A Mechanized Approach Towards Automated Pitch Correction

Dina Brustein
dina.brustein@lps-students.org

David Shustin
david.shustin@gmail.com

Ethan Wu
ethanwu10@gmail.com

Anna Xia
annaxia24@gmail.com

Daniel Chong*
junhodan@gmail.com

Forest Song*
fss26@scarletmail.rutgers.edu

New Jersey's Governor's School of Engineering and Technology
July 26th, 2019

*Corresponding Author

***Abstract*—Music can have many positive impacts on one's health, but it is often difficult to learn an instrument. This project aims to make learning music easier by creating a fast, accurate mechanized tuning device to decrease time spent tuning. The device developed in this project was designed specifically for a guitar. Research has found that this device would appeal to a variety of musicians, making them more confident that their instrument is in tune each time they play it and motivating non-musicians to learn music. To accomplish this objective, a CAD model consisting of a 3D printed peg holder powered by a micro servo was designed. This device was replicated in order to create a total of six tuning modules, which were then attached to the guitar in order to turn the pegs. A Fast Fourier Transform was used in conjunction with windowing functions and an algorithm for fundamental frequency detection in order to extract the frequencies from a user strumming six guitar strings one at a time. Each frequency was then determined to be sharp or flat, and the tuning unit was instructed to adjust the tension of each string accordingly. An additional feature was implemented by way of a web server wherein a user could input custom frequencies to tune the strings to, rather than adopting the standard tuning frequencies. This device was determined to tune faster than 33.0% of guitarists and can later be expanded to instruments other than the guitar.**

I. Introduction

Music education allows musicians, regardless of age, to enjoy many health and mental benefits. The brains of music students show stronger neural connections, more gray matter, higher IQ, better information processing, memory, attention, and motor coordination [1]. Additionally, music therapy, which includes learning to play an instrument, can physically and emotionally help patients with numerous diseases, including dementia, asthma, autism, and Parkinson's disease [2].

One of the factors deterring beginners from continuing to study music is the difficulty of tuning an instrument. A study of 139 participants conducted by the National Association for Music Education found that the average musician needs 4.5 years of experience in order to develop tuning independence [3]. Even for more experienced musicians, tuning takes time that can otherwise be spent practicing their instrument. Furthermore, during group lessons, the teacher must individually tune each student's instrument, which can take up a significant portion of the class time. A self-tuning instrument would therefore allow musicians to spend more time playing and grant them confidence that their instrument is in tune.

The objective of this project was to create an automated tuner that could tune faster than 50% of guitarists and be accurate within five cents of the desired note.

II. Background

A. Market Data

Musicians of string and non-string instruments as well as non-musicians participated in a survey that collected information related to tuning and the idea of a mechanized automated tuner. There were 272 responses in total after the survey was distributed across social media and guitar forums. Through the survey, it was discovered that 61.5% of 109 string instrument players and 87.7% of 106 players of non-string instruments strongly agreed or agreed that having a self-tuning instrument would make them more confident that their instrument is in tune. Furthermore, 61.5% of the string instrument players and 85.9% of the non-string instrument players agreed or strongly agreed that they would like a self-tuning instrument. Among 52 non-musicians who were not actively planning on learning an instrument, 65.3% agreed or strongly agreed that the availability of a self-tuning instrument would make them more likely to learn an instrument. Overall, this data demonstrated significant market demand for an automatic instrument tuner. The full data collected during the survey is found in Appendix A.

B. Music Theory

The spectrum of frequencies audible to the human ear is grouped into the notes C, C♯, D, D♯, E, F, F♯, G, G♯, A, A♯, and B. These notes can occur in multiple octaves, which are numbered from 0 to 9. Therefore, the notes are subscripted with the octave in which they occur, e.g. D_2 or $\mathrm{G}\sharp_4$. The frequency of a note in one octave is equal to half the frequency of the note in the next higher octave.

The difference in frequency between two adjacent notes is referred to as one semitone. Semitones can also be converted into cents, with 100 cents being equal to one semitone. Cents are often used to determine how sharp or flat a note is or the extent to which a pitch is out of tune. Cents can be derived from Equation 1. In the equation, n represents the number of cents by which a note is out of tune, f_1 represents the desired frequency, and f_2 represents the current frequency of the note.

$$n = 1200 \cdot \log_2 \left(\frac{f_2}{f_1} \right) \tag{1}$$

When a note is played on any instrument, the instrument emits a range of several frequencies, one of which is the note's fundamental frequency, the lowest peak emitted in the signal. The other frequencies emitted are harmonics (overtones of the fundamental frequency), which are integer multiples of the fundamental frequency. Depending on the instrument, inharmonic overtones (frequencies higher than the fundamental which are not integer multiples of the fundamental) may also be emitted. String instruments generally have overtones which are very close to harmonic [4]. An automated tuning device will therefore need the ability to distinguish between the fundamental frequency, overtones, and harmonics within a string's auditory spectrum.

The particular relative amplitudes of each of the harmonics constitutes the acoustic "signature," or timbre, of a given instrument. Timbre depends on the emitted sound's harmonic spectrum and each harmonic's timing of appearance [5]. In some instruments, the fundamental frequency may actually be lower in amplitude or absent all together. However, the human brain is still able to perceive it as the same pitch in the correct octave [6]. Therefore, the strongest frequency in the spectrum produced by a string is not necessarily the fundamental frequency.

C. Frequency

String instruments like the guitar, violin, viola, cello, and harp create sounds through the vibrations of their strings. The pitch of a note played on a string instrument corresponds to the frequency of the standing wave created by the string's vibrations. Equation 2 displays this relationship with f as the frequency, T as the tension of the string, L as the length of the string, and m as the string's mass.

$$f_1 = \frac{\sqrt{\frac{T}{m/L}}}{2L} \tag{2}$$

When tuning a string instrument, there are pegs on the instrument which, when rotated, increase and decrease the tension, T, in the string, therefore increasing or decreasing the frequency of the note played when that string vibrates. For this project, the automated tuner was tested on a six stringed guitar with the standard notes of E_2, A_2, D_3, G_3, B_3, and E_4, with frequencies of 82.41 Hz, 110.00 Hz, 146.83 Hz, 196.00 Hz, 246.94 Hz, and 329.63 Hz, respectively.

D. Alternate Guitar Tunings

Certain chords on guitar are difficult to play because the notes they consist of are physically far apart on the guitar, so it is difficult to reach every point on the guitar that must be compressed in order to play the chord. To combat this, guitarists sometimes use alternate tunings, where they tune each string to a frequency other than its standard fundamental frequency. One popular example is Drop-D, where the low E string is tuned to a D note instead of the standard E. This tuning allows guitarists to play power chords [7]. Due to the prevalence of these tunings, several surveyed guitarists commented in the survey that they would not be interested in an automatic tuner unless it could tune to custom tunings.

E. Proportional-Integral-Derivative (PID) Controller

Controllers that manage output typically use feedback loops to make their measured variable, or process variable, reach a specific target value, or setpoint. The most common feedback algorithm, used in over 90% of all control loops, is the proportional–integral–derivative controller (PID) [8].

PID works by multiplying the current error, $e(t)$ (which is computed in Equation 3) using the setpoint $r(t)$ and the process variable $y(t)$), the cumulative error, and the rate of change of the error by constants chosen for the specific application, K_p, K_i, K_d, respectively. The sum of these three terms at time t computes the output of the controller, $u(t)$.

$$e(t) = r(t) - y(t) \tag{3}$$

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,\mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{4}$$

Equation 4 describes a continuous PID controller. However, when working with a computerized loop with time interval Δt, the PID controller actually runs in discrete time, for which continuous time is only an approximation as Δt approaches 0. The discrete PID controller is better modeled by the sequence described in Equation 7, where n is current number of the loop, e_n is the error in the nth loop, and t_n is the time at the start of the nth loop.

$$\Delta t_n = t_n - t_{n-1} \tag{5}$$

$$e_n = e(t_n) \tag{6}$$

$$u_n = K_p e_n + K_i \sum_{k=0}^{n} e_k \Delta t_k + K_d \frac{e_n - e_{n-1}}{\Delta t_n} \tag{7}$$

However, in both the continuous and discrete versions of PID, the "I" term, represented in Equation 7 as $K_i \sum_{k=0}^{n} e_k \Delta t_k$ can "wind up" and reach very large values as an error persists over time. For this reason, the I term is clamped between an upper and lower bound. Setting a maximum and minimum value for the I term prevents the system from winding up over time.

Another problem presented by the I term is the possibility of overshoot. Once the process variable reaches the setpoint, there may still be a considerable value of the I term resulting from a history of error, causing the system to overshoot. Therefore, a constant is set called a tolerance. If the process variable is less than the tolerance away from the setpoint, the I term is set to zero to prevent overshoot.

By choosing the correct tuning constants (K_p, K_i, and K_d) and good tolerance and clamping constants in Equation 7, unreliable, lagged, and turbulence-prone outputs can reach and hold a measured variable to a setpoint.

F. Digital Signal Processing

1) Fourier Transform: A note played by an instrument is not a perfect sine wave. The harmonies that real-world instruments produce can be represented as a sequence of sinusoidal functions with varying frequencies. The sum of the harmonies present in the sound of an instrument's note is the note itself. Therefore, a note can be represented by a Fourier series of its harmonics [9].

In order to perform useful analysis of sound waves, it is necessary to have a way of decomposing a signal into its harmonies. This task is accomplished with a Fourier transform, a mathematical operation that maps a function of time or signal into a function of frequency [10]. Equation 8 shows this definition, where $\hat{f}(\omega)$ is the Fourier transform of the signal $f(t)$ [11].

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} e^{-2\pi i \omega t} f(t)\, \mathrm{d}x \tag{8}$$

The Fourier transform defined in Equation 8, however, describes an operation on continuous functions. Discrete functions, such as those describing the sequence of samples that make up a digital audio file, cannot be manipulated with the Fourier transform. Instead, they are processed with a version of the Fourier transform that can be called on discrete functions, aptly named a Discrete Fourier Transform (DFT). An optimized version of the DFT, a Fast Fourier Transform (FFT), can perform the same function faster. [10].

However, there are limitations to the frequencies that can be detected by a Fourier transform. The Nyquist theorem states that the sampling frequency must be at least twice the frequency of the highest frequency in the signal. Any higher frequencies appear as a lower frequency signal where $f_c \leq \frac{f_s}{2}$, f_c is the frequency of the aliased peak, and f_s is the sampling frequency [12].

2) Windowing Functions: A Fourier transform essentially assumes that the ends of the signal are "connected" to each other. If there are discontinuities created when transitioning between the beginning and end of the signal, they manifest themselves in the frequency spectrum as a frequency well above the Nyquist limit, aliasing into the range of the signal between 0 and half of the sample rate. Therefore, frequencies will appear in the Fourier transform to be "spread out" over multiple frequencies, a phenomenon known as spectral leakage [10].

To alleviate the spectral leakage, windowing can be used to eliminate the discontinuity by decreasing the amplitude of the signal at its two ends smoothly down to zero or close to zero. Specific windowing functions, however, may also change the frequency content of the signal by introducing additional frequencies. The frequency response section of each windowing function contains side lobes that show these additional frequencies and their amplitudes which are introduced for each frequency in the original signal. In addition, the windowing functions also affect the amplitudes of each frequency in the original signal. Based on these characteristics of the individual windowing functions, a specific windowing function suited to the task at hand must be selected [10].

3) Fundamental Frequency Detection: Many algorithms have been developed for estimating the fundamental frequency of a tone from a digital signal. There are algorithms that operate both in the time domain and the frequency domain.

One time domain algorithm is autocorrelation, where the signal is compared with a delayed version of itself, searching for the delay values where there is the largest amount of correlation between the two signals. Since the frequencies produced by a harmonic tone source are all periodic with a frequency equal to the fundamental frequency, the lowest delay showing a high correlation is likely the fundamental frequency [13]. The method for calculating the correlation can differ, and is typically an average of the differences between the signal and its delayed counterpart. A disadvantage of autocorrelation methods is that they tend to perform poorly on sounds composed of multiple tones (multiple different notes being played simultaneously, each with their own fundamental frequency and harmonics).

Another algorithm for fundamental frequency detection, harmonic product spectrum (HPS), operates in the frequency domain. HPS repeatedly compresses the present frequencies by a factor of sequential integers. The algorithm then multiplies the compressed frequency domain with less compressed versions of the same frequency domain, performing a multiplicative overlay. The nth harmonic, f_n is computed as the fundamental frequency, f_0 multiplied by an integer, $f_n = n \cdot f_0,\ n \in \mathbb{Z}$. Therefore, throughout each division of the frequency domain by an integer, a harmonic will align itself with the original fundamental frequency, causing a peak of constructive interference at the fundamental frequency and destructive interference at other frequencies [13]. As a result, peaks remaining in the spectrum after many multiplicative iterations correspond to the fundamental frequencies present in the signal.

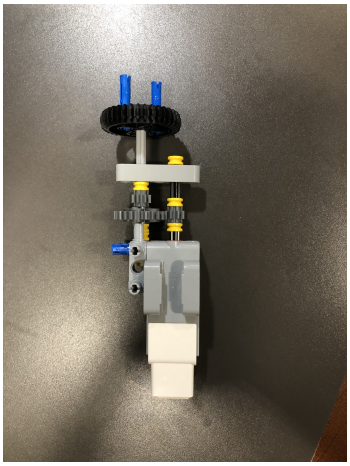

Fig. 1. The LEGO® EV3 Medium Servo Motor prototype

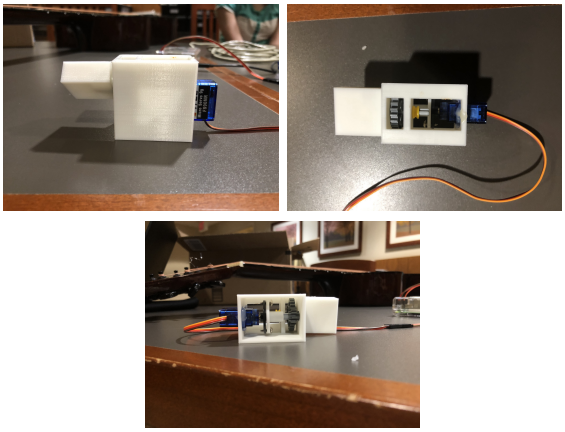

Fig. 2. The first prototype

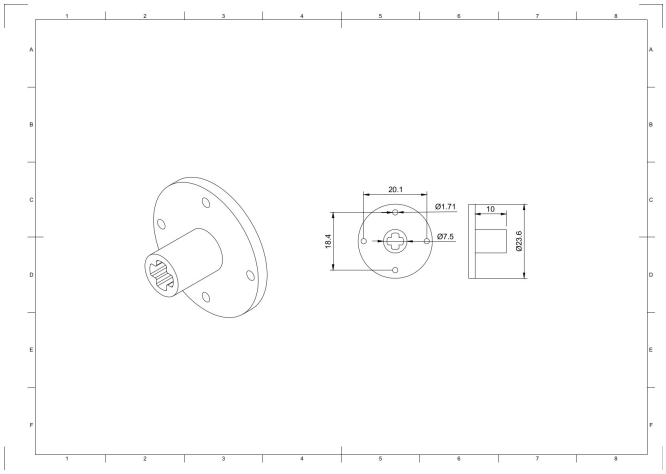

Fig. 3. The servo horn adapter

G. Software

Autodesk Fusion 360 computer aided design (CAD) software was used to create a three-dimensional model of the tuning units and to design the 3D printed parts of the tuning units. Autodesk Fusion 360 was also used to create two-dimensional drawings of the tuning units.

III. Experimental Procedure

A. Physical Components

The automated tuning device (henceforth referred to as the device) is composed of six tuning units connected together to the head of the guitar, a Raspberry Pi with a microphone placed near the guitar, and a USB battery bank to power the Raspberry Pi.

B. Prototyping

In the first steps of the design process, several proofs of concept were built to simulate the proposed design. Experiments were done with 1:1, 3:1, and 9:1 gear ratio on a motor with similar stall torque to the FEETECH FS90MR Servo, the LEGO® EV3 Medium Servo Motor [14], [15]. A gearbox was prototyped on the motor, and gears were added and removed to reach the desired gear ratio.

Whereas the 1:1 gear ratio was too weak to turn the most resistant peg, the low E string's, the 3:1 and 9:1 gear ratios were able to turn the peg. The 3:1 gear ratio, shown in Figure 1, was adopted because of it was more compact and required less complexity than the 9:1 gearbox while achieving the same goal.

After the design was completed, the first prototype of the tuning unit was printed and constructed. The initial design did not include a connection mechanism or attachment point; it consisted of only the enclosure, the hot dog, and the relevant parts for turning the peg.

Assembly was impaired by some mistakes in the CAD. The servo attachment holes were made too large due to human error, so instead of using the proper screws to attach the servo, the servo was attached by pressing it against hot glue on the attachment point.

Once the servo mount was fixed in the CAD model, the remainder of the assembly process continued without trouble. The completed prototype can be seen in Figure 2.

C. Physical Assembly

In order to design a device capable of turning the pegs of a guitar, research was done into an assortment of motors and servos to find one that fits between the pegs of a standard guitar while providing an ample amount of torque. The FEETECH FS90MR metal geared continuous rotation servo was chosen to drive the mechanism. The transmission's gears, bushings, and axles were sourced from the LEGO® Technic set while the other parts of the assembly were created with Autodesk Fusion 360 CAD software and manufactured using an Ultimaker 3 3D printer with PLA filament.

A custom 3D printed piece, shown in Figure 3, was designed to allow for the connection of the axle and the servo horn. This piece featured a $23\,\mathrm{mm}$ diameter disk and a $10\,\mathrm{mm}$ high hollow cylinder that fits a LEGO® axle on the inside.

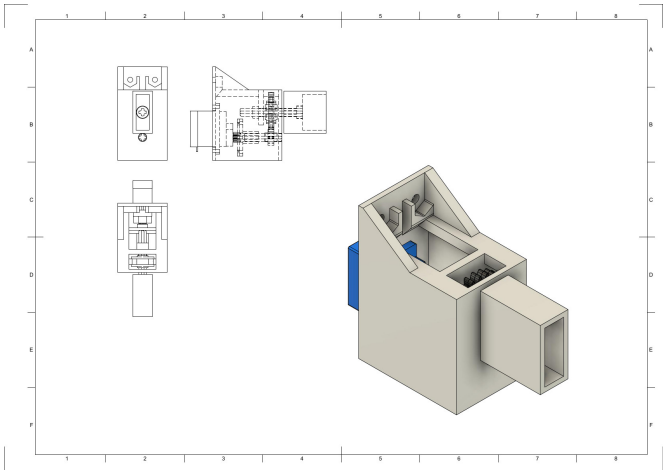

Fig. 4. A fully assembled tuning unit

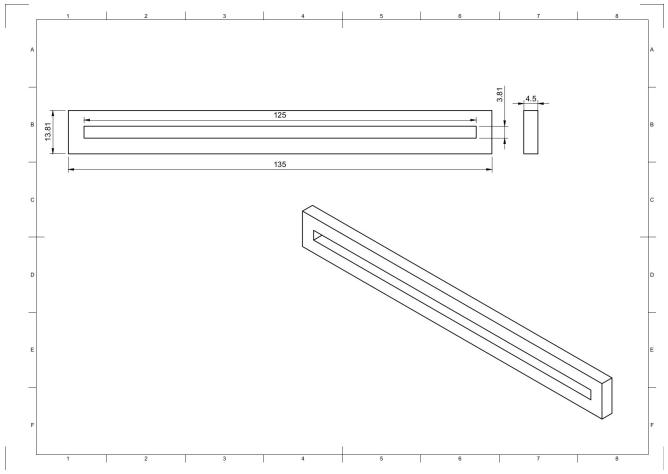

Fig. 6. The rail

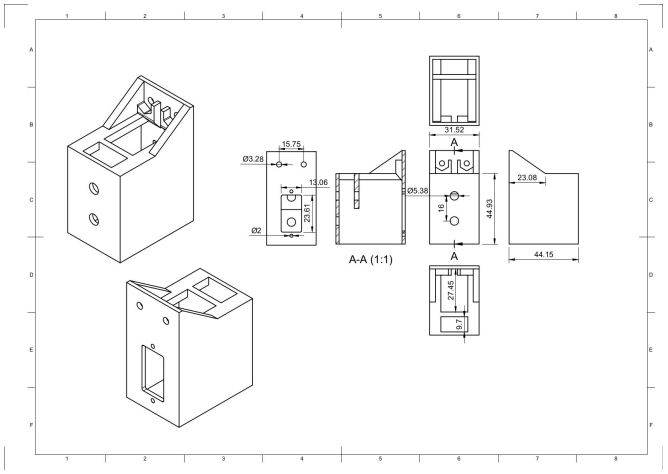

Fig. 5. The enclosure, its isometric views, and a cross-section

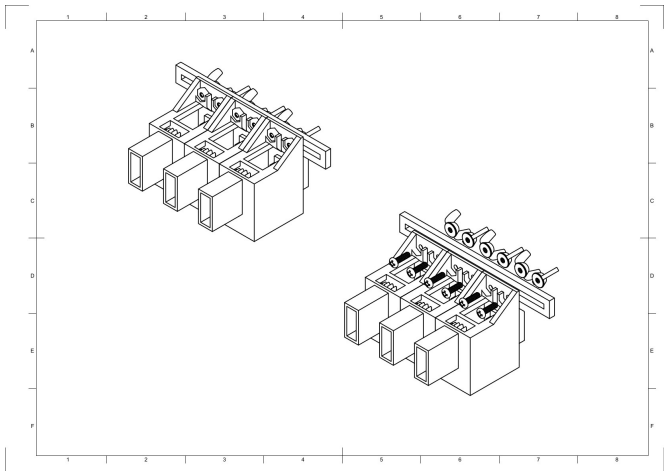

Fig. 7. A complete assembly of 3 tuning units connected by a rail

The design was split into repeatable modules called "tuning units," shown in Figure 4, to accommodate guitars of various geometries.

The transmission was designed as a gearbox made of an 8-tooth gear driving a 24-tooth gear. This transmission achieves a 3 to 1 gear ratio, multiplying the servo's output torque and dividing its speed by a factor of 3. By gearing the servo's speed down, the gearbox has finer control while providing sufficient strength to turn the guitar's tuning peg. The transmission was housed in a 3D printed enclosure with a hole cut out for the servo, along with two more for its mounting screws.

A wall was extruded to provide a second point of contact for the top axle. Two holes were cut from the front to allow the axles to fit through. The top and bottom remained open, allowing the inner assembly to be more accessible and providing enough space for the gearbox. Another element was added to the top of the enclosure with two holes. The holes can be used to pin the modules together, modularizing the system.

Each tuning unit controls one peg on the guitar. Six units were 3D printed along with a connection mechanism to link all six tuning units and mount them on the guitar. The connection mechanism is a rectangular rail, pictured in Figure 6, and it was designed to connect the units into groups of three, with one group on each side of the guitar's head. Another rail of a longer length was designed to accommodate 6 pieces on the same side to allow the tuning of guitars with all pegs on one side.

The connection method can be used in two different ways. Both methods connect each enclosure to the rail using screws. The enclosure design fits hex nuts around its mounting points, allowing the screw to first fit through the rail, into the enclosure, and be enclosed in a nut. The second method, a more user-friendly option (pictured in Figure 7), is to thread the screw through the enclosure first, then through the rail, capping it with a wing nut on the rail's side.

Another 3D printed piece was then designed to grip onto the peg itself. The resulting piece was named the "hot dog" because of its shape. The hot dog, pictured in Figure 8, supports its guitar peg on four sides, rotating it to change the string's pitch. The other end features a hole that can fit a

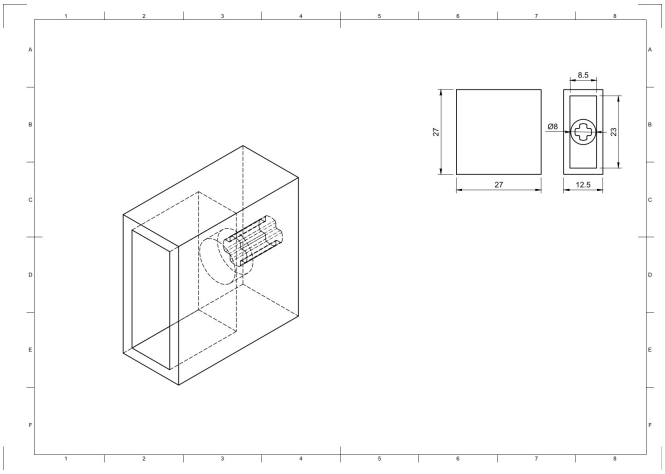

Fig. 8. The hot dog

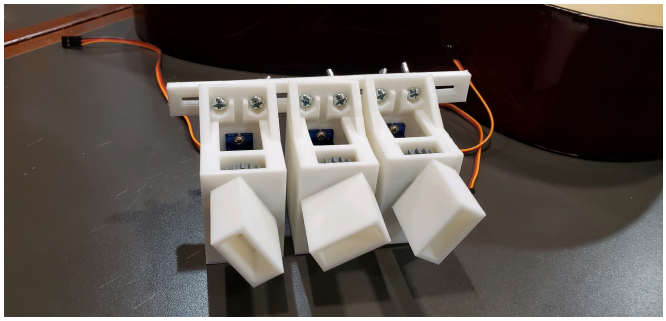

Fig. 9. The completed assembly

LEGO® axle and a corresponding bushing to connect it to the enclosure.

D. Final Mechanism

Once the first prototype was proved to have tuning capabilities, the attachment mechanism depicted in Figure 6 and Figure 7 was created. The new version was printed and assembled with 3 tuning units connected by a rail.

Due to human error in the CAD, the holes were made slightly too small to fit the screws. The holes were widened with a screwdriver and the CAD was corrected for future prints. Once the tuning units were held together, as in Figure 9, the assembly was attached to the guitar's head with velcro as shown in Figure 10.

E. Control Software

The software that controls the device consists of three major components: the web user interface, which has JavaScript running in the user's mobile device and a web server running on the Raspberry Pi, the pitch detector program, and the motor control program. The API backend to the web user interface directly calls functions on the pitch detector program, which runs in the same process. The pitch detector program then communicates to the motor control program via a Unix named pipe.

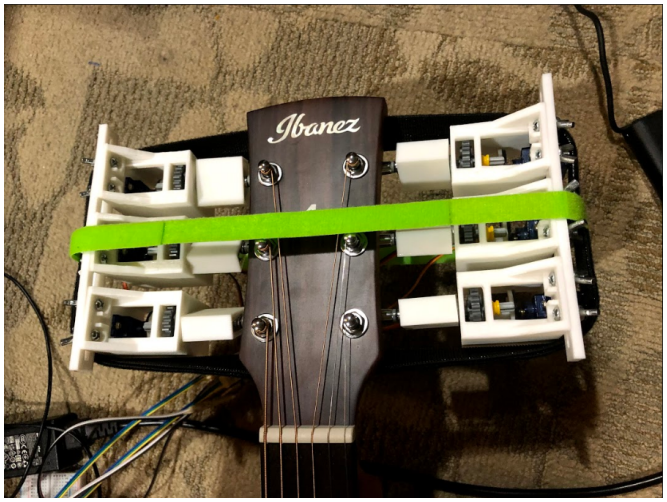

Fig. 10. The completed assembly attached to the guitar head

The continuous rotation servo motors were controlled by from the Raspberry Pi via a C++ program utilizing the C library PiGPIO, used for sending the servo control pulses. The motor control program reads how out of tune each string is in cents from the pitch detector program, and uses it to feed a PID controller, as shown in Equation 7, for each string. Because the process variable, $y(t)$, is the difference in cents between the target frequency and the current frequency, the PID controller aims to minimize the absolute value of the process variable. Therefore, the setpoint, $r(t)$ is equal to zero. The error can then be calculated using Equation 9.

$$e_n = -y(t) \tag{9}$$

The error passes through the PID controller, and the output is used to set the speed of the servos. If a string is detected as not being played, the program commands that servo to stop and hold position.

The pitch detector program runs the pitch detection algorithm, the API backend to the web user interface, and the code responsible for acquiring audio from the microphone connected to the Raspberry Pi each in different threads. For reading audio, pyalsaaudio was used to get audio data through the Advanced Linux Sound Architecture (ALSA) interface. 16 periods of 1024 samples are read before passing the entire 16384-sample buffer over to the pitch detection algorithm, running in a separate thread.

The web user interface server consists of the web server nginx, which serves pre-built files generated by Gatsby for the app and forwards API requests to the Flask-based backend server which communicates with the pitch detector program.

F. Pitch Detection Algorithm

The audio buffer received from the microphone on the Raspberry Pi is first multiplied with a Hanning window using the SciPy library. Next, a real Fourier transform is applied using NumPy. A crude signal-to-noise metric is then calculated

to determine if the guitar is currently playing. This metric is calculated by taking the difference between the minimum and maximum intensities of the frequency spectrum within a frequency window of between 100 and 500 Hz. If the signal-to-noise ratio in this frequency band is below $42\,\mathrm{dB}$, then it is assumed that the guitar is not being strummed, and the device's motors are commanded to switch off.

The frequency spectrum is then subtracted with a "noise profile" generated beforehand to eliminate background self-noise from the microphone and sound system. This profile is generated by sampling 5 buffers from the microphone after waiting through 5 buffers for the microphone to start up. The profile is then saved to disk to be read in for this noise-cancellation step.

Three iterations of the HPS algorithm are then applied: the frequency spectrum is multiplied with versions of itself which are compressed on the frequency axis by factors of 2, 3, and 4. The amplitudes of the processed frequency spectrum are then converted to dBFS.

Frequency bins are then constructed based on the target frequencies that each string is supposed to be tuned to and the first harmonic overtone. Any harmonics that are within 5 cents of a fundamental frequency or overtone of another string are ignored. The points in the frequency spectrum obtained from the Fourier transform are then sorted by the amplitude of the frequency. They are then iterated through in order of descending amplitude, and each of these "peaks" is then assigned to the closest frequency bin. If the frequency of the peak is more than 200 cents away from the frequency of the bin, that peak is not considered. If there already has been a peak assigned to that bin, then the new peak is also not considered because it is of lower amplitude than the peak that was already assigned. Once all frequency bins have been filled, the iteration stops. If the amplitude of the peak is more than $24\,\mathrm{dB}$ below the loudest peak, it is also discarded and the iteration also stops. The bins corresponding to any overtones of strings are then discarded, and the remaining frequency bins contain the fundamental frequencies corresponding to each guitar string that was plucked.

The results of this process on a strum of all six strings of a guitar are shown in Figure 11. Each dashed vertical red-orange line shows the identified pitch of the identified strings, and the dotted green and black lines show the locations of the frequency bins, where the green lines show the bins corresponding to the correct fundamental frequencies of the guitar strings and the black lines show the overtones of those strings.

Each of the peak locations for detected frequencies are then refined using a parabolic interpolation, implemented with an interpolation function found in a sample implementation of pitch detection algorithms found on GitHub [16].

Once the fundamental frequencies of each string are detected, they are then compared against the reference tuning and their pitch offset in cents is then calculated. This pitch offset is then sent to the PID controller as the process variable, as described previously.

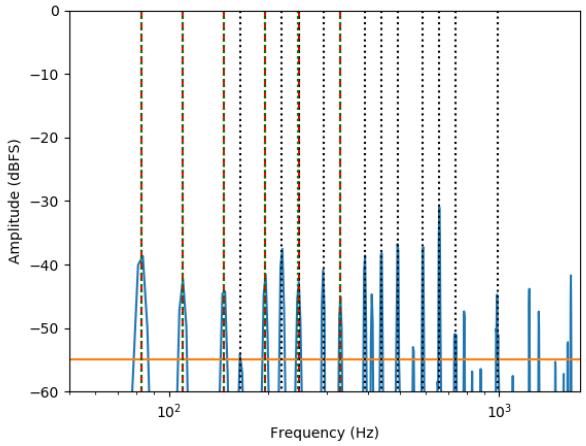

Fig. 11. FFT of guitar strum with overlaid output of frequency bin based pitch detection

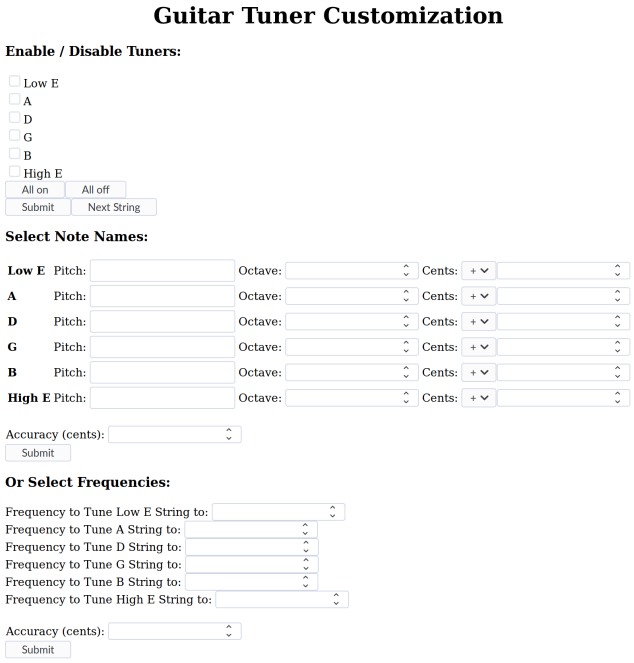

Fig. 12. User interface for tuner customization

G. Web Application Development

A web application, shown in Figure 12, was developed to configure the tunings of the device. The front end was first designed in HTML using React and Gatsby, with text boxes for the user to enter notes or frequencies, as well as the desired accuracy of the mechanism, measured in cents. The notes were then converted to frequencies using JavaScript code. The resulting frequencies were transmitted, along with the requested accuracy, to the mechanism, and this communication was developed using the Axios HTTP request library.

H. User Instructions

In order to set up the device, users must first align the tuning modules in order to reflect the spacing between pegs on their guitar. They must do this only when tuning a guitar for the first time because the connection system allows the user to lock the modules into the correct position.

To tune their instrument, users must input their desired frequencies into the web application. They must then attach each tuning module to a tuning peg. Next, they must individually select what string they are tuning on the web application and pluck that string continuously until the module for that string stops moving.

I. Device Testing

The speed and accuracy of the device was tested by detuning the guitar and recording the frequency that each detuned string emitted. Then, the tuning program was executed as the user plucked each string individually. On the web application, the user was able to disable servos that the user was not currently using to tune. When all the strings were in tune according to the G Strings mobile tuning application, the frequencies of the tuned strings were recorded. The time to tune all the strings was also recorded.

Because one servo was defective, only five strings were tuned. The time to tune all six strings was extrapolated by multiplying the time taken to tune five strings by $^6/_5$.

IV. Results

A. Device Testing

After ten trials, it was determined that the tuner took an average of 95.5 seconds to tune five strings, which was extrapolated to a time of 114.6 seconds to tune all six. This was faster than 33.0% of string instrument players, although the majority of the musicians in the survey were intermediate and advanced, while this device's target audience is comprised of beginners. Furthermore, in a group class, the tuner may become more viable because students can tune their guitars simultaneously instead of the teacher having to tune each guitar in succession.

The absolute error for each trial is seen in Figure 13.

The trials showed that lower strings were generally tuned more quickly and accurately than higher strings, so it is likely that had the servo been functional, the extrapolated time would have been an overestimate because the simple extrapolation method assumes that all strings take the same amount of time to tune.

B. Failure Analysis

Initially, the tuner was intended to tune all six strings at once. The quality of the thresholding function used, however, made the tuner's ability to detect all six peaks inconsistent. The threshold function had a high failure rate, either preventing the peaks corresponding to frequencies of other strings to be detected or classifying too many noise peaks as real data. In addition, harmonics sometimes had higher amplitudes than

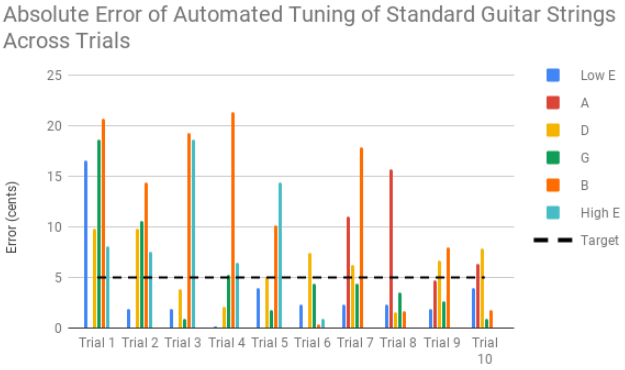

Fig. 13. Graph displaying absolute error of automated tuning (excluding the A string)

TABLE I
PID Parameters

Parameter	Value
K_p	0.75
K_i	0.0004
K_d	20

their fundamental frequencies, causing them to be detected rather than the correct peak.

The quality of the servos was another source of difficulty. When the servos were powered, it was discovered that because of the Raspberry Pi's limited GPIO timing control, the servo could only be driven at 5 speeds in either direction. This limitation was far from the optimal continuous speed range, and presented an obstacle to development of the tuning control. With the correct PID tuning (shown in Table I), however, a tuning accuracy of ± 3 cents was achieved. Furthermore, one servo was defective.

C. Digital Signal Processing

1) Transformation to Frequency Domain: For the Fourier transform, a Blackmann-Harris windowing function was initially chosen based on sample code for various pitch-detection algorithms [16]. It was soon apparent, however, that the windowing function was introducing significant spectral leakage at inharmonic frequencies that were close to the harmonics and fundamentals of certain other strings. Instead, the Hann windowing function was chosen for its small side lobes, which resulted in minimal additional inharmonic spectral content being added [10].

2) Microphone Noise: The microphone chosen to be used on the Raspberry Pi exhibited significant amounts of self-noise (noise intrinsic to the microphone and its associated analog-to-digital conversion circuitry), including a specific peak very near to the frequency of a D_3. This peak was shown to interfere with the detection of the fundamental frequency of the guitar's D string as it is also tuned to a D_3. Therefore, the noise cancellation step was developed.

3) Other Considered Pitch-Detection Algorithms: Originally, the HPS algorithm was not employed for simplicity and

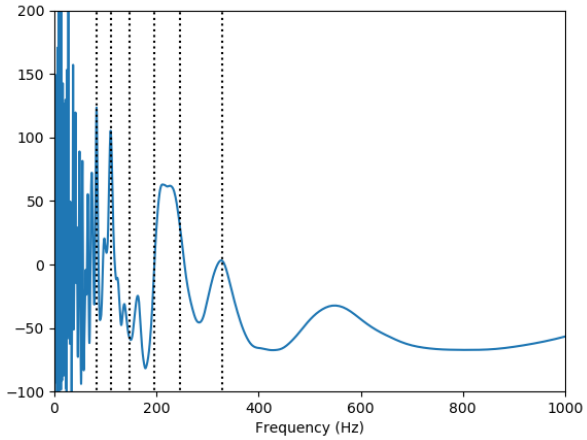

Fig. 14. Autocorrelation output on strum of guitar, with actual frequencies of strings overlaid

the frequency bin categorization algorithm was programmed to search for additional overtones. However, this proved to be rather unreliable in differentiating between fundamental frequencies and harmonic overtones. The HPS was instead implemented to attenuate the amplitudes of these overtones. It also helped greatly reduce the amplitudes of background noise and decreased the width of the peaks corresponding to frequencies emitted by the guitar strings.

An autocorrelation fundamental frequency detection algorithm was not used because autocorrelation based techniques struggled with separating the pitches of the multiple guitar strings playing simultaneously. This is likely because the periodicity of an individual tone becomes much less pronounced when played in combination with another tone. Since autocorrelation depends on observing the period of periodic parts of the signal, this made it difficult for the autocorrelation to find all of the fundamental frequencies of the various strings with any accuracy. Figure 14 shows the output of the autocorrelation on an audio clip of all six strings of a guitar being strummed. Each of the dotted black lines is the actual frequency of one of the strings; it is clear that although 3 strings (the two lowest frequency strings and the highest frequency string) are detected with a fair degree of accuracy, the remaining three strings show no distinguishable peaks and thus no clear correlation at their respective frequencies.

V. Conclusion

Overall, the device had an average tuning time of 114.6 seconds to tune all six strings, which was faster than 33% of string instrument players from the survey. While the goal was for the device to tune faster than 50% of string instrument players from the survey, a majority of the respondents to the survey were intermediate or advanced and the device is geared towards beginners. As the learning curve progresses, the device accuracy will improve as well.

Future works include developing a sturdy structure to mount the tuning units, Raspberry Pi, and wiring to the guitar to make the system more compact. In order for this product to be marketable to users, it would need a better thresholding function to make it more consistent, as well as the aforementioned changes to make it more professional. It would also need to be able to stop servos automatically, as it places a large burden on users to force them to decide when the tuner is done and when it is still going.

One way to drastically reduce tuning time would be to tune all six strings simultaneously, rather than one at a time. This would be feasible with an improved thresholding function.

This device can also be expanded to service a variety of stringed instruments in addition to guitars. The device can especially be adapted to all instruments in the guitar family without radically changing its structure.

Outside of the string family, the survey in Appendix A demonstrates that wind, percussion, and other non-string musicians are even more interested in an automatic tuner than those who play string instruments. Piano in particular is an instrument that is very difficult to tune. Many pianists have to hire tuners in order to tune their instruments, which, on average, takes 1.5 hours and costs about $115 [17], [18]. In the survey, an overwhelming majority—88.5%—of pianists agreed or strongly agreed that they would like a self-tuning instrument. Among musicians who played instruments other than piano or string instruments, 82.2% agreed or strongly agreed. Both these figures are far higher than the 61.5% of string musicians who agreed or strongly agreed.

Appendix A
Survey Data

The following questions were asked in the survey.

Question 1 (asked of all respondents): What instrument do you play? (Figure 15)

Question 2 (asked of all musicians): How would you describe your skill level with your instrument? (Figure 16 and Figure 17)

Question 3 (asked of all string musicians): How long has it been since you last tuned your instrument?

Question 4 (asked of all string musicians): Please tune your instrument (you can use any kind of tuning technology you want) and time how long it takes you to do so. Submit your time as the answer to this question. (Figure 18)

Question 5 (asked of all non-musicians): Are you actively planning to learn an instrument?

Question 6 (asked of all respondents): Which region best describes your geographic location? (Figure 24)

Question 7 (asked of all respondents): What age group are you in? (Figure 25)

Respondents were also asked to rate their level of agreement with the following statements.

Statement 1 (for musicians): I would like a self-tuning instrument. (Figure 19 and Figure 20)

Statement 2 (for musicians): If I had a self-tuning instrument, I would be more confident that my instrument would be in tune each time I played it. (Figure 21 and Figure 22)

Statement 3 (for non-musicians who are not actively planning to learn an instrument): The availability of a self-tuning instrument would make me more likely to learn an instrument. (Figure 23)

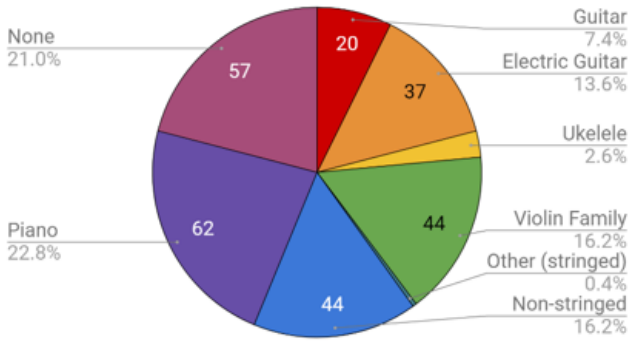

Fig. 15. Responses to Question 1

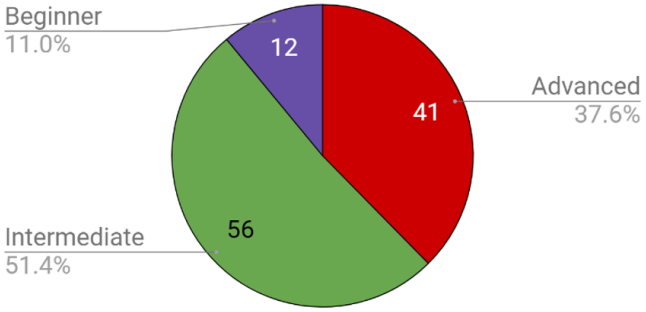

Fig. 16. String instrument players' responses to Question 2

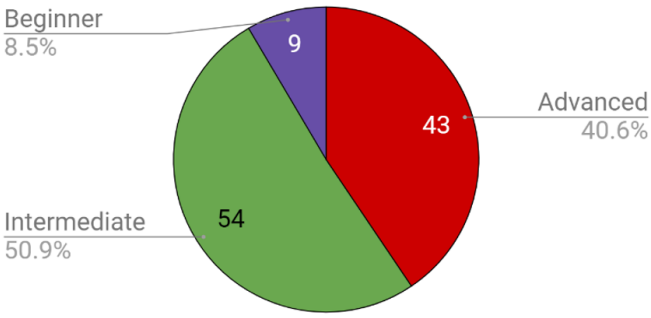

Fig. 17. Other musicians' responses to Question 2

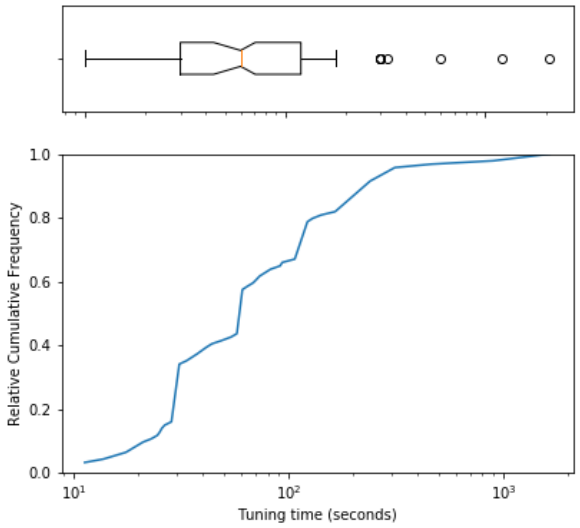

Fig. 18. String instrument players' responses to Question 4

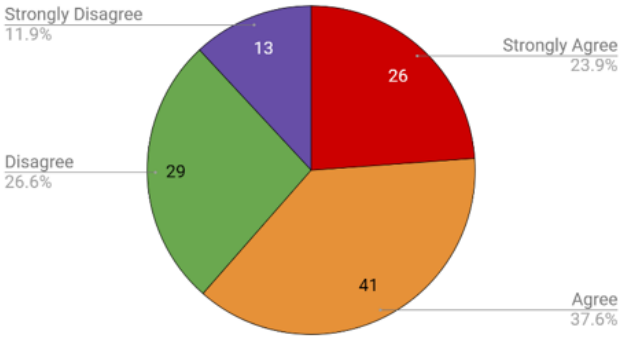

Fig. 19. String instrument players' responses to Statement 1

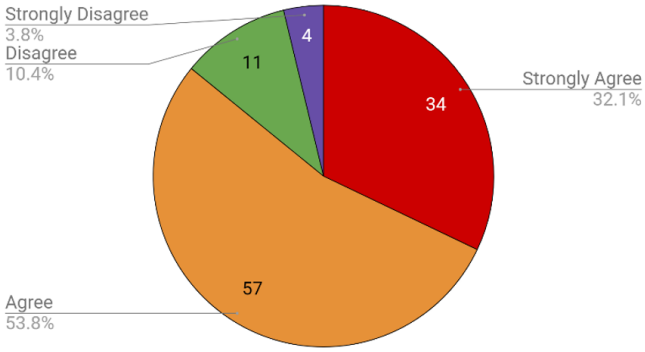

Fig. 20. Other musicians' responses to Statement 1

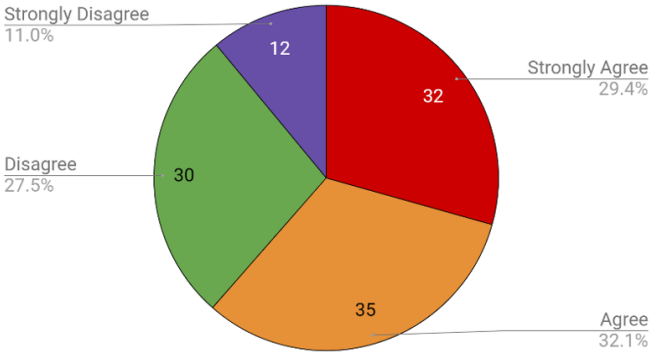

Fig. 21. String instrument players' responses to Statement 2

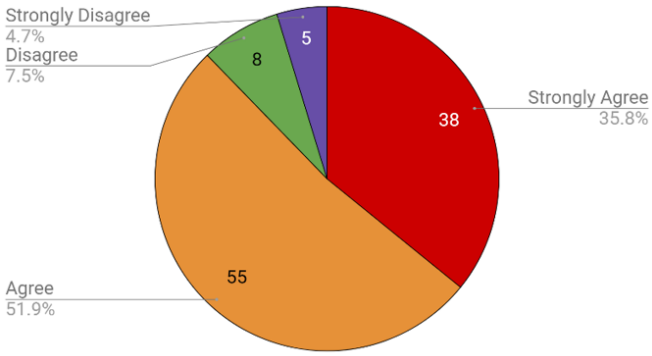

Fig. 22. Other musicians' responses to Statement 2

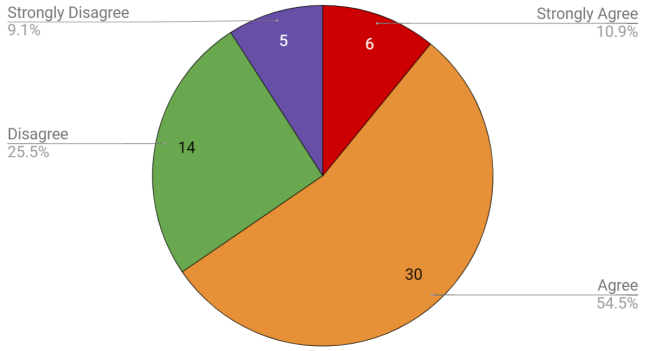

Fig. 23. Responses to Statement 3 by non-musicians who are not planning on learning an instrument

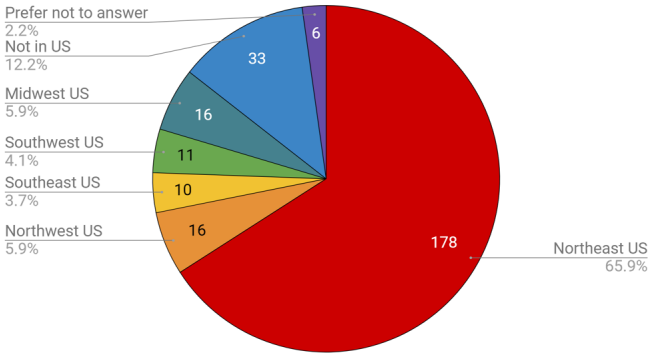

Fig. 24. Responses to Question 6

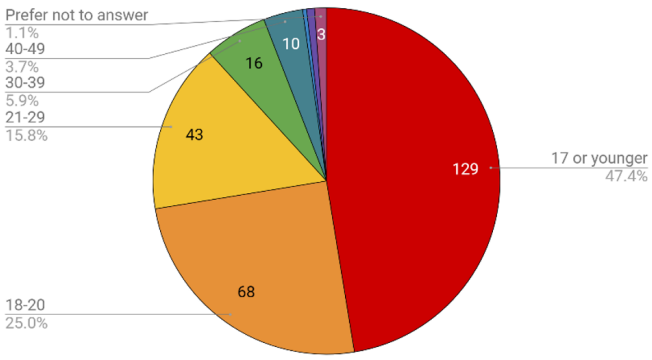

Fig. 25. Responses to Question 7

ACKNOWLEDGMENTS

The authors of this paper gratefully acknowledge the following: project mentor Forest Song and project liaison Daniel Chong for their valuable engineering expertise and mentorship; Andrew Page for providing vital building materials; Residential Teaching Assistant Benjamin Lee for volunteering his guitar to test on; Professor Roy Yates for his invaluable feedback on data collection; Evan Dogariu for his experience and advice on signal processing; Dean Jean Patrick Antoine, the Director of GSET, for his management and guidance; Head Residential Teaching Assistant Michael Higgins and Research Coordinator Helen Sagges for their guidance on conducting proper research; Rutgers University, Rutgers School of Engineering, and the State of New Jersey for the chance to advance knowledge, explore engineering, and open up new opportunities; the Rutgers Makerspace for access to their 3D printer; Lockheed Martin, the New Jersey Space Grant Consortium, and other sponsors for their funding of our scientific endeavors; and lastly, GSET alumni for their continued participation and support.

REFERENCES

[1] The Royal Conservatory for Music, "The benefits of music education," March 2014.
[2] American Music Therapy Association, "Definition and quotes about music therapy," 1998-2019.
[3] M. T. Hopkins, "Teachers' practices and beliefs regarding teaching tuning in elementary and middle school group string classes," *Journal of Research in Music Education*, vol. 61, no. 1, pp. 97–114, 2013.
[4] J. Wolfe. How harmonic are harmonics? The University of New South Wales.
[5] P. J. Donnelly and J. W. Sheppard, "Classification of musical timbre using Bayesian networks," *Computer Music Journal*, vol. 37, no. 4, pp. 70–86, 2013.
[6] B. H. Suits. The missing fundamental. Michigan Tech Physics.
[7] L. Zucker. (2015, April) Lesson: The advantages of alternate tunings.
[8] K. Åström and T. Hägglund, "The future of PID control," *Control Engineering Practice*, vol. 9, no. 11, pp. 1163 – 1175, 2001.
[9] E. W. Weisstein. Fourier series. Wolfram MathWorld.
[10] National Instruments, "Understanding FFTs and windowing."
[11] G. Kaiser, *A Friendly Guide to Wavelets*, ser. Modern Birkhäuser Classics. Birkhäuser Boston, 2010.
[12] B. A. Olshausen. Aliasing.
[13] P. de la Cuadra, "Pitch detection methods review."
[14] LEGO® Education. EV3 medium servo motor.
[15] FEETECH. FEETECH FS90MR (2 pack) - 360° rotation — metal gear continuous rotation robotic servo — by himalayanelixir.
[16] endolith. Frequency estimation methods in Python.
[17] Modern Piano Boston, "How long does it take to tune a piano?" April 2016.
[18] Piano Technician Academy, "How much money do piano tuners make?" November 2017.