Fully Automated Guitar Tuner

ECE 445 Final Report

Benjamin Wang, Brandon Ramos, Cooper Ge
Team 71
TA: Vassily Petrov
5/8/20

Abstract

The problem we chose to address is that tuning guitars is a time consuming and undesirable task for guitar players of all levels. When tuning by ear, it is extremely hard to find the right pitch. The original solution was similar to current commercial tuners; it told you the closest approximate pitch and graphically indicated how sharp or flat you were. Our new solution is an all-in-one tuning solution with minimal user input. Rather than just displaying the input graphically, our device will automatically handle both strumming and tuning to mechanically tune the guitar. The main improvement is convenience for the user.

Table of Contents

1 Second Project Motivation

1.1 Problem Statement

It's an unfortunate fact of life that guitars fall out of tune over time. Playing on pitch is hugely important, but is impossible to do if the instrument itself is out of tune. Those with perfect pitch may do it by ear, but experts estimate only 0.01% to 0.05% of the population has that ability [1], and even then, they must still physically take the time to tune. Everybody else must rely on a tuner. The most common tuners will tell you whether a specific string is sharp or flat, leaving players to physically adjust the tuning knobs and manually pluck the guitar string each adjustment. For beginners, this process may take up to 5 minutes [15]. Cheaper strings and guitars [15], which beginners are more likely to have, will also go out of tune quicker, making their lives even more difficult.

1.2 Solution

Our solution is a complete package that will tune 3 strings of a guitar at a time with minimal user input. The device will be battery powered and compact, making it easily portable. The user would attach it to the head of the guitar and adjust the motor arms so they sit on the tuning pegs. The user would then attach the strumming motor and allow the system to tune the guitar completely hands free. Once finished, the process would repeat on the other three strings, making the tuning process as convenient as possible.

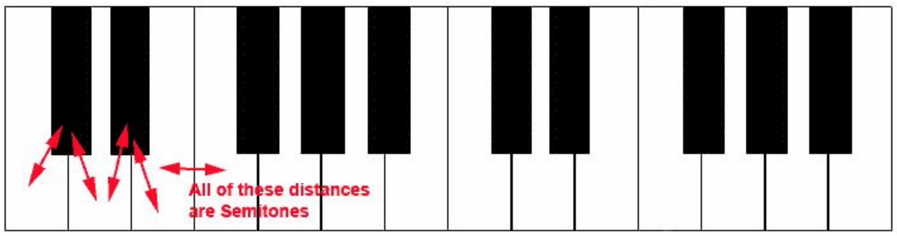
The original electronic guitar tuning solution from the spring of 1999 was simply a microphone-to-amplifier circuit attached to a pre-built, commercial DSP board to indicate simple flat or sharp classifications for each string. Our updated solution not only makes a more robust classification of pitch error, but also contains mechanical systems to fully automate the tuning processes of strumming and adjusting the tuning knobs for three strings at a time. The changes made offer huge advantages in both time-efficiency and convenience for beginner to intermediate level guitar players.

There currently exists a widely commercially available automatic single-string guitar tuner [2]. Our solution is advantageous to the single-string guitar tuner because it can automatically adjust two additional strings at the same time and also presents the novel ability to strum itself for a totally self-contained tuning process loop. Similar to our solution's advantages over the original electronic guitar tuning solution, our solution also has the advantages of time-efficiency and convenience over the market solution.

1.3 High-Level Requirements

- Identify the pitches of the three strings played to within half a semitone each
- Tune guitar within half a semitone
- Tune within within 1 minute

Note: Semitone definition - the smallest interval used in classical Western music, equal to a twelfth of an octave or half a tone. For example, two adjacent keys on a piano.



Semitones can be further divided into *cents,* which is a logarithmic unit of measure. One semitone consists of 100 cents.

1.4 Visual Aid

Shown below in Figure 1.4 is a simplified visual representation of our automatic tuning solution.

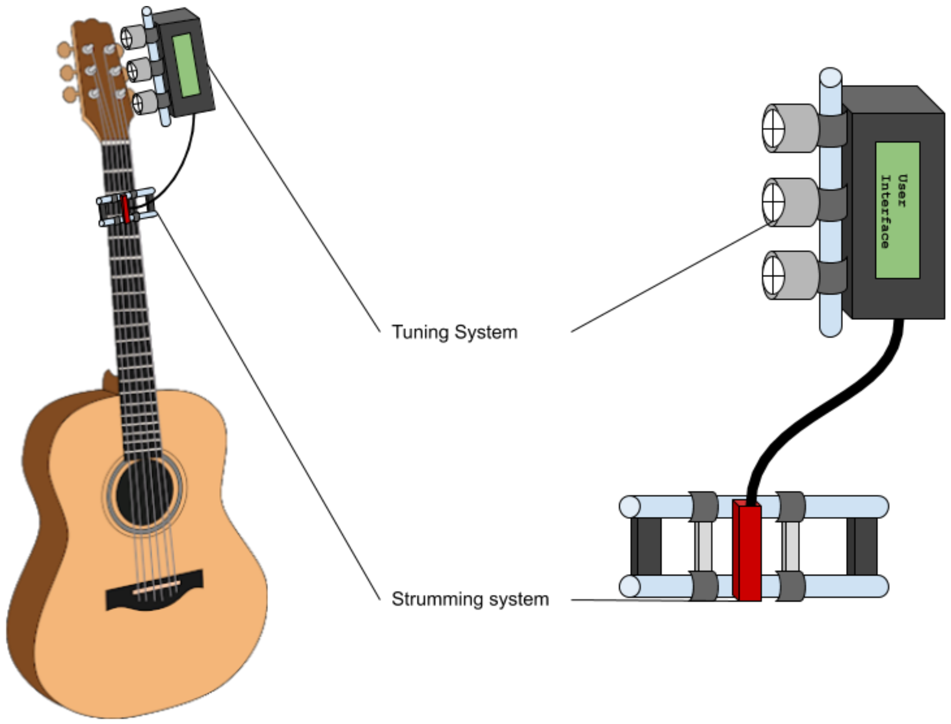


Figure 1.4 Visual Aid

1.5 Block Diagram

We have divided our automatic guitar tuning system into four primary subsystems: the audio recording system to collect the audio information, the mechanical systems to interface with the guitar (further split into the strumming and tuning system), the processing system to both control the mechanical systems and determine how they should be controlled, and the power system to supply the aforementioned systems. All block connections are wired, with power lines and data lines being differentiated by color. The audio line is not a physical connection, and represents sound from the strings being strummed

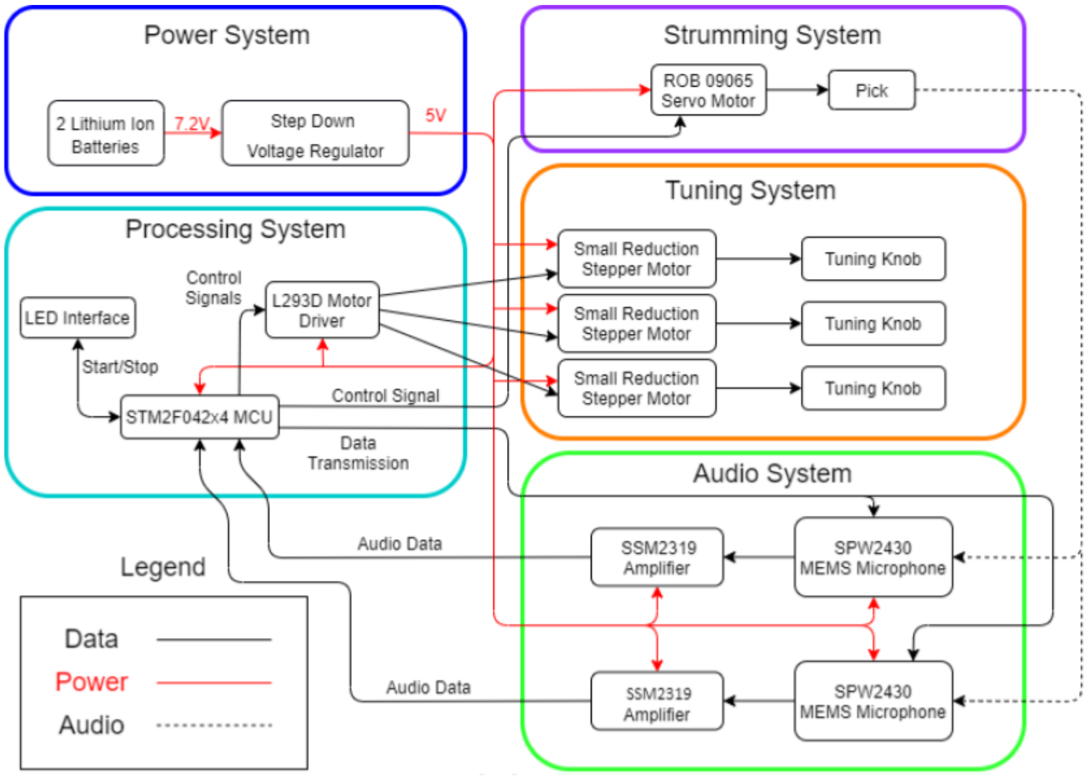


Figure 1.5 Block Diagram

2. Second Project Implementation

2.1 Physical Design

2.1.1 Physical Design Implementation

The functionality of our design depends on being able to interface with a variety of guitars. Unfortunately, there does not exist a standard shape nor size for guitars across brands [9]. The shape of the headstock, the part of the guitar where the tuning knobs are located, similarly have great variation in size and shape. Tuning knobs can be configured all six lined up on one side or three on each side. In the interest of universal guitar compatibility, the tuning motors are adjustable from 1” to 3.5” and the strumming motor adjustably clips to the neck at a generous 2.5” rail length for the strumming range to the standard 1.75”-2” guitar neck width.

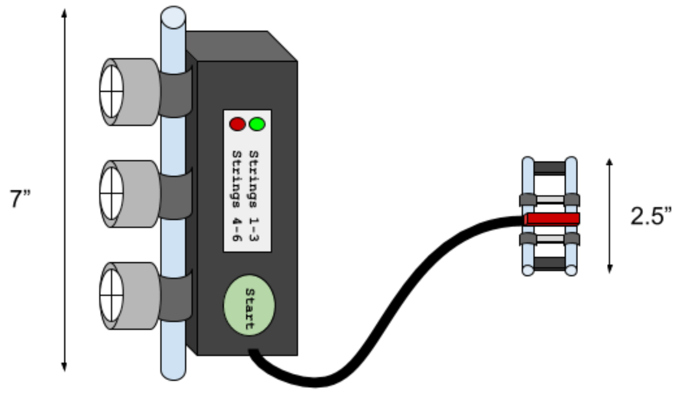


Figure 2.1.1.1 Physical Design Sizing

From a physical design standpoint of our project, the strumming system is the most complex. The system must be able to adjustably clamp onto a guitar neck at an optimal height for the strumming piece to pluck the guitar strings and then must have further adjustments to adjust the range of the strumming motor to only pluck the desired set of strings. These specifications are met by creating an adjustable-width outer clamping system to mount the strumming system securely at a specified height above the guitar strings and having the strumming system motor’s range be limited by movable mechanical stops along the rail on which it runs. Shown in Figure 2.1.1.2 is a more detailed representation of the simplified strumming system and Figure 2.1.1.3 shows a CAD rendering of the rail-servo-pick system.

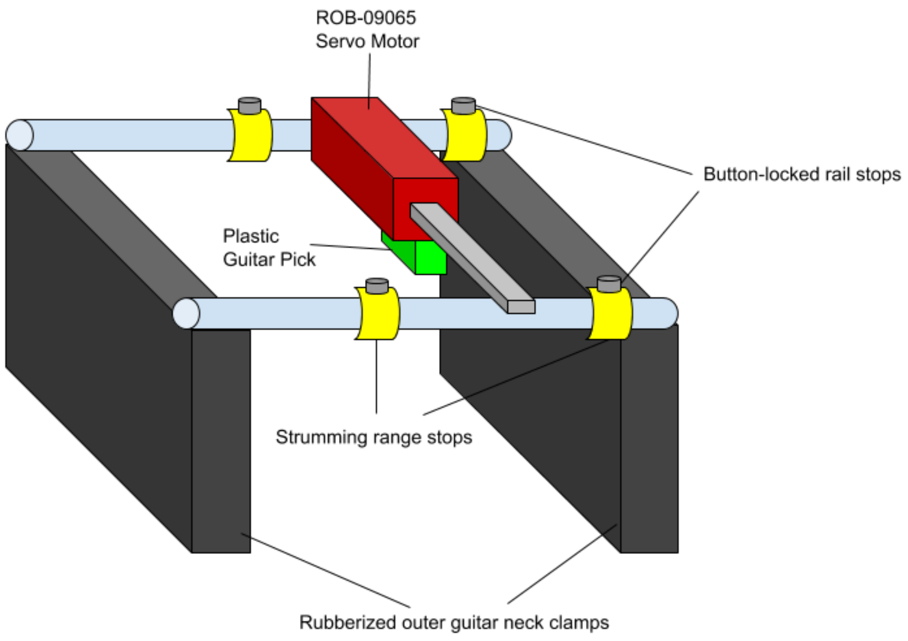


Figure 2.1.1.2 Strumming System Simplified Diagram

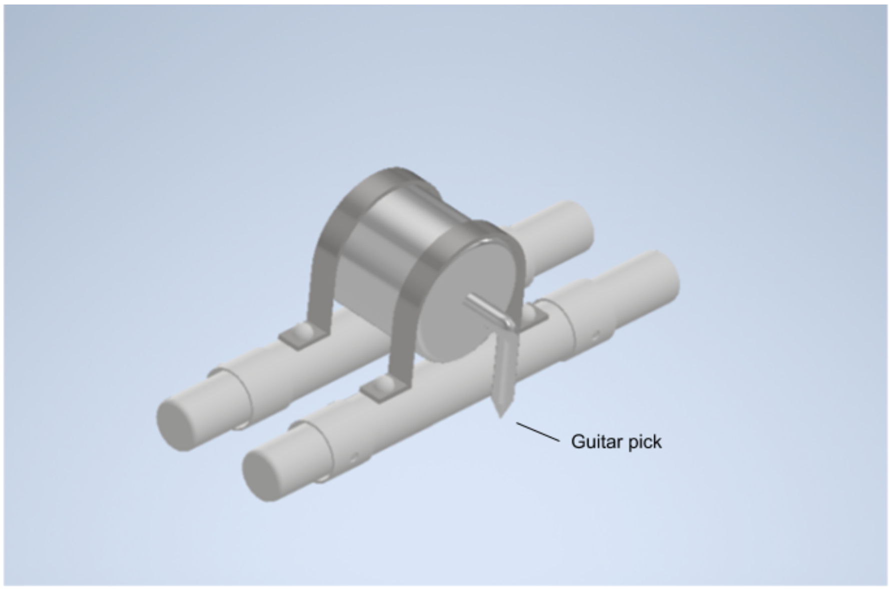


Figure 2.1.1.3 CAD Rendering of Strumming System

2.1.2 Physical Design Implementation Analysis

It is worth considering whether or not our design has enough mechanical precision to accurately perform the gestures needed for tuning efficiently-- namely, strumming and turning the tuning knobs. Strumming is a simple enough problem to solve. Designing a servo motor to run along a rail across guitar strings is simply a matter of ensuring the servo has enough torque. Similarly, the stepping motors for tuning must also have a sufficient maximum torque to apply tension to the guitar strings. The torque ratings for both types of motors are guaranteed by the manufacturer to be sufficient for our use case. For example, the rated torque of the servo motor we are using for the strumming system is 22.22 ounce-inches, which is far more than enough than the estimated 1.4 ounce-inches of torque required to pluck a guitar string [14]. A much more serious consideration, however, must be given to how calculated pitch differences for each string will be translated into tuning knob adjustments.

Tuning knobs adjust string tension by driving a gear in the headstock of the guitar attached to a post that the guitar string gets continually wrapped around. Just as there does not exist a standard guitar size and shape, there does not exist a standard post size to mount the strings onto nor a standard gear size to rotate that post.
The formula for standing waves on strings is as follows:

$$f = \frac{1}{2L}\sqrt{\frac{T}{\mu}} = C\sqrt{x}$$

where f is the frequency of vibration, L is the length of the string, T is the tension of the string, and μ is the constant linear density of the string. The number of rotations required to make a specific pitch adjustment on a tuning knob not only on how much the string length is adjusted by the tuning knob, but also how much each degree of rotation adjusts the length of the string. Luckily, guitar strings are designed to linearly change tension with the square of string length in their respective frequency ranges. By leveraging this fact, we can simplify the calculation of the first harmonic frequency of standing waves to be the following:

$$f = \frac{1}{2}\sqrt{\frac{x}{\mu}}$$

where x is a parameter directly proportional to knob rotations. This means that the relationship of knob rotations to string frequency can be calculated with just two data points. This means that every string can be tuned in approximately three iterations by our design, well under the 1 minute time requirement.

2.2 Algorithm

The algorithm is responsible for identifying the pitches being played, which will allow the microcontroller to compare the detected pitch with the desired pitch, and tune the string accordingly.

2.2.1 Background

Musical notes are the fundamental building blocks of music. There are seven notes, which are commonly notated with the first seven letters of the alphabet - A, B, C, D, E, F, G. These can be further modified by sharps and flats, or, which are denoted by #/*b* respectively, making 12 total notes - C, C#, D, D#, E, F, F#, G, G#, A, A#, B. Each note has a fundamental frequency; for example, middle C is approximately 261.6 Hz. A lower frequency corresponds to a lower pitch, and vice versa. Doubling the fundamental frequency will generate the same note, just an octave higher. For example, 440 Hz is an A, and 880 Hz is also an A. Each note will also have harmonics at integer multiples of its fundamental frequency. For example, a 220 Hz A will have harmonics at 440Hz, 880 Hz, 1320 Hz, 1760 Hz and so on, as shown in the figure below.

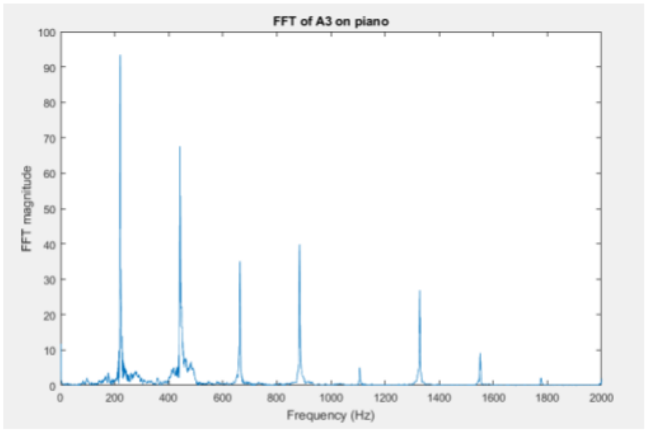


Figure 2.2.1.1 Frequency Composition on Piano

A chord is simply two or more notes played at once, and is how our project sets itself apart. Common pitch detectors, including the ones in commercial tuners, cannot handle multiple inputs at once.

2.2.2 Algorithm Summary

Our algorithm will leverage Fujishima's 1999 paper [5]. His algorithm involves taking the DFT of the signal, generating a spectrum bin table, and using that to sort the DFT into a pitch class profile. He then proceeds to use pattern matching to identify the chord, which will not be necessary for our purposes. His graphic from his paper is reproduced below.

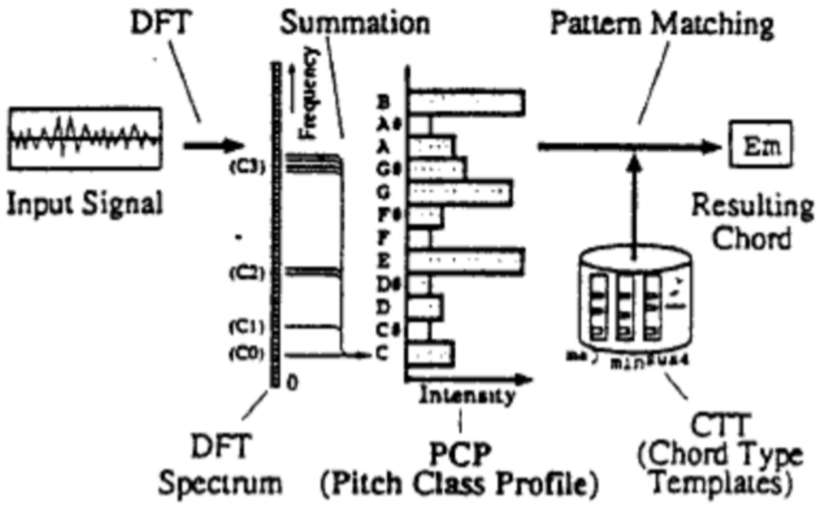


Figure 2.2.2.1 Algorithm Graphic

Our algorithm flow chart is pictured below.

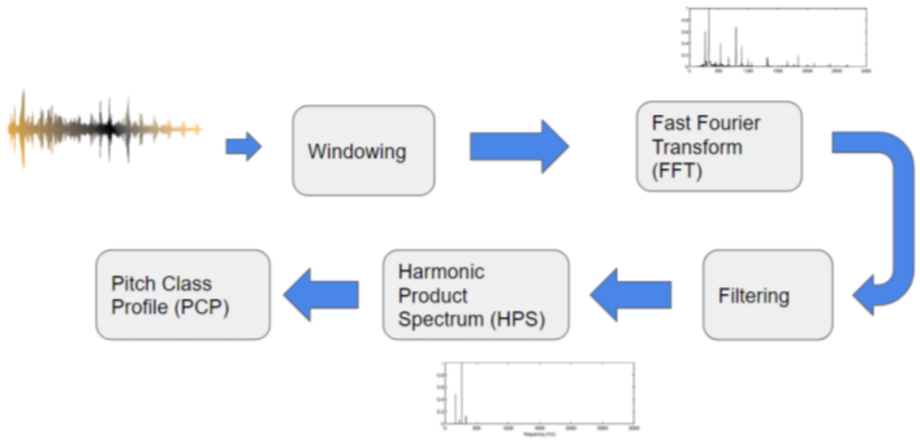

Windowing
Fast Fourier Transform (FFT)
Filtering
Harmonic Product Spectrum (HPS)
Pitch Class Profile (PCP)

Figure 2.2.2.2 Algorithm Flow Chart

Since we cannot feasibly process the entire length of the signal, we must first window the incoming data. In our prototype testing, we found no significant difference between different windows, and chose the hamming window for its side lobe attenuation. We then take the fourier transform of the windowed signal to convert the signal into the frequency domain, and pass it through a bandpass filter to attenuate the frequencies that are not associated with our desired notes or chords. The narrower the band, the less intensive the algorithm would have to be, but we had to make sure we weren't filtering out data that we wanted. We found that for guitar, we could filter out frequencies outside 60 - 2000 Hz. As mentioned earlier, notes have harmonics at integer multiples of their frequency. These harmonics can often overlap with other notes in the chord, making it hard to distinguish what is a fundamental frequency and what is a harmonic.

This is where the harmonic product spectrum comes in. The HPS cuts off higher order harmonics, leaving either just the fundamental frequency, or a lower number of harmonics.
From that, we generate our pitch class profile, a twelve dimension array, which contains the intensity of each pitch with the following equation.

$$PCP(p) = \sum_{l\ s.t\ M(l)=p} ||X(l)||^2$$

, where $M(l)$ is the spectrum bin table. $M(l)$ is defined according to the equation below, where f_s is the sampling frequency, N is the length of the FFT, and f_{ref} is the reference frequency desired for $PCP(0)$.

$$M(l) = -1\ for\ l = 0$$

$$M(l) = round(12 * log_2(\frac{f_s * \frac{1}{N}}{f_{ref}}))mod12\ for\ l = 1, 2, ..., \frac{N}{2} - 1$$

2.2.3 Testing Results

Initial testing for our algorithm proved to be promising, with it successfully identified the correct pitches in a three tone chord from a recording. Some of our outputs are pictured below.

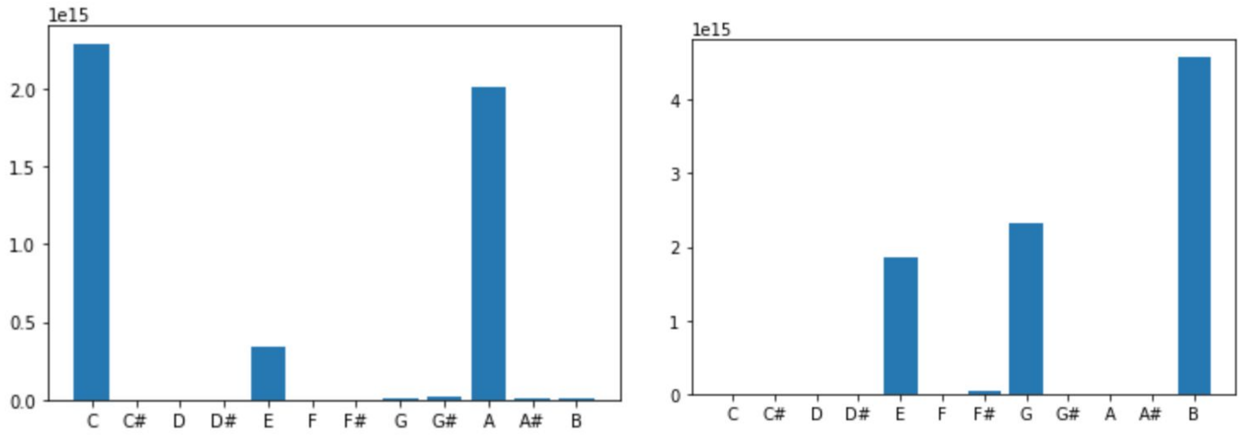


Figure 2.2.3.1 Testing Results 1

2.2.4 Algorithm Improvements

We quickly realized that our first trials did not have enough resolution. If a frequency was halfway between two notes, it would not get accurately represented. While this could be partially fixed by having standard tunings for a guitar stored in memory, it would still be better to have our algorithm be more accurate. We doubled the number of frequency bins, and results from two recordings of the same guitar, tuned and untuned, are pictured below.

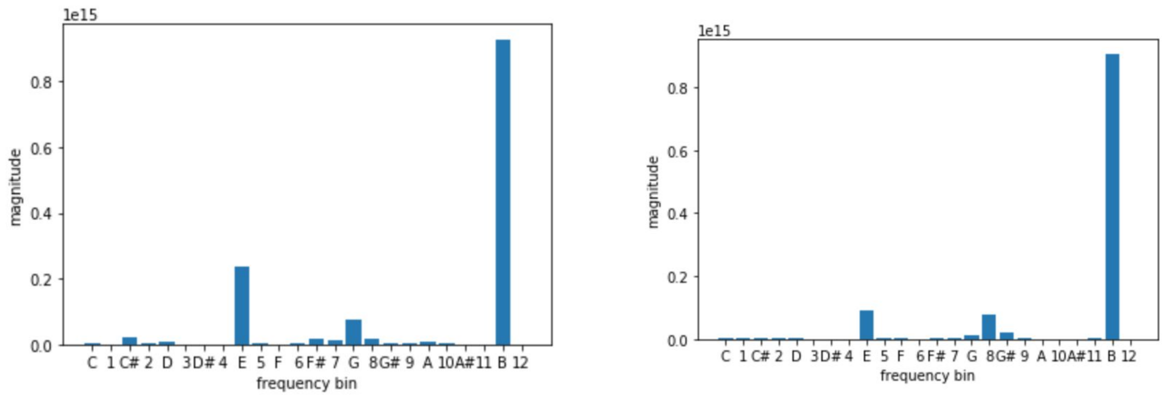


Figure 2.2.4.1 Testing Results 2

2.3 Control Logic

The control logic is critical in making sure the mechanical systems both correctly create audio input for the algorithm and correctly implement the tuning adjustments to the strings. This system is created in software on the processing core according to the diagram shown in Figure 2.3.1. The control logic design specifies the order in which mechanical events occur and implements the tuning calibration of the tuning knob adjustments. Tuning calibration is important because not all guitar tuning knobs have consistent change in string tension per tuning knob rotation. Shown in Figure 2.3.2 is the control logic implemented in pseudo-code.

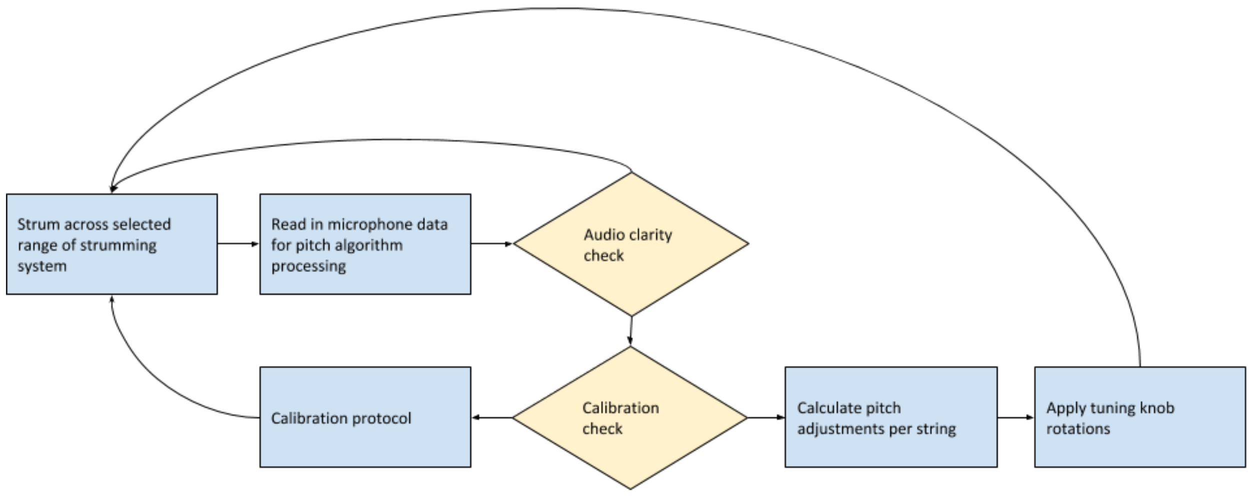
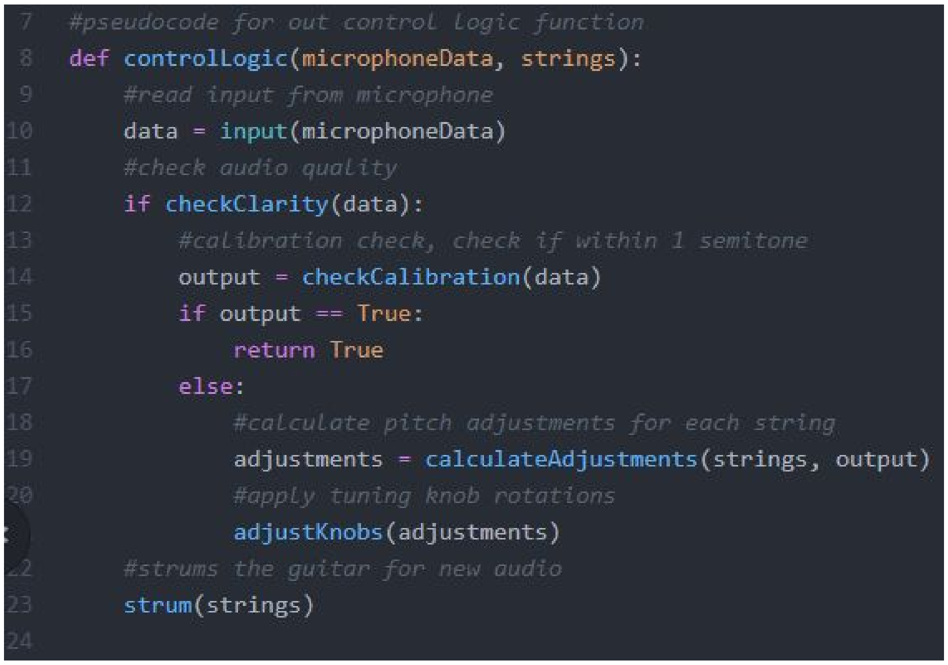

Strum across selected range of strumming system
Read in microphone data for pitch algorithm processing
Audio clarity check
Calibration check
Calibration protocol
Calculate pitch adjustments per string
Apply tuning knob rotations

Figure 2.3.1 Control Logic Flow Diagram

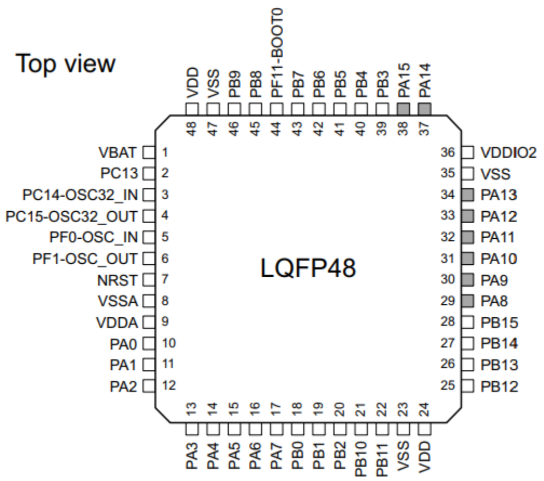


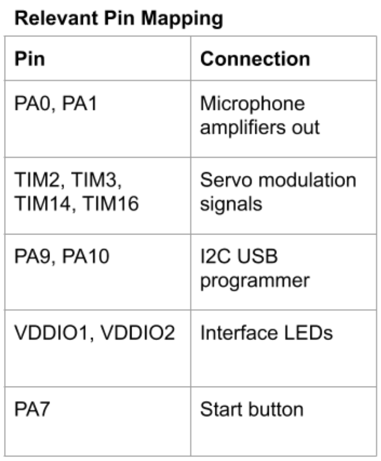
```
#pseudocode for out control logic function
def controlLogic(microphoneData, strings):
    #read input from microphone
    data = input(microphoneData)
    #check audio quality
    if checkClarity(data):
        #calibration check, check if within 1 semitone
        output = checkCalibration(data)
        if output == True:
            return True
        else:
            #calculate pitch adjustments for each string
            adjustments = calculateAdjustments(strings, output)
            #apply tuning knob rotations
            adjustKnobs(adjustments)
    #strums the guitar for new audio
    strum(strings)
```

Figure 2.3.2 Control Logic Pseudocode

2.4 Microcontroller

Due to the computation-heavy nature of our pitch detection algorithm across three strings, we have elected to use the STM32F042x4 MCU in the LQFP48 package. Unlike the original project from Spring '99, our solution does not implement a prebuilt board for computation. We will be leveraging the 48 MHz CPU clock on the microcontroller to implement both the pitch detection algorithm and the control logic. Shown below in Figure 2.4 is the pinout for the surface-mount package and the relevant pin mappings for our design.





Relevant Pin Mapping

Pin	Connection
PA0, PA1	Microphone amplifiers out
TIM2, TIM3, TIM14, TIM16	Servo modulation signals
PA9, PA10	I2C USB programmer
VDDIO1, VDDIO2	Interface LEDs
PA7	Start button

Figure 2.4.1 Microcontroller Pinout [10]

2.5 Audio System

The audio system in our design consists of two MEMS microphones and two corresponding amplifiers. It is responsible for collecting the audio data from the guitar and transporting it to the processing system. In some ways this system also includes the ADC on the microcontroller.

2.5.1 Audio System Implementation

We designed our audio system to establish the connection between the strumming motor system and the processing subsystem so that relevant audio data can be processed into tuning logic. In the interest of the maximization of microphone clarity of the guitar sound, we elected to use the SPW2430 MEMS microphone and SSM2319 audio amplifier in a cardioid configuration. The MEMS technology is favorable for digital noise tolerance and package size [4] and the cardioid configuration of the microphones will allow for effective attenuation of background noise and room reflection [12].

2.5.2 Audio System Implementation Analysis

Our audio system presents a significant bottleneck in our ability to identify the pitch of each guitar string. Theoretically, according to the algorithm we are using outlined in Section 2.2, the only bottleneck on the frequency information we can recover is the sampling rate and number of FFT points used. The DFT formula is given by

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi nk}{N}} ,$$

where N is the number of FFT points. The frequency resolution is given by

$$f_r = \frac{f_s}{N} .$$

f_s is dictated by how fast our microcontroller can sample our microphone. In this case, we can read our two microphones at a sampling rate of up to 1MHz. However, our microphones are only capable of detecting up to 10 kHz, and therefore by the Nyquist theorem, we should sample at a minimum of 20 kHz. We chose an FFT length of 1024, giving us a frequency resolution of 19.53 Hz.

The standard tuning for the six strings on the guitar E2, A2, D3, G3, B3, and E4 have frequencies 82.41 Hz, 110 Hz, 146.8 Hz, 196 Hz, 247.9 Hz, 329.6 Hz. The minimum frequency difference between the strings we must then be able to resolve is between E2 and A2, 27.59 Hz. Additionally, we must be able to resolve pitches within 40 cents. This criteria is less consequential than the 27.59 Hz baseline, however, because the pitches will have harmonics that allow the relative cent scale to be seen on the higher order harmonics which double in frequency every order. Since our frequency resolution is less than the minimum frequency distance between pitches, we will be able to successfully detect pitch for our design.

2.6 Bill of Materials and Cost

Table 2.6.1 Bill of Materials and Cost

Description	Quantity	Vendor	Total Cost
Small Reduction Stepper Motor	3	Adafruit	$14.85
Dual H-Bridge Motor Driver L293D	3	Adafruit	$8.95
Servo - Generic (Sub-Micro Size)	1	Sparkfun	$8.95
SSM2319CBZ-REEL	2	Mouser	$1.80
STM32 F0 ARM Development Board	1	Banggoood	$3.94
MEMS Microphone Breakout - SPW2430	2	Adafruit	$9.90
Capacitor and Resistor Budget	x	x	$20.00
Mechanical Build Budget	x	Machine Shop	$200.00
Pinout/PCB Interface Budget	x	x	$30.00
PCB Printing(General)	1	x	$35.00
			Total: $333.39

Table 2.6.2 Labor Cost Summary

Engineer	Hours Expected	Hourly Rate	Cost + Overhead
Ben	156	$50.00	$19,500.00
Brandon	156	$50.00	$19,500.00
Cooper	156	$50.00	$19,500.00
(Machinist)	20	$40.00	$2,000.00
			Total: $60,500.00

Table 2.6.3 Grand Total Cost

Section	Total
Materials	$333.39
Labor	$60,500.00
Grand Total	$60,833.39

3 Second Project Conclusions

3.1 Implementation Summary

In the implementation of the design of our second project, we have been able to accomplish the following implementations:

- Successful classification of three string frequencies within half a semitone simultaneously
- Thorough prototype of physical design and mechanical calibration
- Pseudocode of control loop for the tuning process

3.1.1 Algorithm Implementation

The implementation of the algorithm is the cornerstone of our project design. Almost every other part of our design is directly connected to it. We were able to successfully detect the ideal pitches of the three strummed strings using our algorithm to an accuracy of half a semitone as shown in Section 2.2. This means that pitch adjustments can be made in the resolution of 25 cents and a properly calibrated system would have a maximum error of only ∓12.5 cents to the ideal standard guitar tuning. Additionally, we gathered real world data of guitar audio to prove our algorithm was able to identify the pitch of both the top and bottom three guitar strings in a room noise. The algorithm was implemented and tested by Cooper Ge.

3.1.2 Physical Design

The most novel and advantageous aspect of our design comes from the implementation of the physical design. We were able to successfully design the mechanical specifications for a working proof-of-concept physical design shown in Section 2.1. By having the mechanical design in place to effectively leverage the electrical components, the foundation is built for how the abstract control logic will physically interact with the guitar and present a solution to the tedious nature of guitar tuning. The physical design was implemented by Brandon Ramos.

3.1.3 Control Loop

The control loop controls the whole tuning process and determines when the process is finished. Every time we start the tuning of the guitar, we call the control loop function to begin tuning. The loop is as follows:

1) Strum guitar strings
2) Check audio quality, if too poor, restrum
3) If audio input is correctly calibrated, exit loop
4) Calculate pitch differences using algorithm
5) Adjust tuning knobs
6) Restart loop

The control loop dictates which subsystem to utilize at each step in the process and makes sure each system is working not only separately but also with each other. The control loop was designed and written by Benjamin Wang.

3.2 Unknowns, Uncertainties and Testing Needed

There are several aspects of our project that we have been unable to complete due to lack of access to the lab. These obstacles primarily involve the testing and development of the hardware required for our design. Even assuming we had the time, money, and part availability to obtain the necessary components of our design, our ability to perform testing or development would be very limited.

Due to lack of machine shop assistance, we have been unable to fabricate the frame or fittings for our tuning system. This means that the calibration of our motors and microphone would be blind since they would not have a relevant testing environment. Even if we wanted to perform blind characterizations of our electrical components, we would simply not be able to receive the parts in time to also have time to test them.

Due to the limitations of our development period, our implementation was limited to only the software and abstract design portions of our project. In our implementation, we are still uncertain as to the mechanical calibration of our motors needed to effectively tune the guitar which our design heavily relies on and the quality of the audio data our audio system will output. Given more time, we would first prototype our design on breadboard and unit test mechanical and audio systems separately by providing their required inputs to ensure each system is bug-free. From there, we would work on calibrating both systems to work with the control loop developed for the microcontroller.

3.3 Safety and Ethics

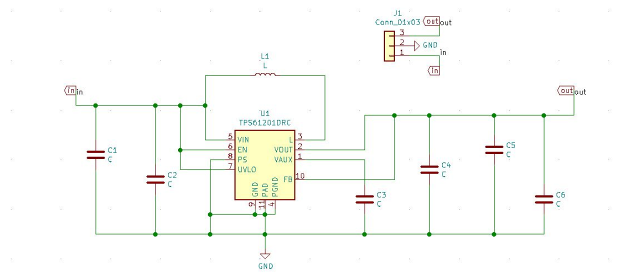
Following the IEEE Code of Ethics, our only issue in regard to Code of Ethics #1, that our design holds the safety of the public as its first priority and informs the public of any endangering aspects [6]. Our project utilizes rechargeable lithium ion batteries, which is our main source of power for our project. Lithium ion batteries have the potential to cause fire hazards from overheating [7] or environmental hazards if disposed of incorrectly [8]. We will monitor the battery temperature using a semiconductor temperature sensor and a LED to notify users when battery temperature goes above 55 degrees Celsius. We will also make it easy to deconstruct our system so that the lithium ion battery can be disposed of properly. Although we are using a microphone to take in audio input, we are directly processing the audio data and not saving it in any way, so this does not violate the code of ethics. An issue raised during our design review was the potential for damage to personal property from our motors. This issue is already somewhat mitigated by our motor selection. They are not strong enough to do any kind of damage to the tuning knob threads or guitar strings. We plan to further address this issue by implementing rotation and speed limits in software.

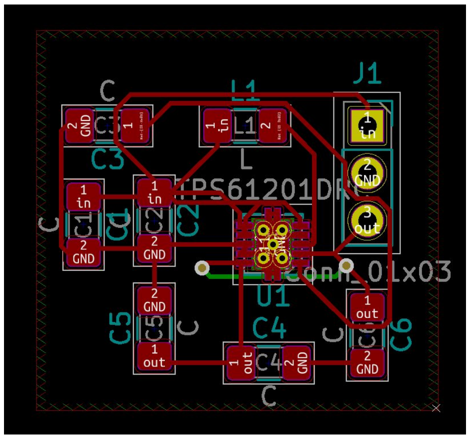
3.4 Project Improvements

If there were a year to complete this project instead of a few weeks, in addition to completing more aspects of our design implementation, some improvements we would look into would be vibrational pitch sensing of the headstock, a more robust user interface, and further improvement of the pitch detection algorithm. We believe these design changes would be worth looking into because vibrational pitch sensing would be almost totally resistant to room noise, our current user interface is absolutely bare-bones. The implementation of our design was significantly compromised by the time constraints of the project and we believe that having a more robust pitch sensing system, more informational and interactive user interface, and higher resolution pitch detection algorithm would provide direct benefits to our design implementation.

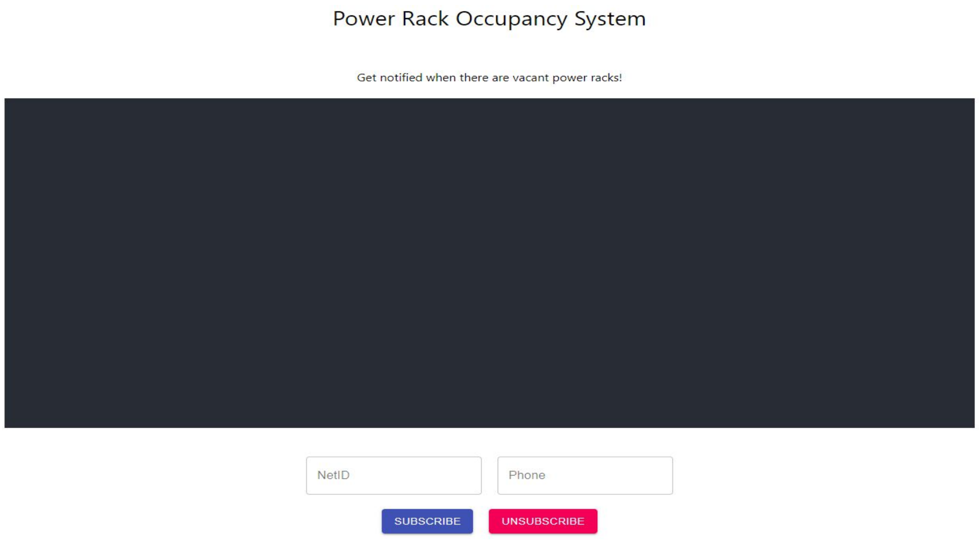
4 Progress made on First Project

After the design review for our first project, we completed the PCB design for the power supply subsystem and successfully interfaced with each of our sensors. The schematic and board layout are pictured. We succeeded in building the backend server framework for our push notification feature using the Twilio API. Whenever we detected a vacant power rack, we would scan through the user database and send a SMS text to each of the phone numbers associated with the users. SMS was confirmed to be sent and Google Cloud Functions were written to communicate with the user database.





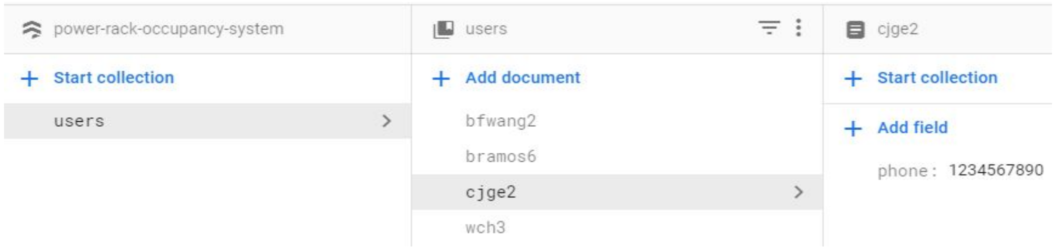
Frontend using ReactJS, allowing users to view available power racks and subscribe/unsubscribe to the notification system.



Power Rack Occupancy System

Get notified when there are vacant power racks!

User database using Google Firebase, keeping track of each user's netid and phone number.



bfwang2
bramos6
cjge2
wch3

phone : 1234567890

5 References

[1] *PubMed,* "Autism-related language, personality, and cognition in people with absolute pitch"*,* 2020. [Online] Available: https://www.ncbi.nlm.nih.gov/pubmed/?term=pitch+autism+brown+folstein. [Accessed: 3-April-2020].

[2] *Roadie,* "Press"*,* 2020. [Online] Available: https://www.roadiemusic.com/press. [Accessed: 3-April-2020].

[3]*Wikipedia,* "Electronic Tuner", 2020. [Online] Available: https://en.wikipedia.org/wiki/Electronic_tuner. [Accessed: 3-April-2020].

[4] *DigiKey,* "Electret Condenser (ECM) vs MEMS Microphone", 2017. [Online] Available: https://forum.digikey.com/t/electret-condenser-ecm-vs-mems-microphone/447. [Accessed: 3-April-2020].

[5] *ICMA,* "Realtime Chord Recognition of Musical Sound", 1999. [Online] Available: https://quod.lib.umich.edu/i/icmc/bbp2372.1999.446/1/--realtime-chord-recognition-of-musical-sound-a-system-using [Accessed: 3-April-2020].

[6] IEEE,"IEEE Code of Ethics", 2020. [Online] Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 3-April-2020].

[7] *ViceProvost,* "The Hazards of Lithium Batteries"*,* 2020. [Online] Available: https://viceprovost.tufts.edu//ehs/files/The-Hazards-of-Lithium-Batteries.pdf. [Accessed: 3-April-2020].

[8] *ABRI,* "Environmental Impacts of Lithium Ion Batteries"*,* 2020. [Online] Available: https://batteryrecycling.org.au/environmental-impact-of-lithium-ion-batteries/.[Accessed: 3-April-2020].

[9] *Guitar Gear Finder* " Everything You Need to Know About Guitar Sizes", 2018. [Online] Available: https://guitargearfinder.com/guides/everything-you-need-to-know-about-guitar-sizes/. [Accessed: 16-April-2020]

[10] *STMicroelectronics* "STM32F042x4 STM32F042x6", 2017. [Online] Available: https://www.st.com/resource/en/datasheet/stm32f042c4.pdf. [Accessed: 16-April-2020]

[11] *Guitar Tuning* “Guitar Tuning”, 2011. [Online]
Available: http://www.guitar-tuning.net/. [Accessed: 16-April-2020]

[12] *AZDEN* “Understanding Microphone Polar Patterns”, 2018. [Online] Available:
http://www.azden.com/blog/understanding-microphone-polar-patterns/.
[Accessed: 17-April-2020]

[13] *Wikipedia* “Machine head”, 2017. [Online]
Available: https://en.wikipedia.org/wiki/Machine_head. [Accessed: 17-April-2020]

[14] *WPI* “Assistive Guitar Plucking Device and User Interface”, 2019. [Online] Available:
https://web.wpi.edu/Pubs/E-project/Available/E-project-042419-130612/unrestricted/PAM_Band_MQP_Final_Paper.pdf. [Accessed: 17-April-2020]

[15] *Woodpecker Multimedia* “Why Guitars are Hard to Tune”, 2016. [Online] Available:
http://www.woodpecker.com/blogs/tuning_guitars.html. [Accessed: 08-May-2020]