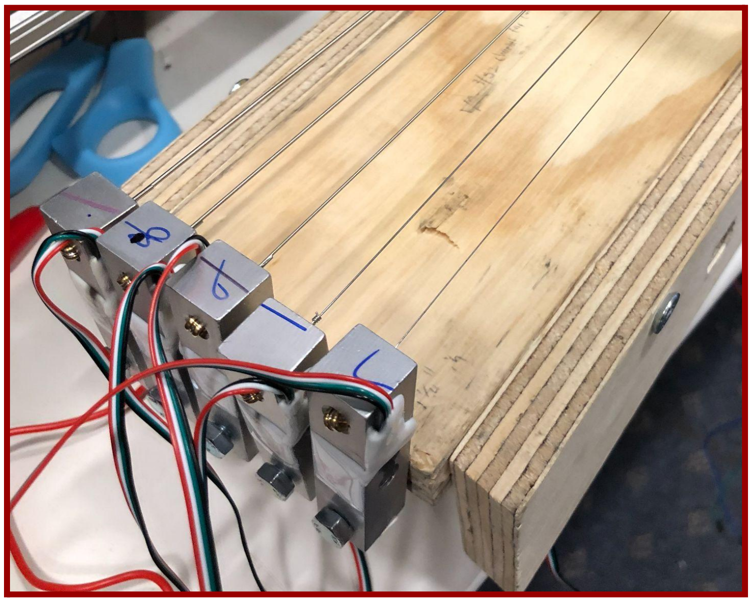


Figure 54: Load Cells with Attached Strings

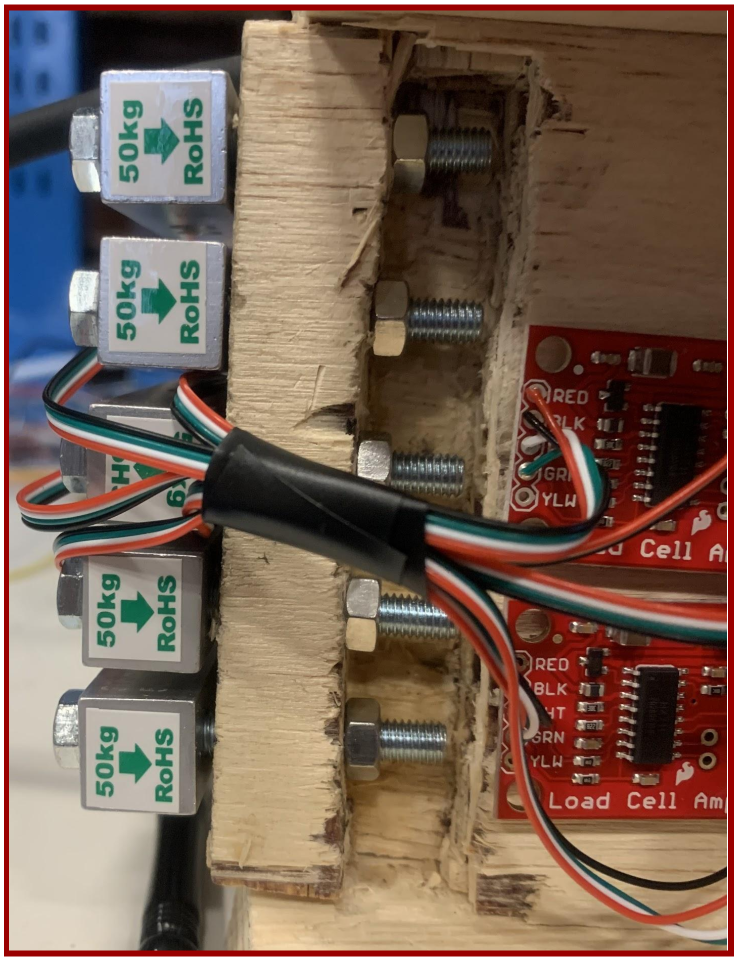


Figure 55: Channel for Securing the Load Cells

The load cells shown in the figure above were attached by creating a channel under the body of the instrument such that the nuts could be attached to secure the bolts. A chisel was used to create the channel. Each string is held in place by the ball at the end of the string, which prevents them from passing through the hole in the load cell.

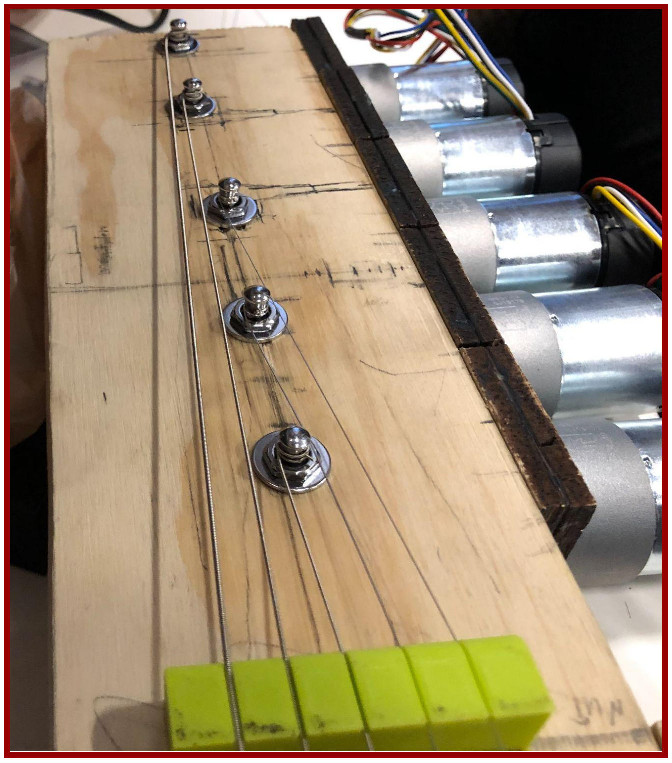


Figure 56: Tuning Machines and Motors

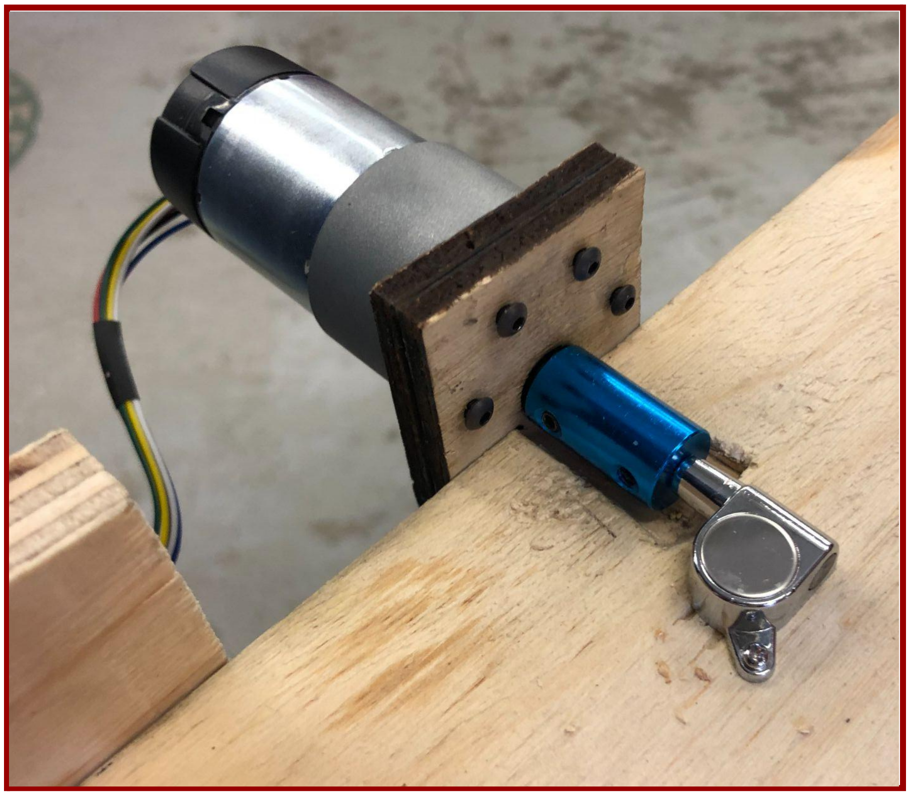


Figure 57: One Mounted Motor Attached to Tuning Machine

Figure 57 shows how the motor and tuning machines were secured. For the motor, laser cut motor mounts were created, which were secured to the instrument with wood screws. A 1 inch coupler was then used to connect the motor to the tuning machine. In order for the module to turn without touching the instrument, a channel was created using a chisel. The other end of the tuning machines with strings attached can be seen above in Figure 56.

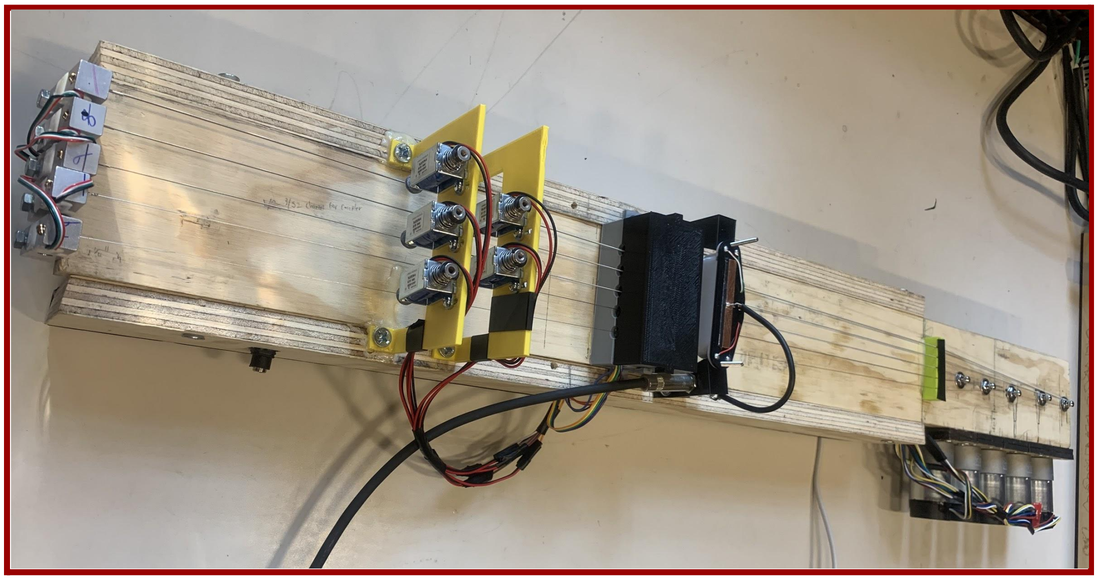


Figure 58: Constructed Instrument

Figure 58 shows the completed instrument. Wood screws were used to secure the yellow solenoid frames, as well as the black pickup frames. There was some concern over the integrity of the side plates degrading by adding more screws, so the photoresistor case was secured with glue instead.

5.1.2.) Electrical System Results

The electrical circuitry was set up on the underside of the instrument, as can be seen in Figure 59. Electrical tape was used to secure jumper connect points, and was used to assist in cable management. A hole was drilled in the instrument to allow for the power jack to connect with the adapter, as shown in Figure 59. All of the PCBs were taped to the wood with a double-sided polymer tape, with the exception of the Arduino DUE and the motor controller modules, which were attached on a mount drilled into the instrument body.

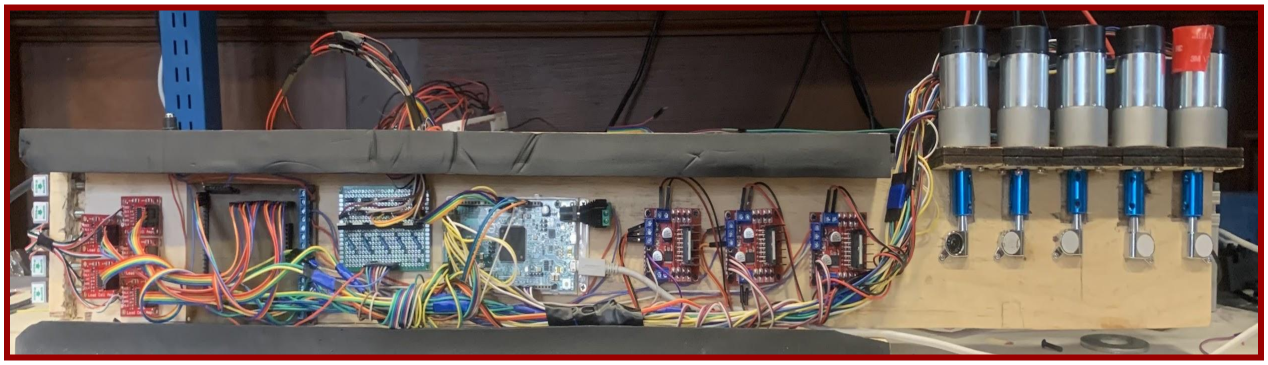


Figure 59: Finished Instrument Electrical Connection

5.1.3) Software Design Results

To create an initial layout for the software, a UML diagram was created. UML is an acronym for Unified Modeling Language, and it is a language used to visualize an object-oriented design in the form of classes, functions, and data. An object-oriented design was chosen for the project to easily represent the objects of the system and their relationships. The UML diagram is shown below in Figure 60.

The different classes shown in the UML diagram below are Main, StringModule, UI, Solenoid, Photoresistor, Motor, LoadCell, LookupTable, and DueFlashStorageHandler. The Main class is responsible for linking the UI, DueFlashStorageHandler, and StringModule together. It also handles all of the system states: initialization, selection, and calibration. The StringModule class is used to create objects for each of the strings and pair a lookup table, photoresistor, solenoid, motor, and load cell with each individual string. The UI class is used to get the tuning information from the web-based user interface. The DueFlashStorageHandler class is used to store the encoder value. Originally, this class was also intended to store the data for the lookup table, but during testing it was discovered that the Arduino Due did not have enough memory available to store the lookup table. To solve the storage problem, the lookup table data was instead stored on the host computer instead of the Arduino Due. The storage of the file was achieved using the file module of Node.js. This UML diagram was followed in terms of class structure, but certain internal class methods were added, changed and removed in the final version of the software.

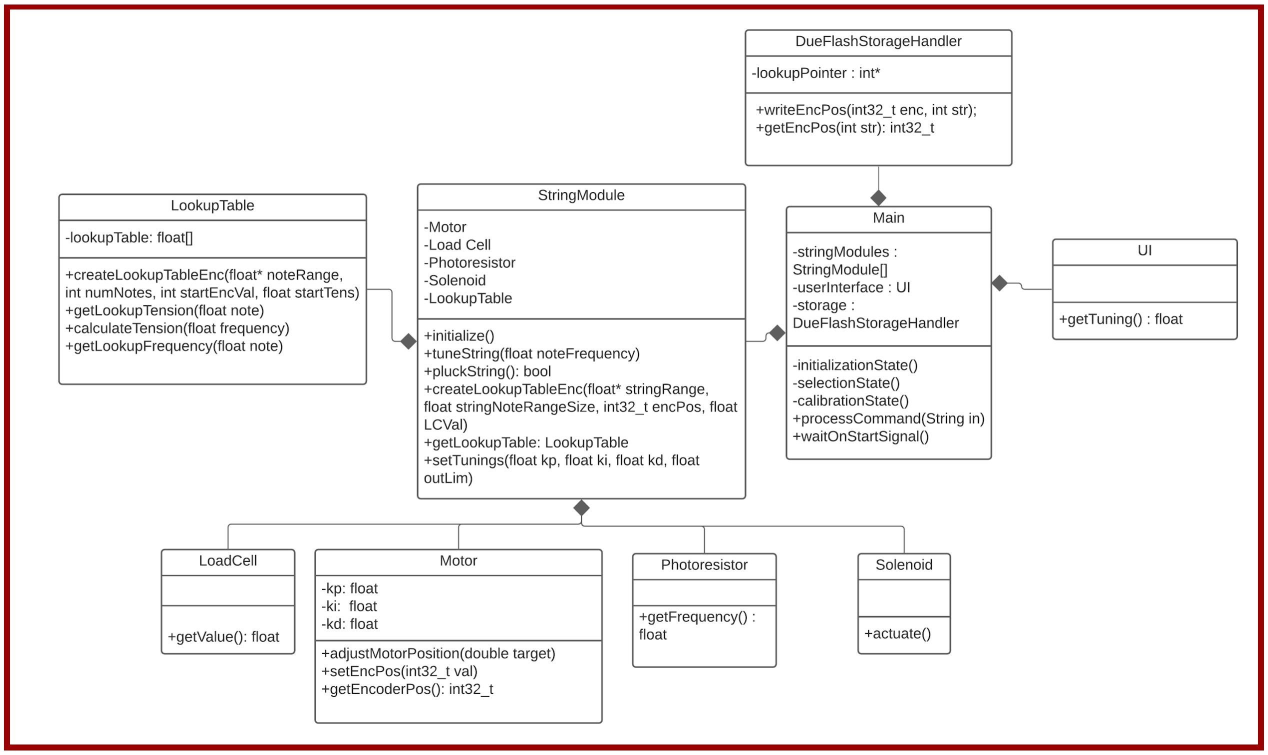
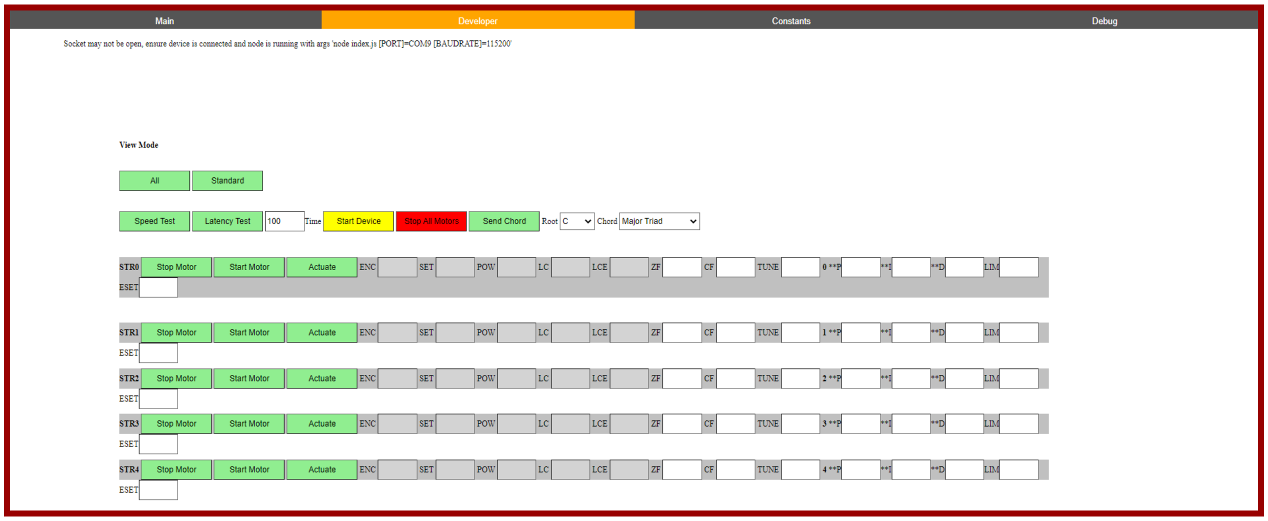

DueFlashStorageHandler
-lookupPointer : int*
+writeEncPos(int32_t enc, int str);
+getEncPos(int str): int32_t
LookupTable
-lookupTable: float[]
+createLookupTableEnc(float* noteRange, int numNotes, int startEncVal, float startTens)
+getLookupTension(float note)
+calculateTension(float frequency)
+getLookupFrequency(float note)
StringModule
-Motor
-Load Cell
-Photoresistor
-Solenoid
-LookupTable
+initialize()
+tuneString(float noteFrequency)
+pluckString(): bool
+createLookupTableEnc(float* stringRange, float stringNoteRangeSize, int32_t encPos, float LCVal)
+getLookupTable: LookupTable
+setTunings(float kp, float ki, float kd, float outLim)
Main
-stringModules : StringModule[]
-userInterface : UI
-storage : DueFlashStorageHandler
-initializationState()
-selectionState()
-calibrationState()
+processCommand(String in)
+waitOnStartSignal()
UI
+getTuning() : float
LoadCell
+getValue(): float
Motor
-kp: float
-ki: float
-kd: float
+adjustMotorPosition(double target)
+setEncPos(int32_t val)
+getEncoderPos(): int32_t
Photoresistor
+getFrequency() : float
Solenoid
+actuate()

Figure 60: Software UML Diagram

The user interface for development, shown below in Figure 61, is different from the user interface in Section 4.4 that is shown to the player. This interface provides buttons for performing certain tests, setting PID and load cell calibration parameters per string, stopping motors in case of faults, and actuating strings. Data is sent and received in ASCII format over serial communication with the instrument, and the UI updates correspondingly. All of the grey boxes are updated automatically when incoming data is received from the instrument. It was necessary to create this interface because the user-friendly interface is not verbose enough for debugging purposes - it does not print encoder values, tension values, or permit more complicated functions.



Socket may not be open, ensure device is connected and node is running with args 'node index.js [PORT]=COM9 [BAUDRATE]=115200'

100 Time

Root C

Chord Major Triad

Figure 61: Development UI

Certain issues were discovered when using the PID library available from the Arduino package manager (“PID_v1”), which includes assuming latency will be constant between updates of the PID. This is a problem because the behavior of the control system will be significantly impacted when there are periods of different latency, because the I and D terms are derived based on an assumed constant latency. The issue was resolved by creating a custom PID library that accounted for variable latencies and has certain changes that prevent the I term from accumulating too much, and the D term from inverting the output.

It was found in the development process that when using the DueFlashStorage library, the Arduino Due microcontroller had trouble storing the lookup table data in its persistent flash storage. This resulted in the flash page being locked, and prevented the uploading of new programs to the device. Using certain tools available on the board manufacturer’s website, such as SAM-BA, the lock was removed. It was decided to either use an external persistent storage device, such as SRAM or DRAM, or implement storage on the development computer. The latter was chosen because delays in acquiring and integrating an external storage chip would outweigh its speed and portability benefits when compared with the Node.js storage option.

The use of ASCII commands instead of MIDI was chosen because MIDI is more difficult to debug. This is because with MIDI, there is no plaintext communication between devices. The lack of plaintext communication would possibly require the functions to be re-interpreted or characters to be encoded in MIDI commands. The cost of using ASCII commands is that more data must be sent over a serial connection to perform the same action, but this was determined to be preferable as most complexity arises from the implementation of algorithms, which would be delayed with a protocol that is less understandable to a human reader.

Multithreading the microcontroller was found to be a difficult task; a ‘protothreading’ library was used to multithread the solenoid actuation. In practice, this implementation exhibited odd behavior and did not simultaneously actuate strings. It was then chosen to use an array of booleans as flags for determining when to switch solenoid states.

5.1.4) Challenges Uncovered During Construction

Over the course of construction, there were several unforeseen challenges that slowed the development of the prototype. The biggest problems arose with the photoresistor case, which had to be reprinted several times. Due to the edges of the body being cut at a slight slant, the side plates extended slightly above the instrument, and were not flush. This, in turn, impacted the height of the mounted modules, which had to be readjusted. Since multiple iterations of these fixtures were required, the sensors and other modules could not be connected to the instrument or tested for a significant amount of time.

Another issue that arose was the warping of the body itself. As tension of an attached string increased, the body began to slightly bend. This caused the other strings to lose tension, as well as slightly change the height of each string. In order to compensate for this, the maximum possible tension set for each string was decreased. This allowed for functionality of the prototype, although the pitch range of the instrument was decreased.

The warping of the body was an unforeseen issue. Had it been planned for, the underside electrical components would be organized such that there is space for a supporting rod. For the sake of time, decreasing the maximum tension of the strings was deemed an acceptable fix. More information on potential improvements or additions that could be made to the instrument can be found in Section 9.

5.2) Testing Methodology

System evaluation is a crucial step in any project’s progress, as it ensures that the system is capable of all that it was constructed to achieve. In order to ensure functionality of the instrument, tests were conducted that examined the three primary areas of concern in relation to the goals of the system - tuning speed, tuning accuracy, and latency.

Below are the testing methods that were utilized to evaluate the machine and its functionality. This testing took place in phases - after the range of each string was determined, the first portion of the tests concerned the operation of a single string. Within this first phase, the tests were divided into three sections - tests that concerned pitch accuracy, tests for speed, and tests for latency. Many of the tests that concerned one subject contained data pertaining to other subjects - please note when a variable is recorded in one test and utilized in another.

5.2.1) Pitch Range Determination

It is essential that the range of each string is understood for instrument functionality to occur. The bottom end of this range was determined by finding the lowest tension value that still allows for the note to sound clear - if the strings are not under enough tension, the string creates a buzzing sound that interferes with the tone of the string. The high end of the range will, ideally, allow for each string to span an octave. This is performed in an effort to ensure that the instrument will be able to form any chord - if each string can cover an octave, any note in a chord could be accessible. If this cannot be achieved, the high end of the string will be set as either the point at which the motor can no longer supply the force to the tuning machine to turn it, or before the string breaks. Please note that these ranges were recorded on an individual string when all adjacent strings were de-tensioned.

5.2.2) Pitch Accuracy

5.2.2.1) Load Cell Testing, Single String

The purpose of this test is to understand how accurately the load cell performs on a single string. The target tension of the load cell will be derived from Equation 12 using a target frequency. The string will then be tuned to this determined tension, and the frequency value will be recorded. This process will be repeated for every semitone interval the string is capable of supporting, not just for the notes played during operation. The recorded values will be target tension, achieved tension, desired frequency, and achieved frequency. The test will be repeated twice on the same string. If there are major discrepancies between the values of the two tests, a third test will be performed in an attempt to understand and reconcile them.

$$f = \frac{C}{\sqrt{T}} \tag{12}$$

Where T= string tension, f = string frequency

$$c = \frac{\frac{m}{L}}{2L} \tag{13}$$

Where m= string mass, L = string length

5.2.2.2) Load Cell Testing, Non-Tuning Strings Tensioned

The purpose of this test is to see if having tension across the rest of the instrument will affect a single string's tension-frequency relationship. Furthermore, the test will display whether or not the changing tension of the string being manipulated has an impact on the tension of the four other resting strings, which should, ideally, remain constant. The four additional strings will be observed at arbitrary tensions, while a target string is adjusted. The process described for load cell testing without additional tension will be repeated as the surrounding strings maintain a constant tension. The recorded values will be the target tension of the changing string, achieved tension of the changing string, desired frequency, achieved frequency, and the tension of the four theoretically unchanging strings. The test will be repeated twice on the same string, first with ascending tuning, then descending. If there are major discrepancies between the values of the two tests, a third test will be performed in an attempt to understand and reconcile them.

5.2.2.3) Photoresistor Testing

The purpose of this test is to determine which of the ten windowing functions built into the FFT library will be the most accurate and effective. Each function will be tested across the range of frequencies a single string can produce, and the most effective windowing function will be utilized. It is expected that different windowing functions will perform better for different ranges of frequencies - no one windowing function will be ideal. The recorded values will be the target frequency, achieved frequency, and the time it took for the tuning operation to be completed (for use in a later test).

Furthermore, this test will determine the accuracy of the FFT algorithm across all frequencies of a single string. The purpose of this part of the test is to see how the algorithm performs with lower notes, as during previous interactions with the system some difficulties have been noticed with this end of the range. The instrument will tune to a target frequency, and the actual frequency will be determined by both FFT and an external tuner. The measurements will be compared for error. The recorded values will be the target frequency, achieved frequency, and the time it took for the tuning operation to be completed (for use in a later test). The test will be repeated twice on the same string. If there are major discrepancies between the values of the two tests, a third test will be performed in an attempt to understand and reconcile them.

5.2.2.4) Encoder on a Single String

This test will determine the relationship of a motor's encoder value and frequency. The results will show whether frequency is constant or changing when returning to a set encoder value. An initial encoder value will be set and recorded, and frequency of the string will be measured. Then, a new encoder value will be input, and frequency will be measured. Finally, the motor will return to the initial encoder value, and frequency will be measured once again. From there, a comparison will be made of the different frequency measurements at the same encoder values.

5.2.3) Timing

5.2.3.1) Motor Speed

The purpose of this test is to see how fast the motor could hypothetically achieve tuning positions without the potentially slowing influences of the rest of the frequency determination system. In order to perform this test, the motor will be run at maximum speed for the times intervals 100ms, 200ms, 300ms, and 10000ms. Then, the interval of the encoder value that was traveled will be recorded. This test will be performed both under no load, with the tuning machine coupled to the motor without a string, and finally, with a string. The recorded values from this test will be the encoder interval jump performed, and the motor speed under the different previously specified conditions. This test will be repeated twice on the same motor. If there are major discrepancies between the values of the two tests, a third test will be performed in an attempt to understand and reconcile them.

5.2.3.2) Load Cell Amplifier Polling Rate Examination

The load cell will be polled for an arbitrary time interval long enough to produce viable data while the string's tension is static, and record the time between polling instances. The average of the time between readings will be taken, which is the unmodified polling rate. The load cell amplifier will be modified by cutting a "RATE" input wire, which is connected to ground. This "RATE" input, when connected to VCC, will let the load cell amplifier poll at the rate of 80Hz, while connecting to ground makes said amplifier poll at 10Hz. By default, the Sparkfun board utilized in the project had a default of connecting the "RATE" wire to ground. Therefore, by cutting the wire going to ground, the 80Hz polling rate on the HX711 amplifier can be achieved [73]. After the wire was cut, the same process was repeated to determine the modified polling rate.

5.2.3.3) Revolutions and Encoder Relationship

This test will be used to determine the amount of encoder ticks it takes to cause one revolution of the tuning machine. A zip tie will be attached to the tuning machine, and will be used for visual reference for counting revolutions. An initial encoder value will be measured, and then the motor will be run while revolutions are observed. Finally, the final encoder value will be measured, and the rate of encoder ticks per revolution can be calculated.

5.2.3.4) Revolution and Semitone Change Relationship

This test is used to determine the average revolutions of the motor needed to shift up a semitone, tested on an entire string's range. This data can be used to determine the cut off criteria for the motor selection. The test is performed on the 172:1 geared motor used to tune the D2 string. The initial encoder value is measured before the motor is tuned up by each semitone. Lastly, the final encoder value is measured, which is used to calculate the number of revolutions taken by each semitone change. The revolution changes is determined by the following Equation 14:

$$r = \frac{n_{enc}}{gear \times res_{enc}}$$

(14)

Where r = revolution per semitone

n_{enc} = number of encoder ticks

gear = motor gear ratio = [172]

res_{enc} = encoder resolution = [64]

5.2.4) Tuning Operation Latency

The purpose of this test is to understand the latency of the system. A potential source of latency for the system was the rate of polling for the HX711 load cell amplifier. In order to measure the latency, the values recorded will be the number of times the load cell amplifier was polled, the start and end pitch, and the time it took for the system to complete the tuning operation of adjusting from the start pitch to the end pitch. An equal amount of trials will be performed with the 10 Hz amplifier configuration and the 80 Hz amplifier configuration.

5.3) Testing Results

5.3.1) Pitch Range Determination

During this test, it was discovered that the body of the instrument is capable of supporting the full tension-based force the strings apply to it of 161.1lbs when the strings are tuned to their maximum pitches as displayed in the table below, but the wood bows as more force is applied. This results in changes to the string tensions, which brings all other strings out of tune once one is tuned to a specific tension. It additionally modifies the string length, affecting the calculation that creates the tension to frequency mappings.

The string ranges for the instrument were successfully determined during this test. It was found that for the four lowest strings, the motor is incapable of supplying enough force to break the string, resulting in a motor stall. However, the highest string broke during the determination of the highest note it is capable of reaching. This implies that the range of the bottom four strings need not be restricted in an effort to avoid notes that may break the string, as it had been found to be impossible with the current makeup of the system. The instrument successfully covers the three octave pitch range that it was designed for.

The maximum tension in pounds that each string can be brought to was also determined. For most strings, this was the most the motor could tension before stalling.

Table 7: String Range Determination

String Name	Low Range End	High Range End	High Range Determination Method	Maximum Tension (lb)

E♭1	E♭1	B♭2	Motor Stall	34.8
B♭1	B♭1	F3	Motor Stall	43.2
D2	D2	B3	Motor Stall	42.4
A2	A2	F4	Motor Stall	22.3
D3	D3	G4	String Breakage	18.4

5.3.2) Pitch Accuracy

5.3.2.1) Load Cell Testing, Single String

It is important to note that this test was performed with an external tuner to determine the actual frequency of the string. For this test, that tuner was a phone application called Pano Tuner [78]. It was found out that the string constant, in reality, varies when the string tension changes. From the data gathered in the three trials, it can be seen that the general trend of the string constant is decreasing. As can be seen in Figure 62, using the data fitting trendline function in Microsoft Excel, the 2nd degree polynomial displayed in Equation 13 offered a precise fitting of the plotted data.

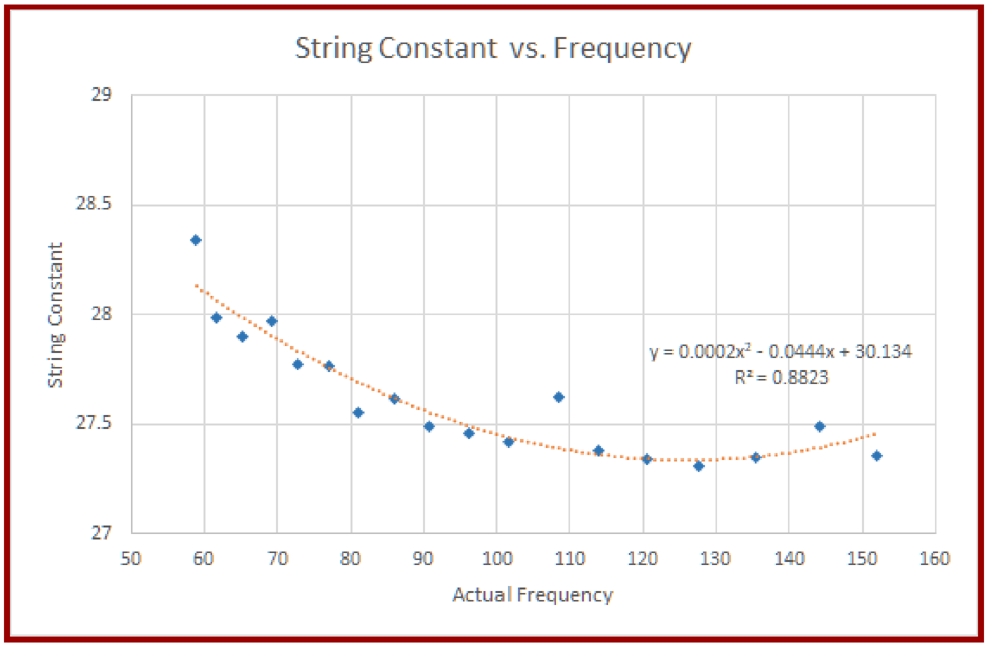


Figure 62: String constant vs. Frequency with 2nd degree polynomial fitted trendline:

$$y = 0.0002x^2 - 0.0444x + 30.134 \tag{13}$$

On average, the string constant was determined to be 27.617. The top half of the data range and bottom half of the data range constants were also extracted from the gathered data and their averages were taken, which were 27.823 and 27.413, respectively. Then, the different averages were utilized in Equation 12 to determine what frequency would theoretically be reached if each of the constants were used. The cent differences between these calculated frequencies and the recorded frequencies were then found, which can be observed in Figure 63 below. It was noticeable from the graph that the low range average constant is more suitable for low range notes from B♭1 to D♭2, while the overall average constant fits from D2 to G2, and the high range constant fits well for pitches from A♭2 to F3.

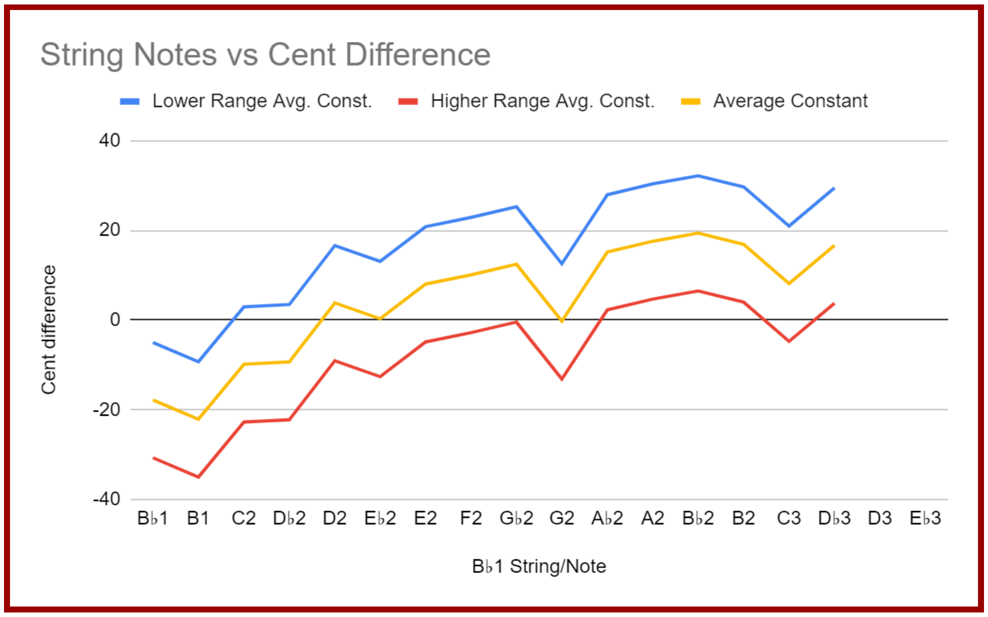


Figure 63: Cent difference comparison using different average string constants

One possible explanation for this inconsistency of the value of the load cell is the bowing of the instrument. However, that might not have been possible due to the fact that only one string - the B♭1 string - was fully tensioned during testing, which doubtfully caused any significant torque to make the instrument bow that much. Another explanation is the strain the tensioned string puts on the narrow channel holding the load cell, which causes it to bend and alter the length of the string, thus producing less accurate note and load cell constant.

5.3.2.2) Load Cell Testing, Non-Tuning Strings Tensioned

During this testing process, it was discovered that the readings on the load cells on other strings changed significantly when the tension on the B♭1 string altered. The general trend, which can be seen in Figure 64 and Figure 65, is that other strings were detensioned as the B♭1 string tension increased, and tension on the four other strings returned to its original tension when the B ♭ 1 string was loosened. It was observed that the larger the gauge of the string is, the lower the frequency produced will be, and the changes in the load cell reading will be more significant. In this case, the E ♭ 1 string experienced the biggest change, while the load cell reading on D3 string barely changed at all.

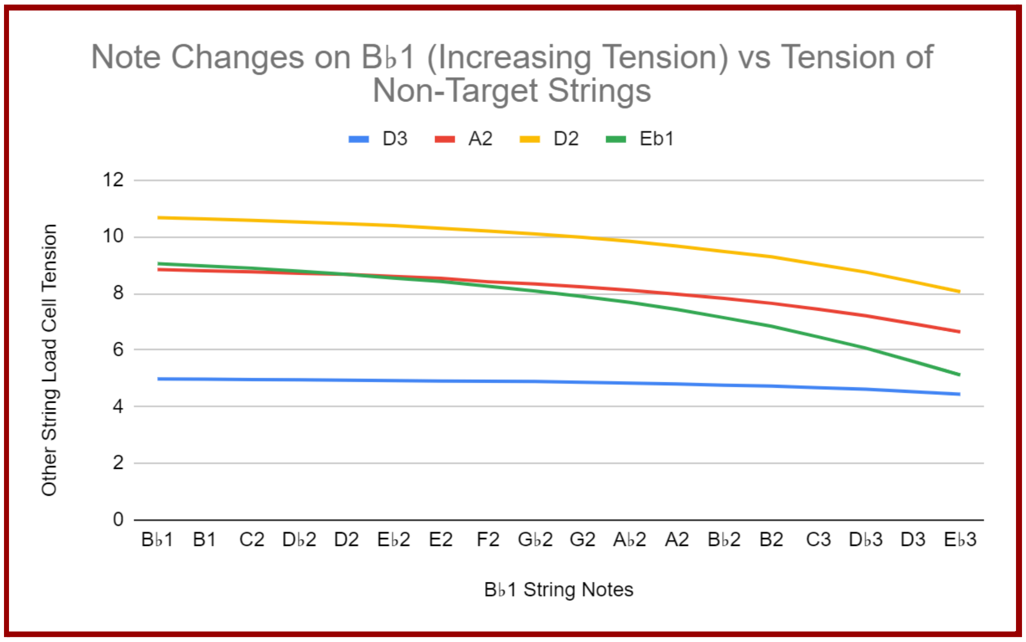


Figure 64: Note changes on B ♭ 1 string (increasing tension) vs load cell values of non-target strings

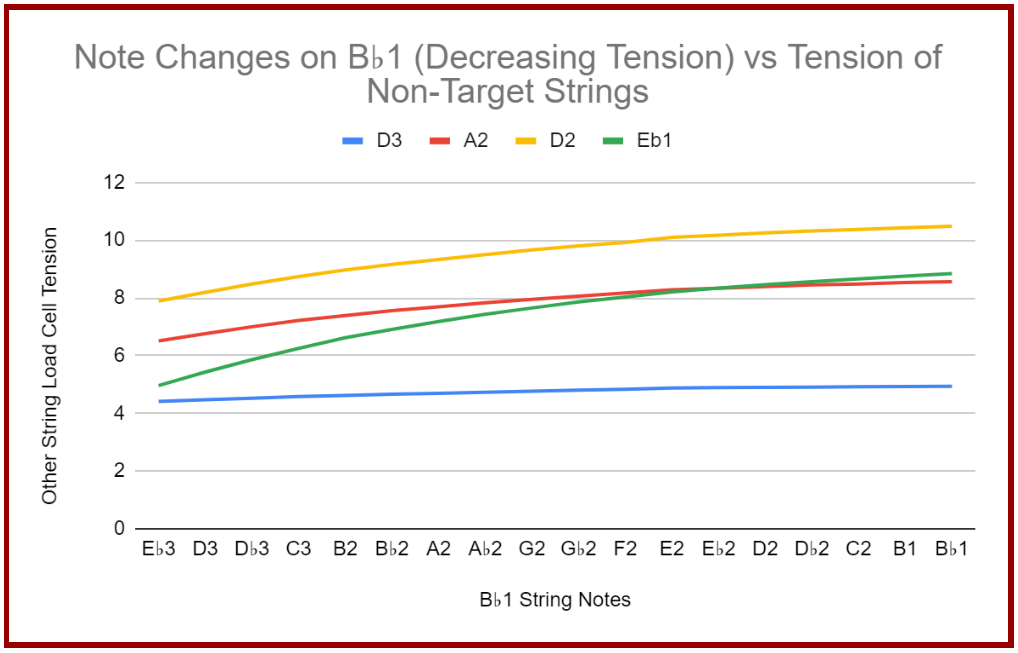


Figure 65: Note changes on B ♭ 1 string (decreasing tension) vs load cell values of non-target strings

The cause for these observations could also be accounted for by the same explanation in the previous single string load cell testing in Section 5.3.2.1. This change in the load cell's reading might also be the reason for the excessively lengthy tuning time, since the readings on the load cells change every time one of the strings is being tuned, causing the set points in the motor PID control to recalibrate after each computation. It was found that while tuning operations were occuring, the force of the string caused the load cells to put strain on the channel they were mounted in, and the base was not strong enough to prevent this strain from occurring. One solution to overcome such bending on the load cell channel is having a metal plate installed on the load cell channel. This would not only reinforce the narrow wooden channel on which the load cells were mounted, but also improve the appearance of the instrument by concealing the bolts and nuts used for attaching the load cells to the instrument body.

5.3.2.3) Photoresistor Testing

The data below is produced by the average of windowing FFTs after filtering data outside 0.92 and 1.08 times the target frequency. Raw measurements were produced by actuating strings with solenoids, and determining frequency with FFT. The raw data was printed to the console and parsed by a Java program which performed the filtering and averaging operations. The rightmost column is the average of all post-filtering frequencies. It was found in this test that FFT is inaccurate at low frequencies, but accurate at higher frequencies. At lower frequencies, a displaced harmonic of the fundamental is often found, which is beyond the design parameters of six cents difference between target and actual frequency. The higher the frequency, the more valid measurements are produced with FFT and the photoresistor. Note that anything below the red line in the following graphs means the accuracy fit design parameters of 6 cents, and that windowing types which aren't shown on a particular graph mean that they were filtered out using the above method.

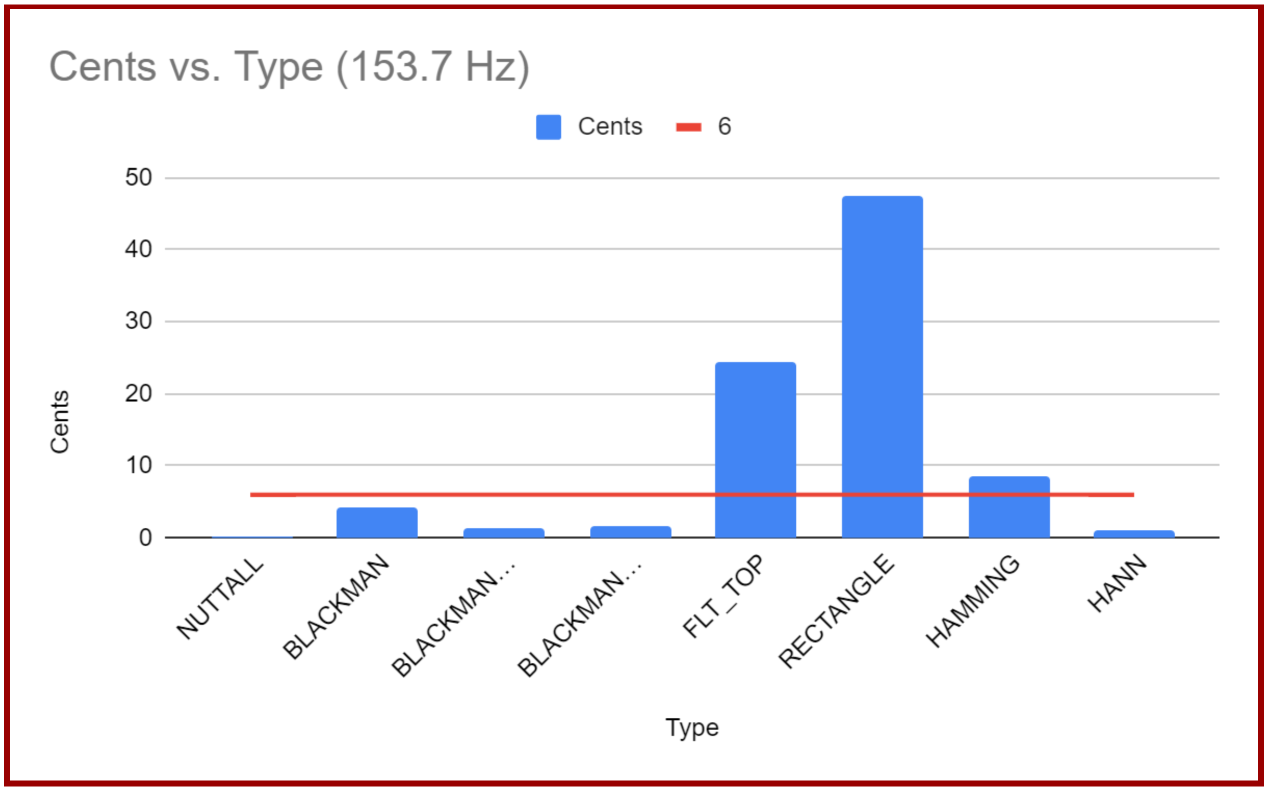


Figure 100: 153.7 Hz Windowing

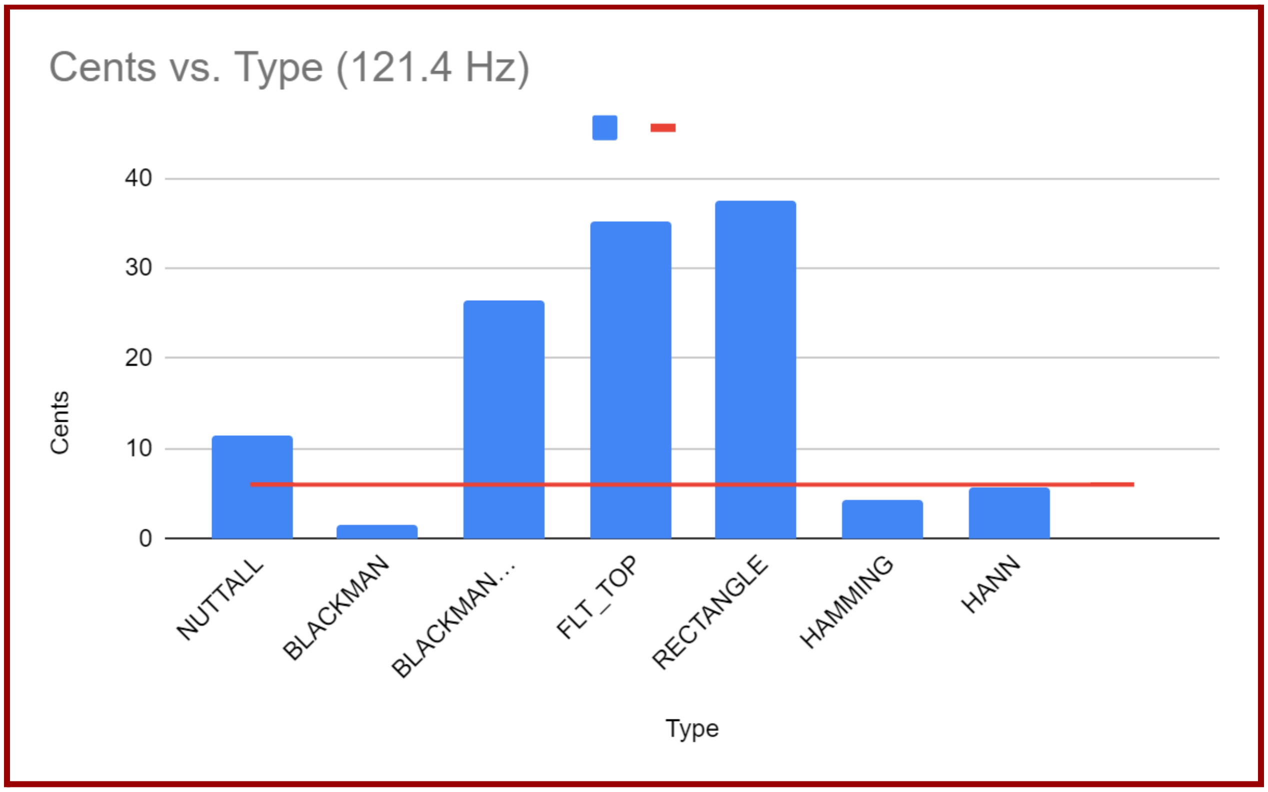


Figure 101: 121.4 Hz Windowing

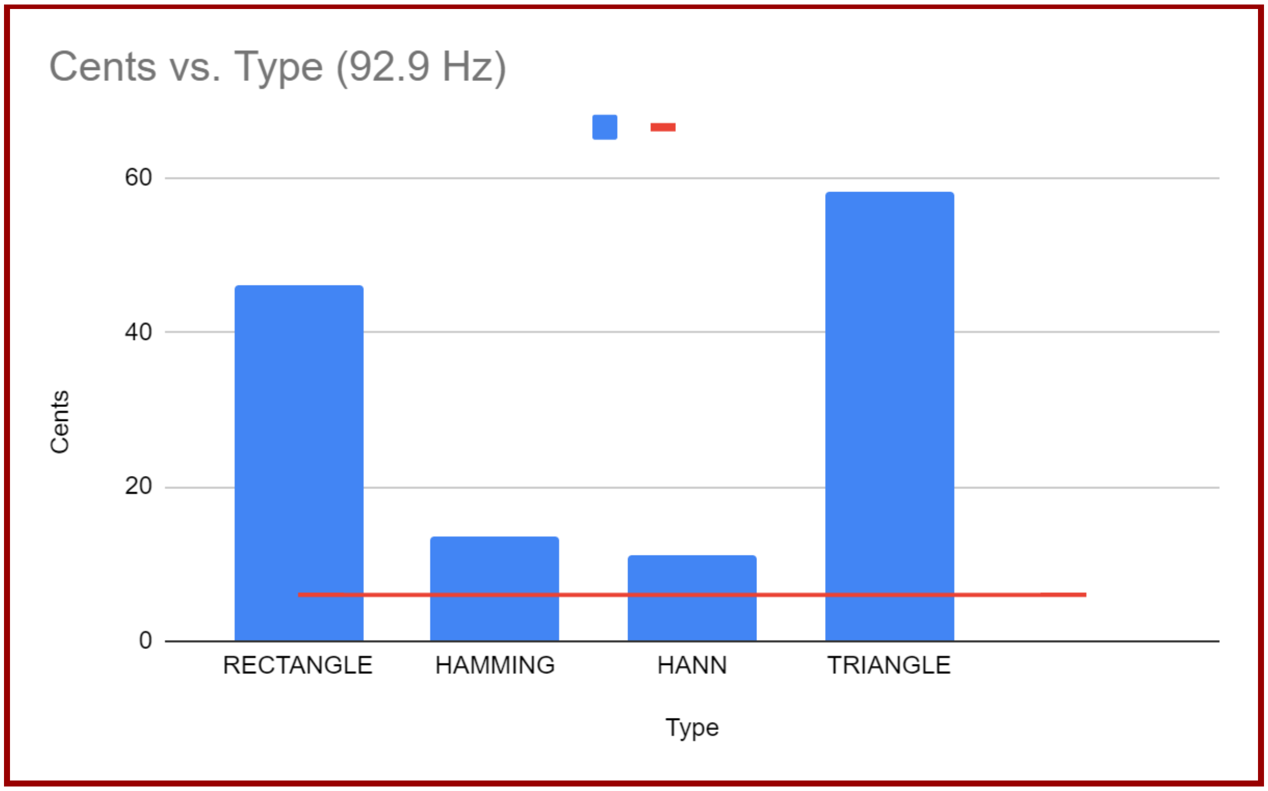


Figure 103: 92.9 Hz Windowing

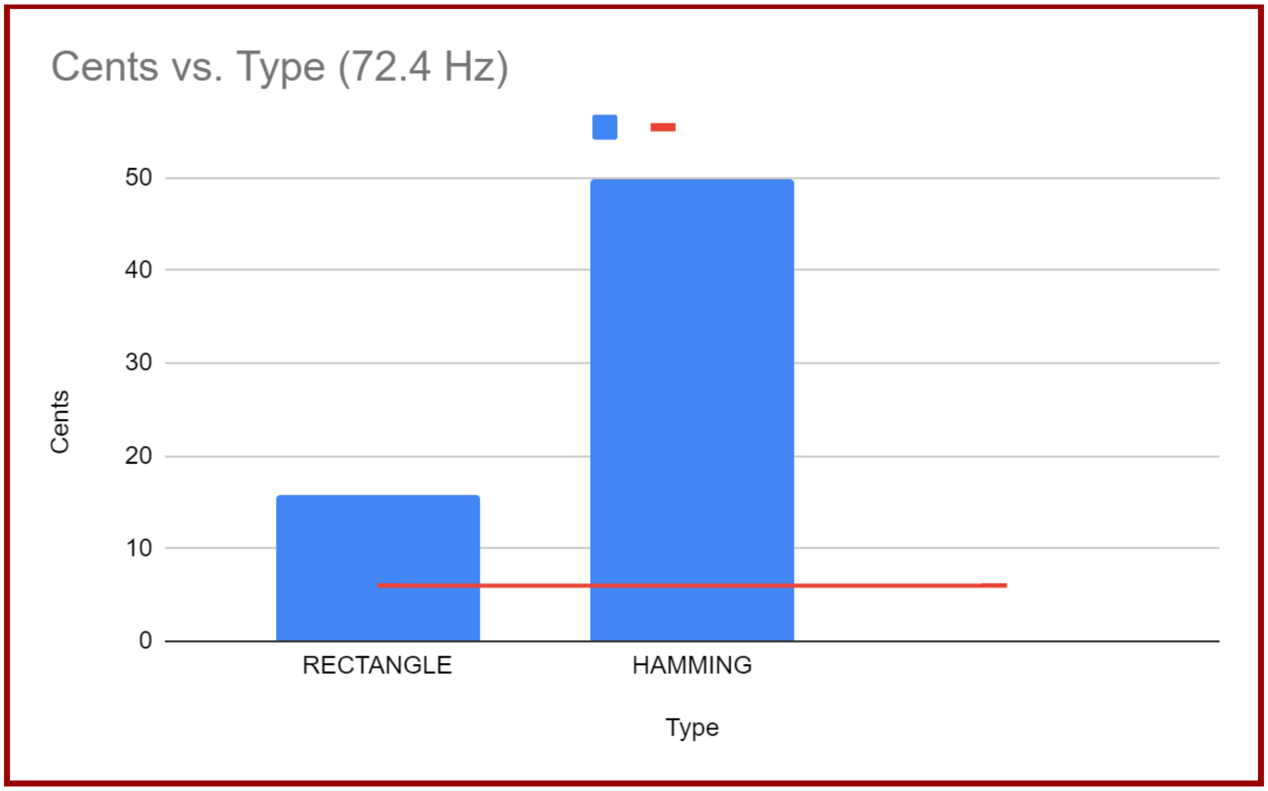


Figure 104: 72.4 Hz Windowing

5.3.2.4) Encoder on a Single String

For this test, it was found that there was slight variability in frequency when repeatedly returning the motor to specific encoder values. The figure below shows the various frequency readings for encoder position 0. It can be observed that the values for frequency vary across the similar encoder positions. The values were not measured at exactly position 0 due to the slight inaccuracy of the encoder. However, along the x-axis, different frequencies were recorded for the same encoder values. This test shows that simply setting an encoder value is not a viable strategy for achieving desired frequencies, thus justifying the other modules.

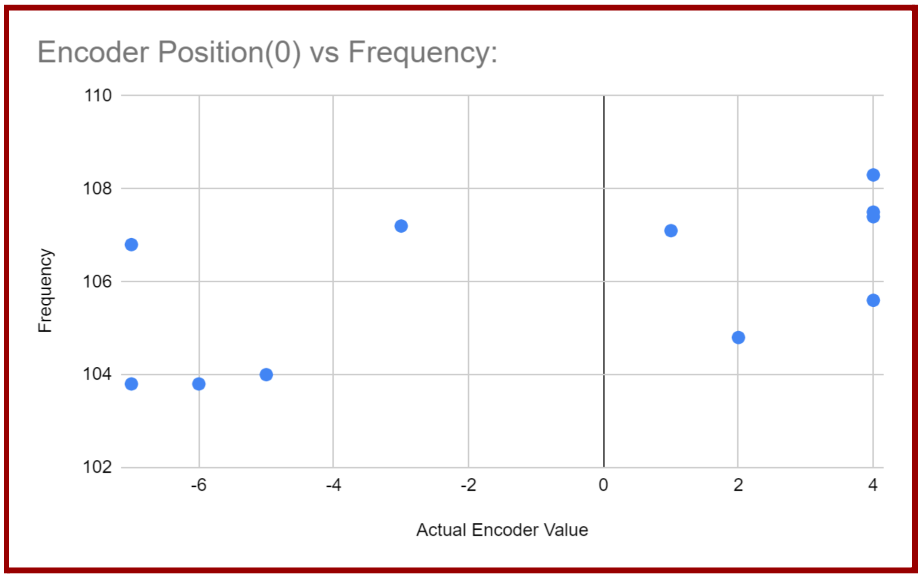


Figure 66: Encoder Position 0 vs. Frequency

5.3.3) Timing

5.3.3.1) Motor Speed

For this test, three trials were performed on the A2 string. In trial one, the motor did not have anything attached to the shaft and it was not under load. In trial two, the motor was coupled to the tuning machine, and in trial three, the motor was coupled to the tuning machine and the tuning machine was attached to a tensioned string. The values recorded from the test were the start and end encoder values, the start and end frequencies, the time interval in milliseconds that the motor was running at maximum speed, and the size of the interval jump between the start and end encoder value. The time intervals used for the tests were 100 ms and 200 ms for all of the trials, and additional intervals of 300 ms and 10,000 ms were used for trials one and two. The 100 ms and 200 ms times where chosen as they were the maximum proposed tuning times dependent on tuning interval. The fact that it took a shorter time to move a large pitch distance in this test versus the note tuning tests is due to accuracy not being a factor in this test. The encoder interval jump, the time interval, and the gear ratio of the motor were used to calculate the RPM.

The data from the 200 ms testing for trials one, two, and three are graphed in Figures 67, 68, and 69. Please note that the graphs for the 200ms tests were chosen to represent the data - the graphs for the additional tests display the same trend. In the graphs, the blue dots represent endpoints that the motor reached during the testing for each trial. From reviewing the data, it was observed that an increase of the load on the motor across the three trials resulted in a decrease in the motor speed. In trial three, the encoder position also indicated tension where lower encoder values were lower tensions and higher values were higher tensions. It was observed in trial three that the motor speed decreased as the tension in the string increased and that the motor speed increased as the tension in the string decreased. Figure 67 depicts five results from the motor test - the resulting RPM and encoder value coordinates after the 200ms operation are represented by blue dots, so that the difference between each of the results can be seen. The rest of the graphs are in the same format.

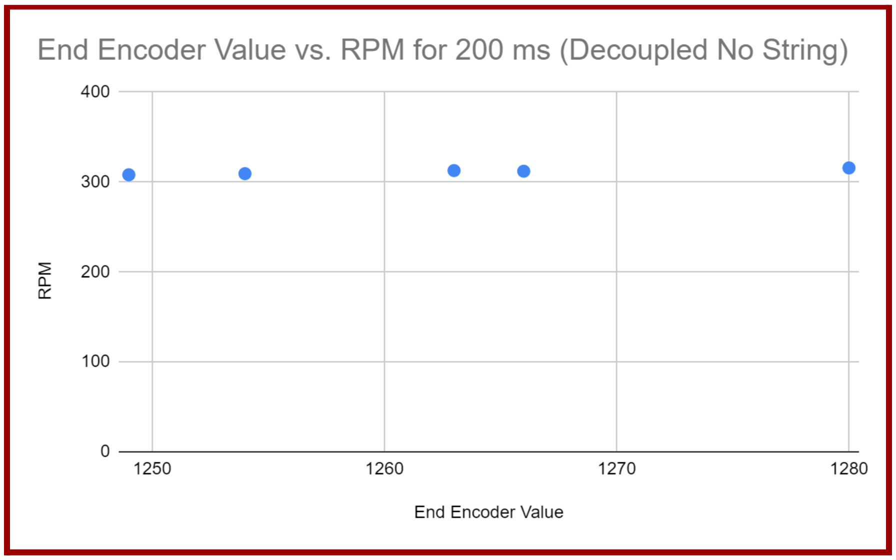


Figure 67: Graph of Encoder Value vs. RPM for 200 ms time with motor under no load

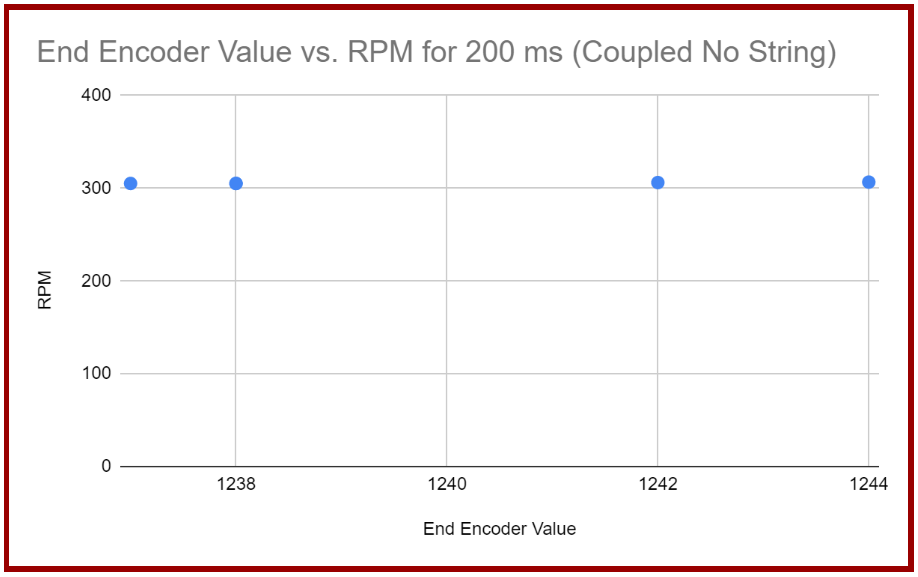


Figure 68: Graph of Encoder Value vs. RPM for 200 ms time for motor under load of tuning machine

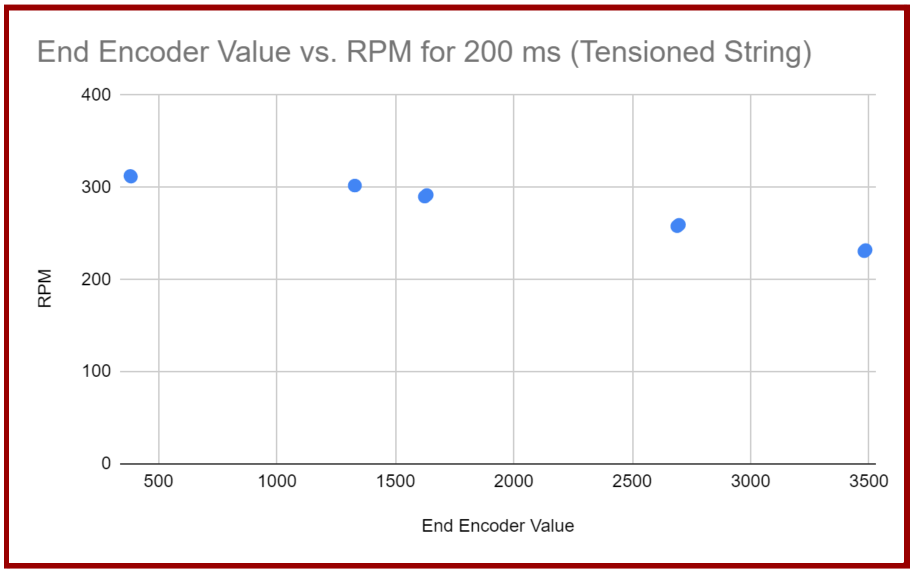


Figure 69: Graph of Encoder Value vs. RPM for 200 ms time with motor under load of tuning machine and tensioned string

5.3.3.2) Load Cell Amplifier Polling Rate Examination

Upon completion of the testing, it was found that while the frequency of polling is higher for the modified amplifier, there was a reduced quantity of polling instances detected when compared to the unmodified amplifier. While the 10Hz load cell may not be collecting data at as frequent of a rate as the modified load cell, thus potentially reducing the accuracy of the data, the unmodified amplifier communicated its value far more frequently than its modified counterpart.

These results posed a new question of whether or not one load cell configuration is more accurate than the other. If this was known, a cost benefit analysis of the exchange of accuracy for speed could be performed. However, the potential need for this test was not identified until a week before the project was slated to reach its conclusion. It is recommended that this test be conducted and a decision on which format of load cell to utilize be made in future.

Table 8: Load Cell Amplifier Polling Rate Data

	Unmodified Amplifier	**Modified Amplifier**
Average Number of Polling Instances During Tuning	96.047878049	11.4852734
Average Frequency of Polling	10.41137633	87.0681807

5.3.3.3) Revolutions and Encoder Relationship

For this test, five revolutions of the tuning machine were observed. The initial encoder value was 0, and the final recorded value was 84,500. By dividing 84,500 encoder ticks by 5 revolutions, the result is a rate of 16,900 encoder ticks per revolution of the tuning machine.

5.3.3.4) Revolution and Semitone Change Relationship

For this test, it was observed that the average tuning revolution per semitone changed depending on which note the motor is required to tune from. As a general trend, the number of revolutions to tune a semitone increases as the pitch increases, which is shown in Figure 105.

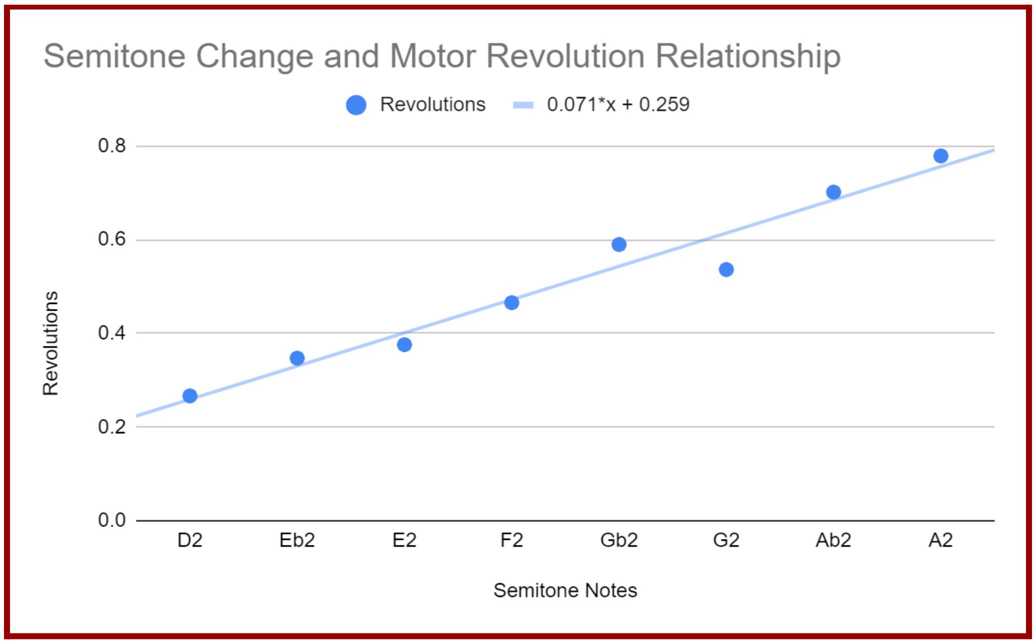


Figure 105: Semitone Change and Motor Revolution Relationship

The test was performed on the D2 string, starting from its lowest note that it is named for. The minimum requirement was based on the results from F2, as the extremes of the string ranges theoretically will not be utilized often. From Figure 105, it can be seen that the shortest revolution the motor has to travel is about 0.25 revolutions. However, an average of 0.5 revolutions or greater is required in order to tune a semitone at middle-range notes on the string. Since most tuning operations should occur within the middle-range of pitches, both so that the string's durability lasts longer, the 0.5 revolution per semitone, or 300RPM was determined to be the cut off speed requirement for motors.

5.3.4) Tuning Operation Latency

For this test, four total trials were performed on the Eb1 string and the start and end pitches tested ranged from 0 to 5. Two of the trials were performed with the load cell amplifier in the 10Hz configuration and two trials were performed with the amplifier in the 80Hz configuration. The results from the four trials are shown in the graphs below. The graphs in Figure 70 and Figure 71 show the relationship between the times the load cell amplifier was polled and the time of the tuning operation. The graphs in Figure 72 and 73 show the relationship between the pitch interval traveled, measured in semitones, and the time of the tuning operation. From comparing the data from the 10Hz amplifier testing and the 80Hz amplifier testing it was observed that the number of times polled and the tuning time for the 80Hz amplifier was less than it was for the 10Hz amplifier, therefore resulting in a decrease in latency for the system.

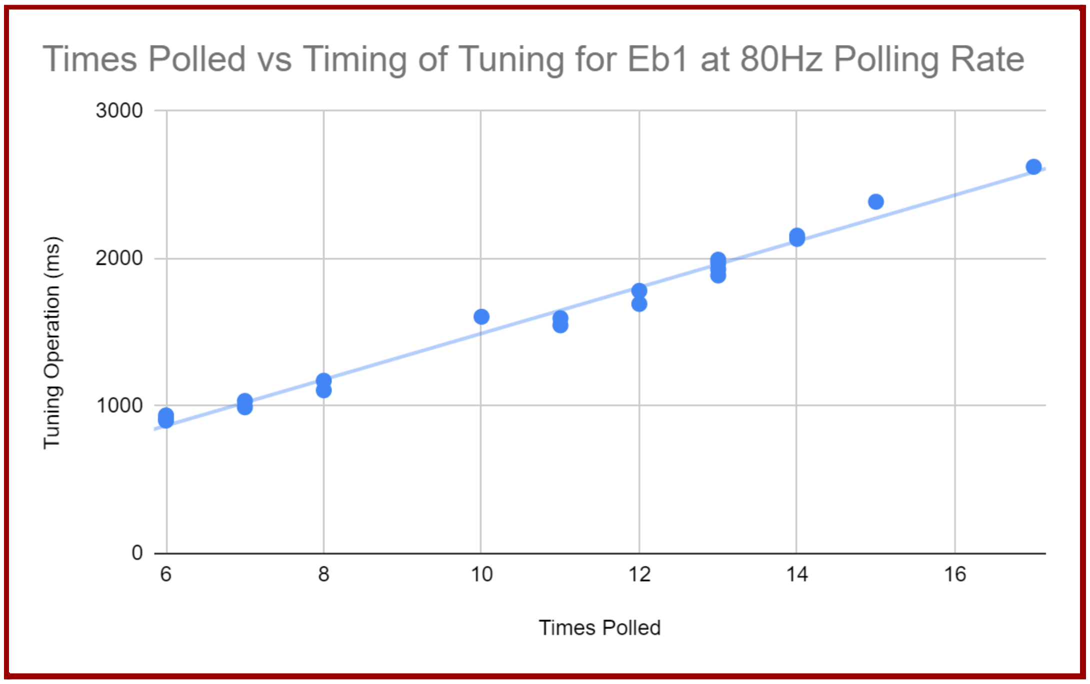


Figure 70: Graph of Times Polled vs. Timing of Tuning for 80Hz amplifier configuration

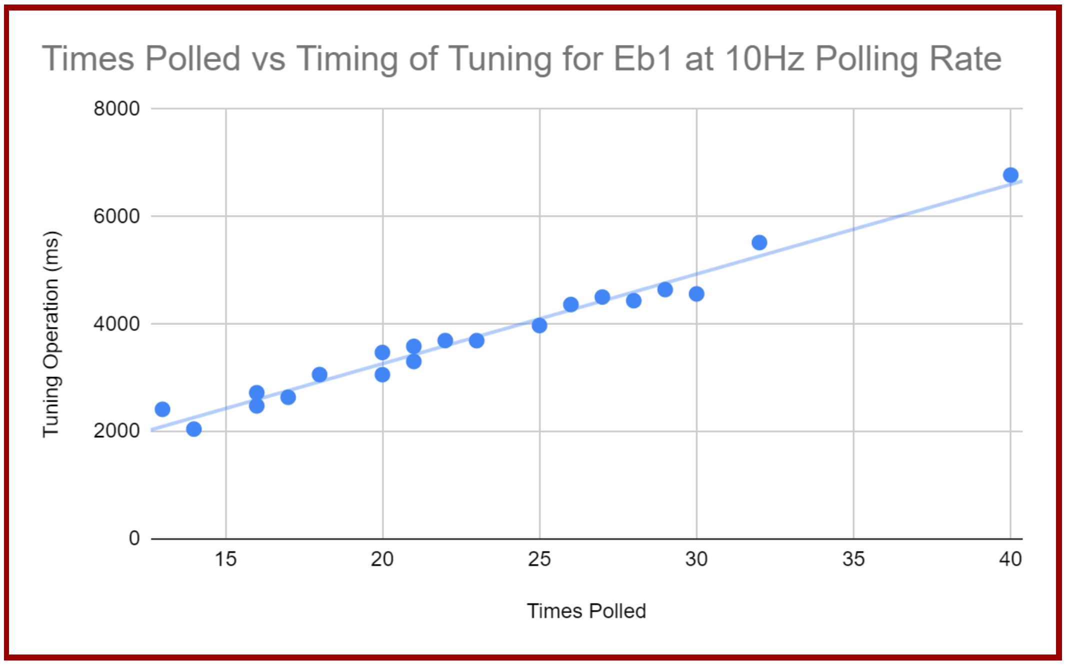


Figure 71: Graph of Times Polled vs. Timing of Tuning for 10Hz amplifier configuration

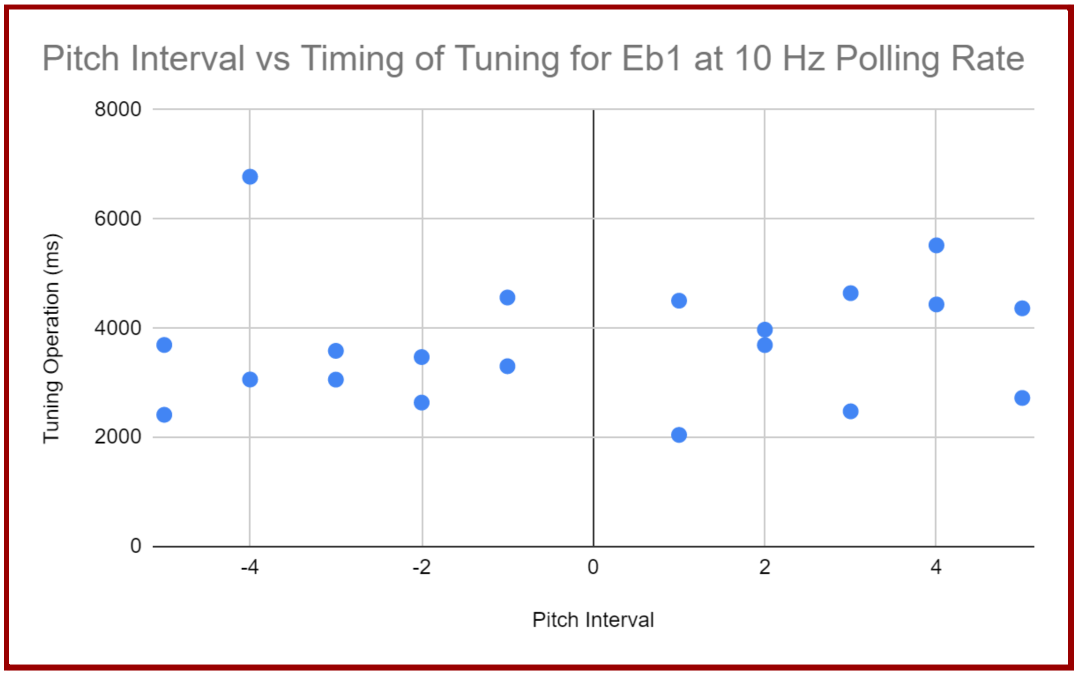


Figure 72: Graph of Pitch Interval vs. Timing of Tuning for 10Hz amplifier configuration

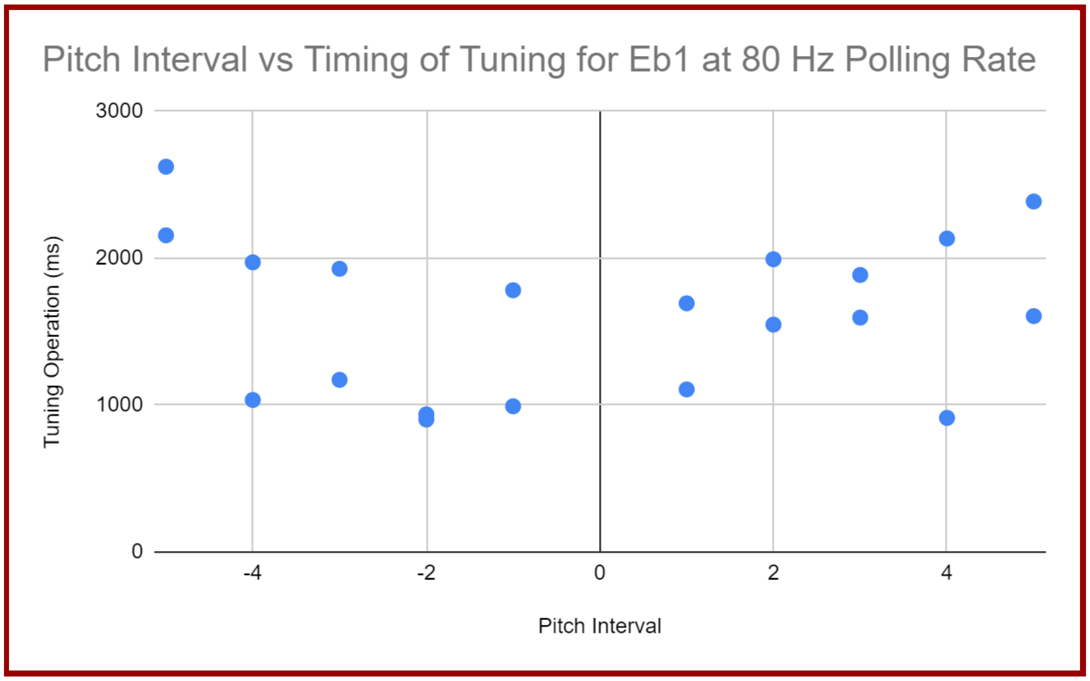


Figure 73: Graph of Pitch Interval vs. Timing of Tuning for 80Hz amplifier configuration

5.3.5) Serial Communication Latency

Serial communication latency was determined by sending a signal from the host computer to the microcontroller and back, and taking the difference between the timestamp when sent and when received. The round-trip time was determined to be between 3 and 6 milliseconds, averaging 4.5. This makes one way communication around 2.25 milliseconds.

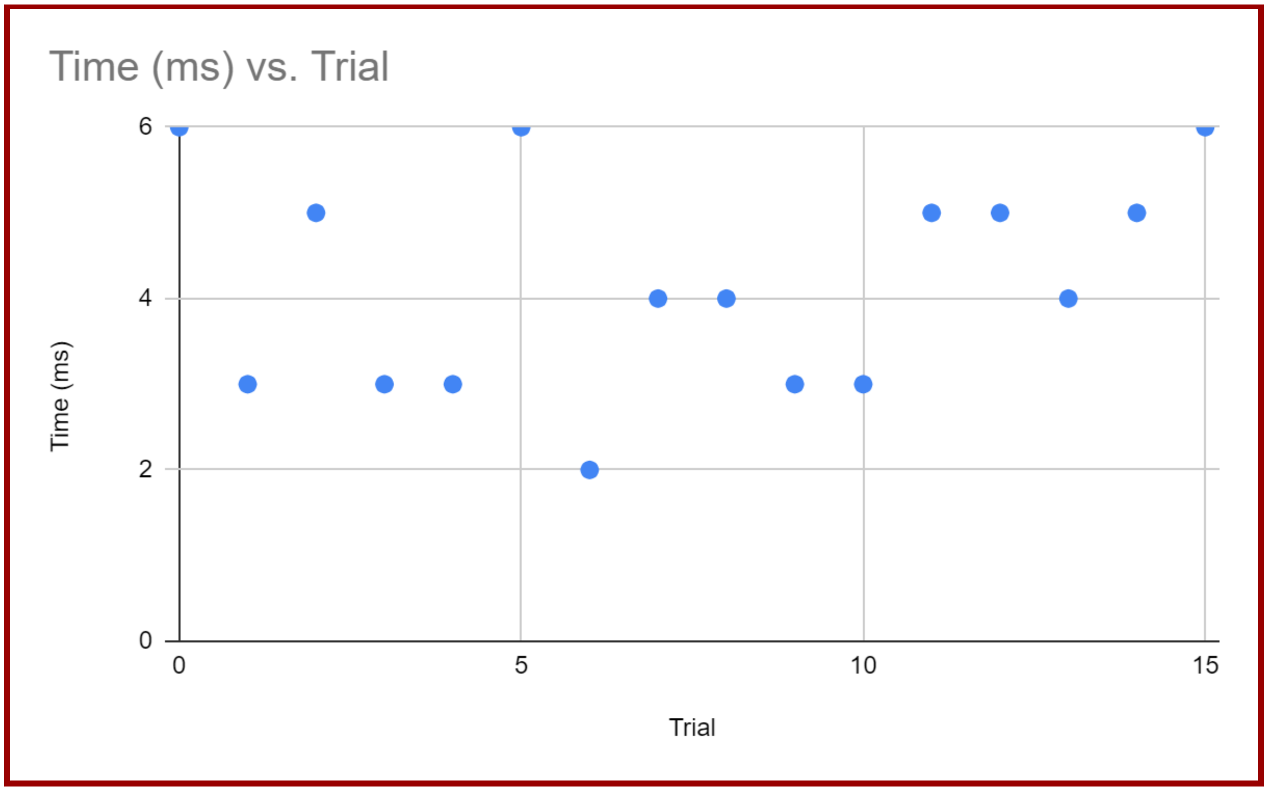


Figure 104: Serial Communication Latency Data

5.3.6) Control Loop Latency

The control loop latency for various states is listed below. 20 samples of how long it took to run through the loop function were recorded and averaged, taking the minimum and maximum times. These results indicate that a tuning operation can take between 1.8-5.8 ms of additional time per cycle. The frequency of cycles at the minimum period shows what the most effective polling rate of a load cell amplifier would be. This is determined to be 200 Hz.

State	**Control loop time (ms)**	**Min Time (ms)**	**Max Time (ms)**
tuning one string 80 Hz	5.916666667	5	7
not tuning	3.2	3	4
tuning one string 10 Hz	7.333333333	6	9
tuning all 5 strings	13.875	13	16

Table 11: Control Loop Latency Table

5.4) Analysis of Results

It was determined that the device takes between 1000 to 1800 milliseconds to tune a semitone on an 80 Hz amplifier. This averages at 1400 ms, and is not significantly changed by the direction of the semitone movement. With a 10 Hz amplifier, the tuning took between 2000 and 5000 milliseconds, averaging 3500 ms without significant difference by the direction of the tuning. When using a 10 Hz amplifier, the amplifier was polled an average of 23.5 times per tuning operation. The 80 Hz amplifier, despite being faster, was being polled less - at 10.4 times per operation. The algorithm works by moving to an encoder value, checking a load cell tension with expected, and re-adjusting the encoder value. The observation that a faster polling load cell more than halves the tuning time, indicates that responsiveness is an issue. It was found that it takes around 200 milliseconds to tune up or down an approximate semitone, using just PID control with an encoder. It is probable that the additional average 1200 ms of time for the 80 Hz amplifier could be reduced further with an amplifier that polls at a faster rate.

The algorithm used for the tuning operation assumes that the load cell is the best method of recording data that can be processed to produce a string's frequency. Disregarding mechanical issues that cause one string's tension to affect the body of the instrument itself, the load cell is seen experimentally to be a better indicator of frequency than the encoder position. Due to the poll rate of the load cell, it cannot be used as the primary source of error for PID control because measured error will always lag behind the control system. The encoder then has to be used as the primary source of error, even though it is derivative of the load cell. This requires the load cell to be checked at least once after the encoder reaches its setpoint, which tends to cause the expected encoder position to change because the encoder is not an accurate indicator of frequency (see 5.3.2.4). This repeated polling and encoder PID operation may result in the significant time costs previously mentioned.

When analysing the testing, it was also discovered that there is no standard time interval for a semitone-based tuning operation to be performed. One of the causes for this is the time taken to travel a certain interval varies depending on the pitch travelled. Elapsed time tuning one interval at high range notes is longer than time taken to tune the same interval, but at lower notes. This is because more tension is needed at higher notes to create more significant changes in pitch, while the opposite phenomenon is observed with lower notes. Another explanation for the varied tuning time is the changing motor speed under different tensions. As observed in Section 5.3.3.1, motor speed decreases as the string tension increases and vice versa, which makes the tuning time determination harder to standardize. Furthermore, the inaccuracy in the motor encoder's position introduces another problem to the system. Some possible explanations for this include the backlash in the motor, missing interrupts, and changes in the physical system itself. Interrupts can be missed when they occur while the program is in another interrupt routine. Since multiple motors may be moving at once, with each encoder having its own interrupt routine, this is a possibility. This problem causes tuning inaccuracy, causing the need for string recalibration using other modules. Additionally, the instrument body bowing situation as well as the strain the strings exert onto the load cell channel also adds another layer of complexity to determining the tuning time over different interval jumps due to the complication of how the strings interact with each other and the body itself.

The variable tension of strings as adjustments were made had a large, negative impact on the overall effectiveness of the instrument. The results displayed in Section 5.2.2.2 show that as a single string was adjusted, the tension of the other four strings drastically changed, thus changing the frequencies. This is likely caused by the bowing of the instrument as tension increases, such that the vibrating length of the non-target strings decreases. Because of this, the testing for chord functions was not conducted, and the chord functionality was not further developed. The changing tensions was an unforeseen issue, and the possible solutions were not feasible for this project. Those solutions are described in Section 7.1.1.

Given this data, certain assumptions that drove the control system of the instrument are affected. The first realization is that the physical structure is inordinately affected by string tension, which reduces load cell accuracy. The second is that, possibly due to this, the photoresistor produces the most accurate measure of frequency using FFT at higher frequencies. FFT was not performed on a string during the tuning process. These realizations have not fundamentally changed the design of the system. Although the load cell is less accurate than expected, using the photoresistor is not a viable alternative as it would have to sample during the tuning operation while the string is vibrating. The FFT would then output an average frequency between the frequency at the start and end of its sampling. Additionally, the FFT is only shown as effective using a filtering method as described in 4.4.3., and when the average of measurements are taken. If measurements are invalidated, it would affect the speed of the system. This is because it takes a certain amount of time to perform the FFT itself. The best change possible (to the control system only) in view of this gathered data is to replace the load cell 'constant' with a function that is affected by the instrument's overall tension and the current string's tension. An example of this function was derived in the results section.

5.5) Design Requirement Analysis

Below is an examination of how well the prototype achieved its goals.

Pitch Range

The goal was for the instrument to be capable of producing pitches that spanned three octaves. Since the instrument is capable of producing a lowest note of E ♭ 1 and a highest note of G4, the instrument is able to successfully produce over three octaves of pitches. Each string is able to produce more than an octave's worth of notes, so any pitch on this scale should be achievable, rendering chord creation possible.

Weight/Dimension

The device weighs 9.0 lbs. The dimension of the constructed wood base of the instrument, sans the attached electrical component is 35.5 inches x 5 inches x 2.25 inches. The finished prototype with all of the components attached has a dimension of 36 inches x 7.5 inches x 5.5 inches. The instrument is portable, and can be transported through doors with ease, which was a design requirement. The weight is also much less than the twenty-five pound requirement.

Dynamic Range

The instrument has been equipped with an EMF pickup. Due to the nature of this sound output, the volume of the instrument can be varied depending on the need of the instrument user via an external amplifier. Therefore, the 60dB range is achieved as the instrument can produce sound at any volume.

Tuning Speed

For a single string, the device has an average tuning speed of 3500 milliseconds for a semitone. This was far removed from the design requirement of 100 milliseconds. When tested with a load cell amplifier with higher polling frequency, this time was more than halved. The variation is likely attributable primarily to the load cell polling rate, and secondarily to the efficiency of the PID control system. The PID by itself took around 200 milliseconds to tune up or down an approximate semitone using an encoder position only. ‘Approximate semitone’ is used because the encoder is not an accurate indicator of position as found in the testing results section, although two encoder positions were used that changed the string pitch by a semitone on the first trial. More analysis is available in the ‘Analysis of Results’ section.

Tuning Accuracy

It was found that due to the bowing of the base, the accuracy of the load cell is highly variable. As is described in Equations 11 and 12, it is assumed that the vibrating length of the string will remain constant in order for the tension-frequency relationship to remain consistent. However, due to this wood bowing, the constant changes in a way that was graphed in the results discussion section. It was found that if the average constant measured across all tests was used to predict the frequency of a string given the read load cell value, there was an average of a 14.19 cent difference between the predicted and actual frequencies. The target goal of 6 cents accuracy was achieved in 3 frequency determinations of 18. As can be seen in the results section, utilizing the average of the upper range of tension for the calculation of the higher strings’ constant and vice versa for the lower strings’ results in a more accurate calculation of the observed string frequency.

Furthermore, the photoresistor accuracy was tested and demonstrated that the FFT method is more accurate for the higher frequency strings. Plus, different FFT windowing functions performed better for different strings; the NUTTALL function was preferable at higher frequencies, resulting in a 0.2 cent difference at 153.7Hz, the BLACKMAN function performs better at middling frequencies, and the HANN function provides the best approximation at lower frequencies. With an 11.3 cent difference between the predicted and recorded frequencies, this accuracy falls short of the design goals across the entirety of the instrument's range. However, it can be demonstrated by the higher frequencies that the principal of the methods explored for this project can be developed to functionality further.

Input Capabilities

The design requirement was for the instrument to be able to accept MIDI-based instructions from a user interface which would cause the instrument to tune to a certain chord or tune to individual strings based on the preference of the user. This requirement was partially met because while the user interface is functional for tuning the instrument to chords or tuning individual strings, the user interface sends ASCII commands instead of MIDI-based commands.

Latency

As can be seen from the rest of the results analysis, it takes between 1000-1800ms to perform a tuning operation. We found an average serial communication latency of 2.3 milliseconds. This was determined by sending a command from a host PC to the microcontroller, which then sent a response. The timestamps of when the command was sent and the response received were compared and the difference measured. This determined the round-trip-time (RTT) of 3-6 milliseconds. The average was 4.5 ms, which makes one-way communication 2.3 ms assuming both devices communicated at the same rate. It takes the device from 3 to 4 milliseconds to execute one loop while in an idle state. The device was also found to increase its latency by between 2 and 6 milliseconds per string in a tuning process. Control loop latency increases affects the behavior of control systems, and can make it less responsive to changes in information. From the testing results it was determined that while tuning all strings the system updates the control system less than a third as often as when only one string is being tuned.

6) Market Impact Analysis

The prototype created for this project could have a significant market impact if developed into a product. It would allow an easy way to play an instrument, which so many people desire. This instrument could serve those who are incapable of playing a traditional string instrument; for example, if an individual has lost the use of their left hand, they would be unable to form chords in the traditional manner on a guitar. This autonomously tuning instrument could provide that individual with the ability to play music again. Furthermore, the instrument could be enjoyed by people who are interested in music, but have limited musical aptitude. The ease of playing the instrument via computer interface allows for anyone to play chord progressions or melodies. The instrument ensures that every note is played accurately - all that is required of the user is note selection. Unfortunately, due to the restrictions of the prototype built, testing and evaluation of this concept could not be completed. However, if the recommendations for improvement detailed in the following section are addressed, the system could fill this market gap.

Finally, an advanced musician could potentially benefit from the instrument. Having a web-based control for the instrument would allow for complex songs to be preprogrammed. The instrument could be capable of producing basic chord progressions, and of tuning its individual strings rapidly to emulate the ‘finger picking’ style of guitar playing. Furthermore, this would allow a musician to play in a style that would not work on a regular guitar - complex and opposing sliding operations could be performed on multiple strings simultaneously, for example. Operations such as these could not be performed on a traditional instrument, and could lead to the creation of groundbreaking musical sounds.

7) Recommendations for Future Work

7.1) Improvements

7.1.1) Instrument Recommendations

One recommendation for improvement of the instrument is to construct a more structurally sound body. None of the members of the team had experience in materials science or mechanical engineering, and unfortunately the fact that the wood of the instrument would warp when put under the tension the strings impose was not considered. Furthermore, the way in which the load cells were mounted caused them to be able to move - they were mounted in an not reinforced, shallow channel in the wood of the instrument. When the strings exerted force onto their respective load cells, the tugging caused their positions to shift. The instrument was not brought to full tension until the middle of C term, so this fact was not discovered until it was too late to pursue a new plan of action and is left to be improved in the next iteration of the project. These two issues significantly reduce the string's range, and affect the rate at which multiple strings tune themselves.

While the instrument is currently capable of producing chords, the complications introduced to multi-string tuning operations prevented the chord module of the user interface from being fully implemented. Further testing was not conducted on the formation of chords due to the knowledge of the impact the previously stated weakness would incur upon the operation. If the base construction issues are resolved, it should not be complex to build upon the work completed in this project to create successful chord formations. Furthermore, the machine is currently communicating via ASCII commands and not with MIDI. It is advised that the communication method be switched for future iterations of this project.

Another recommendation is that more work be done for the motor control algorithm. The load cell could be used for the source of error, rather than an encoder reading. The control could also be run at constant intervals to create uniformity in the control and make experimenting with the parameters easier.

For solenoid actuation, it may be preferable to make an external circuit that manages their actuation, as difficulty was encountered when using the Arduino's proto threading libraries to run them, and chose instead to actuate them based on a control loop. This method, unfortunately, depends on the time taken to perform the rest of the program's operations.

7.1.2) Testing Recommendations

Recommendations for future testing have been documented. The latency introduced by the user interface was not tested during the duration of the project. One of the reasons behind this decision was that communications to the machine were being sent via ASCII commands, not MIDI commands. Furthermore, due to the physical limitations on the system, it would be difficult to tell how much of the timing errors potentially discovered were due to the latency of the interface or the system issues. Because of the potential differences in these communication forms, testing was not completed.

Based on analysis of the results from the polling rate and tuning operation latency testing, the decision was made to not modify the original amplifiers on the instrument to the 80Hz configuration based only on its impact on tuning time. Another test to determine the accuracy of the 80Hz amplifier compared to the 10Hz amplifier would be necessary before making the decision to modify all of the amplifiers because there may be a tradeoff between amplifier polling rate and accuracy.

Finally, the tests involving the photoresistor should be repeated using both solenoid actuation as their method for activation, then with a user plucking the string. This test would determine whether or not the direction of string activation - horizontal or vertical - has an impact on the way the photoresistor processes the string frequency. The methods written for the project were intended to be performed in this manner, but this did not come to pass.

7.2) Additions

On top of the improvements that could be made to the prototype design, there is one area where an addition would drastically improve the instrument - an independent control module. The prototype design currently utilizes an external computer to control the string adjustment. This, although functional, is inconvenient for a musician. It would be much easier to just have a lone, functioning instrument, rather than needing to set up a laptop alongside it.

It would also be possible to add more strings to the instrument in order to increase the pitch range. For this prototype, five strings were used such that the goal for pitch range was achieved. However, more strings could be added to the design. This would require a larger body, the acquisition of a new microcontroller with a greater number of I/O pins, and a remodel of the photoresistor case to accommodate each added string.

Appendix A: Relevant Musical Definitions

<u>Semitone</u>

A semitone, commonly referred to as a half step, is the smallest musical interval one would typically play. This tonal span is also defined as the musical distance from a white key to the neck black key on a piano(Mount, 2009). There are twelve of these semitones in an octave, as displayed in Table Twelve below.

Table 9: Semitone Intervals on a C-Scale

C	C#/D ♭	D	D#/E ♭	E	F	F#/G ♭	G	G#/A ♭	A	A#/B ♭	B

<u>Cent</u>

A cent is a unit of measurement for pitch. A semitone is made up of 100 cents, making a full octave 1200 cents.

<u>Monochord</u>

A monochord is a musical instrument that has one string. The tone of this string is manipulated by changing the position of the instrument's moveable bridge, manipulating the tension that the string is placed under. [79]

Appendix B: Circuit Board Design

Section Two: Prototype Soldered Boards

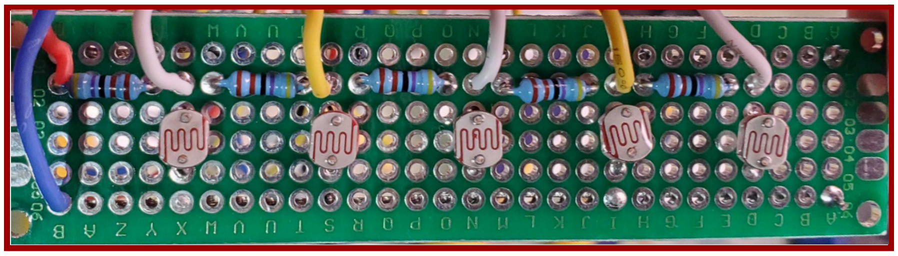


Figure 74: Photoresistor Soldered Prototype Board

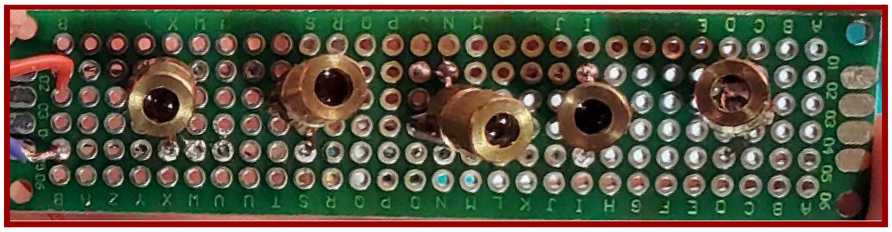


Figure 75: Laser LED Soldered Prototype Board

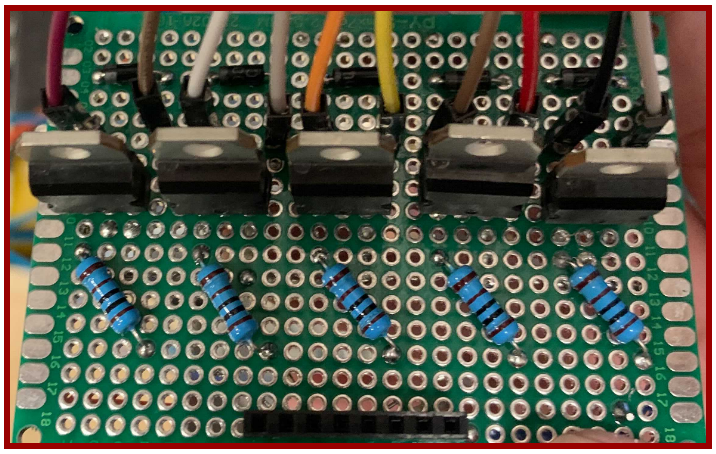


Figure 76: Solenoid Controller Soldered Prototype Board

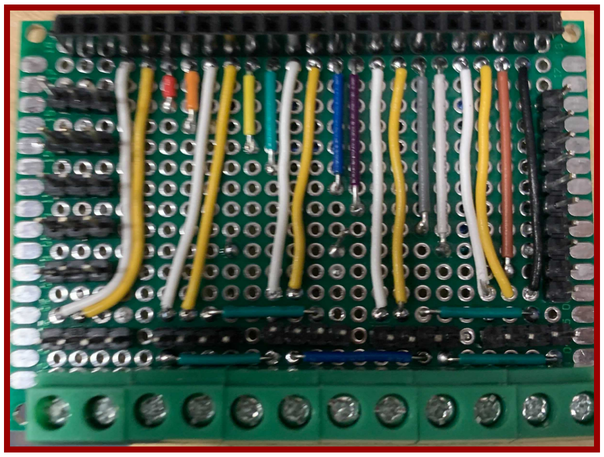


Figure 77: Power Connection Soldered Board

Section Three: Finalized Boards

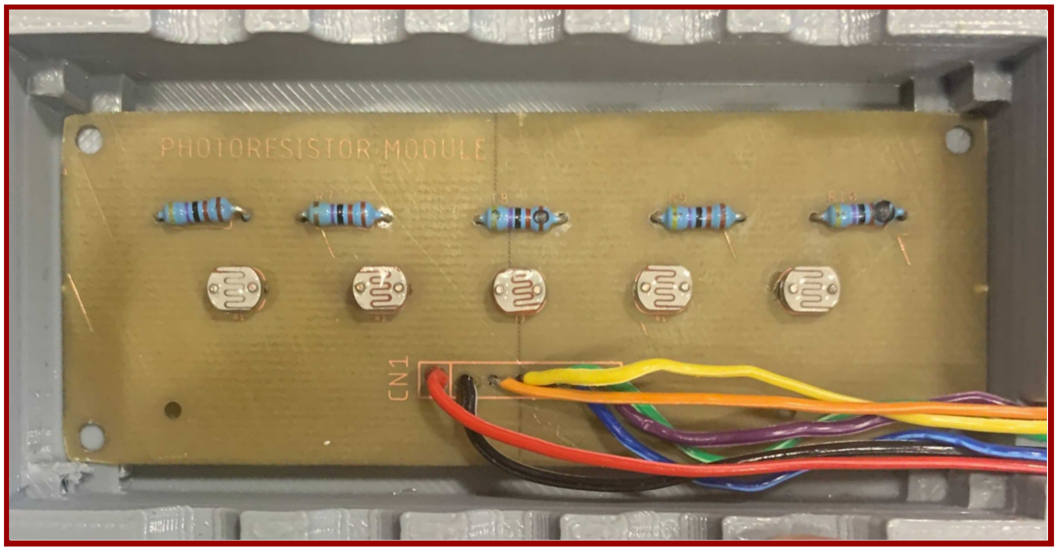


Figure 78: Photoresistor PCB Board

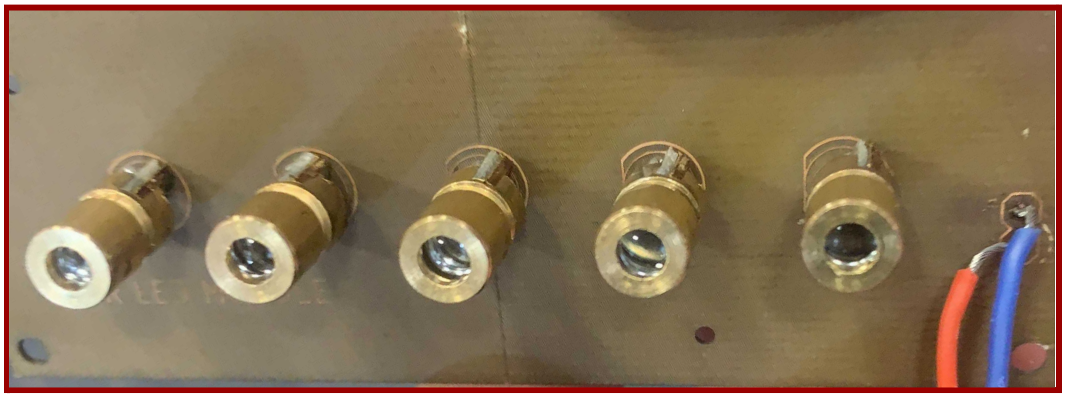


Figure 79: Laser LED PCB Board

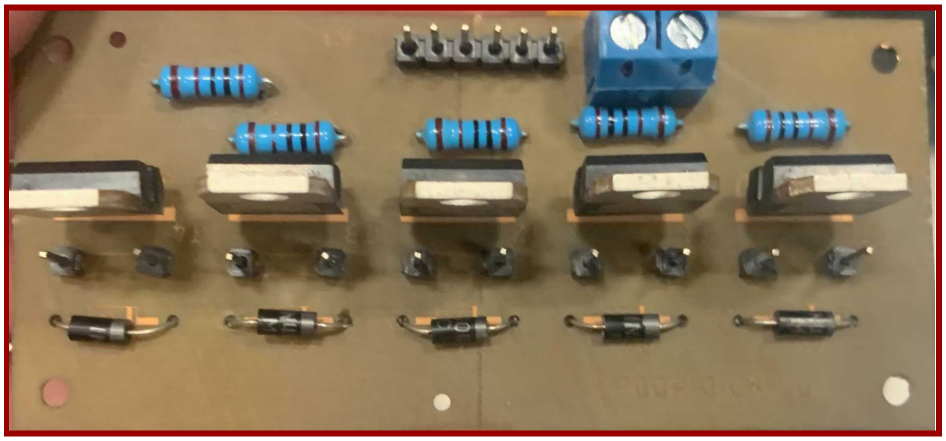


Figure 80: Solder Controller PCB Board

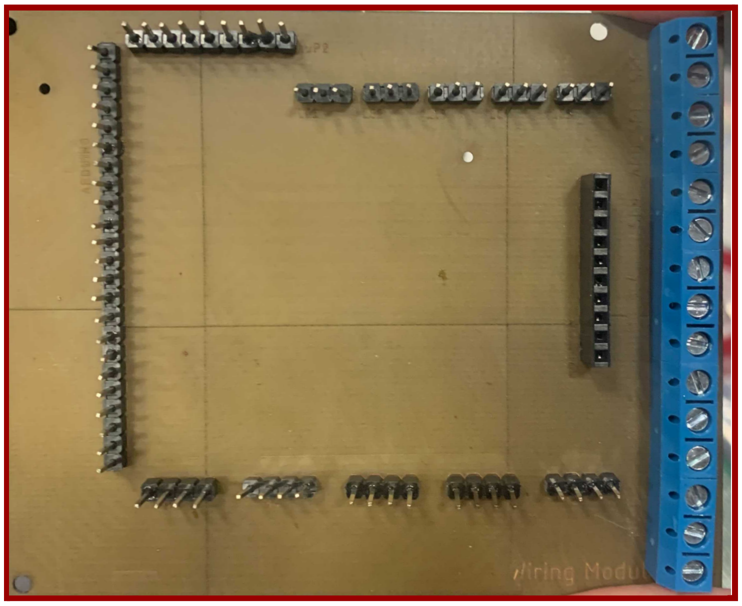


Figure 81: Power Connection PCB Board

Appendix C: Instrument Concept & Final Design Images

Section One: Instrument Conceptual Images

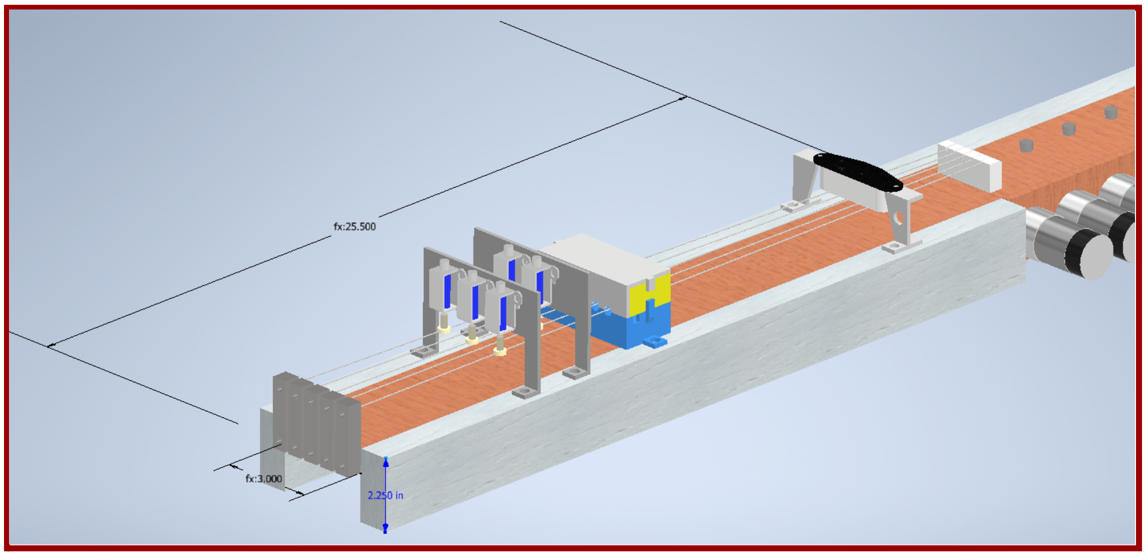


Figure 82: Orthographic View of Full Instrument

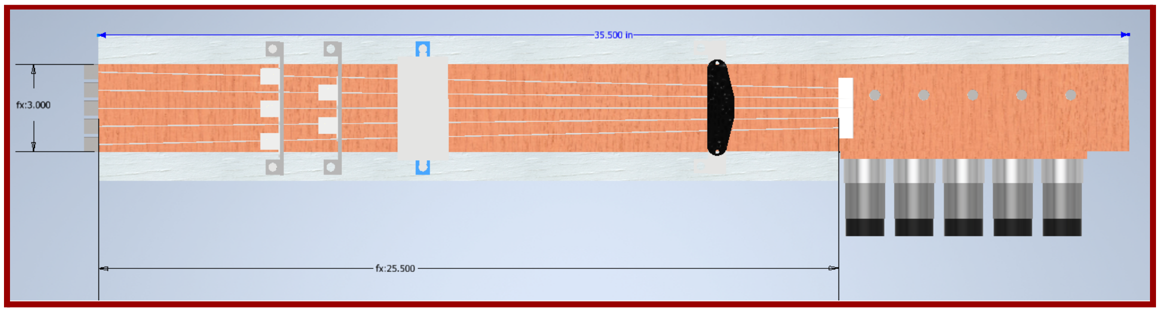


Figure 83: Top View of Full Instrument

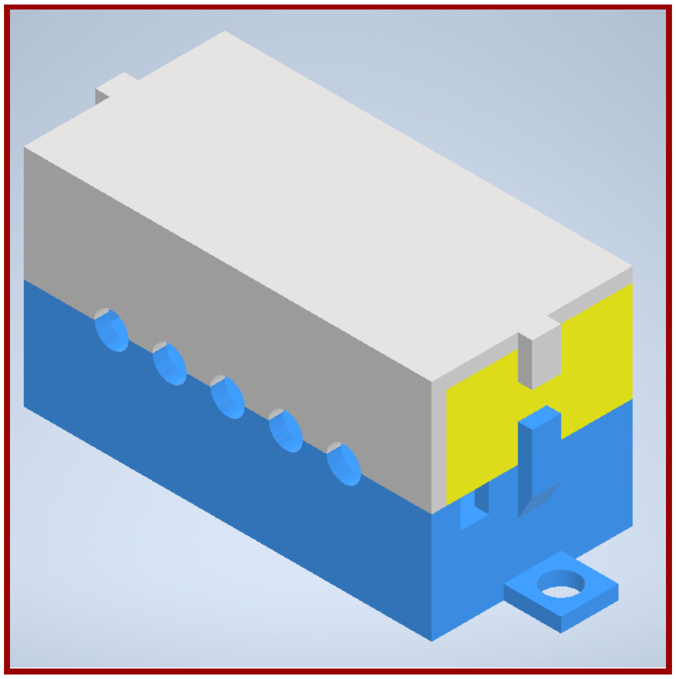


Figure 84: Photoresistor Module Case

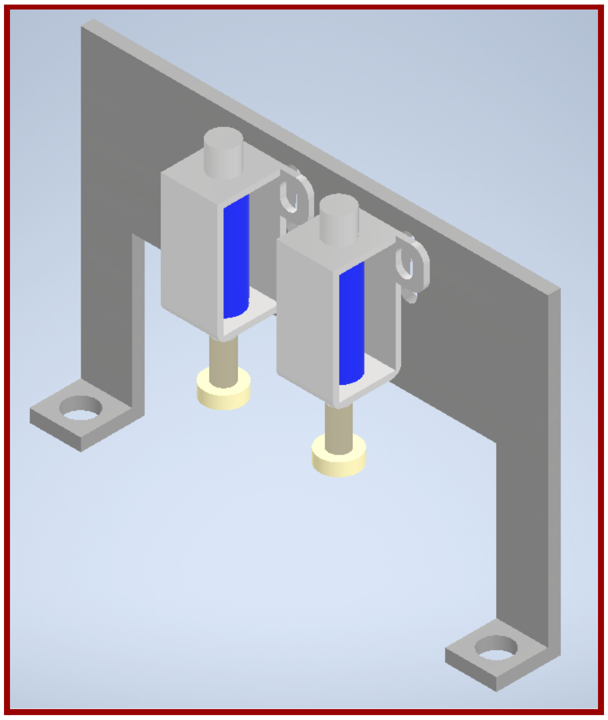


Figure 85: Solenoid Frame

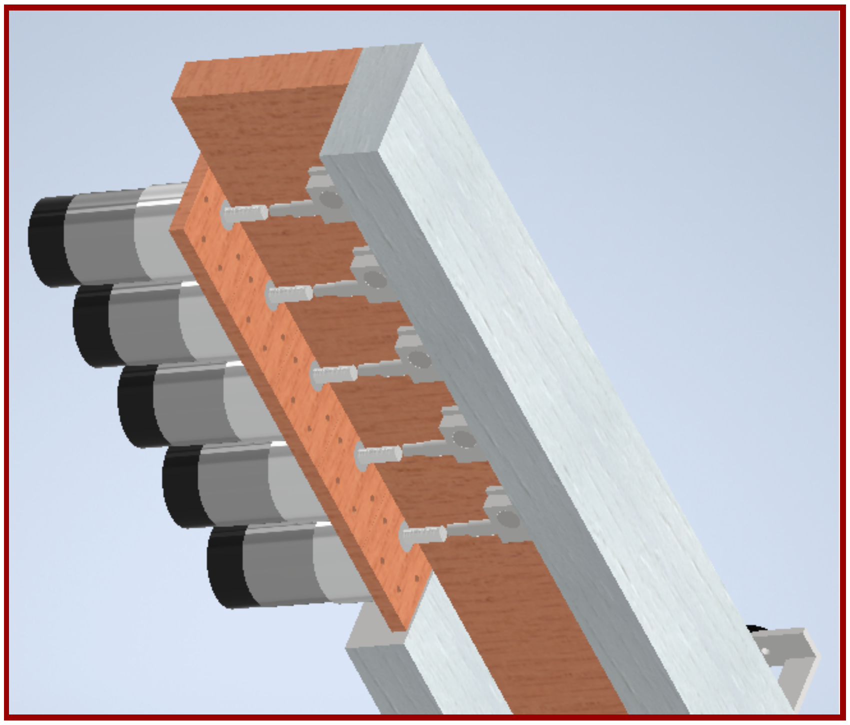


Figure 86: Underside View of Motor Module

Section Two: Final Instrument Images

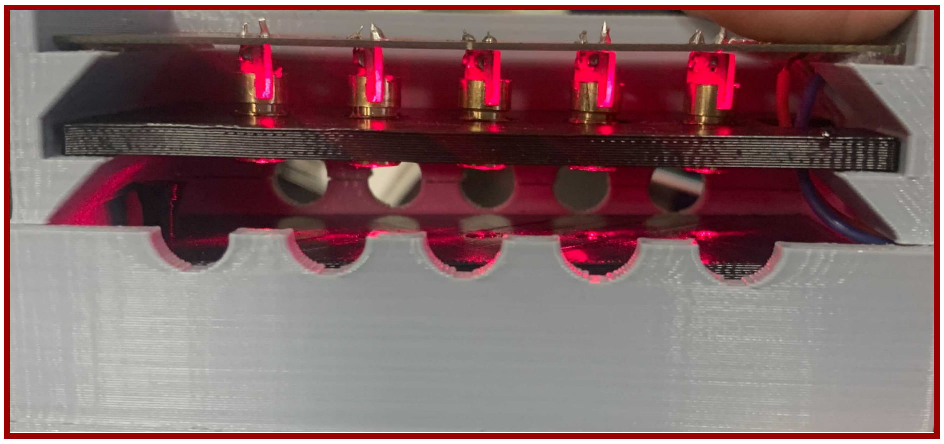


Figure 87: The Inside of the Photoresistor Casing

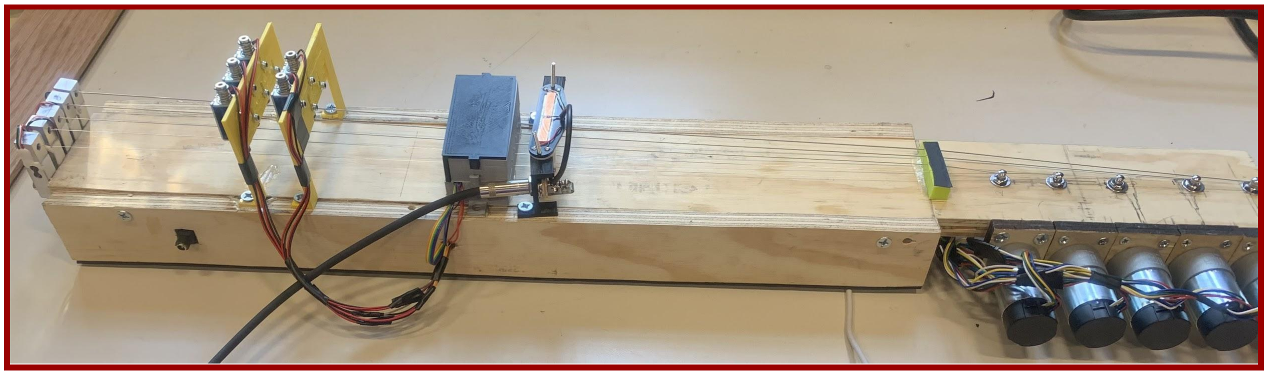


Figure 88: Final Prototype Side View

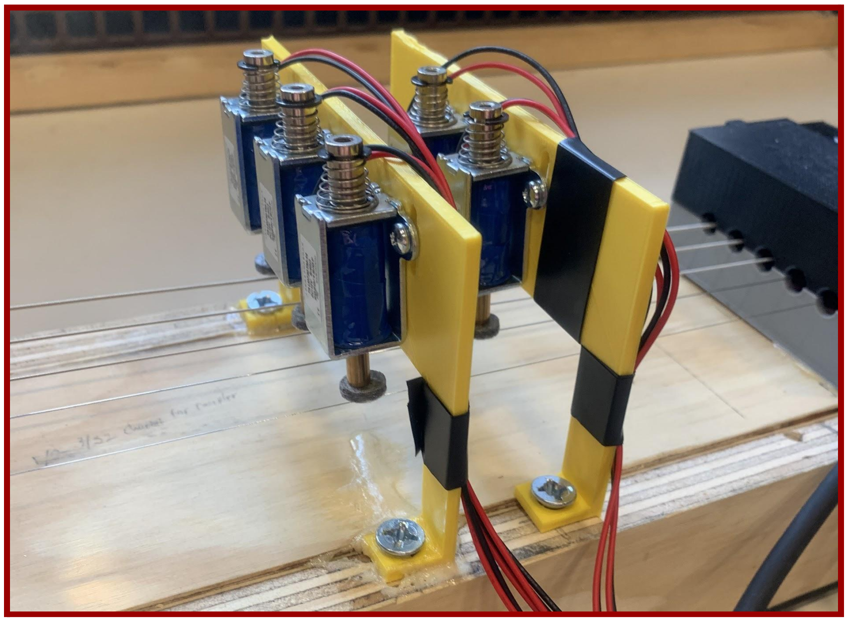


Figure 89: Final Solenoid Module

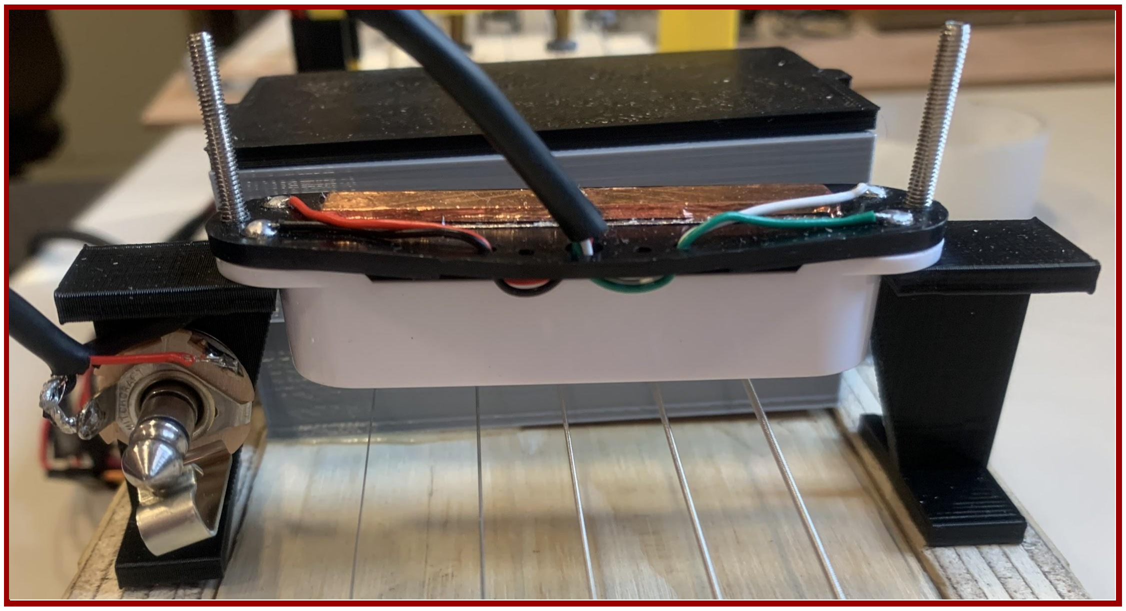


Figure 90: Final Mounted Pickup

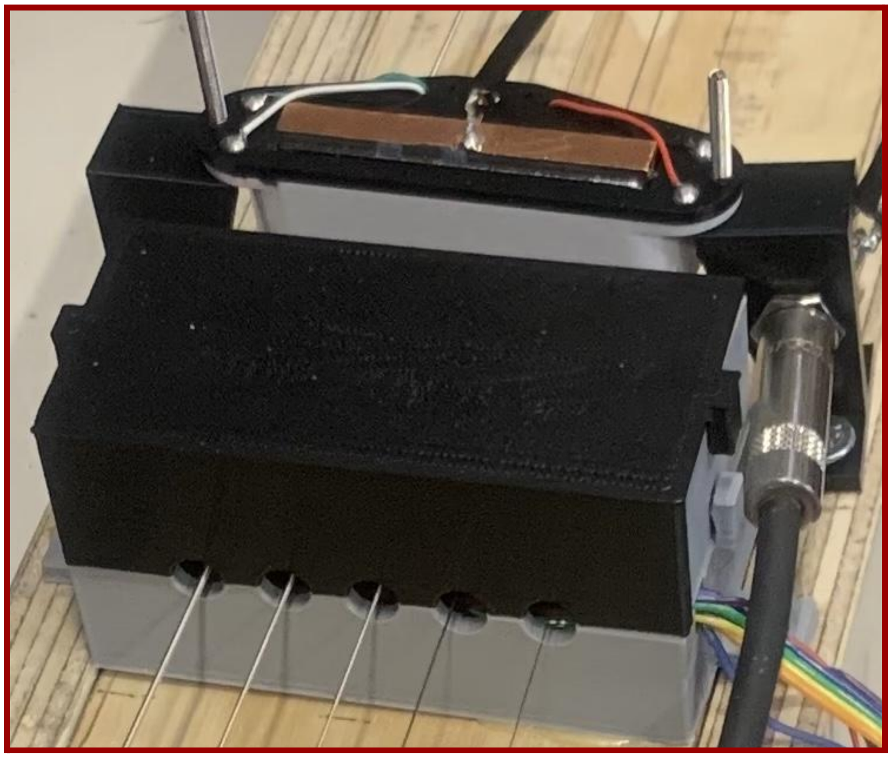


Figure 91: Final Prototype Casing

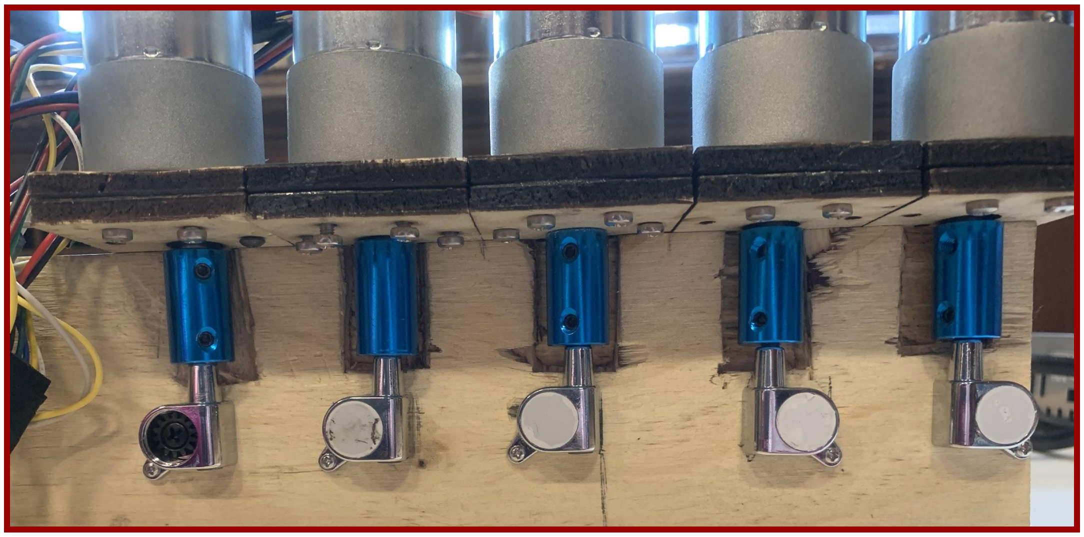


Figure 92: Final Prototype Motors and Tuning Machine

Appendix D: Bill of Materials

Bill 1: A & B Term

Strain Gauge/Load Cell				
Name	Qty	Unit Cost	Cost	Link
Micro Load Cell 50 kg	1	7.00	7.00	https://www.robotshop.com/en/micro-load-cell-50-kg.html
HX711 Load Cell Amplifier	1	9.95	9.95	https://www.sparkfun.com/products/13879
Hall Effect Sensor				
Name	Qty	Unit Cost	Cost	Link
VG481V1	2	$2.17	$4.34	https://www.digikey.com/en/products/detail/honeywell-sensing-and-productivity-solutions/VG481V1/10712141
MP101401	1	$4.73	$4.73	https://www.alliedelec.com/product/zf-electronics/mp101401/70207356/
55100-3H-02-D	1	$13.40	$13.40	https://www.digikey.com/en/products/detail/littelfuse-inc/55100-3H-02-D/764208
Linear Actuator				
Name	Qty	Unit Cost	Cost	Link
Linear Actuator 4"	1	36.56	36.56	https://auto-express.co/products/linear-actuator-4-heavy-duty-with-brackets-stroke-225-pound-max-lift-12-volt-dc
H-bridge for Arduino	1		62.98	https://www.progressiveautomations.com/products/lc-80
DC Motors				
Name	Qty	Unit Cost	Cost	Link
172:1 Metal Gearmotor	1	36.95	36.95	https://www.pololu.com/product/4808/specs
Standard Gearmotor - 6 RPM (3-12V)	1	24.95	24.95	https://www.sparkfun.com/products/12472
Hbridge	1	2.35	2.35	https://www.sparkfun.com/products/315
EM Pickup				
Name	Qty	Unit Cost	Cost	Link
40 Caliber Pickups	1	$45	$45	http://ubertar.com/hexaphonic/one_and_two_string_pickups.html
		TOTAL:	$248.21	

Bill 2: C & D Term

Name	Qty	Unit Cost	Total Cost	Link		
Motor						
19:1 Metal Gearmotor 37Dx68L mm 12V with 64 CPR Encoder	5	39.95	199.75	https://www.pololu.com/product/4751		
Motor 4mm to 6mm Coupling	5	5.99	29.95	https://www.amazon.com/uxcell-Coupling-L25xD14-Coupler-Connector/dp/B07PBBNQPK/ref=pd_di_sccai_1?pd_rd_w=FVLk0&pf_rd_p=c9443270-b914-4430-a90b-72e3e7e784e0&pf_rd_r=CZDAN19SZ16T8F0TMAN3&pd_rd_r=2d54b5cc-8c84-4924-8abc-4754007a2fb8&pd_rd_wg=lULgD&pd_rd_i=B07PBBNQPK&psc=1		
L-Bracket						
Pololu Stamped Aluminum L-Bracket Pair for 37D mm Metal Gearmotors	5	7.95	39.75	https://www.pololu.com/product/1084		
Guitar Pickup						
Dual Hot Rail Pickup	1	12.48	12.48	https://www.amazon.com/Musiclily-Output-Guitar-Single-Humbucker/dp/B07V329NSV/ref=sr_1_3?dchild=1&keywords=rail%2Bguitar%2Bpickup&qid=1612310158&sr=8-3&th=1#customerReviews		
Output Jack	1	6.99	6.99	https://www.amazon.com/Pure-Tone-Full-contact-Mounting-Hardware/dp/B01N94UUD0/ref=sr_1_2_sspa?dchild=1&keywords=output+jack+guitar&qid=1612311019&sr=8-2-spons&psc=1&smid=A3LIBWKT3TINIM&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUFKN1dCRk9IQjY4UksmZW5jcnlwdGVkSWQ9OTAyMjA4OTAxTFNXNDcxOTEwRlY2JmVuY3J5cHRlZEFkSWQ9OTAwMTExNDYxUVUxQ1pYQUNLSzlSJndpZGdldE5hbWU9c3BfYXRmJmFjdGlvbj1jbGlja1JlZGlyZWN0JmRvTm90TG9nQ2xpY2s9dHJ1ZQ==		
Strain Gauge/Load Cell						
Name	Qty	Unit Cost	Cost	Link		
Micro Load Cell 50 kg	5	7.00	35.00	https://www.robotshop.com/en/micro-load-cell-50-kg.html		
HX711 Load Cell Amplifier	5	9.95	49.75	https://www.sparkfun.com/products/13879		
M5 Bolts	5	0.8	4.00	https://www.homedepot.com/p/M5-0-8-x-20-mm-Class-8-8-Zinc-Plated-Hex-Bolt-2-Pack-801368/204273658		
M5 Locknuts	5	0.57	2.85	https://www.homedepot.com/p/2-Pieces-M5-0-8-Zinc-Plated-Metric-Nylon-Lock-Nut-803678/204274144		
Arduino						
Arduino Due	1	40.3	40.3	https://store.arduino.cc/usa/due		
Solenoids						
Push Pull Solenoid	5	12.3	61.5	https://www.amazon.com/Adafruit-Industries-412-Push-Pull-Solenoid/dp/B00R5CICX8/ref=sr_1_5?dchild=1&keywords=solenoid+push+pull&qid=1612975909&sr=8-5		
TIP102 BJT	5	8.99	44.95	https://www.amazon.com/Bridgold-TIP102-Bipolar-Darlington-Transistor/dp/B07TWMT52G/ref=sr_1_3?crid=29567VVTXZ0YU&dchild=1&keywords=tip102+npn+transistor&qid=1612979594&sprefix=tip102%2Caps%2C158&sr=8-3		
1N4004 Diode (150 Pack)	1	5.88	5.88	https://www.amazon.com/AUKENIEN-150Pcs-Rectifier-Electronic-Silicon/dp/B08RHKOG9N/ref=sr_1_1_sspa?crid=2FK8TVVKF7EAG&dchild=1&keywords=1n4004+diode&qid=1612979706&sprefix=1n4004%2Caps%2C159&sr=8-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzM1BKN01TSkRCTkxFJmVuY3J5cHRlZElkPUEwNjk1NDM3M0RERUQ1VU0xUTFNQyZlbmNyeXB0ZWRBZElkPUEwMDIyMzMwMVIzMVZaQjk0MFBWVyZ3aWRnZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=		

1K Ohm Resistor (100 Pack)	1	5.99	5.99	https://www.amazon.com/EDGELEC-Resistor-Tolerance-Multiple-Resistance/dp/B07QG1V4YL/ref=sr_1_1_sspa?crid=31E9OSKLK9G2U&dchild=1&keywords=1k+resistors&qid=1613415281&sprefix=1k+res%2Caps%2C161&sr=8-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEyUDdZT1pOS1o3SzlFJmVuY3J5cHRlZElkPUEwNTM4OTg4Mk5ONlNOTkxaWkdGQiZlbmNyeXB0ZWRBZElkPUEwNTQyMTU4MUJKMk1OMUhVR1o3MyZ3aWRnZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=		
Power Supply						
12V Power Supply	1	22.99	22.99	https://www.amazon.com/dp/B08764MPNT/ref=cm_sw_r_cp_api_glt_fabc_Y1ZSG32GRNAKG92DZZCY?_encoding=UTF8&psc=1		
Laser Diodes						
Laser Diode (10 Pack)	1	5.99	5.99	https://www.amazon.com/gp/product/B0166JFLES/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1		
Photoresistors						
Photoresistor (30 Pack)	1	4.75	4.75	https://www.amazon.com/gp/product/B01N7V536K/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&psc=1		
Load Cell & Amp						
Load Cell	2	7	14	https://www.robotshop.com/en/micro-load-cell-50-kg.html		
Load Cell Amp	2	9.95	19.9	https://www.digikey.com/en/products/detail/sparkfun-electronics/SEN-13879/6202732?utm_adgroup=Amplifiers&utm_source=google&utm_medium=cpc&utm_campaign=Shopping_Product_Sensors%2C%20Transducers&utm_term=&utm_content=Amplifiers&gclid=CjwKCAjwxuuCBhATEiwAIIIz0aNQNtX86Xfa38qlnB-SBIalZ3eRUOTgJAuXj1Ae02W-7SEKmGhBKxoCFPIQAvD_BwE		
Miscellaneous						
Jumper Wires	1	7.49	7.49	https://www.amazon.com/EDGELEC-Breadboard-1pin-1pin-Connector-Multicolored/dp/B07GCZVCGS/?_encoding=UTF8&pd_rd_w=DyeoU&pf_rd_p=49ff6d7e-521c-4ccb-9f0a-35346bfc72eb&pf_rd_r=V0DG2C8W1GE6TX0X1WP9&pd_rd_r=ed072d81-07cc-4869-a398-2dcc135c67c3&pd_rd_wg=8AtKx&ref_=pd_gw_ci_mcx_mr_hp_d&th=1		
Felt Pads	1	5.49	5.49	https://www.amazon.com/gp/product/B07JCJMP21?pf_rd_r=VJXKBHACYV7X3SGRCM4B&pf_rd_p=5ae2c7f8-e0c6-4f35-9071-dc3240e894a8&pd_rd_r=499ffa98-7110-445f-a882-54d8dae547e3&pd_rd_w=Z1Fsu&pd_rd_wg=5nt0P&ref_=pd_gw_unk		
Guitar Strings	2	14.25	28.5	https://www.amazon.com/Ernie-Ball-Super-Slinky-Nickel/dp/B00CAUYNCO/ref=sr_1_2?crid=5HX5WD5M5YS1&dchild=1&keywords=ernie+ball+electric+guitar+strings&qid=1616523792&sprefix=ernie+ball%2Caps%2C165&sr=8-2		
				TOTAL	$648.25	
				BILL 1 TOTAL	$248.21	
				GRAND TOTAL	$896.46	
				REMAINDER	$105.33	

Appendix E: Budget Analysis

For the entirety of this project, WPI allotted $250 per student, allowing a project budget of $1250. Almost all of this budget was necessary to complete the project, with only $105.33 left over in the end. Appendix D displays the bill of materials ordered throughout the length of the project. Only $248.21, about 20 percent of the budget, was spent in A and B term. This period of the project was dedicated to testing various potential components for the final design. Most of the budget was spent in C term, where the prototype was constructed. The most expensive part of the construction was the five metal gearmotors, totaling $199.75. Other various parts added to a total of $648.25 spent in C and D term.

The provided budget was sufficient for the completion of this project. Although the total came close to utilizing the entire budget, there is still a significant remainder now that the project has concluded. The budget was considered when ordering parts, and orders were made conservatively such that money wasn't wasted. There was never a part that couldn't be ordered due to budget constraint.

Appendix F: Code Examples

Please note - the entire code was submitted alongside this paper. Please refer to it as well.

Example One: Frequency Determined

This code determines the frequency using the photoresistor signal of all strings and then compares it to an array of previous values. If they are similar it is sent to the adjustment algorithm.

```
if (stopFlag) {
   for (int j = 0; j < STRING_MODULE_AMT; j++) {
     for (int i = 0; i < SAMPLE_AMT; i++) {
       vImag[i] = 0;
     } //hann OVER > nuttall OVER > hamming UNDER == blackman > rectangle
     // hann or blackman_harris
     FFT.Windowing(samples[j], SAMPLE_AMT, FFT_WIN_TYP_BLACKMAN_HARRIS,
FFT_FORWARD);

     FFT.Compute(samples[j], vImag, SAMPLE_AMT, FFT_FORWARD);

     FFT.ComplexToMagnitude(samples[j], vImag, SAMPLE_AMT);

     double x = FFT.MajorPeak(samples[j], SAMPLE_AMT, SAMPLE_PER_SEC);
     SerialUSB.print("module ");
     SerialUSB.print(j);
     SerialUSB.print(" freq ");
     SerialUSB.println(x, 6);
     SerialUSB.print("Expected ");
     float expect = stringModules[j]->getExpectedFrequency();
     SerialUSB.println(expect);
     SerialUSB.print("Sim threshold ");
     bool simThresh = abs(x - prevVals[j]) < SIMILARITY_THRESHOLD;
     SerialUSB.println(simThresh);
     if (simThresh) {
       activeStrings[j] = stringModules[j]->onPRDetect(x); // set to true if tension changes
     }
     prevVals[j] = x;
   }
   sampled = 0;
   stopFlag = false;
   for (int i = 0; i < STRING_MODULE_AMT; i++) {
     if (activeStrings[i]) {
       state = CALIBRATION;
     }
   }
  }
```

Example Two: Frequency Adjustment Algorithm

Algorithm receives an actual frequency. It first checks that this frequency is valid, then checks if the load cell is within the permitted margin of error - meaning that the system actually thinks it's tuned. If so, the tension is changed by multiplying the quotient of the expected and actual frequencies by the desired tension.

```
bool StringModule::onPRDetect(float act_freq) {
  bool fThresh = abs(getExpectedFrequency() - act_freq) <
LEGITIMATE_FREQ_THRESHOLD;
  SerialUSB.print("fThresh ");
  SerialUSB.println(fThresh);
  if (fThresh) { // throw away junk values
    float exp_tens = lookupTable.getLookupTension(lastTarget);
    float exp_freq = lookupTable.getLookupFrequency(lastTarget);
    loadVal = loadCell.getValue();
    float freq_div = exp_freq / act_freq;
    bool lThresh = abs(exp_tens - loadVal) < PERMITTED_LC_ERROR;
    SerialUSB.print("exp tens ");
    SerialUSB.print(exp_tens);
    SerialUSB.print(" act ");
    SerialUSB.println(loadVal);
    SerialUSB.print("lThresh ");
    SerialUSB.println(lThresh);
    if (lThresh) { // load cell must be close enough to intended in order to change it
      lookupTable.adjustLookupTable(lastTarget, freq_div * exp_tens);
      SerialUSB.print("Adjusted string module from ");
      SerialUSB.print(exp_tens);
      SerialUSB.print(" to ");
      SerialUSB.println(freq_div * exp_tens);
      return true;
    }
  }
  return false;
}
```

Example Three: Tuning Algorithm

If the setpoint changes, the algorithm determines the current load cell value and the encoder value derived from it. Then it resets some runtime variables. During the control loop, the motor uses PID to reach the setpoint, if the difference between expected and actual encoder value is below a threshold, it polls the load cell which is checked to be within the threshold of the desired tension value. If so, it increments successes and once two successes occur in a row, it returns true and shuts down the motors. If the load cell is not within error, it adjusts the desired encoder value by a factor proportionate to the difference between expected and actual tension.

```
bool StringModule::tuneStringAlt(float noteFrequency) {
  desiredValue = lookupTable.getLookupTension(noteFrequency);
  if (lastTarget != noteFrequency) {
    if (loadCell.isReady()) {
      loadVal = loadCell.getValue();

      setPoint = lookupTable.getLookupTableDataEnc(noteFrequency);
      SerialUSB.print("eDes ");
      SerialUSB.println(setPoint);
      SerialUSB.print("lc ");
      SerialUSB.println(loadVal);
      SerialUSB.print("tDes ");
      SerialUSB.println(desiredValue);
      lastTarget = noteFrequency;
      succCount = 0;
      time1 = millis();
    }
  } else {
    motor.adjustMotorPosition((double)setPoint);
    int diff = abs(motor.getEncoderPos() - setPoint);
    if (diff < SHIFT_THRESHOLD) {
      if (loadCell.isReady() && millis() - time1 > POLL_INT) {
        motor.off();
        loadVal = loadCell.getValue();
        time1 = millis();
        SerialUSB.print("lc ");
        SerialUSB.println(loadVal);
        float lcDiff = desiredValue - loadVal;
        if (abs(lcDiff) < PERMITTED_LC_ERROR) {
          succCount++;
          motor.off();
          if (succCount > SUCC_COUNT_EXP) {
            return true;
          }
          return false;
        } else {
          lookupTable.adjustLookupTableEnc(noteFrequency, motor.getEncoderPos() + lcDiff *
ENC_TENS_COEFF);
          setPoint = lookupTable.getLookupTableDataEnc((int)noteFrequency);
          SerialUSB.print("eDes ");
          SerialUSB.println(setPoint);
        if (succCount > 0) succCount--;
      }
    }
  }
}
  SerialUSB.print("epos ");
  SerialUSB.println(motor.getEncoderPos());
  SerialUSB.print("pow ");
  SerialUSB.println(motor.getOutput());
  if (succCount > SUCC_COUNT_EXP) {
    motor.off();
    return true;
  }
  return succCount > SUCC_COUNT_EXP;

}
```

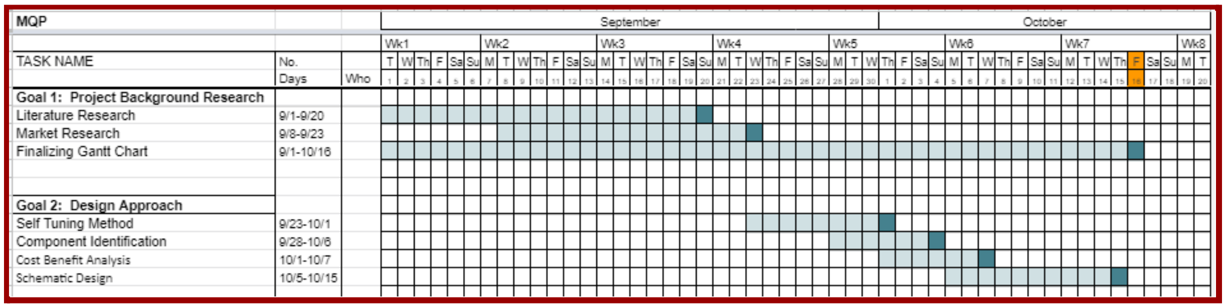
Example Four: PID

The device uses a custom-made PID - initially, a library was attempted for use, but it was based on using a constant sample interval which is not realistic when the number of operations change depending on how many strings are being simultaneously tuned. The I and D term contribute power proportionate to the time elapsed between iterations of the control loop. Since the output was limited to be 35% of the actual motor power in order to avoid breaking the string, the I term was not used until the error is below a certain error threshold because the maximum power is often reached.

```
bool Motor::adjustMotorPosition(double target) {
  //  SerialUSB.print("ab ");
  setpoint = target;

  input = enc->read();
  SerialUSB.print("setpoint ");
  SerialUSB.println(setpoint);
  double error = setpoint - input;

  double timeInterval = (millis() - timestamp) / 2.0;
  double dInput = (input - lastInput);
  // If encoder values are changing by a lot over a short period, don't use i term.
  if (dInput < MAX_DIFF) {
    outputSum += ki * error * timeInterval / 10;
  } else {
    outputSum /= 2;
  }

  outputSum = constrain(outputSum, -outputLimit, outputLimit);
  output = error * kp;
  output = constrain(output, -outputLimit, outputLimit);
  SerialUSB.print("outp ");
  SerialUSB.println(output);
  SerialUSB.print("a ");
  SerialUSB.println(((double)(timeInterval)));
  SerialUSB.print("b ");
  SerialUSB.println(((double)(kd * dInput)));
  double dCont = ((double)(kd * dInput)) / ((double)timeInterval);
  dCont = constrain(dCont, -outputLimit, outputLimit);
  if (abs(error) > 100) dCont /= 2;
  // Weight d term based on how much time has elapsed.
  output += outputSum - dCont;
  SerialUSB.print("outp2 ");
  SerialUSB.println(output);
  output = constrain(output, -outputLimit, outputLimit);
  if (!enabled) {
    outputSum = 0;
  }
  pwmOut(output);
  //  pwmOut(output);
  timestamp = millis();
  SerialUSB.print("epos ");
  SerialUSB.println(input);
  SerialUSB.print("pow ");
  SerialUSB.println(output);
  lastInput = input;
}
```

Appendix G: Timeline

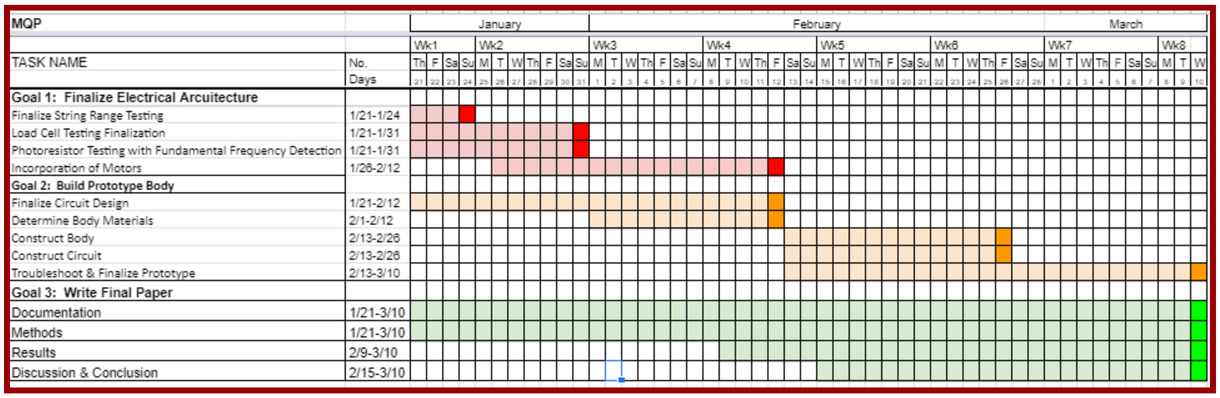
A & B Term



MQP

TASK NAME	No. Days	Who
Goal 1: Project Background Research		
Literature Research	9/1-9/20	
Market Research	9/8-9/23	
Finalizing Gantt Chart	9/1-10/16	
Goal 2: Design Approach		
Self Tuning Method	9/23-10/1	
Component Identification	9/28-10/6	
Cost Benefit Analysis	10/1-10/7	
Schematic Design	10/5-10/15	

Figure 93: A-Term Project Timeline

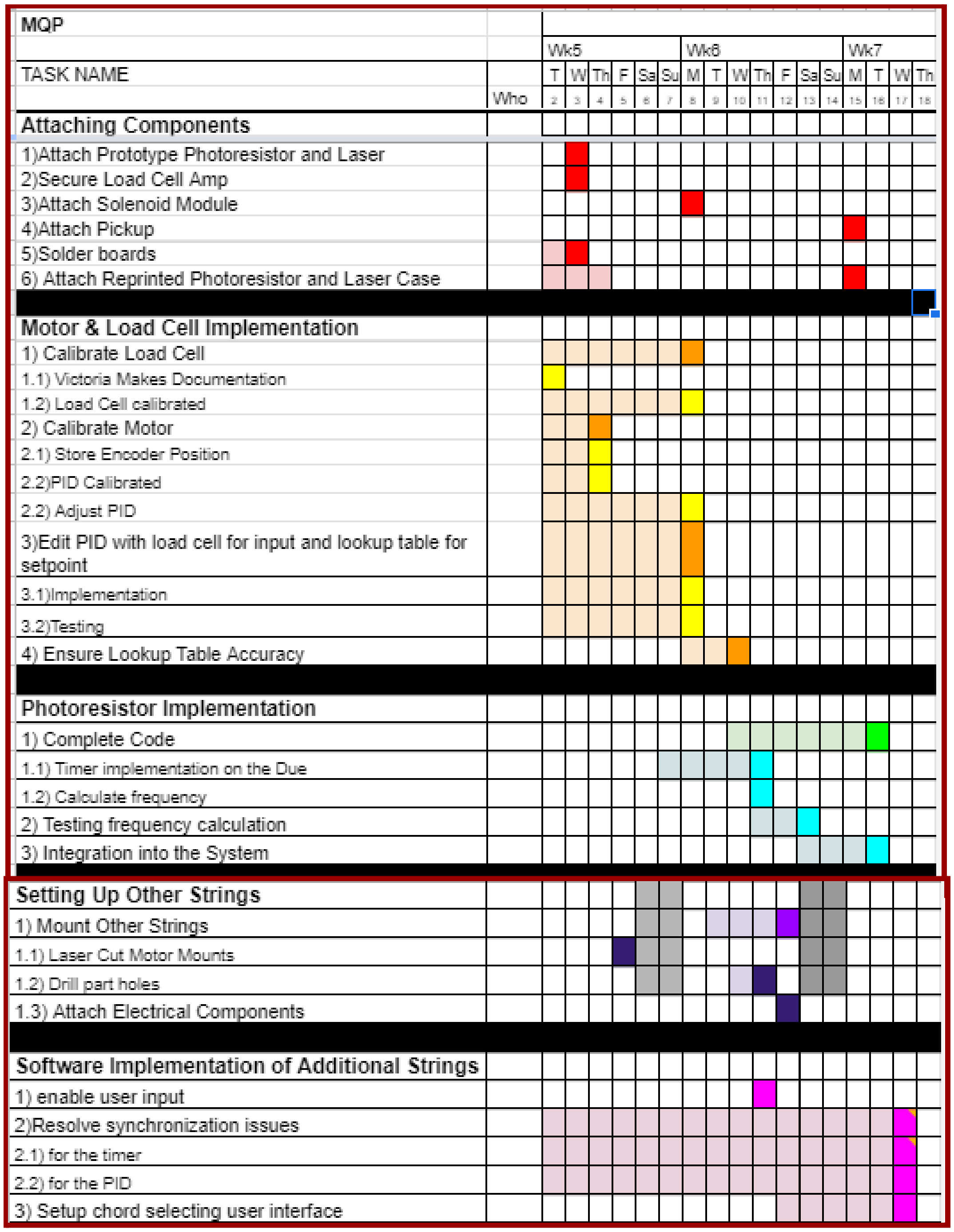


MQP

TASK NAME	No. Days
Goal 1: Finalize Electrical Arcuitecture	
Finalize String Range Testing	1/21-1/24
Load Cell Testing Finalization	1/21-1/31
Photoresistor Testing with Fundamental Frequency Detection	1/21-1/31
Incorporation of Motors	1/26-2/12
Goal 2: Build Prototype Body	
Finalize Circuit Design	1/21-2/12
Determine Body Materials	2/1-2/12
Construct Body	2/13-2/26
Construct Circuit	2/13-2/26
Troubleshoot & Finalize Prototype	2/13-3/10
Goal 3: Write Final Paper	
Documentation	1/21-3/10
Methods	1/21-3/10
Results	2/9-3/10
Discussion & Conclusion	2/15-3/10

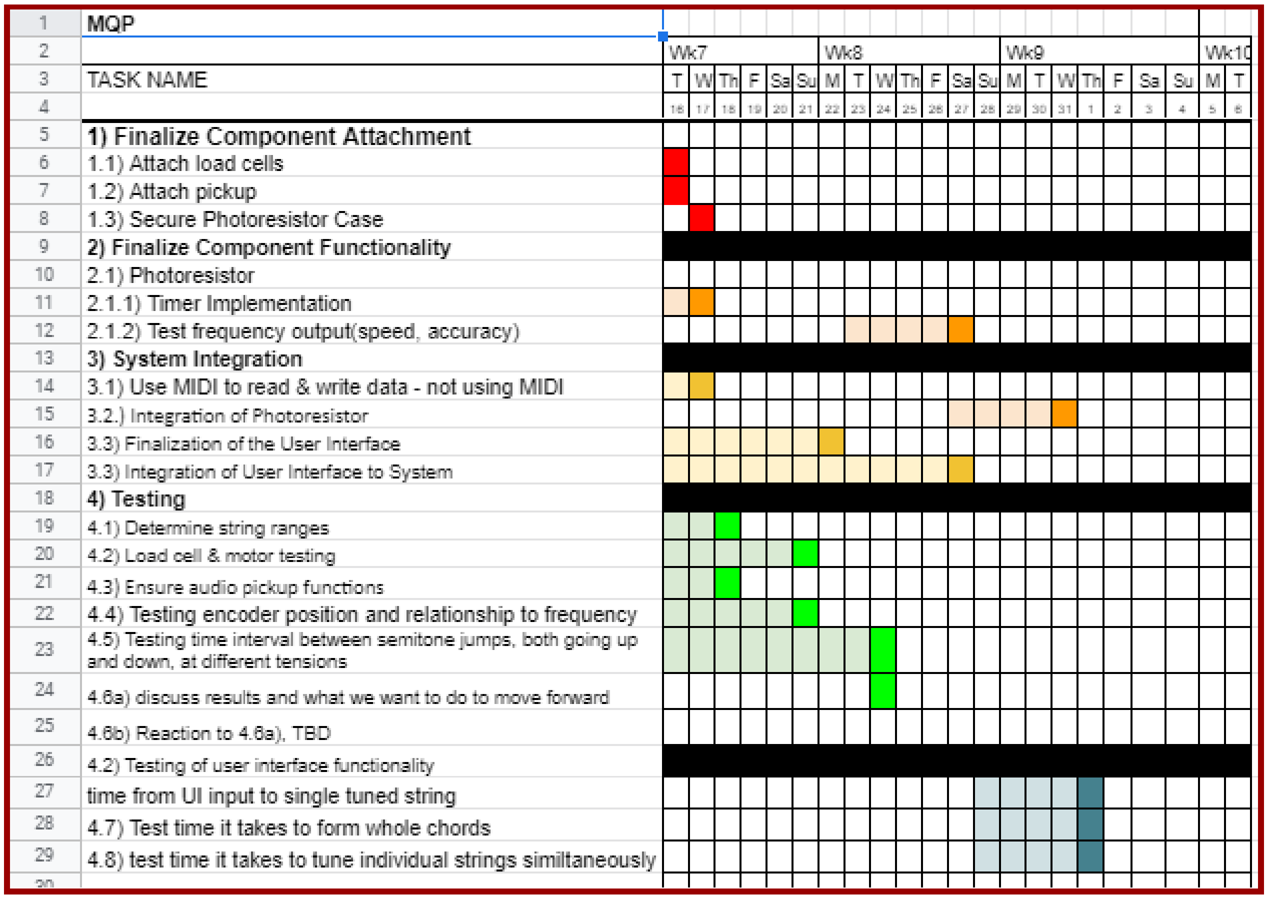
Figure 94: B-Term Project Timeline

Above are the Gannt charts created for A and B Terms in 2020. A Term consisted of mostly research, as well some initial component selection and schematic design. The goals of this timeline were achieved - the approach for the self tuning method, component selection for the sensor modules, and portions of the complete schematic were completed during this term. In B Term, testing of various components was conducted. The Gantt chart above, constructed during the first week of B term, shows that the original plan was to begin the construction of the prototype in B term. However, due to many unforeseen delays and Covid-19 restrictions, construction did not begin until C term. See Appendix I for more input on the impacts of Covid-19.



MQP																		
		Wk5						Wk6							Wk7			
TASK NAME		T	W	Th	F	Sa	Su	M	T	W	Th	F	Sa	Su	M	T	W	Th
	Who	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Attaching Components																		
1)Attach Prototype Photoresistor and Laser			🟥															
2)Secure Load Cell Amp			🟥															
3)Attach Solenoid Module								🟥										
4)Attach Pickup															🟥			
5)Solder boards		░	🟥															
6) Attach Reprinted Photoresistor and Laser Case		░	░	░											🟥			
Motor & Load Cell Implementation																		
1) Calibrate Load Cell		░	░	░	░	░	░	🟧										
1.1) Victoria Makes Documentation		🟨																
1.2) Load Cell calibrated		░	░	░	░	░	░	🟨										
2) Calibrate Motor		░	░	🟧														
2.1) Store Encoder Position		░	░	🟨														
2.2)PID Calibrated		░	░	🟨														
2.2) Adjust PID		░	░	░	░	░	░	🟨										
3)Edit PID with load cell for input and lookup table for setpoint		░	░	░	░	░	░	🟧										
3.1)Implementation		░	░	░	░	░	░	🟨										
3.2)Testing		░	░	░	░	░	░	🟨										
4) Ensure Lookup Table Accuracy								░	░	🟧								
Photoresistor Implementation																		
1) Complete Code										░	░	░	░	░	░	🟩		
1.1) Timer implementation on the Due							░	░	░	░	🟦							
1.2) Calculate frequency											🟦							
2) Testing frequency calculation											░	░	🟦					
3) Integration into the System													░	░	░	🟦		
Setting Up Other Strings						▒	▒						▓	▓				
1) Mount Other Strings						▒	▒		░	░	░	🟪	▓	▓				
1.1) Laser Cut Motor Mounts					⬛	▒	▒						▓	▓				
1.2) Drill part holes						▒	▒			░	⬛		▓	▓				
1.3) Attach Electrical Components												⬛						
Software Implementation of Additional Strings																		
1) enable user input											🟪							
2)Resolve synchronization issues		░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	🟪	
2.1) for the timer		░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	🟪	
2.2) for the PID		░	░	░	░	░	░	░	░	░	░	░	░	░	░	░	🟪	
3) Setup chord selecting user interface												░	░	░	░	░	🟪	

Figure 95: First C Term Timeline



1	MQP																						
2		Wk7						Wk8							Wk9							Wk10	
3	TASK NAME	T	W	Th	F	Sa	Su	M	T	W	Th	F	Sa	Su	M	T	W	Th	F	Sa	Su	M	T
4		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6
5	**1) Finalize Component Attachment**																						
6	1.1) Attach load cells	🟥																					
7	1.2) Attach pickup	🟥																					
8	1.3) Secure Photoresistor Case		🟥																				
9	**2) Finalize Component Functionality**	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
10	2.1) Photoresistor																						
11	2.1.1) Timer Implementation	░	🟧																				
12	2.1.2) Test frequency output(speed, accuracy)								░	░	░	░	🟧										
13	**3) System Integration**	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
14	3.1) Use MIDI to read & write data - not using MIDI	░	🟨																				
15	3.2.) Integration of Photoresistor												░	░	░	░	🟧						
16	3.3) Finalization of the User Interface	░	░	░	░	░	░	🟨															
17	3.3) Integration of User Interface to System	░	░	░	░	░	░	░	░	░	░	░	🟨										
18	**4) Testing**	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
19	4.1) Determine string ranges	░	░	🟩																			
20	4.2) Load cell & motor testing	░	░	░	░	░	🟩																
21	4.3) Ensure audio pickup functions	░	░	🟩																			
22	4.4) Testing encoder position and relationship to frequency	░	░	░	░	░	🟩																
23	4.5) Testing time interval between semitone jumps, both going up and down, at different tensions	░	░	░	░	░	░	░	░	🟩													
24	4.6a) discuss results and what we want to do to move forward									🟩													
25	4.6b) Reaction to 4.6a), TBD																						
26	4.2) Testing of user interface functionality	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
27	time from UI input to single tuned string													░	░	░	░	🟦					
28	4.7) Test time it takes to form whole chords													░	░	░	░	🟦					
29	4.8) test time it takes to tune individual strings similtaneously													░	░	░	░	🟦					

Figure 96: Second C-Term Timeline

Above are two Gantt charts developed for C Term. The initial plan was more ambitious than the latter, and attempted to complete more work in a small time frame than proved to be realistic. Unfortunately, due to COVID-19, complications arose during the production of the prototype, and the completion of the project was delayed. Several steps had to be postponed from the initial deadline of the first Gantt chart. The project officially concluded in D term, although the initial plan was to finish in C term. More details on the complications of COVID-19 can be found in Appendix H.

-

Appendix H: Effects of COVID-19

Working on a senior project such as this during the coronavirus pandemic was a significant challenge. During A term, all meetings were conducted remotely. Each team member was given a certain number of components or topics to research as the initial project proposal was created. The components that were deemed viable from the A term research were then divided up among the team members to be focused on during B term. Work was completed remotely during this term for various reasons including the comfort level of teammates. Since this made collaboration and communication more difficult, progress was slowed. Furthermore, aside from the restricted lab access in the first 2 weeks into C term, only three of the five members of the team had access to the lab during this period, which was when the majority of lab construction took place. The prototype finalization process is also slowed since two members could access the lab during D term.

Because of this severe division in labor and tasks, there were occasions when one team member would not be able to fully solve a problem on their own, and receiving help to solve those problems took much longer than it would have in a traditional collaborative lab setting. Asking for assistance in debugging code, writing code synchronously with someone to upload to a test rig on a different location, and simply reaching out for help when needed was more challenging than it may have been if this project was completed in a more traditional environment.

One difficult unit to integrate into the system due to remote work was the photoresistor. The teammate in charge of the electrical functionality was remote during C term. After the electrical components were finalized and functional, boards were soldered and mailed to the team on campus. Due to complications that arose due to the remote nature of work throughout the pandemic, the process of incorporating the working photoresistor module and interpreting the data coming from it was determined to be a low priority. The funcional boards were available for incorporation for over a month; however, functionality of the motors and load cell were rightfully prioritized, as the system can technically function properly without the incorporation of this unit. Furthermore, when the photoresistor case was constructed, those with access to the lab came to the conclusion that a piece of the photoresistor case was expendable without consulting those who were not in the lab, and therefore did not attach it. During the testing process, it came to light that this shield was integral to the function of the sensor - it makes the laser light beam thin enough that light does not bleed over the sides of the string, preventing the photoresistor from believing the string is constantly plucked. Fortunately, the shield was recovered and attached. However, this example displays a few of the difficulties in communication that were only furthered by the impacts of the pandemic. The COVID-19 pandemic further impacted this issue because not all team members were able to come into the lab - if the team member who had done the photoresistor testing was able to see the machine in person this year, this mistake would have been discovered far sooner and the issue may not have occurred. See Section 5 for the results of the incorporation of the photoresistor module. Please see the below photo of the at-home test rig that was utilized during C Term for the photoresistor by a remote team member.

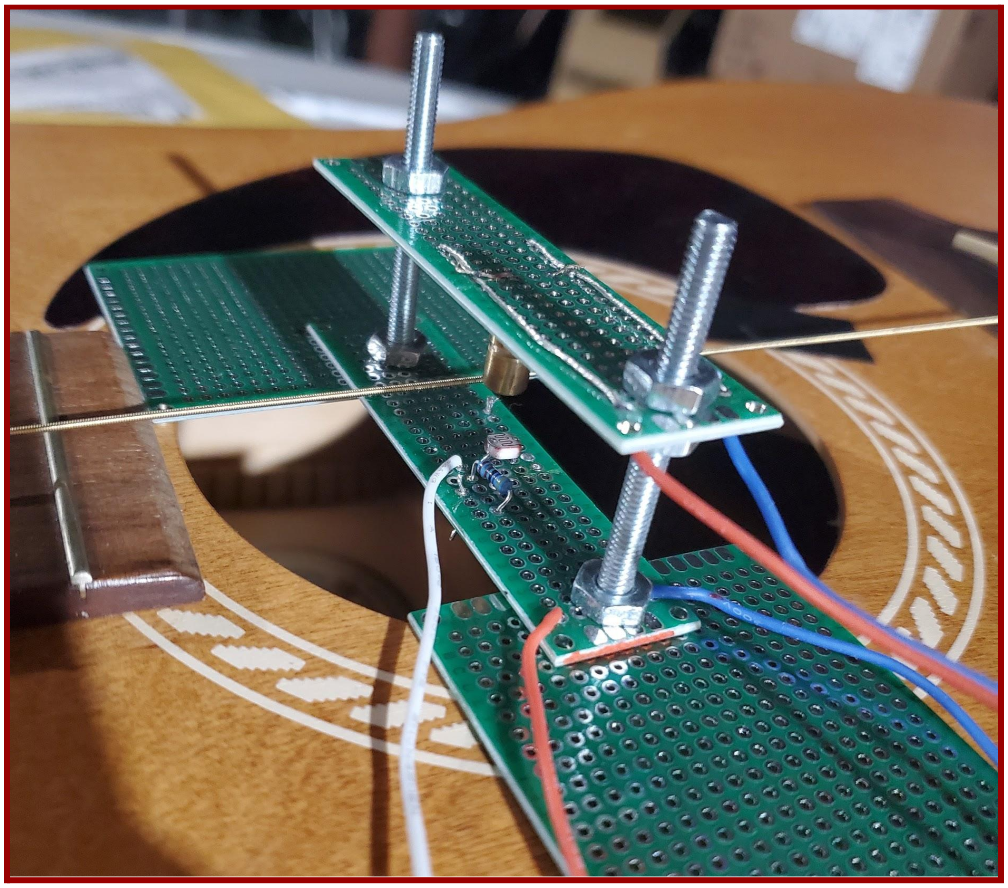


Figure 97: At-Home Test Rig for the Photoresistor

Another difficulty in the earlier stage of testing is the delay in DC motor procurement. Originally, previous calculations showed that the 172:1 gear reduction ratio DC motor, whose torque was 350 oz-in, was sufficient for operating the tuning machine. Although this motor satisfied the tuning accuracy requirement via PID, the speed of 56 RPM was too slow for the 100ms per semitone speed target. Due to the delay in the parts acquisition timeline, the motor was tested during the last few weeks of B term, thus causing those working on motor procurement to not realize this problem until the end of B term. This pushed back the DC motor finalization until the start of C term - the new motors had been ordered through the on-campus procurement process, and as no team members had access to campus during winter break, the parts could not be retrieved until then.

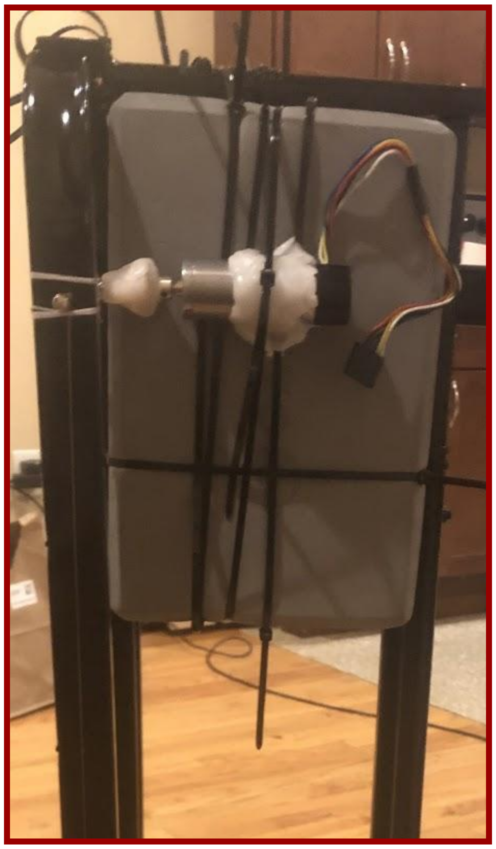


Figure 98: At-Home Test Rig for the DC Motor

Figure 99: At-Home Test Rig for the Load Cell

Appendix I: Authorship

Table 10: Authorship

Section Title	Written By	Edited By
Abstract	Garrett Smith, Fiona Doyle	Fiona Doyle
Acknowledgements	Fiona Doyle	Victoria Thornton
Table of Images	Fiona Doyle	Ryan Hennigan
Table of Tables	Fiona Doyle	Vien Phuong T. Le, Ryan Hennigan
Table of Equations	Fiona Doyle	Ryan Hennigan
1)Introduction	Victoria Thornton, Fiona Doyle	Fiona Doyle, Vien Phuong T. Le, Victoria Thornton
2)Background		
2.1) Physical Properties of String Instruments	Fiona Doyle	Vien Phuong T. Le, Victoria Thornton
2.2) Current Autonomous Tuners on Market	Fiona Doyle	Vien Phuong T. Le
2.2.1) The Roadie 2 Automatic Guitar Tuner	Fiona Doyle	Vien Phuong T. Le
2.2.2) TronicalTune PLUS	Fiona Doyle	Vien Phuong T. Le
2.3) Current Autonomous Tuning Instruments	Fiona Doyle	Vien Phuong T. Le
2.3.1) Electro-Thermal Tuning	Fiona Doyle	Vien Phuong T. Le, Victoria Thornton
2.3.2) Milwaukee Servoelectric Guitar	Fiona Doyle	Vien Phuong T. Le

2.3.3) Cyther V3	Fiona Doyle	Vien Phuong T. Le
2.3.4) STARI	Fiona Doyle	Vien Phuong T. Le, Victoria Thornton
2.3.5) Swivel 2	Fiona Doyle	Victoria Thornton
2.3.6) Automatic Pitch Processing for Electric Stringed Instruments	Fiona Doyle	Victoria Thornton
2.4) Fundamental Frequency Determination & Tuning Algorithms	Garrett Smith	Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le
3) Design Requirements	Garrett Smith, Fiona Doyle	Ryan Hennigan, Vien Phuong T. Le, Victoria Thornton
4) Overall System Design	Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le, Victoria Thornton	Vien Phuong T. L, Fiona Doyle
4.1) Operation Process	Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le, Victoria Thornton, Garrett Smith	Fiona Doyle
4.2) Software Design	Garrett Smith, Victoria Thornton	Fiona Doyle
5) Design Process	Fiona Doyle	Vien Phuong T. Le
4.3) Component Consideration & Procurement	Fiona Doyle	Vien Phuong T. Le
4.3.1.1) Hall Effect Sensor	Vien Phuong T. Le	Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le
4.3.1.2) Electromagnetic Coils	Ryan Hennigan	Fiona Doyle, Vien Phuong T. Le
4.3.1.4) Photoresistor	Fiona Doyle	Vien Phuong T. Le

4.3.1.5) Position Sensitive Detector	Fiona Doyle	Vien Phuong T. Le
4.3.1.6) Piezoelectric Sensor	Vien Phuong T. Le	Fiona Doyle, Vien Phuong T. Le, Victoria Thornton
4.3.2.1) Load Cell	Victoria Thornton	Fiona Doyle, Vien Phuong T. Le
4.3.3) Comparison	Fiona Doyle, Victoria Thornton	Vien Phuong T. Le
4.3.4) Motors	Vien Phuong T. Le	Fiona Doyle, Vien Phuong T. Le
4.3.4.1) DC Motor	Vien Phuong T. Le, Victoria Thornton	Fiona Doyle, Vien Phuong T. Le, Victoria Thornton
4.3.4.2) Linear Actuator	Vien Phuong T. Le, Victoria Thornton	Fiona Doyle, Vien Phuong T. Le, Victoria Thornton
4.3.4.3) Stepper Motor	Vien Phuong T. Le	Fiona Doyle, Vien Phuong T. Le
4.3.4.4) Servo Motor	Vien Phuong T. Le	Fiona Doyle, Vien Phuong T. Le
4.5.4.5) Motor Comparison	Vien Phuong T. Le	Fiona Doyle, Vien Phuong T. Le, Victoria Thornton
4.3.5) Motor Control	Vien Phuong T. Le	Fiona Doyle, Garrett Smith
4.3.7) Microcontroller	Fiona Doyle	Ryan Hennigan, Vien Phuong T. Le
4.3.8) Power Consumption	Fiona Doyle	Vien Phuong T. Le
4.4) System Operation Process Design	Garrett Smith, Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le, Victoria Thornton	Fiona Doyle, Vien Phuong T. Le
4.4.1) Initialization of the Device	Garrett Smith, Ryan Hennigan, Vien Phuong T. Le, Victoria Thornton	Fiona Doyle, Vien Phuong T. Le

4.4.2) Load Cell Tuning Process	Garrett Smith, Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le, Victoria Thornton	Fiona Doyle, Vien Phuong T. Le
4.4.3) Photoresistor Tuning Process	Garrett Smith, Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le, Victoria Thornton	Fiona Doyle, Vien Phuong T. Le
4.4.4) Custom PID Development	Garrett Smith	Fiona Doyle
4.5) User Interface Design	Victoria Thornton	Fiona Doyle
4.6) Instrument Body Design	Ryan Hennigan, Fiona Doyle, Vien Phuong T. Le	Fiona Doyle, Vien Phuong T. Le
4.7) Electrical Architecture Design	Vien Phuong T. Le, Ryan Hennigan	Fiona Doyle
5) System Construction & Evaluation	Fiona Doyle	Vien Phuong T. Le
5.1.1) Physical System Results	Ryan Hennigan, Vien Phuong T. Le	Fiona Doyle, Vien Phuong T. Le
5.1.2) Electrical System Results	Ryan Hennigan, Vien Phuong T. Le	Fiona Doyle, Vien Phuong T. Le
5.1.3) Software Design Results	Garrett Smith	Victoria Thornton
5.1.4) Challenges Uncovered During Construction	Ryan Hennigan, Vien Phuong T. Le	Fiona Doyle, Vien Phuong T. Le
5.2) Testing Methodology	Fiona Doyle	
5.2.1) Pitch Range Determination	Fiona Doyle	Vien Phuong T. Le
5.2.2.1) Load Cell Testing, Single String	Fiona Doyle	Fiona Doyle, Vien Phuong T. Le

5.2.2.2) Load Cell Testing, Non-Tuning Strings Tensioned	Fiona Doyle	Fiona Doyle, Vien Phuong T. Le
5.2.2.3) Photoresistor Testing	Fiona Doyle	Vien Phuong T. Le
5.2.2.4) Encoder on a Single String	Ryan Hennigan	Fiona Doyle, Victoria Thornton
5.2.3.1) Motor Speed	Fiona Doyle	Ryan Hennigan, Victoria Thornton
5.2.3.2) Load Cell Amplifier Polling Rate Examination	Fiona Doyle, Vien Phuong T. Le	Vien Phuong T. Le
5.2.3.3) Revolutions and Encoder Relationship	Fiona Doyle	Ryan Hennigan
5.2.3.4) Revolution and Semitone Change Relationship	Vien Phuong T. Le	Fiona Doyle, Garrett Smith
5.2.4) Tuning Operation Latency	Victoria Thornton	Fiona Doyle, Garrett Smith
5.3.1) Pitch Range Determination	Fiona Doyle	Vien Phuong T. Le
5.3.2.1) Load Cell Testing, Single String	Vien Phuong T. Le	Fiona Doyle
5.2.2.2) Load Cell Testing, Non-Tuning Strings Tensioned	Vien Phuong T. Le	Fiona Doyle
5.3.2.3) Photoresistor Testing	Garrett Smith	Fiona Doyle
5.3.2.4) Encoder on a Single String	Ryan Hennigan	Fiona Doyle
5.3.3.1) Motor Speed	Victoria Thornton	Fiona Doyle

5.3.3.2) Load Cell Amplifier Polling Rate Examination	Fiona Doyle	Fiona Doyle
5.3.3.3) Revolutions and Encoder Relationship	Ryan Hennigan	Fiona Doyle
5.3.3.4) Revolution and Semitone Change Relationship	Vien Phuong T. Le	Fiona Doyle, Garrett Smith
5.3.4) Tuning Operation Latency	Victoria Thornton	Fiona Doyle
5.4) Analysis of Results	Garrett Smith, Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le, Victoria Thornton	Fiona Doyle
5.5) Design Requirement Analysis	Garrett Smith, Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le, Victoria Thornton	Vien Phuong T. Le
6) Market Impact Analysis	Fiona Doyle	Victoria Thornton
7.1.1) Instrument Recommendations	Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le	Fiona Doyle, Vien Phuong T. Le, Victoria Thornton
7.1.2) Testing Recommendations	Fiona Doyle	Vien Phuong T. Le, Victoria Thornton
7.2) Additions	Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le	Fiona Doyle, Vien Phuong T. Le
Appendix A: Relevant Musical Definitions	Fiona Doyle, Ryan Hennigan	Vien Phuong T. Le, Fiona Doyle
Appendix B: Circuit Board Design	Vien Phuong T. Le	Vien Phuong T. Le, Fiona Doyle
Appendix C: Instrument Concept & Final Design Images	Ryan Hennigan, Vien Phuong T. Le	Vien Phuong T. Le, Fiona Doyle

Appendix D: Bill of Materials	Ryan Hennigan	Vien Phuong T. Le, Fiona Doyle
Appendix E: Budget Analysis	Ryan Hennigan	Vien Phuong T. Le, Fiona Doyle
Appendix F: Code Examples	Garrett Smith	Fiona Doyle, Victoria Thornton
Appendix G: Timeline	Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le, Victoria Thornton, Garrett Smith	Vien Phuong T. Le, Fiona Doyle
Appendix H: Effects of COVID-19	Fiona Doyle, Ryan Hennigan, Vien Phuong T. Le, Victoria Thornton	Vien Phuong T. Le, Fiona Doyle

Bibliography

[1] Nave, R. (n.d). Vibrating String. Retrieved from http://hyperphysics.phy-astr.gsu.edu/hbase/Waves/string.html

[2] N.a. (n.d). The Structure of the Piano Design of the Strings Enriches the Sound. Retrieved from https://www.yamaha.com/en/musical_instrument_guide/piano/mechanism/mechanism004.html

[3] Living Pianos. (2018, November 13). Yamaha Model GC1 Baby Grand Piano. Retrieved from https://livingpianos.com/pianos/yamaha-model-gc1-baby-grand-piano-for-sale-6081089/

[4] Passy. (2013, September 01). Guitar Mathematics. Retrieved from http://passyworldofmathematics.com/guitar-mathematics/

[5] N.a. (2019, July 18). How to Use a Capo: Guitar Capo Chart. Retrieved from https://yousician.com/blog/what-is-a-guitar-capo

[6] N.a. (2020, April 17). Irving Sloane Bass Tuning Machines. Retrieved from https://uptonbass.com/shop/irving-sloane-bass-tuning-machines/

[7] Harper, D. (2020, January). *Why Do Guitars Go Out Of Tune?* https://www.happynewguitarday.com/why-do-guitars-go-out-of-tune/

[8] Roadie Automatic Guitar Tuner. (n.d.). Retrieved September 8, 2020, from https://www.roadiemusic.com/roadie2

[9] TonicalTune PLUS (n.d.). Retrieved September 8, 2020, from https://www.tronicaltune.com/product-category/tronicaltune-plus/?v=3a52f3c22ed6

[10] Gibson's woes continue with $50 million robot tuner lawsuit: Guitar.com: All Things Guitar. (2018, November 13). Retrieved September 8, 2020, from https://guitar.com/news/gibsons-woes-continue-with-50-million-robot-tuner-lawsuit/

[11] Gilmore, D. A. (2003). *U.S. Patent No. US 6,559,369 B1*. Washington, DC: U.S. Patent and Trademark Office.

[12] Ridden, P. (2015, May 02). A closer look at Don Gilmore's self-tuning piano system. Retrieved from https://newatlas.com/gilmore-self-tuning-piano-system/21425/

[13] Mone, G. (2009, March 27). Homemade Guitar Hero. Retrieved from https://www.popsci.com/entertainment-amp-gaming/article/2009-03/homemade-guitar-hero/

[14] Baxter, K. (n.d.). The Milwaukee Servoelectric Guitar Project. Retrieved from http://servoelectricguitar.com/oldindex.php

[15] Prihar, E. (2017). *Dynamically Tuning String Instrument*. Worcester Polytechnic Institute. Retrieved from

[16] S. Trail, G. Tzanetakis, L. Jenkins, M. Cheng, D. MacConnell and P. Driessen, "STARI: A self tuning auto-monochord robotic instrument," *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Victoria, BC, Canada, 2013, pp. 405-409, doi: 10.1109/PACRIM.2013.6625511.

[17] Murphy, J., Kapur, A., & Carnegie, D. A. (2019). Swivel 2: A Systems Overview and Evaluation of a New Mechatronic Slide Guitar. Published by New Zealand School of Music Victoria University of Wellington, Wellington, New Zealand. Retrieved from https://docplayer.net/100874474-Swivel-2-a-systems-overview-and-evaluation-of-a-new-mechatronic-slide-guitar.html

[18] Stevenson, A. J. (2004). *U.S. Patent No. US 2004/0187673 A1*. Washington, DC: U.S. Patent and Trademark Office.

[19] N. a. *Digital Frequency Meter*. (2018, July). Electrical 4 U. Retrieved from https://www.electrical4u.com/digital-frequency-meter/

[20] *What is Schmitt Trigger | How It Works*. (n.d.). https://howtomechatronics.com/how-it-works/electrical-engineering/schmitt-trigger/

[21] What is Schmitt Trigger, How it Works and Applications. (2019, May 7). Retrieved from https://components101.com/articles/schmitt-trigger-introduction-working-applications

[22] *Synchronous Counter*. (n.d.). Retrieved October 13, 2020, from https://www.electronics-tutorials.ws/counter/count_3.html

[23] *Digital Frequency Meter*. EEEGUIDE.COM. (2018, January 17). https://www.eeeguide.com/digital-frequency-meter/.

[24] M. C. Abrams, G. C. Toon, and R. A. Schindler, "Practical example of the correction of Fourier-transform spectra for detector nonlinearity," Appl. Opt. 33, 6307-6314 (1994)

[25] Patel, A. (n.d.). *EasyFFT: Frequency transform for Arduino*. Arduino. https://create.arduino.cc/projecthub/abhilashpatel121/easyfft-fast-fourier-transform-fft-for-arduino-9d2677

[26] Aasvik, M. (2017, August 10). *What Is FFT and How Can You Implement It on an Arduino?* Norwegian Creations. https://www.norwegiancreations.com/2017/08/what-is-fft-and-how-can-you-implement-it-on-an-arduino/

[27] Dallas, G. (2014, May). *Signal Processing, Fourier Transforms and Heisenberg*. Georgemdallas. https://georgemdallas.wordpress.com/2014/05/14/wavelets-4-dummies-signal-processing-fourier-transforms-and-heisenberg/

[28]Guzman, J. (2018, March). *Fast and Efficient Pitch Detection: Bitstream Autocorrelation*. Cyfi Research. https://www.cycfi.com/2018/03/fast-and-efficient-pitch-detection-bitstream-autocorrelation/

[29] Vigil, K., & Graham, J. (2008). Octave Identification System for the Guitar. Retrieved from https://www.lcps.org/cms/lib4/va01000195/centricity/domain/2608/octaveid.pdf

[30] "EKU Health and Safety." *EKU School of Music*, music.eku.edu/sites/music.eku.edu/files/ekuhealthandsafety.pdf.

[31] Schmidt, Jesper, et al. "Sound Exposure of Symphony Orchestra Musicians." *The Annals of Occupational Hygiene*, 2011, doi:10.1093/annhyg/mer055.

[32] Pujol, R. (2018, June). *Human Auditory Range*. Cochlea. http://www.cochlea.org/en/hear/human-auditory-range

[33] Harmful Noise Levels. (n.d.). Retrieved from https://www.uofmhealth.org/health-library/tf4173

[34] D.B. Loeffler, "Instrument Timbres and Pitch Estimation in Polyphonic Music Archived 2007-12-18 at the Wayback Machine" Master's Thesis, Department of Electrical and Computer Engineering, Georgia Tech. April (2006), from https://web.archive.org/web/20071218232401/http://etd.gatech.edu/theses/available/etd-04102006-142310/

[35] J. M. Geringer; M.D. Worthy, "Effects of Tone-Quality Changes on Intonation and Tone-Quality Ratings of High School and College Instrumentalists", Journal of Research in Music Education, Vol. 47, No. 2. (Summer, 1999), pp. 135–149. From https://www.jstor.org/stable/3345719

[36] N.a. (2021) Tuneform Convert BPM to Milliseconds Retrieved from https://tuneform.com/tools/time-tempo-bpm-to-milliseconds-ms

[37] N.a. (2017, February 03). Whats Your Maximum Picking Speed? Take the Poll! Retrieved from https://troygrady.com/2016/08/22/survey-on-picking-speed/

[38] Tacuna Systems. (n.d.). *The Versatile Strain Gauge Load Cell*. Retrieved October 5, 2020, from https://tacunasystems.com/knowledge-base/the-versatile-strain-gauge-load-cell/

[39] Al-Mutlaq, Sarah. (n.d.). Load Cell Amplifier HX711 Breakout Hookup Guide—Learn.sparkfun.com. Learn.Sparkfun.Com. Retrieved April 25, 2021, from https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide

[40] P. (n.d). 50KG Micro Load Cell. Retrieved from https://www.robotshop.com/en/micro-load-cell-50-kg.html

[41]Nave, R. (n.d.). Hall Effect. Retrieved from http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/Hall.html

[42] Emerald, P. (1998). Non-Intrusive Hall-Effect Current-Sensing Techniques Provide Safe, Reliable Detection and Protection for Power Electronics. Retrieved from https://www.allegromicro.com/en/insights-and-innovations/technical-documents/hall-effect-sensor-ic-publications/non-intrusive-hall-effect-current-sensing-techniques-for-power-electronics

[43] Ratna. (2018, April 25). Hall Effect Sensors - Work, Types, Applications, Advantages and Disadvantages. Retrieved from https://electricalfundablog.com/hall-effect-sensors-work-types-applications-advantages-disadvantages/

[44] Dickinson, R., & Bentley, W. (2013). Managing External Magnetic Field Interference When Using ACS71x Current Sensor ICs. https://www.allegromicro.com/en/insights-and-innovations/technical-documents/hall-effect-sensor-ic-publications/managing-external-magnetic-field-interference-acs71x-current-sensor-ics.

[45] N.a (n.d) VG481V1 Retrieved from https://www.digikey.com/en/products/detail/honeywell-sensing-and-productivity-solutions/VG481V1/10712141

[46] Wilkinson. (n.d). Wilkinson M Series WOH Classical Open Style Ceramic Humbucker Pickups Set for 7-String Electric Guitar, Black. Retrieved from

https://www.amazon.com/Wilkinson-Classical-Humbucker-7-String-Electric/dp/B08LDMS82X/ref=sr_1_3?dchild=1&keywords=7 string pickups&qid=1615763834&sr=8-3

[47] MCIGICM. (n.d). MCIGICM 30 Pcs Photoresistor Photo Light Sensitive Resistor, Light Dependent Resistor 5 mm GM5539 5539. Retrieved from https://www.amazon.com/MCIGICM-Photoresistor-Sensitive-Resistor-Dependent/dp/B07PF3CWW9

[48] Microsemi-Watertown. (1992, July). THE PIN DIODE CIRCUIT DESIGNERS' HANDBOOK. Retrieved from https://www.ieee.li/pdf/essay/pin_diode_handbook.pdf

[49]Massari, Gottardi (22 April 2006) "A new CMOS electro-optical modulator based on the charge pumping phenomenon", Proc. SPIE 6189, Optical Sensing II, 61890A Retrieved from https://doi.org/10.1117/12.662480

[50] N.a. (2021) ECE 110 Introduction to Electronics: MOSFETS REtrieved from https://courses.engr.illinois.edu/ece110/sp2021/content/courseNotes/files/?MOSFETs

[51] Massari, Gonzo, Gottardi (2002 October) High Speed Digital CMOS 2D Optical Position Sensitive Detector. Retrieved from https://www.researchgate.net/publication/224617763_High_speed_digital_CMOS_2D_optical_position_sensitive_detector

[52] Cui, Song & Soh, Y.C.. (2010). Linearity Indices and Linearity Improvement of 2-D Tetralateral Position-Sensitive Detector. Electron Devices, IEEE Transactions on. 57. 2310 - 2316. 10.1109/TED.2010.2051862.

[53] Avnet Abacus. (2021). Piezoelectric pressure sensors. Retrieved from https://www.avnet.com/wps/portal/abacus/solutions/technologies/sensors/pressure-sensors/core-technologies/piezoelectric/

[54] N.a (2021) ECSTUFF4U Retrieved from https://www.ecstuff4u.com/

[55] Lynch-Aird N, Woodhouse J. Frequency Measurement of Musical Instrument Strings Using Piezoelectric Transducers. *Vibration*. 2018; 1(1):3-19. https://doi.org/10.3390/vibration1010002

[56] Team, E. (2019, August 06). What Is A Piezoelectric Sensor? Retrieved from https://www.electronicsforu.com/resources/piezoelectric-sensor-basics

[57] Achilles, D. (2000). *Tensions of Guitar Strings*. Retrieved from https://courses.physics.illinois.edu/phys406/sp2017/Student_Projects/Fall00/DAchilles/Guitar_String_Tension_Experiment.pdf

[58] Ahrens, T., Capo, A., & Wong, E. (2014). *Automatic Guitar Tuner*. Retrieved from https://www.ece.ucf.edu/seniordesign/fa2014sp2015/g01/Final_SD2_Proj.pdf?fbclid=IwAR0Fr4OHynTbNTo_BOigghG0wgXH2d5176mIkPeH0EwwbqkY3f3kbFwIoiY

[59] Kohara Gear Industry Co., Ltd (2015). *Know About Gear Transmission Torque*. Retrieved from https://khkgears.net/new/gear_knowledge/the-first-step-of-mechanism-design-using-gears/know-about-gear-transmission-torque.html

[60] N.a. (n.d). *Pololu - 19:1 Metal Gearmotor 37Dx68L mm 12V with 64 CPR Encoder (Helical Pinion)*. Retrieved from https://www.pololu.com/product/4751

[61] Driker, M. (2019). *Design Essentials: Linear Actuators with Thermo-Compensation*. Machine Design. Retrieved from https://cdn.baseplatform.io/files/base/ebm/machinedesign/document/2019/04/machinedesign_9885_0517_linearact_pdflayout.pdf

[62] TiMOTION Technology. (2017). *Advantages and Drawbacks of Hydraulic, Pneumatic, and electric lift cylinder actuator*. TiMOTION Technology. https://www.timotion.com/en/news/news_content/blog-articles/general/advantages-and-drawbacks-of-pneumatic,-hydraulic,-and-electric-linear-actuators?upcls=1481266229&guid=1499762723

[63] N.a. (n.d). *Linear Actuator 4" Heavy Duty with Brackets Stroke 225 Pound Max Lift 12 Volt DC.* Retrieved from https://auto-express.co/products/linear-actuator-4-heavy-duty-with-brackets-stroke-225-pound-max-lift-12-volt-dc

[64] Labriola, D. (2005, January 1). *Why open-loop steppers lose steps, and how to solve the problem*. Retrieved from https://www.machinedesign.com/motors-drives/article/21833271/why-openloop-steppers-lose-steps-and-how-to-solve-the-problem

[65] Collins, D. (2017, November 24). *Microstepping for Stepper Motors*. Retrieved from https://www.linearmotiontips.com/microstepping-basics/

[66] ATO.com. (2020, November 27). *How to prevent stepper motor losing step?* Retrieved from https://www.ato.com/how-to-prevent-stepper-motor-losing-step

[67] N.a (n.d). *Pololu - Stepper Motor: Unipolar/Bipolar, 200 Steps/Rev, 42×48mm, 4V, 1.2 A/Phase*. Retrieved from https://www.pololu.com/product/1200

[68] N.a (n.d). *Pololu - Power HD High-Torque, High-Voltage Digital Servo 1218TH*. Retrieved from https://www.pololu.com/product/2147

[69] N.a. (n.d.). *Qunqui L298N Motor Controller*. Retrieved from https://www.amazon.com/Qunqi-Controller-Module-Stepper-Arduino/dp/B014KMHSW6

[70] N.a. (n.d.). *Adafruit Push-Pull Solenoid*. Retrieved from https://www.mouser.com/ProductDetail/Adafruit/412/?qs=GURawfaeGuBVsKnEJX3AiQ%3D%3D

[71] N.a. (n.d.) *Arduino Due - A000062*. Retrieved from https://grobotronics.com/arduino-due-a000062.html?sl=en

[72] London, J. (2012, January 6). *Three Things Linguists Need to Know About Rhythm and Time in Music*. Retrieved from https://www.researchgate.net/publication/270818914_Three_Things_Linguists_Need_to_Know_About_Rhythm_and_Time_in_Music/link/5acbae364585151e80aa2224/download

[73] N.a. (n.d.). *SparkFun Load Cell Amplifier - HX711 - SEN-13879 - SparkFun Electronics. Sparkfun.com*. Retrieved 30 April 2021, from https://www.sparkfun.com/products/13879.

[74] Koen, S. (2020, June 19). *The Big O Notation - Towards Data Science*. Medium. https://towardsdatascience.com/the-big-o-notation-d35d52f38134

[75] *Autocorrelation - an overview | ScienceDirect Topics*. (2019). Science Direct. https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/autocorrelation#:%7E:text=Basically%2C%20the%20autocorrelation%20function%20describes,with%20neighboring%20portions%20of%20itself.

[76] Dressler, & Streich. (2007). Tuning Frequency Estimation Using Circular Statistics. *ISMIR*, 357–360. https://archives.ismir.net/ismir2007/paper/000357.pdf

[77] Condes. (2021). *Arduino FFT Library*. GitHub. https://github.com/kosme/arduinoFFT

[78] Kaleloft LLC. (2019). *Pano Tuner* [Mobile App]. Google Play Store.

[79] Britannica, T. Editors of Encyclopaedia (2020, June 14). Monochord. Encyclopedia Britannica. https://www.britannica.com/art/monochord

[80] Pololu (2020, January). *Pololu - 19:1 Metal Gearmotor 37Dx68L mm 12V with 64 CPR Encoder (Helical Pinion). Pololu Robotics &amp; Electronics*. Retrieved from https://www.pololu.com/product/4751.

[81] Association for Advancing Automation. (2018, February 12). A Comparison of DC Linear Actuators with DC Linear Motors. Automate. https://www.automate.org/tech-papers/a-comparison-of-dc-linear-actuators-with-dc-linear-motors

[82] Phidgets Inc. (2018, November 27). *Load Cell Primer—Phidgets Support*. Phidgets. https://www.phidgets.com/docs/Load_Cell_Primer#What_is_a_Wheatstone_bridge.3F

[83] STMicroelectronics (2000, January). *L298 Dual Full-Bridge Driver*. Retrieved from https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

Additional Works Consulted

- J. D'Addario & Company, Inc. *Technical Reference for fretted instrument string tensions*. Retrieved September 20, 2020 from https://www.daddario.com/globalassets/pdfs/accessories/tension_chart_13934.pdf
- Russell, Daniel A. 2011, *The Vibration of a Fixed-Fixed String*, Retrieved from www.acs.psu.edu/drussell/Demos/string/Fixed.html.
- N.a. (n.d). 3.3 Vibrating Strings. Retrieved from https://www.phys.uconn.edu/~gibson/Notes/Section3_3/Sec3_3.htm
- Halfpenny, Eric. (n.d.) *Stringed Instrument.* Encyclopedia Britannica, Retrieved from: https://www.britannica.com/art/stringed-instrument#ref53707
- IGI Global. (n.d.) *What is Dynamic Tuning.* Retrieved from: https://www.igi-global.com/dictionary/dynamic-tuning/8549
- "Ranges of Orchestral Instruments." Range of Instruments, www.orchestralibrary.com/reftables/rang.html.
- Jain, Aditya, et al. "A Comparative Study of Visual and Auditory Reaction Times on the Basis of Gender and Physical Activity Levels of Medical First Year Students."

International Journal of Applied and Basic Medical Research, vol. 5, no. 2, 2015, p. 124., doi:10.4103/2229-516x.157168.

- Tom's Guitar Projects. (2014). *Electric Guitar Output Voltage Levels.* Retrieved from: http://tomsguitarprojects.blogspot.com/2014/12/electric-guitar-output-voltage-levels.html
- Ubertar Hexaphonic. (n.d.) *One and Two String Pickups.* Retrieved from: http://ubertar.com/hexaphonic/one_and_two_string_pickups.html
- Borel, B. (2012, July 3). *Inventions: The Piano That Tunes Itself*. Popular Science. https://www.popsci.com/technology/article/2012-06/2012-invention-awards-self-tuning-piano/
- Cort, J. (2011, March). *Maximum isometric finger pull forces*. ScienceDirect. https://www.sciencedirect.com/science/article/pii/S0169814110001137
- *Improving PID Controller Performance*. (2019, March 5). National Instruments. https://www.ni.com/en-us/innovations/white-papers/08/improving-pid-controller-performance.html
- Middlesworth, M. (n.d.). *NIOSH Lifting Equation*. Ergo Plus. https://ergo-plus.com/niosh-lifting-equation-single-task/
- *The MIDI Standard*. (n.d.). University of Michigan. Retrieved October 25, 2020, from https://www.eecs.umich.edu/courses/eecs373/Lec/StudentF18/MIDI%20Presentation.pdf
- *What are the Maximum Power Output and Data Transfer Rates for the USB Standards?* (n.d.). https://resources.pcb.cadence.com/blog/2020-what-are-the-maximum-power-output-and-data-transfer-rates-for-the-usb-standards
- Woolf, P. (2009). 9. PID Control. In *CHEMICAL PROCESS DYNAMICS AND CONTROLS*. openmichigan.
- Roberts, Evan J. "How Often Do Piano Tuning Pins Need Replacing?" *Roberts Pianos*, 16 July 2018, www.robertspianos.com/design/piano-tuning-pins-need-replacing/.
- Overstock™. "Online Shopping - Bedding, Furniture, Electronics, Jewelry, Clothing & More." *Overstock.com*, www.overstock.com/Food-Gifts/Right-Handed-6mm-Bore-Hole-Guitar-Tuning-Peg-Machine-Head-Tuner-Silver-Tone-Beige/29923141/product.html.

- Krutz, Anton. “A Complete Guide to Tuning Your Instrument.” *K.C. Strings Violin Shop*, 6 Apr. 2020, www.kcstrings.com/blog/a-complete-guide-to-tuning-your-instrument/.
- Duffy, Mike. “What Are Tuning Pegs and Why Do You Need Them?” *Fender Guitars*, www.fender.com/articles/tech-talk/what-are-tuning-pegs-and-why-do-you-need-them.
- Brain, M. (2002, July 01). How Electric Guitars Work. Retrieved from https://entertainment.howstuffworks.com/electric-guitar.htm
- Ramsey, G., & Pomian, K. (2020, July 10). Physics of Stringed Instruments. Retrieved from https://exploresound.org/2017/02/physics-stringed-instruments/
- Brain, M. (2021, February 22). How Acoustic Guitars Work. Retrieved from https://entertainment.howstuffworks.com/guitar.htm
- N.a. (n.d). 2.1 Pitch, Frequency, Period, Loudness, Timbre. Retrieved from https://www.phys.uconn.edu/~gibson/Notes/Section2_1/Sec2_1.htm#:~:text=The technical term for pitch,440 oscillations in 1 second.&text=A 440 = 440 oscillations per,440/sec = 440 Hz.
- “Semitone.” *Merriam-Webster.com Dictionary*, Merriam-Webster, https://www.merriam-webster.com/dictionary/semitone. Accessed 14 Mar. 2021.