DREXEL UNIVERSITY
Senior Design
Electrical and Computer Engineering

PROGRESS REPORT
Fall 2018-2019

Team Number ECE-29

Automatic Violin Tuner

Team Members

Name	Department	Email
Yoshin Govender	ECE (Electrical Engineering)	yg353@drexel.edu
Anthony Santoro	ECE (Electrical Engineering)	acs385@drexel.edu
Michael Barnes	ECE (Electrical Engineering)	mrb372@drexel.edu
Steven Ngan	ECE (Electrical Engineering)	sjn36@drexel.edu

Team Advisor(s)

Name	Department/Company	Email
Dr. Kevin Scoles	ECE	scoleskj@drexel.edu

Group Leader signature: Yoshin Govender

Advisor Signature (or attach e-approval) : Kevin J Scoles

Abstract

Violin tuning is one of the foundations for playing with consistent and accurate intonation. Tuning, whether it is done through the violin pegs or fine tuners, changes the frequency and pitch of the notes. From a young age, violinists are taught this skill and subsequently require years of constant exposure in order to develop the skill set to tune independently. This device, an automated tuner for the violin's fine tuners, is designed to help younger musicians expedite this skill to develop the capabilities for recognizing the correct pitch intuitively. By comparing the out of tune pitch to the pitch corrected by the device, students can compare what strings are supposed to sound like with proper intonation. In addition, most elementary and middle schools have class dedicated for instrumental music, typically once a week for elementary school and once a day for middle school. One of the hoped outcomes of this device is a reduction of time spent tuning so more time will be allocated to practice and rehearsal.

A fully-automated testbench, full system identification, pitch detection as well as development on the primary actuator have been major milestones thus far. The pitch of the violin was able to be recorded and analyzed with the use of an Arduino Mega. The relationship between frequency and turns in degrees of the fine tuner were measured and derived as well. It was concluded that within the area of operation, the data collected exhibited a linear behavior. The next steps of the project will consist of developing an efficient closed-loop control system for autonomously adjusting the fine-tuning screws. Currently, two microcontrollers are being operated to manage both sensors and various hardware. With the final prototype in mind, this dual-MCU configuration may result in a bulkier product, so in the future, a more powerful microcontroller in conjunction with a Raspberry Pi will be used instead. In addition to cutting down physical hardware, this will streamline the system and allow for better real-time performance.

Table of Contents

List of Figures

List of Tables

Problem Description

One of the first techniques that is taught to beginner musicians, primarily violinists/violists, is how to tune. The violin is four stringed instrument that can be tuned via two methods: through its pegs for major tuning and through its fine tuners for more precise and controlled tuning. These fine tuners are located on the bottom portion of the violin, resting above the tail piece. Each fine tuner controls its respective string (G, D, A and E). Violinists depend on each of the four strings to be correctly in tune to play with accurate intonation. With everchanging environments consistently affecting the intonation of such a delicate instrument, tuning is required every time the violin is played. From tuning

forks, humming pitches, to Suzuki Method-type approaches[1], tuning is a skill that takes years to master. Students usually seek the guidance of their teacher to tune correctly for the first few years and while this is one of the first techniques that is taught in stringed instrument playing, it is perhaps one of the most crucial and difficult skills to perfect. This is especially true for young players. This proposed handheld device is aimed to help beginner violinists tune by automatically adjusting the violin's fine tuners to its desired pitch. The target audience is aimed mainly towards younger and less experienced violinists to help improve musical pitch detection to possibly obtain what is known as perfect pitch.

The preliminary steps that were considered involving similar inventions to the proposed device. Next was to conduct research on the various types of controllers that were on the market. Many factors were taken into consideration such as cost, functionality and simplicity. In addition, the methods of pitch detection were explored during this process. Many forms of detection were considered such as a condenser microphone or a pick-up microphone. Following that, a test rig was required to document change between the tension and pitch with the screw positioning. Pitch detection algorithms were coded in MATLAB to record and analyze data. Concurrently, a control system is being designed to implement and model the data that has been acquired.

The final prototype of the tuner will encompass a piezo microphone that is able to attach and detach to the bridge of the violin. This device will be handheld and lightweight enough to be portable in all instrument cases. Users will be required to physically attach the coupler of the device to a violin's fine tuner, one at a time. A pluck, or pizzicato, of one string will generate enough vibrations from the fingerboard to the bridge, subsequently registering a frequency for the microphone to read. The tuner will detect its current fundamental frequency and thus will run through algorithms for the motor to turn clockwise or counter clockwise thereby tightening or loosening the screw. As a result, the relative pitch of the string being tuned will become either more sharp or more flat. This autonomous tuner will be programmed for motor to tune to fixed frequencies that equate to the notes of the four strings, G, D, A and E (196Hz, 293.7Hz, 440Hz and 659.3Hz respectively).

Proposed Work and Deliverables

The idea for how this device would operate was based around a few key factors. First, the fine tuners themselves do not require a significant amount of torque to turn, opposed to the violin pegs. Thus, it would be very unreasonable to design a device that could tune a violin via its pegs. Secondly, the size of violin fine tuners is universally similar across the different sizes of the violin itself. A young violinist with a ¼ sized violin would be able to use this device similarly to a full-sized violin. In addition, this is true across the stringed instrument family including violas.

A condenser microphone was first considered to record pitch. However due to noisy environments, a piezo pickup microphone was decided to be a more sustainable choice. Moreover, the microphone would be attached to the bridge, the main link between the

[1] The *Suzuki Method* is a teaching curriculum and philosophy targeted mainly towards children created by Japanese violinist Shinichi Suzuki. He based the concept of language acquisition to the learning of music and called this method the mother-tongue approach. It reinforced the idea that learning is most effectively done through repetition, encouragement and parent responsibility.

vibrations of the string to the body of the instrument. It would therefore pick up more accurate and precise vibrations of the instrument.

The first task was to conduct preliminary background checks on existing inventions or anything similar to the proposed device. The most similar existing system that relates to the proposed project is the *Roadie 2*. This is an automatic tuner designed for guitar pegs and utilizes a digital display which indicates what instrument it is tuning as well as the note it is currently trying to tune to. The user interface allows for the user to select different types of settings through an analog scroll wheel. The proposed device is somewhat inspired by the *Roadie 2* however the violin tuner will utilize a touch screen display rather than analog parts. In addition, the violin automated tuner will pick up the fundamental frequency of the violin string when it is plucked and will tune in accordance to the registered pitch.

Currently, a test rig which records the fundamental frequency in conjunction to the rotation of the fine tuner (in degrees) has been completed. In increments of 90 degrees, the fine tuner was rotated via stepper motor while the string's fundamental frequency were recorded. This automation script was produced though MATLAB. The rig consisted of a 2x4 wooden plank used to emulate the basic mechanics of a violin as to not damage a real violin. This was eventually tested on a real violin to cross verify various concerns regarding tension differences. The results proved to stay consistent across both the test rig as well as the real violin. After analyzing and reviewing the correlation from the data points gathered, it was determined that within the region of interest, the relationship between frequency and rotation in degrees is linear. The next major step is to derive a control system using the data from model testing that will communicate to the stepper motor.

The main measures of success consist of speed, accuracy, usability and convenience. One of the main outcomes from the invention of this device is to reduce the amount of time spent tuning in elementary and middle school classrooms. This can only be achieved if the device itself is able to tune the strings within a reasonable amount of time. The goal is for the tuner to read, interpret and execute the stepper motor within seconds for each string. Accuracy of the tuning device is yet another gauge for a successful solution to the problem at hand. The device will tune to a degree that will be considered "in tune". For example, the tuner should be able to tune an "A" string, 440 Hz +/- 2 Hz. Usability and convenience will be a significant factor for the design of this project. The target audience is towards younger musicians so the simplicity will be heavily taken in account. The user interface should be simple and easy to use so that everyone would be able to operate the device without ambiguity.

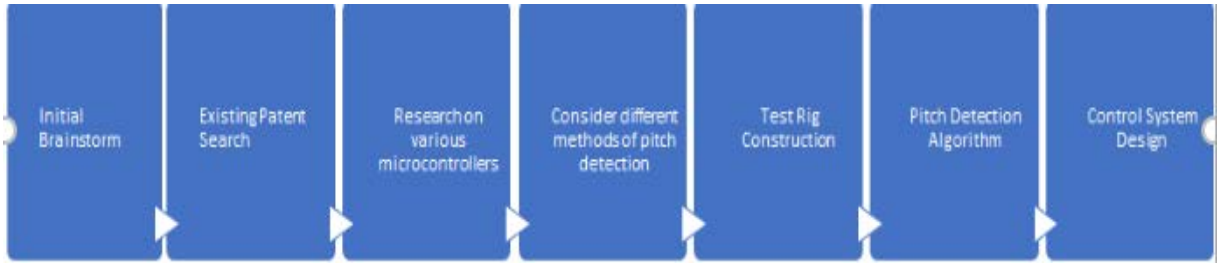

Figure 1: Work Flow Diagram for Fall'18

Completed Work

Model Derivation

Before any control system could be designed, it was necessary to obtain the performance characteristics of the violin strings themselves. In the case of violins, sound is generated through physical resonance of the strings. The pitch of the strings is determined by its length **L**, its tension $\mathbf{F_T}$, its wavelength **λ**, and its linear density **μ**. In a string, the frequencies are generated through standing waves, with are made up of a series of half-wavelengths dispersed throughout the string. This relation is determined by the following condition:

(Equation 1)

$$L = n\left(\frac{\lambda}{2}\right) \qquad n = 1, 2, 3 \ldots$$

Therefore, the natural frequencies of vibration are

(Equation 2)

$$f_n = n\left(\frac{v}{2L}\right)$$

From this, it can be ascertained that the string produces several harmonics along with its main pitch. This series of harmonics is what produces the timbre, or characteristic sound, of the violin.

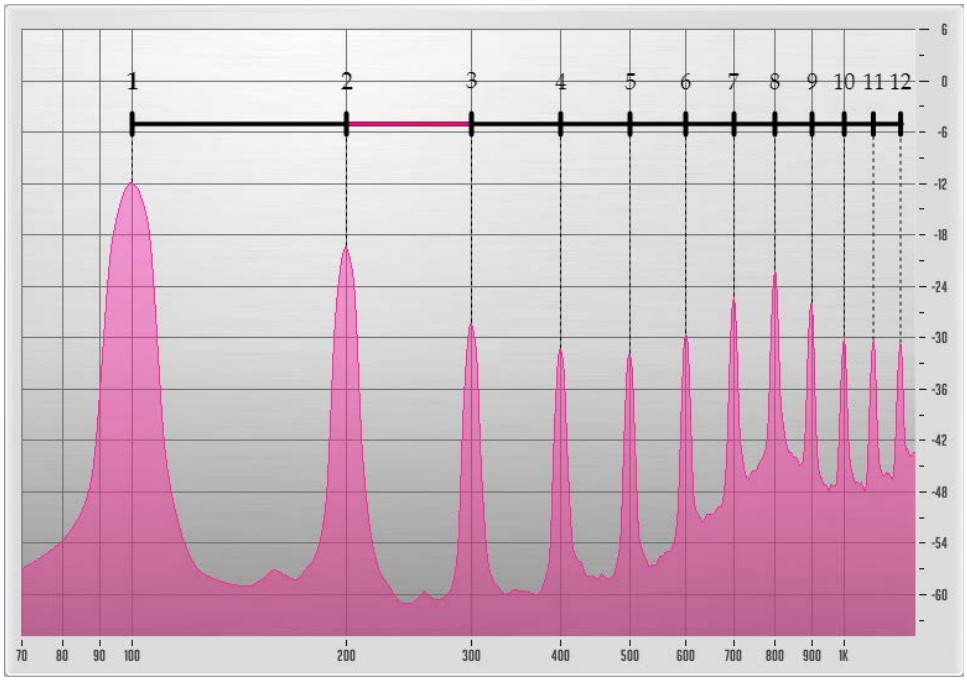

Figure 2: Harmonics of an audio signal

The frequency that this project is concerned with is the fundamental frequency, which is the frequency that will be heard by listeners and players of the instrument. The fundamental frequency is the frequency of the first harmonic, where n = 1. Based on this information, along with the previous stated equations, the frequencies which a string produces is modeled by:

(Equation 3)

$$f_n = \frac{n}{2L}\sqrt{\frac{F_T}{\mu}}$$

Wherein the pitch, or fundamental frequency is

(Equation 4)

$$f_1 = \frac{1}{2L}\sqrt{\frac{F_T}{\mu}} = \frac{\sqrt{\frac{TL}{m}}}{2L}$$

A huge object of concern was the fact that, based on this model, the system would be nonlinear. Depending on how significant it is, this could potentially make controller design very difficult. To investigate this, an experiment was run that would reveal how the system would perform under standard operation. The experimental setup featured a rig that suspended a violin string between a fine tuner and tension load cell. On the bridge of the setup, a piezo contact microphone was applied to detect pitch, and the fine tuner was actuated using a NEMA 17 stepper motor. On each iteration of the experiment, the motor would turn the screw a set amount, the tension of the string would be measured, and a short recording of the string being bowed would be recorded and analyzed. The automation was performed using MATLAB and an Arduino Mega.

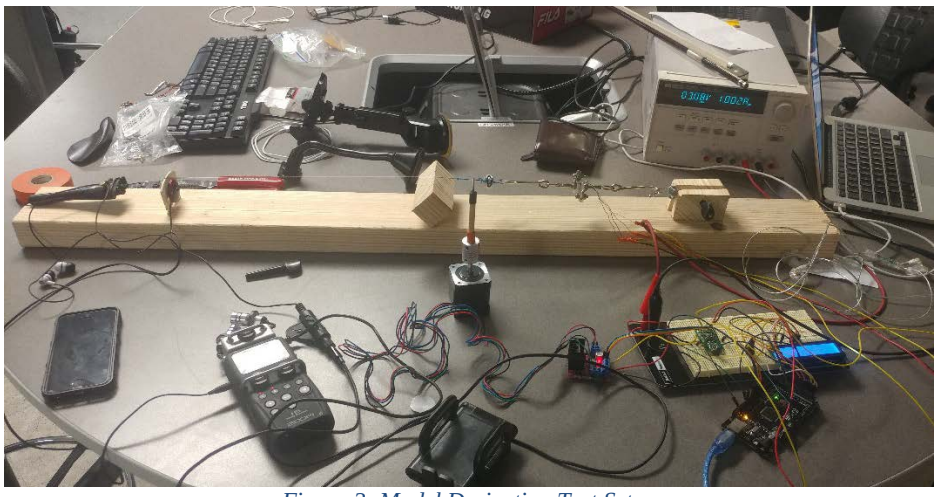

Figure 3: Model Derivation Test Setup

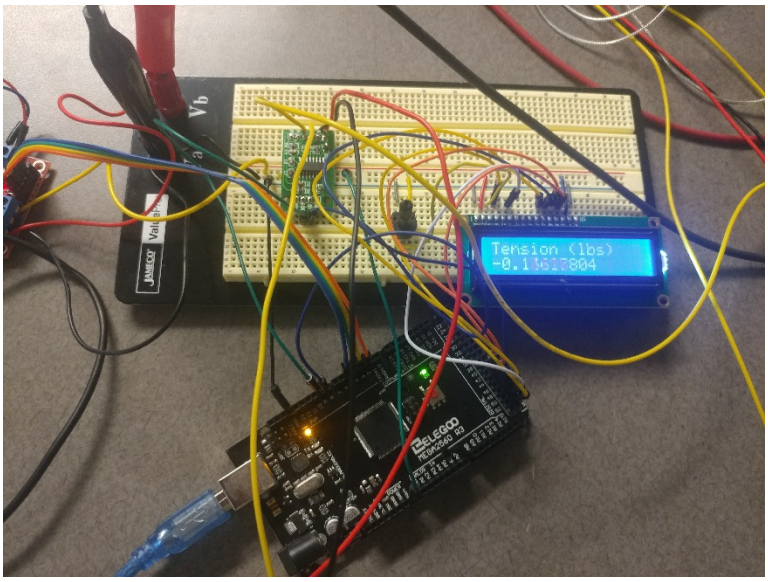

Figure 4: Arduino Test Rig

Once the data was in MATLAB, each of the audio samples were analyzed using the Log-Harmonic Summation technique (Hermes, 1988). This algorithm was chosen because it was the most well-suited to determining exactly how off-pitch the audio signal was without excessive noise or excessive quantization.

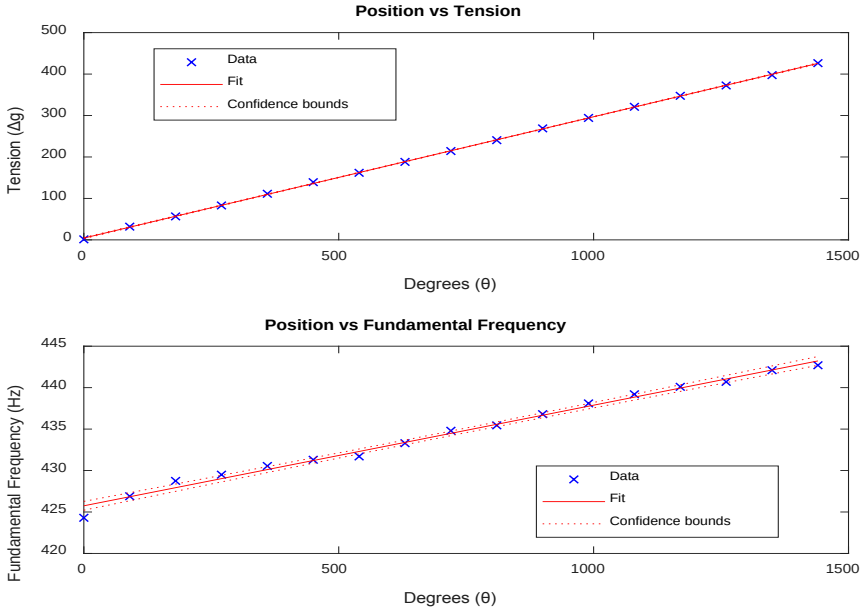

Figure 5: Model Test Experiment Results

The results of the experiment (shown in Figure 5) showed some promising results. While mathematically, the system itself is non-linear, the system response within the range of the fine-tuner screw is fairly linear, coming in with an R-squared value of **0.991**. This means that as far as the control system is concerned, the plant can be reasonably linearized, and a simple controller can be used.

Control Overview

Closed-Loop System

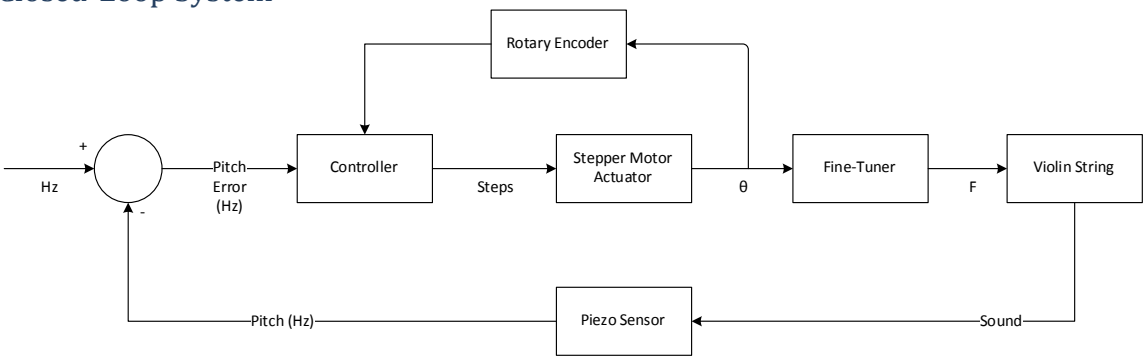

Figure 6: Generalized Closed-Loop Control System

The device will perform the fine-tuner adjustments using a system similar to the block diagram shown in Figure 6. In this diagram, the fine-tuner knob and violin string represents the process to be controlled. A user defines the intended pitch of the string as the input to the system. The actual pitch of the string will be calculated from an audio stream from a piezo microphone in real time, and used to determine the error. The control system will then compensate for the error by adjusting the fine-tuner knob using a stepper motor. The rotary encoder just serves as a form of failure detection, in case the motor gets stuck or is not performing as expected.

As far as the controller type goes, there are a few options. Ideally, the device would implement some sort of observer. However, because the plant changes constantly from string to string and instrument to instrument, this would overcomplicate things. Instead, the device will use an observer-less system such as a PID or a standalone state-feedback controller. Normally with PID controllers, there is a concern of excessive overshoot and integrator windup that could cause the system to fail. However, since the motor will not be moving very fast, and because the primary concern is steady state error, this should not be a problem.

Pitch Detection Method

Choice of pitch detection model

For implementing the pitch detection of an instrument, there were many pre-existing models that the detection could be based on. The choice to narrow down the selection was made based on the given constraints observed in the project thus far. The first of these constraints was the hardware that is being utilized for the project. In the planning and purchasing stages the decision to go with the STM32 “Blue Pill” microcontroller was made for the frequency detection circuit. The primary reasons for this choice were that the STM32 board had a higher core clock speed, higher DAC resolution (12-bit), and a very low cost to purchase around 2 dollars. ST Microcontrollers are widely used for DIY projects in the realm of engineering so finding relevant references and documentation for creating procedures such as pitch detection would not prove to be an

obstacle. With the hardware picked out the constraint of data aggregation and analysis needed to be met. The methods for controls that would be utilized for the project required the pitch detection algorithm to be able to be sent a request, record audio detected, find fundamental frequency, and send out that fundamental frequency back to the primary controller requesting the information in a very small window of time so as not to slow the control loop to a point where errors can occur. The third constraint would be the physical collection of the sound produced by the violin. The frequencies produced by the strings of the violin will not only produce a peak at the resonant frequency of the string itself but also harmonic peaks at multiples of the frequency; this can be observed for a 100 Hz signal in Figure 6. The presence of these harmonics can skew pitch detection algorithms that are not prepared to account to them as they may be recorded as another periodic peak of the fundamental frequency.

Therefore, whatever method that would be chosen for this task would have to conform with the hardware, speed, and harmonic constraints given by the project. The desired method that was chosen was Autocorrelation. The autocorrelation algorithm implemented on the STM32 came out to only two fundamental lines of code used for analysis, this allowed for lightweight performance and fast determination of fundamental frequency. In addition to this, given the operation of the autocorrelation algorithm the fundamental frequency can be found accurately no matter how severe the harmonic distortion may be in the recorded data.

Operation of Autocorrelation Algorithm

The general operation of the autocorrelation algorithm measures the level of similarity that a specific set of recorded data has with a time delayed version of itself. Initially, the data being analyzed is duplicated and convoluted with itself sample by sample while being summed. The product of this operation is used to set a similarity threshold and after this is set the duplicated signal is shifted by a time delay of one sample time and the product of the time delayed samples are taken again with the original set of samples and a new sum is produced. This sum is compared to the threshold value to determine if the cycle has repeated. If the sum is comparable to the set threshold the current delay implemented on the duplicated signal is recorded as the period of the fundamental frequency. This process can be observed in Figure 7.

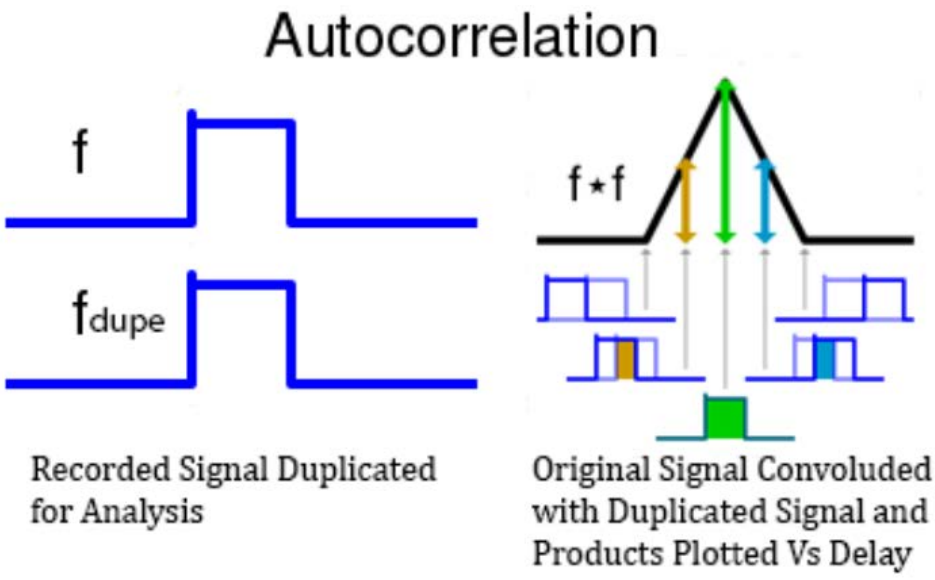

Figure 7: Autocorrelation Method Diagram

Hardware Configuration

There are four primary control aspects of this project that need to be accounted for: pitch detection, actuation, closed-loop control, and user interfacing. Such a process would be extremely difficult to pull off using a single-threaded design, as a few of these operations are real-time dependent and need to be performed asynchronously. To address this problem, there are three different configurations that would accomplish the task, each with their own pros and cons.

Optimal Configuration

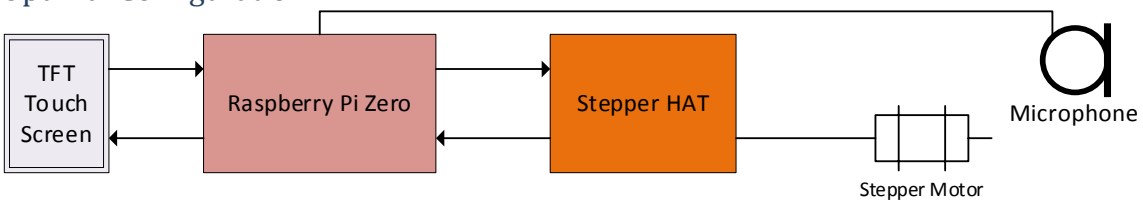

Figure 8: All-in-one System on Chip Configuration

Ideally, the entire project would be based around a single processor. A good option for this is the Raspberry Pi. The device is considerably powerful, and while the zero model is only single-cored, it is fairly robust for multi-threaded operations. If multiple cores were needed, then the quad-core Raspberry Pi Compute module would be more than sufficient. This configuration is superb because every single hardware component is being directly controlled by a single CPU, meaning the whole operation and control process would be extremely efficient. This would also allow for very easy software and firmware updates to be released for customers.

With that said however, there are a few major problems with this design. The first is that the Raspberry Pi, out of the box, is very bad at real-time operations. This is due to the lack of an Real Time Clock (RTC), and more prominently due to the fact that the current

Linux and BSD kernels (Debian Stretch) do not support real-time. The second problem is that the Pi Zero cannot perform the pulse control needed to operate a stepper motor. These problems can each be fixed though a couple of hardware and software tweaks. The SOC can control stepper motors with the addition of a pulse generator such as the AD9850 or the Adafruit Servo HAT[4]. The issue of the clock can be fixed simply by adding a real-time clock to the device through the GPIO pins. The final problem of the operating system is a bit more complicated. While there are versions of Linux that support real-time, they are poorly optimized. Because of this, it would be better to completely migrate away from the Linux kernel entirely to an Real-Time Operating System (RTOS) like RTEMS [5]or ChibiOS[6].

While this would be a great option from and performance, volume, and price perspective, it would be very difficult to pull off in the given project timeline, as there is very sparse support for working with a Raspberry Pi RTOS, and a huge portion of the development timeline would need to be allocated simply to figuring out how to do basic things that are not well-documented anywhere online.

Simple-Development Configuration

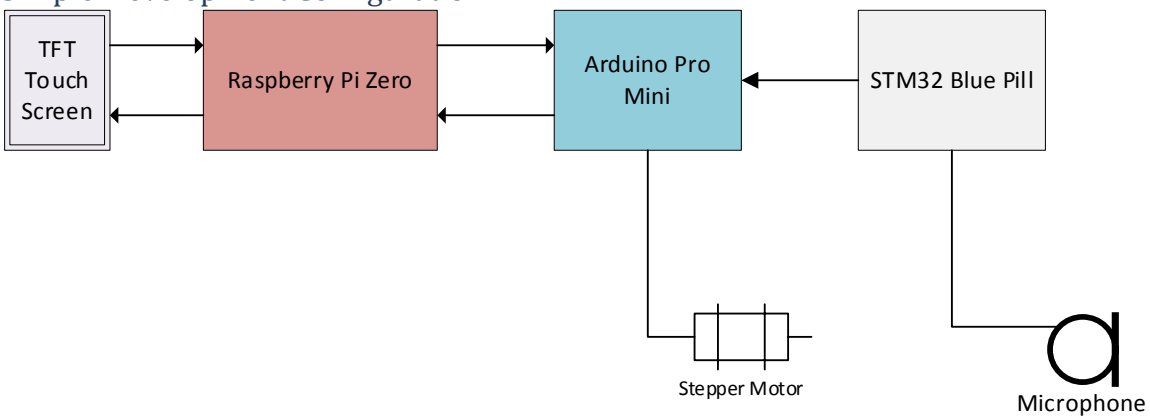

Figure 9: Dual-Microcontroller Configuration

Alternatively, the device can incorporate multiple microcontrollers, each of which have a dedicated purpose. In the design shown above in Figure 9, a 32-bit STM32 Microcontroller serves as the pitch detection chip. The sole purpose of this controller is to perform the autocorrelation algorithm on the incoming audio signal to derive the fundamental frequency. The closed loop control process, as well as the stepper motor control, can then be performed by an Arduino pro mini, and receive the pitch information over serial. The advantage of this configuration is that development would be quick. The Raspberry Pi and Arduino are both very well-documented, and well-supported, which would get the team to a working end-product more painlessly.

Unsurprisingly however, this design is inelegant and requires more hardware. The addition of two microcontrollers can make it more difficult to keep the product size small.

[4] Name given to expansion boards for the Raspberry Pi
[5] Real-Time Executive for Multiprocessor Systems. Open-source software designed for embedded systems, originally used for military systems.
[6] Development environment for embedded applications including an RTOS, HAL, peripheral drivers, support files and tools.

In addition, this model adds in two additional signal nodes which will require additional serial protocols to be developed between all three. The reason why two microcontrollers are needed is that a single Arduino is simply not powerful enough to handle all of the hardware and sensors on its own. On the other hand, the STM32 is quite powerful, and features a 72 MHz clock, which should be fast enough to handle everything reliably. Unfortunately, these controllers are very poorly supported, and there is a complete lack of C++ stepper motor libraries available.

Balanced Configuration

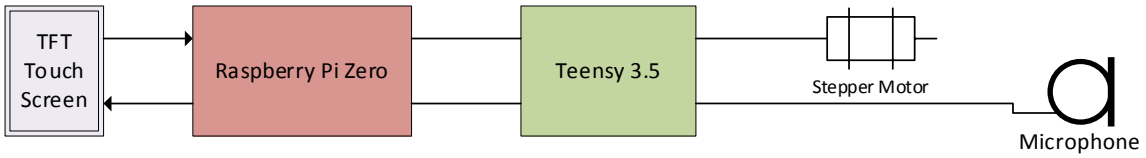

Figure 10: Single-Microcontroller Configuration

A sufficient balance can be achieved through the use of a relatively new microcontroller called the Teensy[7]. This 32-bit controller is extremely powerful, and designed for complex operations, particularly those that involve audio processing. In addition, the board is fully Arduino-compatible and very well-supported. This design would allow for just a single microcontroller to be used along with the Raspberry Pi, which would require fewer communication hops and less circuitry.

Teensy controllers are unfortunately not as cheap as the Atmel and STM32 based controllers used in the other configurations, as they are sold for around $20 for the lower-end 3.2 model. This price increase however could be justified if it can simplify the design.

For now, the team decided to opt for the Simple-Development configuration simply because it would allow for a speedier build, but the balanced configuration is also being looked into as a viable option for the prototype in winter term.

Work Schedule & Proposed Timeline

See Appendix A

[7] A USB-based 32-bit ARM Microcontroller made by PJRC

Industrial Budget

Table 1: Industrial Costs for Large Scale Manufacturing

	Expense	Cost pre Unit	Units	Total
Employee Costs	Electrical Engineer Salary (Yearly)	$65,000.00	4	$260,000.00
	Overhead (30% of Salaries)	$19,500.00	4	$78,000.00
	Employee Benefits (15% of Overhead)	$2,925.00	4	$11,700.00
	Health Insurance (/Person/Year)	$5,179.00	4	$20,716.00
	Total Cost (Year)			$370,416.00
Rent	Office Rent (/sq. ft)	$24.00	2000	$48,000.00
	Utilities (Yearly)	$4,300.00	1	$4,300.00
	Insurance (Yearly)	$1,200.00	1	$1,200.00
	Office Supplies (Yearly)	$2,500.00	1	$2,500.00
	Total Cost (Year)			$56,000.00
Hardware	Computer	$1,200.00	4	$4,800.00
	Power Supply	$500.00	1	$500.00
	Signal Generator	$400.00	1	$400.00
	Multi Meter	$400.00	1	$400.00
	Soldering Iron	$200.00	1	$200.00
	Oscilloscope	$900.00	1	$900.00
	Raspberry Pi Zero	$5.00	3	$15.00
	Teensy	$20.00	3	$60.00
	Arduino Pro Mini	$10.00	3	$30.00
	Miscellaneous (Components, 3D Printing, etc.)	$300.00	1	$300.00
	Total Cost (Year)			$7,605.00
Software Licenses	MATLAB	$2,150.00	2	$4,300.00
	Microsoft Office Professional	$440.00	4	$1,760.00
	Adobe Acrobat Professional	$450.00	4	$1,800.00
	OrCAD Design Suite	$10,660.00	1	$10,660.00
	Total Cost (Year)			$18,520.00
Subtotal (year)				$452,541.00
Risk (15% of Subtotal)				$67,881.15
Total Budget for One Year				$520,422.15

Out-of-Pocket Budget

Table 2: Current Out-of-Pocket Budget for Fall Term

Item Name	Unit Price	Quantity
USB to TTL Adapter 2-pack	$9.38	1
Learning Board Module for Arduino (STM32F103C8T6 2-pack)	$10.99	1
40 kg Tension Load Cell and HX711	$13.99	1
Nema 17 Stepper Motor Bipolar	$13.99	1
Contact Microphone Pickup 2-pack	$12.99	1
Violin Accessory Kit	$24.37	1
Perf Boards	$5.30	1
Single-Row headers	$5.30	1
Heat-Shrink Joiners	$5.30	1
Retractable USB cable	$10.60	1
DPDT Toggle Switch	$2.12	1
Total	$114.33	

Societal, Environmental or Ethical Impacts

Elementary and middle schools only have a short time slot or "period" dedicated to instrumental class (typically an hour or so). Much of the time used in class is dedicated towards tuning. One of the goals of this device is aimed to mitigate time spent tuning to allow more time for rehearsal and practice. Students can tune for themselves at the start of rehearsals without the aid of an instructor. This device will be constructed to help student musicians obtain "an ear" for tuning so violinists will gain the skillset to eventually tune independently.

It would not be surprising to come across critics of this proposed device. Understandably so, some may comment that using such a device long term will hinder the musician's ability to tune independently. While not its intended purpose, some musicians may rely on this tuner rather than utilizing their musical tuning ability. Tuners can be extremely helpful but like any other tool, it must be used correctly. To be clear, this device is targeted towards very young and beginning violinists to help build the foundation to tune with perfect (or close to) pitch. A more experienced violinist should not be using this device to tune.

The physical components that are required to build this automated tuner should not pose any significant environmental impacts. The majority of the electronics found in the device are standard components such as a small stepper motor, Arduino boards as well as wires found in most electronic devices.

Design Constrains

This device should be lightweight and portable for travel. Ideally, this tuner can fit inside all violin/viola cases and will be lightweight enough so there will be no damage to the violin. The target audience for this device ~~is~~ young students. Thus, operation simplicity of the tuner will be a major factor in the design process. Beginner violinists/violists should be able to use this device without the aid of an adult. Random samples will be tested upon

the completion of the project to gather the feedback of real musicians across the different age groups.

In addition, the device must contain torque control or a limiter so that it senses that the fine tuner is turned all the way in. The motor should cease to rotate as to not damage the fine tuner. The power source that will drive the motor must be suitable for reasonable usage and practical. Moreover, the coupler (shown in Figure 11 and Figure 12) that will connect to the fine tuner must be able to adapt to other violin tuners. The tuner accuracy will be within an acceptable and consistent range.

Summary & Conclusion

Overall, the progress on the autonomous violin tuner has been substantial in the three months of working. The test rig violin was built to conduct various experiments so that a real violin would not be damaged. Because of our results, it was found that our testing was able to be done on a real violin without any potential damage. For experimental purposes, a working control system was derived based on research done pertaining to pitch detection, various algorithms of fundamental frequency and actuator control. The test rig allowed for tension and frequency relationships to be modeled and experimented on. A linear relationship was found between the frequency (in Hertz) and the fine tuner rotations (in increments of 90 degrees). In addition, the string tension differential was recorded and analyzed.

Moving forward, more powerful microcontrollers will be considered to ascertain the goal of the device being lightweight, portable and easily operational. The linear nature of the range in which the violin's pitch is concerned will be modeled with the actuator to preform automated tuning.

References

Akellyirl. (2017, January 8). *Reliable Frequency Detection Using DSP Techniques.* Retrieved from Tech Stories: http://www.akellyirl.com/reliable-frequency-detection-using-dsp-techniques/

Cooper, I. (2015). *Visual Physics Online.* Retrieved from Waves, Musical Instruments, Strings: http://www.physics.usyd.edu.au/teach_res/hsp/sp/mod31/m31_strings.htm

Hermes, D. J. (1988). Measurement of Pitch by Subharmonic Summation. *The Journal of the Acoustical Society of America*, 257-264.

INDUSTRIES, B. (2019, 11 30). *Roadie Automatic Guitar Tuner*. Retrieved from https://www.roadietuner.com/roadie2: https://www.roadietuner.com/roadie2/

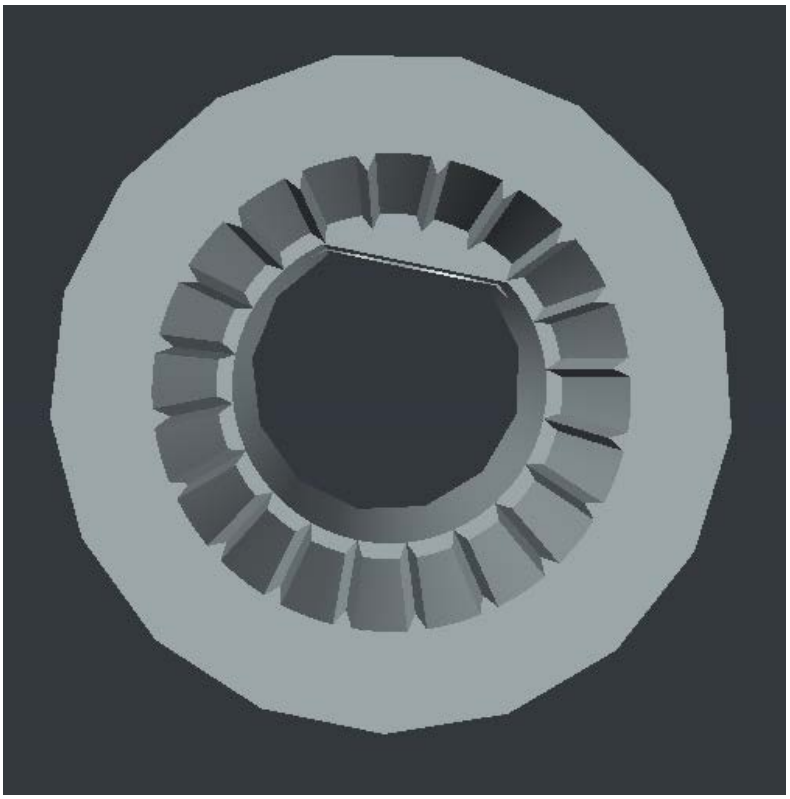

Figure 11: Fine-Tuner Coupler (Slanted Design)

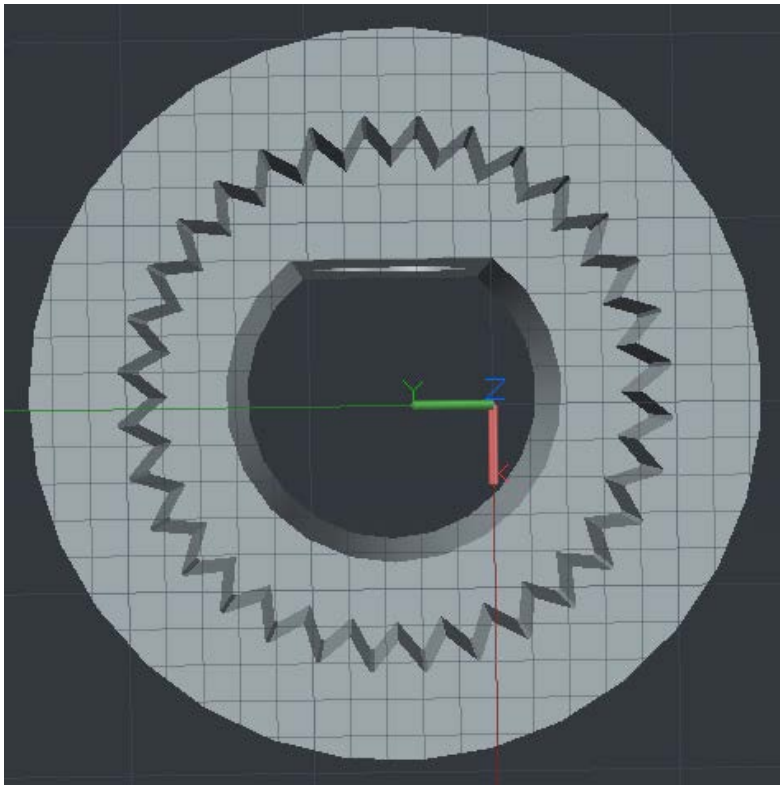

Figure 12: Fine-Tuner Coupler (Set-diameter)

Appendix A: Gantt Chart

GANTT CHART FALL TERM '18

TASK NAME	START DATE	END DATE	DURATION (WORK DAYS)	WEEK 1 (9/24-9/28)					WEEK 2 (10/01-10/05)					WEEK 3 (10/8-10/12)					WEEK 4 (10/15-10/19)					WEEK 5 (10/22-10/26)					WEEK 6 (10/29-11/02)					WEEK 7 (11/05-11/09)					WEEK 8 (11/12-11/16)					WEEK 9 (11/19-11/23)					WEEK 10 (11/26-11/30)					WEEK 11 (12/03-12/07)				
				M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F
Initial Brainstorm																																																					
Initial Brainstorm Process	9/24	9/28	5																																																			
Hold First Meeting	10/1	10/1	1																																																		
Decide Quailty Goals	10/1	10/2	2																																																		
Evaluate Interest In Project	10/3	10/5	3																																																		
Review With Advisor	10/12	10/12	1																																																		
Submit to Coordinator	10/18	10/18	1																																																		
Background Research																																																					
Exisiting Patent Search	10/22	10/26	5																																																		
Research on String Physics of the Violin	10/26	11/1	5																																																		
Research on Potential Microcontrollers	10/30	11/5	5																																																		
Research on Code Involving Pitch Detection	10/31	11/6	5																																																		
Control System and Construction																																																				
Test Rig Design & Build			0																																																		
Model Derivation			0																																																		
Pitch Detection Development			0																																																		
Control System Design			0																																																		
Presentation Preparation for Fall Term																																																					
Discuss Presentation with Advisor	11/26	11/26	1																																																		
Prepare Presentation and Report	4/8	4/10	2																																																		
Rehearse Presentation	11/28	12/3	4																																																		
Revise Presentation	12/4	12/7	4																																																		