DREXEL UNIVERSITY
Senior Design
Electrical and Computer Engineering

PROGRESS REPORT
Winter 2018-2019

Team Number ECE-29

Self-Automated Violin Tuner

Team Members

Name	**Department**	**Email**
Yoshin Govender	ECE	yg353@drexel.edu
Mike Barnes	ECE	mrb372@drexel.edu
Steven Ngan	ECE	sjn36@drexel.edu
Anthony Santoro	ECE	acs385@drexel.edu

Team Advisor(s)

Name	**Department/Company**	**Email**
Dr. Kevin Scoles	ECE	scoleskjdrexel.edu

Group Leader's Signature : Yoshin Govender

Advisor Signature (or attach e-approval) : Kevin 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.

Since the fall term, major changes have been made both in regards to hardware and firmware. An updated design incorporating a new microcontroller, H-bridge driver, pitch detection algorithm, microphone and motor has been implemented. A more powerful compact circuit was designed as a result. One of the biggest changes was the transition from the Arduino to a USB-based microcontroller called the Teensy 3.5. The sound detection algorithm has been changed from simply just autocorrelation to a more accurate one called YIN. This algorithm is able to estimate fundamental frequency for both speech and music which provides the best use for this application. The motor has been changed to fit ergonomic needs that balance torque and speed requirements. Lastly, various microphone circuits have been tested, and a new design chosen to improve performance as well as convenience. With all the different updates and upgrades since the Fall term, a working prototype has been running to tune each of the 4 strings of the violin.

Exploring future features, an LCD display will be considered to prompt the user to select various tuning profiles. A Raspberry Pi will be a suitable selection to regulate the user interface as well as the controls architecture of the tuner. A sustainable power source from batteries will be explored that factors in lifetime as well as usability.

Table of Contents

List of Figures

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). 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.

Operational Description

This automatic tuner will receive audio data through a condenser or contact microphone, and will use that data to fully tune a violin by articulating the fine-tuner screws using a stepper motor. This device will have multiple profiles that will enable it to be used on different instruments with different string tunings. The user will be able to select an instrument preset and then tune each string one by one by placing the device on the fine-tuner screw and plucking the string. When the particular string emits the pitch that is correct for that instrument profile, the closed-loop tuning process will stop and instruct the user to move the device to the next fine-tuner. When all four of the strings are sufficiently tuned, the user is informed and the entire process is complete. Figure 1 is a state diagram that will help visualize this procedure.

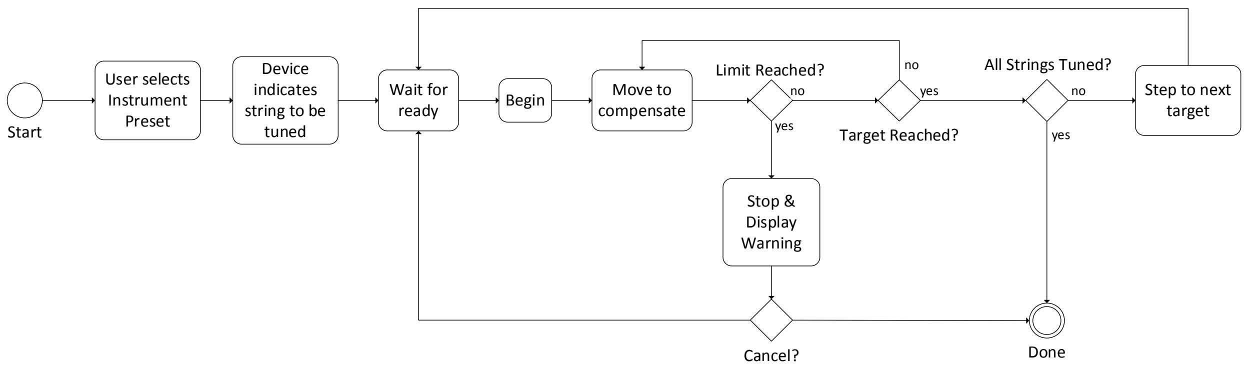


Figure 1: Operational State Diagram

Completed Work and Changes of Scope

New Hardware Architecture

From a high-level, there are four main sections of the hardware architecture; the front-end, the controller, the actuators and the sensors. The sensor for this device is a piezo contact microphone that is driven by an amplification circuit that scales the signal to between 0 and 1.2 V. The actuator is a small-form geared stepper motor that is driven by an H-bridge. Everything involved in the control system is operated by a Teensy 3.5 microcontroller, which receives commands over serial. The top-most layer is a Raspberry Pi zero, which manages the user interface (UI) and sends control commands to the microcontroller unit (MCU) below it.

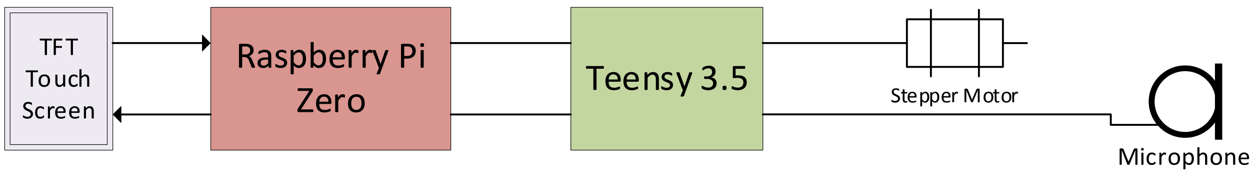


Figure 2: Single-Microcontroller Hardware Architecture

This architecture was chosen because it allowed for more agility in the design process. The original plan was to have a color TFT screen with a graphical user interface available to the user, and with this architecture, the option of scaling back and opting for a simpler methodology would also be available, even if the presence of a raspberry pi would be deemed unnecessary. Therefore, taking this approach would enable the team to be more flexible in how much time could be put into the front end of the device.

Actuator Selection

Selection of an appropriate motor and driver board is critical, as very specific actuator parameters are required for a device such as this. First, the chosen actuator must be small enough to fit inside a handheld device, and should be light enough for comfortable operation. Next, the actuator needs to have precision control on the position & velocity levels, and should be able to move accurately at low speeds. The motor needs to both move fast enough to complete the tuning process in a reasonable time (around 10-15 seconds at most) and be capable of delivering enough torque to be able to turn each of the fine tuners of the violin. Finally, the actuator must have a low-enough power draw that it can be used with a lithium-ion battery. For the actuator, it was necessary to experiment with different motor and driver combinations.

Motor

In the early stages of the project, and particularly during the development of the close-loop system, the motor of choice was a Nema 17 stepper motor. This motor was used at first simply because of how well it worked and how easy it was to use. Being a standard for CNC devices, this stepper motor is bipolar and easy to drive, and is capable of operating at high torques and high speeds while still maintaining exact, consistent positional precision. However, this motor could not be used in a handheld device due to its large size, large weight, and high power-draw.

For the size and form factor of the product, a geared DC motor with encoder feedback could be a viable choice, since it would be able to deliver very high speed and high torque from a relatively small form-factor. As an added bonus, this configuration would allow the encoder to be used as an additional safety measure in case something goes wrong during the tuning process. Unfortunately the motor utilized for this task (an OSEPP RB-Oel-120 with 1:45 gearing) proved to be terrible for fine position control due to its large dead-band, and even with extensive PID tuning could not be used in any reliable fashion. The best compromise for size, torque, speed, and power was found through small reduction stepper motors. Unlike the Nema 17, these steppers featured gearing that would allow higher torque to be achieved at the cost of speed. The best balance was found through a model with 1/16 gearing, which was capable of meeting the speed and torque requirements for the device.

Driver Board

Because a microcontroller cannot operate a motor outright, it requires an intermediary piece of hardware to help transform its digital signals into waveforms that are capable of driving them. Driver boards are a critical selection, as they are often the biggest determination of how well a motor performs. The simplest of these drivers are integrated circuits known as H-Bridges. Two variations of this circuit were used during experimentation, a Texas Instruments Dual Full-Bridge Driver (L298N) and a Texas instruments Quadruple Half-H Driver (L293D). These controllers are extremely common thanks to their ability to drive a wide range of DC and stepper motors. Unfortunately, these driver boards come with their own problems. The L298N is robust, powerful, and even features a built-in 5V regulator. However, like the NEMA 17 motor that it was used with, the controller proved to be too large and too power-hungry, in addition to having a dangerous amount of heat dissipation.

The L293D managed to solve some of these problems. The chip is small, (about 19 by 6 millimeters) and has fairly low power draw (160 mA at idle, 1A at peak speed). The L293D does not have a built-in regulator, but that was not an issue since there will be a dedicated power regulation circuit anyway. The problem with H-Bridges however is that while they excel with bipolar stepper motors, they struggle with unipolar stepper motors. The presence of a center-tap between the internal coils resulted in problematic system resonance. Not only did this cause the motor to miss steps, it also resulted in high frequency noise that worked its way to the audio detection pin of the microcontroller, which caused incorrect pitch values to be registered. For the time being, the problem was solved by placing a low-pass filter in the circuit, which blocked out the noise and allowed the device to operate smoothly.

Pitch Detection Method

The operation of autocorrelation, which the main algorithm of pitch detection from last term, measures the level of correlated data from a signal with a delayed copy of itself as a function of delay. To refresh, data from a signal is duplicated and convoluted with itself while being summed simultaneously. While YIN is based on autocorrelation, this particular algorithm allows for several more advanced and desirable features. Moreover, there is no upper limit on the frequency search range which makes this algorithm best suited for musical applications. YIN utilities 6 additional steps which each encompasses a specific task to further reduce errors rates. Respectively, the steps are autocorrelation method, difference function, cumulative mean normalized difference function, parabolic interpolation and finally the best local estimate. In this study (De Cheveigne, 2001), a sample signal was examined and the following series of steps were implemented to reduce error rates. The autocorrelation function (ACF) "chooses" the highest non-zero lag peaks by searching an extensive range of lags represented by the horizontal arrow as shown below:

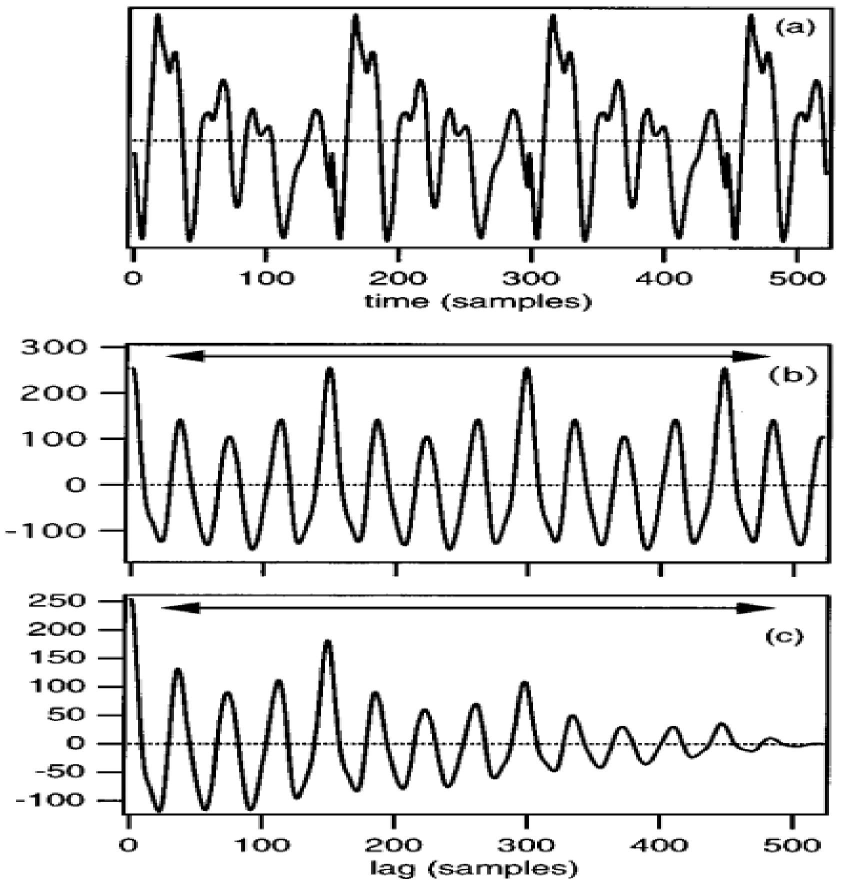


Figure 3. Top: Sample Waveform. Middle: Autocorrelation Implemented. Bottom: Autocorrelation Tapered to Zero

$$r_t(\tau) = \sum_{j=t+1}^{t+W} x_j r_{t+\tau} \qquad (1)$$

Despite the commonality and ease of use of autocorrelation, this method introduces too many errors for many applications. The next steps are implemented so that improvements of accurate pitch detection is maximized. The ACF is then replaced by the difference function:

$$d_t(\tau) = \sum_{j=1}^{W} (x_j - x_{t+\tau})^2 \qquad (2)$$

Values of τ will be searched for which the function equals zero. The squared sum may be expanded to a function in terms of the auto correlation function:

$$d_t(\tau) = r_t(0) + r_{t+\tau}(0) - 2r_t(\tau) \qquad (3)$$

The first two terms are energy terms which indicates that the second energy term varies with τ, implying that the maxima of rt(τ) and minima dt(τ) may sometimes not coincide or overlap. Because of this, error rates fell from 10% to 1.95% for unbiased autocorrelation. The difference function paves for the next step: cumulative mean with a normalized difference function. The cumulative mean normalized difference function is proposed to replace the difference function:

$$d'_t(\tau) = \begin{cases} 1, if\ \tau = 0 \\ \dfrac{d_t(\tau)}{\frac{1}{\tau}\left[\sum_{j=1}^{\tau} d_t(j)\right]}, otherwise \end{cases} \qquad (4)$$

The new function is tabulated by dividing each value of the previous values by its average over shorter lag values. It differs from dt(τ) in that it is 1 indexed rather than starting at 0. This new function allows the elimination of errors that are “too high”. The error rate falls from 1.95% to 1.69%. Various benefits are include eliminating upper frequency limits of the search range, zero-lag dip will be ignored and to normalize the function for the next error reduction step.

The next step is to establish an absolute threshold to choose the smallest value of τ that gives a minimum of *d'*. If none are found, the global minimum is chosen instead. Using the sample data of this study and the threshold of 0.1, a reported error rate dropped from 1.69% to 0.78%. In essence, an absolute threshold is established in order to determine a list of signals admitted to a set. In the case of a period that is not a multiple of the sampling period, the estimate may be incorrect by up to half the sampling period. This problem is addressed by parabolic interpolation. A parabola is used to period estimate of $d_t(\tau)$ and any neighboring values that fit by the parabola. This step is independent from the other steps however it heavily relies on the spectral properties of the autocorrelation function. It can be found that at certain phases of the period that coincides with a high value of $d'(T_t)$. The best local estimate allows for the function to determine a vicinity of each analysis point to estimate a value. An algorithm is set to search a minimum of $d'_\theta(T_\theta)$ for θ within a small interval of:

$$\left[t - \frac{Tmax}{2}, t + \frac{Tmax}{2}\right] \qquad (5)$$

where (T_θ) is the estimate at time θ and Tmax is the largest expected period.

YIN has proven to preform best among various databases with small error rates. To summarize, the steps that are taken help ensure that estimates are stable and not fluctuate on the time scale of the fundamental period. While this signal that is being analyzed through the YIN algorithm is not done by a violin, similar results will yield as shown with the various tests below.

Version	Gross error (%)
Step 1	10.0
Step 2	1.95
Step 3	1.69
Step 4	0.78
Step 5	0.77
Step 6	0.50

Table 1. Error Rate for Each YIN Step

Microphone Circuit

For the purposes of testing the core functionalities of the project's tuning and control software quarter the standard Elegoo microphone circuit, pictured in Figure 13, was implemented in the circuit to handle the process of transforming the audible sounds to an observable analog signal. While the circuit was functional, it provided minimal gain control and isolation from other audible noise sources in the vicinity. As an alternative a new transducer was chosen and analog circuit designed for the circuit. The transducer has to be one that will only pick up sounds pertaining to the pitch of the violin being tuned, this could be a common source of error due to the likelihood of other instruments undergoing tuning within close proximity of the user of the device. In order to provide the necessary isolation, a piezoelectric contact microphone was chosen. This transducer, when affixed to the body of the violin, will pick up only the pitch being created by the strings being tuned while isolating the surrounding ambient noises.

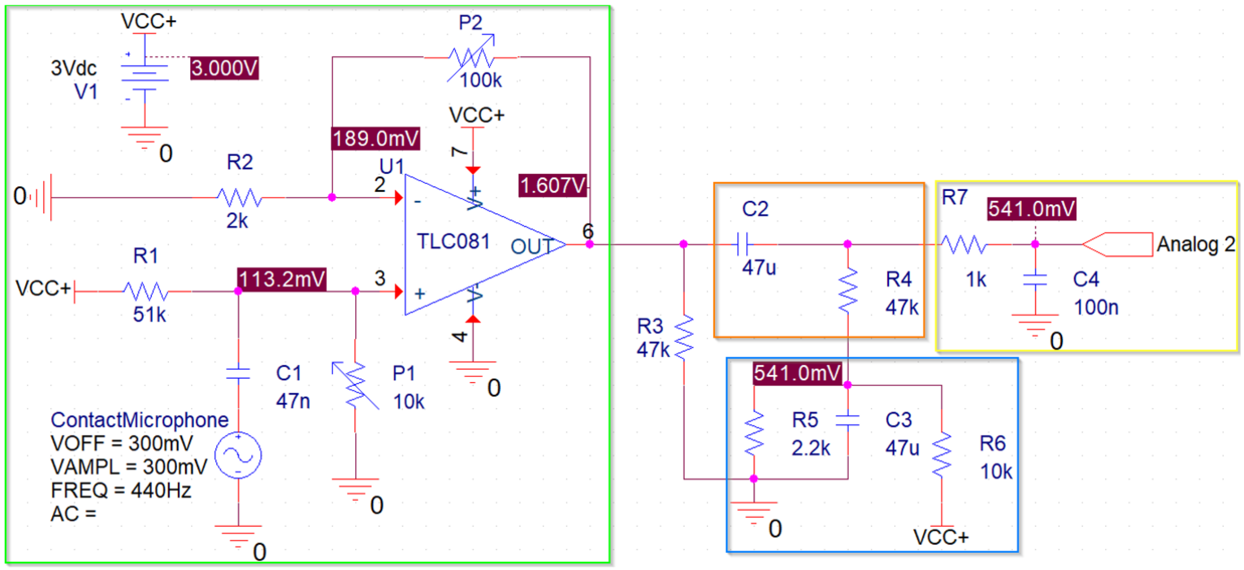


Figure 4 Analog Microphone Circuit

The bridge-pickup microphone is the input of the violin and hooked up to the first stage of the microphone circuit, a differential amplifier. This amplifier was designed to produce a gain sufficient to swing the signal to cover the full input range of the Teensy's ADC. Doing this will allow for the signal to be processed with the highest resolution. The schematic of the differential amplifier stage can be seen in the green box of Figure 1. The circuit consists of the operational amplifier TLC080A1, one capacitor, two resistors, and two potentiometers. The output of the contact microphone will be a differential signal, this is referenced to ground by attaching one of the signal wires to ground and the other to the input of the circuit. The operational amplifier takes the signal from the contact microphone after it passes through C1. In addition to the microphone signal being fed into the amplifier input a DC offset is also added. The C1 capacitor will block any DC voltage from going across the microphone so as not to damage it when the signal is offset. The 51KΩ resistor R1 and 10KΩ potentiometer P1 are responsible for DC biasing the input signal going to the op-amp to offset the signal in order to shift the negative swing of the signal to being only positive; in the case of a 3.3V supply the offset can be set anywhere from 0-0.541V when changing the value of P1. This shift is necessary due to the op-amp only being supplied a positive voltage therefore it will not have the ability to swing the output signal below 0V. This voltage divider circuit then feeds the DC offset audio signal into the non-inverting input of the op-amp. The 2 KΩ resistor R2 and 100KΩ potentiometer P2 control the gain of the op-amp and can allow for a variable gain anywhere from 0-50V/V when changing the value of P2. The ideal values for the potentiometers P1 and P2 should be ones that allow for the offset of the signal to be slightly higher than the magnitude of the negative swing and gain to keep the peak to peak voltage of the signal just under 1.2V. The value of both the offset and gain will generally stay the same for most string instruments and have been included in the circuit to ensure the device can accommodate any orchestral string instrument that may use it.

After the differential amplifier stage the signal should be only positive with a peak to peak voltage around 1.2V. At this point the signal will enter a high-pass filter, which can be seen inside of the orange box in Figure 1. This is constructed as a standard passive RC high-pass filter consisting of a series capacitor C2 and a shunt resistor R4. The cutoff frequency for this filter was selected at close to 1Hz which can be seen in Equation 1.

$$f_{c\_HP} = \frac{1}{2\pi R_4 C_2} = \frac{1}{2\pi * 47k\Omega * 47\mu F} = 0.072Hz \qquad (2)$$

This will effectively eliminate any DC offset output from the op-amp and allow any instrument frequencies to pass through. This is crucial in order to accomplish the next stage which is another voltage divider circuit that adds a DC offset to the signal, which is shown in a blue box in Figure 1. This offset is accomplished by choosing the ratio of resistors R5 and R6 and will center the signal at about 0.6V in order to ensure the amplified signal be able to have the maximum voltage swing possible, ideally from 0V to 1.2V, without clipping the signal when being read into the Teensy's analog 2 pin.

With the signal now swinging from 0V to 1.2V it enters the last stage of the microphone circuit, a low-pass filter. This filter is constructed as a standard passive RC low-pass filter with a series resistor R7 and shunt capacitor C4. The circuit is designed with a cutoff frequency of about 1600Hz and the selection of the resistance and capacitance values was then chosen to match up with this selected cutoff as seen in Equation 2.

$$f_{c\_LP} = \frac{1}{2\pi R_7 C_4} = \frac{1}{2\pi * 1k\Omega * 100nF} = 1591.55Hz \qquad (3)$$

This filter will ensure that any noise that may have been potentially picked up by digital hardware such as the Teensy, motor controller, or external electromagnetic interference will be filtered out and not affect the audio signal going into the Teensy's analog pin.

Control System

PID Control

The tuning process of this system can be considered a closed-loop, error-driven feedback system. In this context, the output of the system Y(s) is the actual measured pitch of the violin string and the input of the system R(s) is a set point which acts as the target. Error-driven here means that the movement of the actuator is determined by the difference E(s) between the measured output and the setpoint. Figure 1 shows a block diagram of this kind of system.

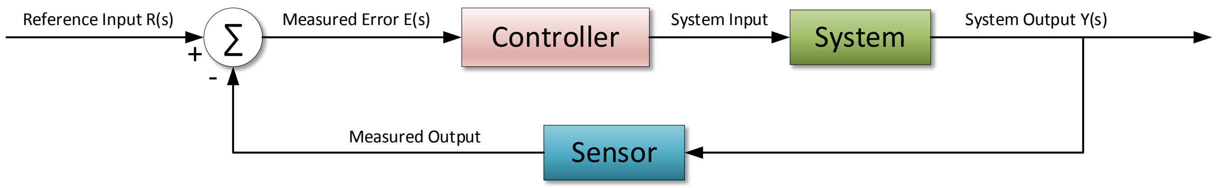


Figure 5: Block Diagram of a Negative Feedback Control System

A common type of controller used in this configuration is known as a PID controller (shown in Figure 2). This control mechanism works by generating an actuator signal that is the sum of the system's error, integral, and derivative multiplied by constant gains; K_p, K_i, and K_d (Ang, Chong, & Li, 2005).

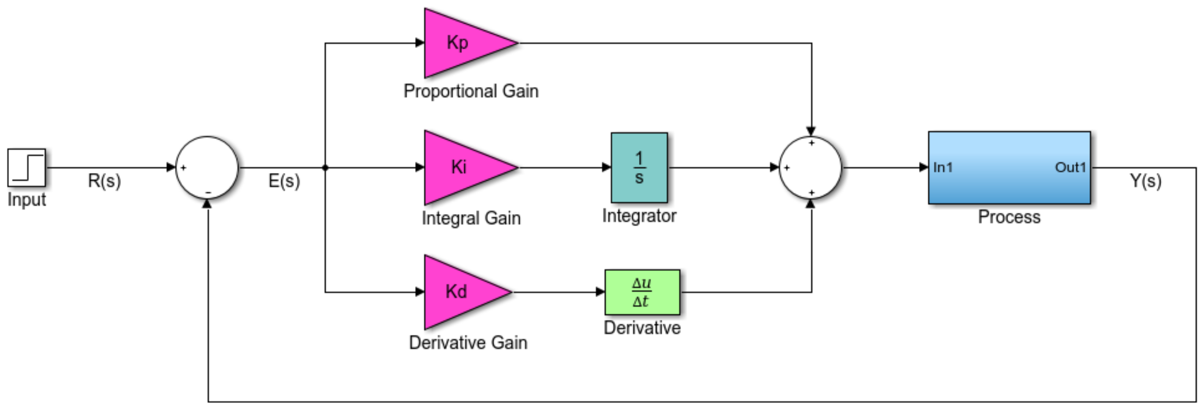


Figure 6: PID Block Diagram

These three parameters can be tuned to achieve different rise times, settling times, and damping ratios. From a high-level, the proportional gain K_p can be magnitude of pull towards the setpoint. High proportional gains will lead to a faster response time, however, setting this value too high can lead to excessive overshoot, and undamped characteristics such as oscillation. To counteract this, the derivative gain K_d can be thought of as a vector that pulls away from the setpoint, and can help dampen the system. Proportional and derivative control alone works well, but it can lead to the buildup of an error offset e_0. The integral gain K_i can account for this by ensuring that the system does not spend too much time on one side of the setpoint. The implementation of this controller in the Laplace and time domains are given in (8) and (9).

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

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

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\mathrm{d}\tau + K_d \frac{d}{dt} e(t) \qquad (5)$$

These equations hold for a continuous time system; however, a microcontroller operates in discrete time. The system must make iterative calculations one after another at a fixed rate Δt. Here, first-order derivatives are made by backward finite differences. Equation 8 shows the PID equation migrated to the discrete domain, where the value k represents the calculation number, $T_i = \frac{K_p}{K_i}$, and $T_d = \frac{K_d}{K_p}$.

$$u(t_k) = u(t_{k-1}) + K_p \left[\left(1 + \frac{\Delta t}{T_i} + \frac{T_d}{\Delta t}\right) e(T_k) + \left(-1 - \frac{2T_d}{\Delta t}\right) e(t_{k-1}) + \frac{T_d}{\Delta t} e(t_{k-2}) \right] \qquad (6)$$

Equation 8 shows that the integral and derivative components are nothing but sum and difference equations. Because of this, the code for this type of controller is extremely simple, and can be abstracted into the pseudocode below, wherein the function *calculate()* is executed at a frequency of $\frac{1}{\Delta t}$.

```
previous_error = 0
error_sum = 0

void calculate(){
    error = setpoint - measured_value
    error_sum += error
    output = Kp*error + Kd*(error - previous_error) + Ki*(error_sum)
    previous_error = error
}
```

Closed-Loop Tuning System

The tuning process itself is at the very heart of the violin tuner and encapsulates its core functionality. It operates using a PID compensation system discussed above, and uses the YIN algorithm (4) to calculate the fundamental frequency of the violin string in real time. In this system, the process is the violin string whose input is the position of the fine-tuner screw and output is the fundamental frequency in Hertz (Hz), the input frequency is the setpoint, and the actuator is the stepper motor.

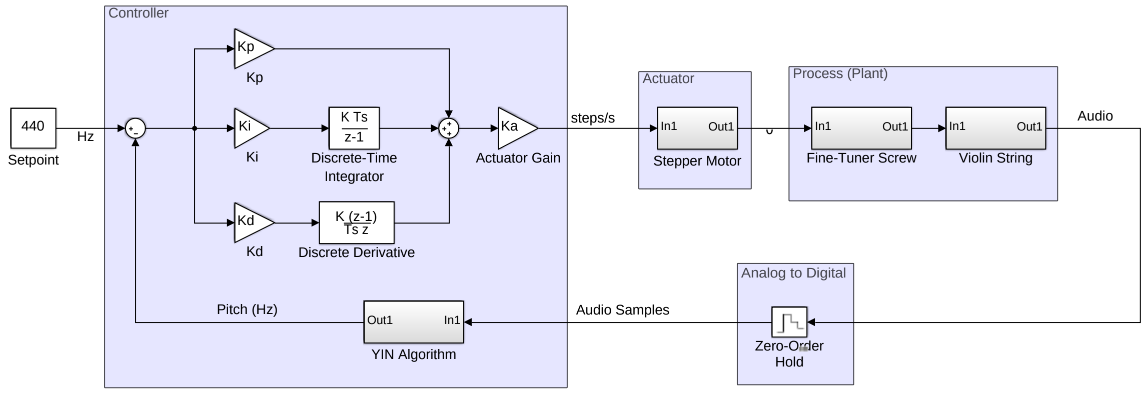


Figure 7: Automatic Tuner System Block Diagram

Applying this algorithm to the hardware was simply a matter of scheduling the PID calculation to occur as fast as possible, and adjust the stepper motor's angular velocity in real time. Unfortunately, due to the derivative factor, PID controllers are very susceptible to noise, which is quite problematic since the YIN algorithm has the tendency fluctuate a bit, especially frequencies approaching 1 kHz. In order for a controller like this to work, the incoming data needs to be smoothed. Figure 7 shows a comparison between raw pitch data and the same data put through a gaussian[1] smoothing filter. This noise is problematic for full-PID control because the derivative block would amplify the noise. Because of this, only proportional control is being used for the time being.

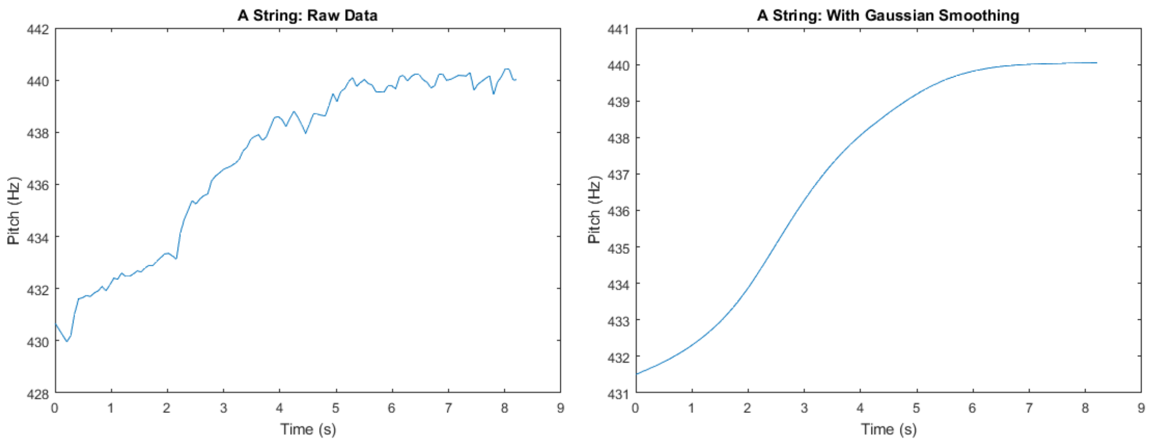


Figure 8: Comparison between unfiltered and filtered data

To evaluate the performance of this control system, each string was tuned to their proper value from above and below the setpoint to generate a set of step response plots. The G, D, and A string were all initialized from a frequency 10 Hz away from the target, while the E string was initialized from 20 Hz away due to its steep tuning characteristics. For ease of analysis, each of these response curves have the gaussian filter applied to the pitch data.

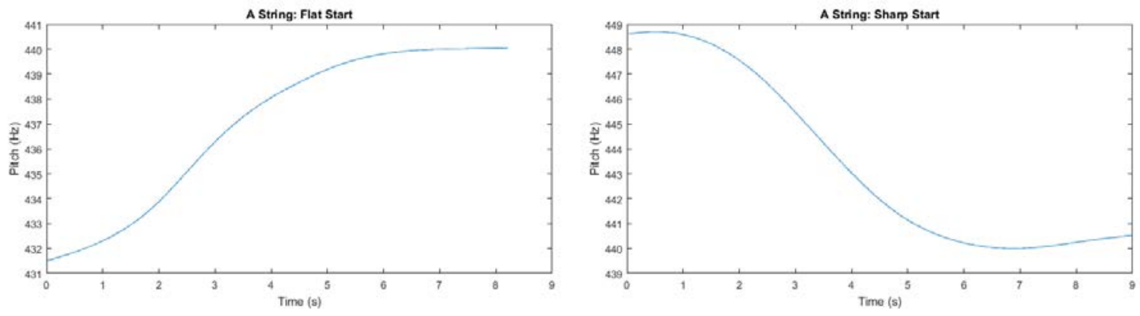


Figure 9: A String Step Response Plots

1

A Gaussian filter is an infinite impulse response filter that smooths data through a weighted moving average. It was chosen since it provides a good balance between the time and frequency domains.

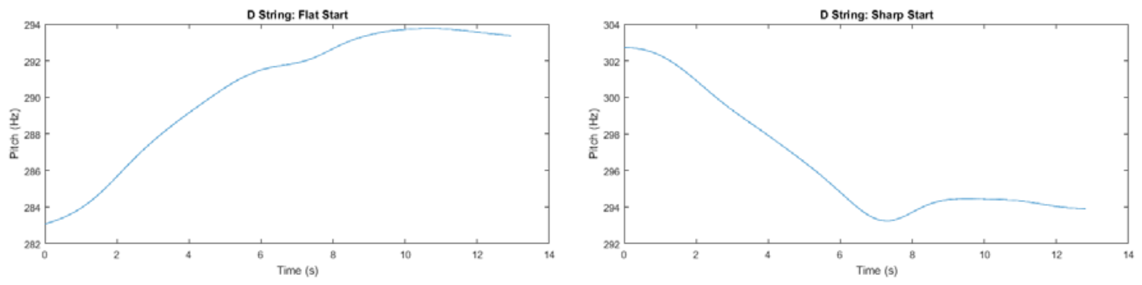


Figure 10: D String Step Response Plots

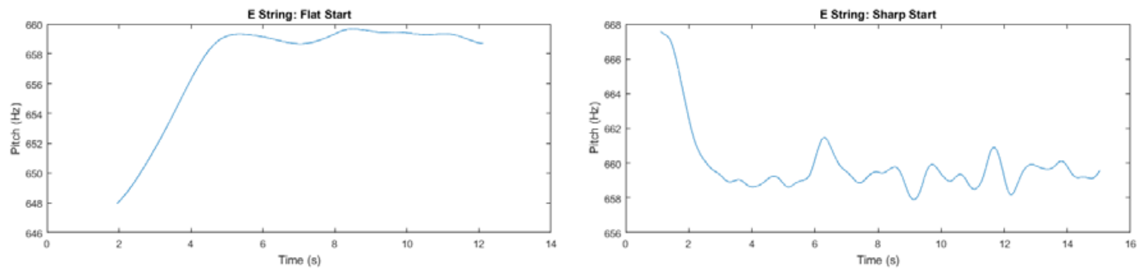


Figure 11: E String Step Response Plots

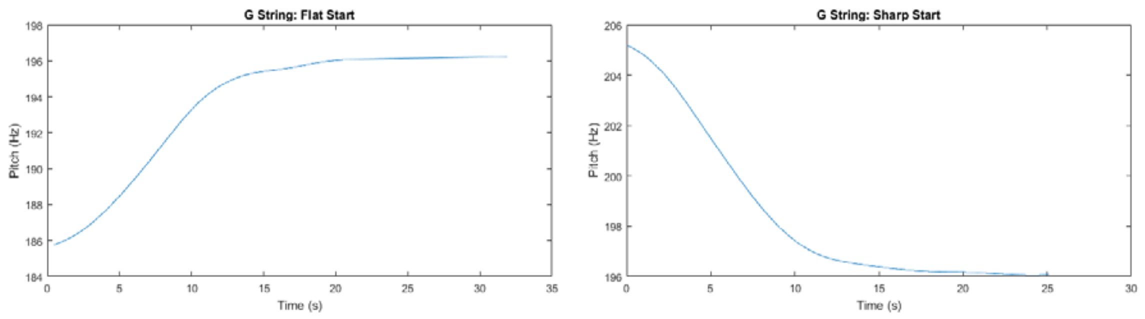


Figure 12: G String Step Response Plots

The step response plots above illustrate some of the issues that come with tuning different types of stings. The lowest two strings, A and G, had the smoothest response curves, while the higher two strings, particularly the E string, fluctuated dramatically. This is because the audio frequency spectrum is logarithmic, and changes on the higher end of the spectrum are more dramatic than on the lower end. What this means is that a given rotational change of a fine-tuner results in a greater change in pitch for higher frequencies than for lower frequencies. This could be potentially problematic when it comes to system performance, so in the future, this problem can be mitigated by filtering sensor data in real-time and potentially implementing micro-stepping for certain strings. On the other hand, a completely opposite characteristic can be observed in the response data for the G string. As listed in Table 1, this string takes 10.1 seconds to rise towards the setpoint, while the A string only took 3.9 seconds. Here, the same issue of a logarithmic frequency spectrum also applies, as far more angular displacement is necessary to achieve the same degree of frequency change.

Table 2: Tuning Process Rising Step Info

	Rise Time (s)	Settling Time (s)	Settling Min (Hz)	Settling Max (Hz)	Overshoot (Hz)	Peak (Hz)	Peak Time (Hz)
A String	3.92	5.50	439.20	440.05	0.01	440.05	8.22
D String	6.83	8.59	292.74	293.76	0.02	293.76	10.72
E String	2.13	4.47	658.31	659.66	0.05	659.66	8.49
G String	10.10	15.73	195.00	196.24	0.12	196.24	31.20

Existing Obstacles and Proposed Solutions

Algorithm Issues

As previously mentioned, the Yin algorithm produces a noise in the frequency readings, and that kind of telemetry is not very useable for complete PID control. While gaussian smoothing is a good algorithm for smoothing already measured data for analysis purposes, it is fairly inefficient for real-time operation. A far better algorithm for that would be a Kalman filter, which would allow for corrections in sensor deviation to be applied to a telemetry stream in real-time. A Kalman filter is an algorithm that takes a series of time-dependent measurements which contain statistical noise and produces estimates that are more accurate than those based on a single measurement. Once this kind of filter is in place, the full capabilities of PID control can be utilized.

Another problem is state estimation. In its current form, the device works very well when bowed constantly while tuning, because it receives a constant stream of sensor data from which it can derive its current error e_0. However, it would be impractical for a user to be able to hold the device on the violin and bow at the same time, so the final version needs to accept plucks as the primary form of input. While plucking does currently work, in order for it it to work at well as bowing, an in-the-loop estimator will be needed to predict how far the fine-tuner will need to rotate to get to the target pitch. This can be achieved by deriving a linear model for each string using the target hardware, and measuring the pitch recorded at several rotational intervals.

Driver Board Issues and Motor Resonance

The most common small-formfactor These types of motors are more suited to a Darlington array such as the ULN2003, which consists of seven NPN Darlington pairs and common-cathode clamp diodes that can rapidly switch inductive loads, an operation paradigm that is much better suited for unipolar motors.

Audio Input Changes

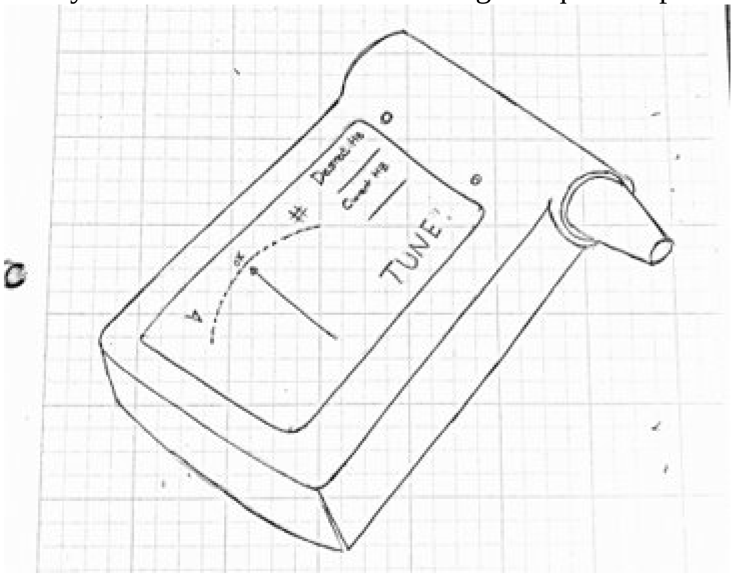
For the purposes of testing the core functionalities of the project’s tuning and control software quarter the standard Elegoo microphone circuit was implemented in in the circuit to handle the process of carrying the audible sounds to an observable analog signal. The use of the circuit was deemed to be temporary due to the minimal gain control and noise filtering the circuit provided as well as having poor isolation from other audible noise sources in the vicinity. Because of these reasons a microphone circuit was designed in order to deliver a low noise analog signal that would be able to accommodate any common transducer. The transducer itself had to be carefully selected so as to not pick up surrounding sounds not pertaining to the pitch of the violin, which could be a common source of error due to the likelihood of other instruments undergoing tuning within close proximity of the user of the device. In order to provide the necessary isolation for the transducer a piezoelectric contact microphone was chosen. This transducer, when affixed to the body of the violin will be able to pick up only the resonating pitch being created by the strings while isolating the surrounding ambient noises.

Power and Batteries

Powering the final product will need a 5 V regulator and a battery that can be recharged. A flat lithium ion polymer (LIPO) battery will be used. The rationale behind the battery selection is their size and max current capability. In the spring term we will be finalizing our hardware choices and making an energy budget for the tuner. Using the energy budget to select the a LIPO batterie that can output the required current by selecting the continuous discharge rate C for our needs. For charging the battery we need a circuit that will not only properly charge the battery but also have passthrough ports to operate the tuner while charging the battery. The first solution that we came up with was a LM2577 boost converter and a TP4056 protection circuit. This combination is big and does not allow for passthrough functionality. The current charge and regulator circuit being used is the PowerBoost 1000C. This circuit gives us everything we are looking for a 5 V regulator a charge/protect circuit and passthrough capability.

Physical Enclosure

The physical enclosure needs to be small enough to be hand held and not feel cumbersome. The case should be intuitive to use so anyone can use it when they first pick it up. The constraints to this are the motor and electronics. A custom PCB is a way to easily condense the size of the electrical components. The motor is something that can be smaller, but we would need to verify the new motor will have enough torque and precision to accurately tune the violin.

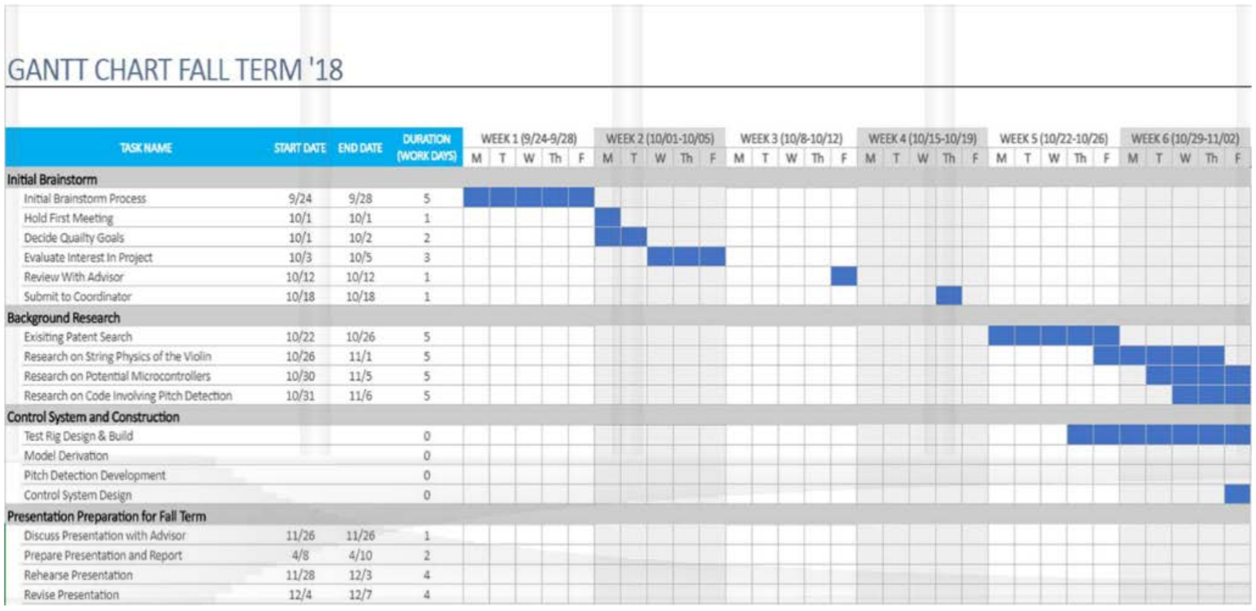


Front-End and User-Interface

The last addition to the system that is undergoing production currently is the Raspberry Pi, a small functional computer that will handle bridging the gap between an intuitive user interface and the low-level controls architecture of the tuner itself. Figure 2 demonstrates this bridge and it’s placement in the system. This will be done by having the Pi display a basic text-based user interface on an LCD controlled by a rotary encoder for sending user input. Any user input will then be interpreted in order to send the appropriate commands to the teensy via serial commands. The core functions being implemented will include the ability to select different stringed instruments, select a custom tuning profile, show live tuning progress for each string, and communicate any errors to the user. These commands will be sent serially from the Pi to the Teensy utilizing an Arduino library called CmdMessenger. This library allows for all commands that may be sent to be pre-defined on both the Pi and Teensy with the ability to handle each command based on the specific variables that are sent with it and alter device parameters and the message sent back accordingly, effectively creating a state machine shared between the two devices.

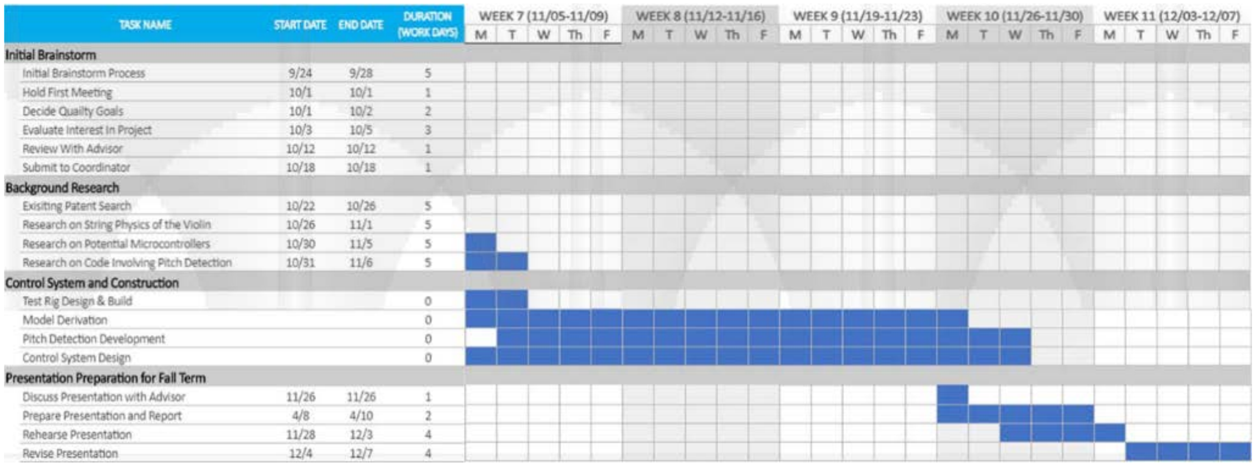
Once the functionality of the serial communications has been implemented between the Pi and Teensy and a basic text-based user interface has been created allowing for the user to operate the device to its full functionality, exploration into a graphical user interface will be considered. This will most likely consist of a larger LCD display capable of displaying images and animated graphics to the user in addition to text. This will be useful for fast and even more intuitive communication of data such as menu options, tuning progress, and error reporting. Accomplishing this will allow for the product to be more easily used by the intended target audience which is young and inexperienced musicians.

Work Schedule & Proposed Timeline



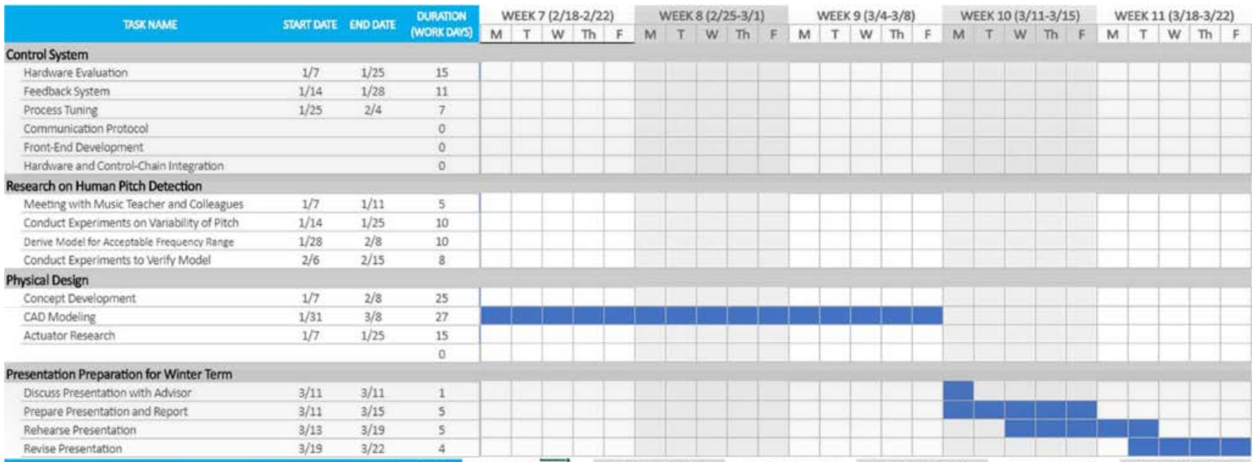
GANTT CHART FALL TERM '18

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)				
				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																														

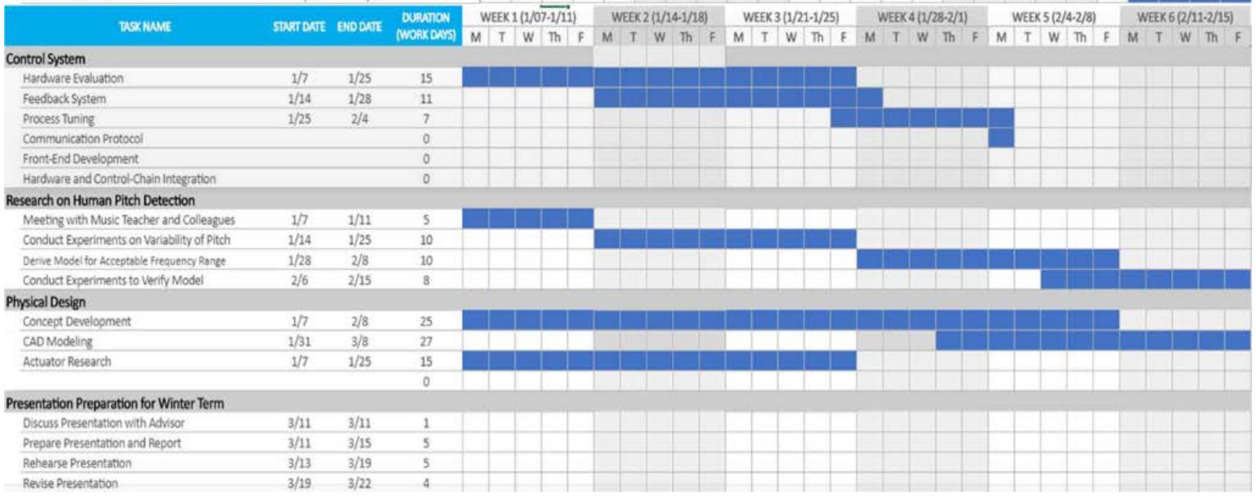


TASK NAME	START DATE	END DATE	DURATION (WORK DAYS)	WEEK 7 (11/05-11/09)					WEEK 8 (11/12-11/16)					WEEK 9 (11/19-11/23)					WEEK 10 (11/26-11/30)					WEEK 11 (12/03-12/07)				
				M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F
Initial Brainstorm																												
Initial Brainstorm Process	9/24	9/28	5																									
Hold First Meeting	10/1	10/1	1																									
Decide Quality 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																						🟦	🟦	🟦	🟦

Winter Term

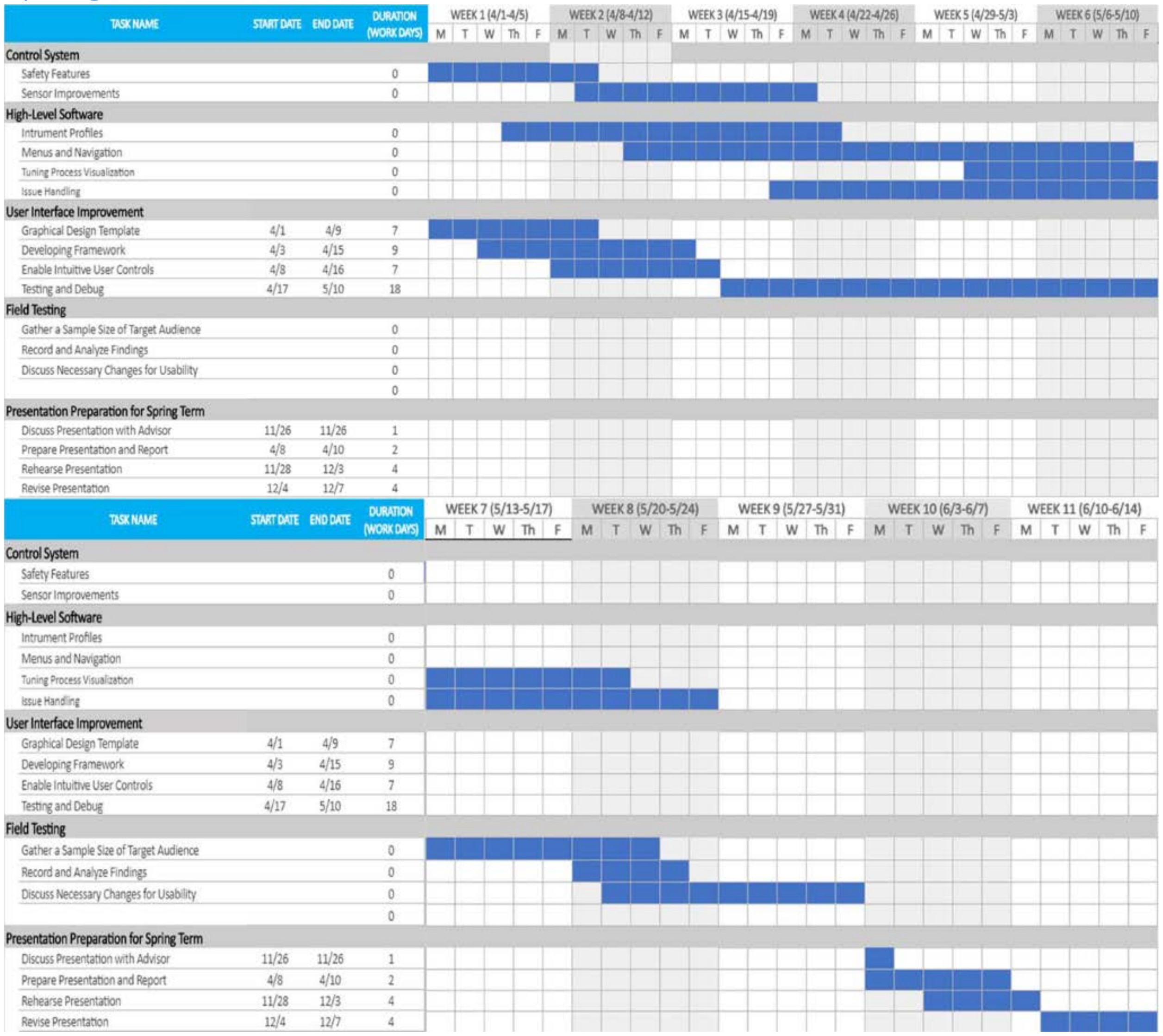


TASK NAME	START DATE	END DATE	DURATION (WORK DAYS)	WEEK 7 (2/18-2/22)					WEEK 8 (2/25-3/1)					WEEK 9 (3/4-3/8)					WEEK 10 (3/11-3/15)					WEEK 11 (3/18-3/22)				
				M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F	M	T	W	Th	F
Control System																												
Hardware Evaluation	1/7	1/25	15																									
Feedback System	1/14	1/28	11																									
Process Tuning	1/25	2/4	7																									
Communication Protocol			0																									
Front-End Development			0																									
Hardware and Control-Chain Integration			0																									
Research on Human Pitch Detection																												
Meeting with Music Teacher and Colleagues	1/7	1/11	5																									
Conduct Experiments on Variability of Pitch	1/14	1/25	10																									
Derive Model for Acceptable Frequency Range	1/28	2/8	10																									
Conduct Experiments to Verify Model	2/6	2/15	8																									
Physical Design																												
Concept Development	1/7	2/8	25																									
CAD Modeling	1/31	3/8	27	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦										
Actuator Research	1/7	1/25	15																									
			0																									
Presentation Preparation for Winter Term																												
Discuss Presentation with Advisor	3/11	3/11	1																🟦									
Prepare Presentation and Report	3/11	3/15	5																🟦	🟦	🟦	🟦	🟦					
Rehearse Presentation	3/13	3/19	5																		🟦	🟦	🟦	🟦	🟦			
Revise Presentation	3/19	3/22	4																								🟦	🟦	🟦



TASK NAME	START DATE	END DATE	DURATION (WORK DAYS)	WEEK 1 (1/07-1/11)					WEEK 2 (1/14-1/18)					WEEK 3 (1/21-1/25)					WEEK 4 (1/28-2/1)					WEEK 5 (2/4-2/8)					WEEK 6 (2/11-2/15)				
				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
Control System																																	
Hardware Evaluation	1/7	1/25	15	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦															
Feedback System	1/14	1/28	11						🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦														
Process Tuning	1/25	2/4	7															🟦	🟦	🟦	🟦	🟦	🟦	🟦									
Communication Protocol			0																					🟦									
Front-End Development			0																														
Hardware and Control-Chain Integration			0																														
Research on Human Pitch Detection																																	
Meeting with Music Teacher and Colleagues	1/7	1/11	5	🟦	🟦	🟦	🟦	🟦																									
Conduct Experiments on Variability of Pitch	1/14	1/25	10						🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦															
Derive Model for Acceptable Frequency Range	1/28	2/8	10																🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦					
Conduct Experiments to Verify Model	2/6	2/15	8																							🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Physical Design																																	
Concept Development	1/7	2/8	25	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦					
CAD Modeling	1/31	3/8	27																			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Actuator Research	1/7	1/25	15	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦															
			0																														
Presentation Preparation for Winter Term																																	
Discuss Presentation with Advisor	3/11	3/11	1																														
Prepare Presentation and Report	3/11	3/15	5																														
Rehearse Presentation	3/13	3/19	5																														
Revise Presentation	3/19	3/22	4																														

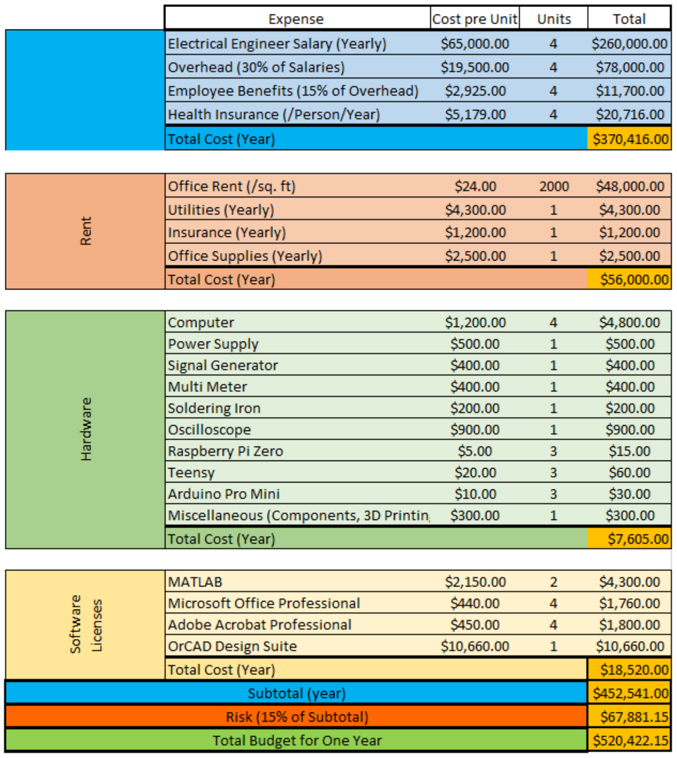
Spring Term



TASK NAME	START DATE	END DATE	DURATION (WORK DAYS)	WEEK 1 (4/1-4/5)					WEEK 2 (4/8-4/12)					WEEK 3 (4/15-4/19)					WEEK 4 (4/22-4/26)					WEEK 5 (4/29-5/3)					WEEK 6 (5/6-5/10)				
				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
Control System																																	
Safety Features			0	■	■	■	■	■	■	■																							
Sensor Improvements			0							■	■	■	■	■	■	■	■	■	■														
High-Level Software																																	
Intrument Profiles			0				■	■	■	■	■	■	■	■	■	■	■	■	■	■													
Menus and Navigation			0									■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
Tuning Process Visualization			0																							■	■	■	■	■	■	■	■
Issue Handling			0															■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
User Interface Improvement																																	
Graphical Design Template	4/1	4/9	7	■	■	■	■	■	■	■																							
Developing Framework	4/3	4/15	9			■	■	■	■	■	■	■	■	■																			
Enable Intuitive User Controls	4/8	4/16	7						■	■	■	■	■	■	■																		
Testing and Debug	4/17	5/10	18													■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Field Testing																																	
Gather a Sample Size of Target Audience			0																														
Record and Analyze Findings			0																														
Discuss Necessary Changes for Usability			0																														
			0																														
Presentation Preparation for Spring 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																														

TASK NAME	START DATE	END DATE	DURATION (WORK DAYS)	WEEK 7 (5/13-5/17)					WEEK 8 (5/20-5/24)					WEEK 9 (5/27-5/31)					WEEK 10 (6/3-6/7)					WEEK 11 (6/10-6/14)				
				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
Control System																												
Safety Features			0																									
Sensor Improvements			0																									
High-Level Software																												
Intrument Profiles			0																									
Menus and Navigation			0																									
Tuning Process Visualization			0	■	■	■	■	■	■	■																		
Issue Handling			0	■	■	■	■	■	■	■	■	■	■															
User Interface Improvement																												
Graphical Design Template	4/1	4/9	7																									
Developing Framework	4/3	4/15	9																									
Enable Intuitive User Controls	4/8	4/16	7																									
Testing and Debug	4/17	5/10	18																									
Field Testing																												
Gather a Sample Size of Target Audience			0	■	■	■	■	■	■	■	■																	
Record and Analyze Findings			0						■	■	■	■																
Discuss Necessary Changes for Usability			0							■	■	■	■	■	■	■	■	■										
			0																									
Presentation Preparation for Spring 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																						■	■	■	■

Industrial Budget



	Expense	Cost pre Unit	Units	Total
	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 Printin	$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

Item Name	Unit Price	Quantity
USB to TTL Adapter 2-pack	$9.38	1
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 Acessory Kit	$24.37	1
IEEE Locker Fall Term	$20.00	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
IEEE Locker Winter Term	$30.00	1
Teensy 3.5	$27.00	1
MP6500 Stepper Motor Driver	$10.68	2
5VDC 32-Step 1/16 Gearing	$7.94	2
Teensy 3.5	$27.00	1
Total	$255.57	

Global, Cultural, Societal, Environmental, and Economic 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.

The physical components that are required to build this automated tuner should not pose any significant environmental impacts. All the electronics found in the tuner are standard off the shelf components such as a small stepper motor, Arduino boards as well as wires found in most electronic devices, they are all certified by ROHS and the like. The housing is made out of ABS plastic, this plastic is a thermoplastic and can be easily recycled by chipping, melting, and extruding into a new product.

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.

Musical discouragement is a common attitude to have in one’s career whether professional or amateur. Starting out, many beginner violinists struggle to find the encouragement to continue for multiple years, especially without a teacher. Typically, this stems from many reasons but most commonly, due to not producing a producing a quality sound. Naturally, this automatic violin tuner will help establish a fundamental basis for proper violin playing. With patience and diligence students are hoped to prolong their playing with the necessary fundamentals such as proper tuning mastered.

The product would allow for an easier entry into the realm of orchestral string instruments. This could draw more beginner musicians to be drawn into orchestral sting instruments when starting out allowing for there to be an increase in experienced string players when they reach adulthood. This has the potential to have and economic impact as the market for string instruments and their accessories expands.

Summary & Conclusions

The progress on the autonomous violin tuner has been going at a steady pace and many components have been tentatively finalized. The motor was selected to be a stepper motor with a 1/16 gearing ratio for the required torque. A motor controller the L293D Quadruple Half-H this chip has a small footprint, a low power consumption, and works well with our selected motor. To use the piezo pickup microphone with the teensy an amplifier circuit needed to be constructed this circuit will work with both a condenser and piezo microphones. The YIN detection algorithm is being used for pitch detection.

This term has been focused on component selection and finalization. In the final term we will be focusing on laying out the PCB, designing the tuner housing, and constructing a final product prototype.

References

Ang, K. H., Chong, G., & Li, Y. (2005). PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 559-576.

De Cheveigne, A. K. (2001, May 1). *YIN, A Fundamental Frequency Estimator for Speech*. Retrieved from audition: http://audition.ens.fr/adc/pdf/2002_JASA_YIN.pdf

Appendix A: Design Constraints Summary

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. 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 way that the coupler is designed allows for a "fail save" to protect the fine tuners. The couplers' depth is a length designed to seize contact if the fine tuner screw is used up as to avoid causing any damage. Conversely, the device should recognize if the screw is "too loose". A frequency differential may be a suitable solution. If the microphone does not register change in frequency it may be concluded that either the screw is used up or the opposite: the screw is too far out.

Naturally, because of the string properties of the violin, it took a more considerable amount of time to tune the lower strings (G and D) compared to the higher strings (A and E). On average, it took 15-20 seconds to tune the G string whereas it only took approximately 5-7 seconds to tune the E string. In future design considerations, selective motor speeds may be explored depending on which tuning profile is selected.

Appendix B: Additional Diagrams

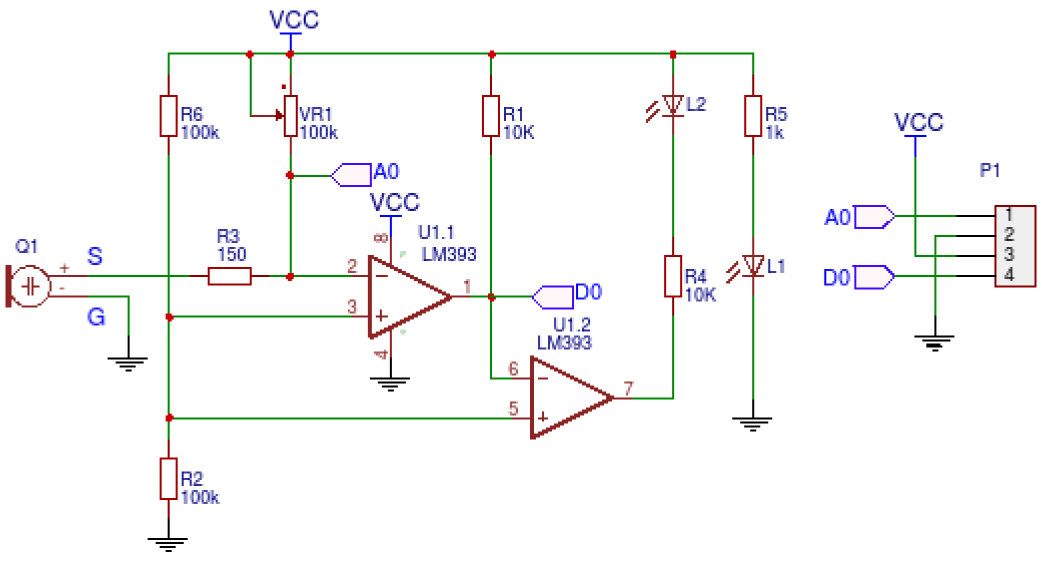


Figure 13 Standard Arduino Microphone Circuit

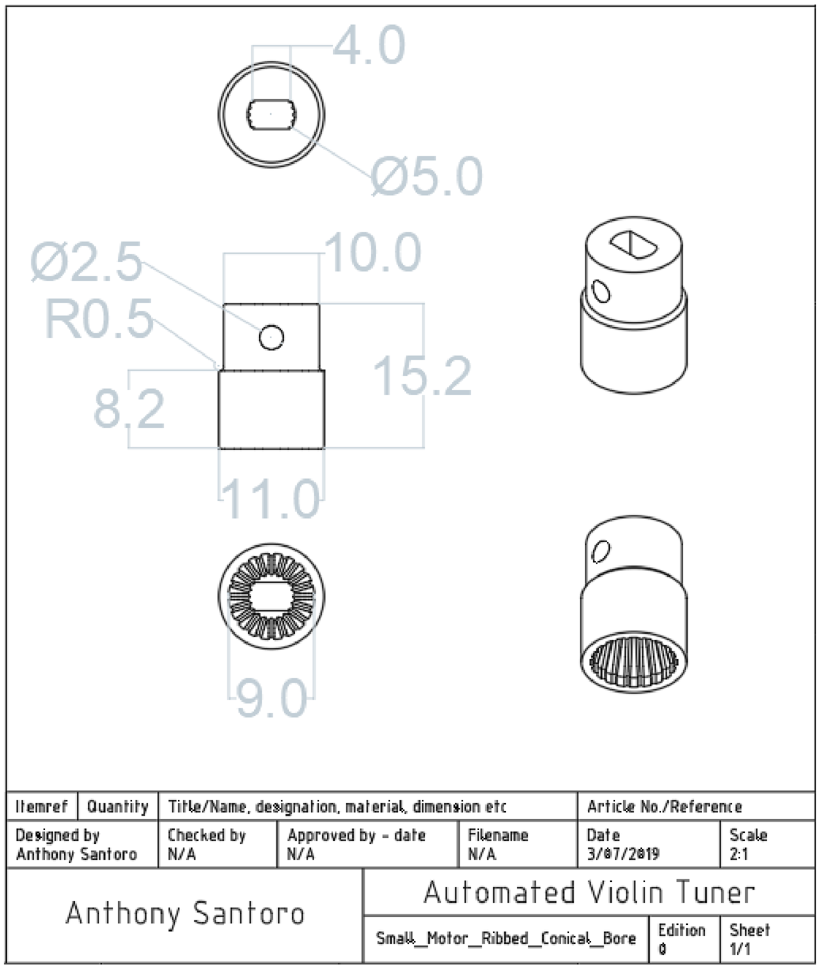


Figure 14: Fine-Tuner Motor Coupler

Appendix C: Teensy Firmware

```
#include <Audio.h>
#include <SPI.h>
#include <Wire.h>
#include <CmdMessenger.h>

CmdMessenger cmdMessenger = CmdMessenger(Serial);
enum TuningTargets{STRING_G3, STRING_D4, STRING_A4, STRING_E5};
enum TuningStates{
      NOT_READY,
      READY,
      TUNING,
      DONE
    }tuningState;

#define kSerialSpeed 115200

float note;
float prob;

bool RUN = true;
```

```
void setup() {
  audioSetup();
  motorSetup();
  scheduleRoutines();
  attachCommandCallbacks();

  Serial.begin(kSerialSpeed);
}

void loop() {
  cmdMessenger.feedinSerialData();
  performPolledRoutines();
}

/////////////////////////////////////////////////////

#include <AccelStepper.h>

#define speedLimit 500

//ULN2003//
AccelStepper stepper(4, 2, 3, 5, 4);

float stepperSpeed;

void motorSetup(){
  stepper.setMaxSpeed(speedLimit);
}

void runStepper(){
  stepper.setSpeed(stepperSpeed);
  stepper.runSpeed();
}

//////////////////////////////////////////////////////

#include <PID_v1.h>

TuningTargets tuningTarget = STRING_D4;

double Setpoint, Input, Output;
double p_Kp=2, p_Ki=5, p_Kd=1;
PID tuningPID(&Input, &Output, &Setpoint, p_Kp, p_Ki, p_Kd, DIRECT);

float FREQ_TARGET;
float focusBand = 3;

AudioInputAnalog              adc1; //adc(A2);
AudioAnalyzeNoteFrequency     notefreq1;
AudioConnection               patchCord1(adc1,0, notefreq1,0);

long lastSampleTime = 0;
bool noteAvailable = false;

void setTuningTarget(TuningTargets target){
  switch(target){
    case STRING_G3: {
      FREQ_TARGET = 196.0;
      break;
    }
    case STRING_D4: {
      FREQ_TARGET = 293.7;
      break;
    }
    case STRING_A4: {
      FREQ_TARGET = 440.0;
      break;
    }
    case STRING_E5: {
      FREQ_TARGET = 659.3;
      break;
    }
  }
}

void audioSetup(){
  AudioMemory(30);
  notefreq1.begin(.15);
  tuningPID.SetMode(AUTOMATIC);
  tuningPID.SetOutputLimits(-speedLimit,speedLimit);
  setTuningTarget(STRING_E5);
}
void detectPitch(){
  //Serial.println("TEST");
  if (notefreq1.available()) {
    noteAvailable = true;

    lastSampleTime = millis();
    note = notefreq1.read();
    prob = notefreq1.probability();
    Serial.printf("%3.2f\n", note);
    //Serial.printf("Note: %3.2f | Probability: %.2f\n", note, prob);

  }else{
    noteAvailable = false;
  }
}

void tuneString(){
  if (noteAvailable){
    if (note>FREQ_TARGET-50 && note <FREQ_TARGET+50){
      noInterrupts();
      stepperSpeed = map(note, FREQ_TARGET-focusBand, FREQ_TARGET+focusBand,
speedLimit, -speedLimit);
      interrupts();
    }
  }
  if ((millis()-lastSampleTime) > 100){
    noInterrupts();
    stepperSpeed = 0;
    interrupts();
  }
}

void runStateMachine(){
  switch(tuningState){
    case NOT_READY: {
      break;
    }
    case READY:{
      break;
    }
    case TUNING:{
      tuneString();
      if (note>FREQ_TARGET-1.0 && note<FREQ_TARGET+1){
        tuningState = DONE;
        //cmdMessenger.sendCmd(kAcknowledge, "Done");
        noInterrupts();
        stepperSpeed = 0;
        interrupts();
        Serial.println("Done");
      }
      break;
    }
    case DONE:{
      break;
    }
  }
}

////////////////////////////////////////////////////////////

#include <Metro.h>
#define MOTOR_INTERVAL 10 //Defined in Microseconds

#define LCD_REFRESH_INTERVAL 100 //Defined in Milliseconds
bool LCD_REFRESH_TOGGLE = true;

bool PITCH_DETECT_TOGGLE = true;

IntervalTimer myTimer;
Metro lcdMetro = Metro(LCD_REFRESH_INTERVAL);

void scheduleRoutines(){
  myTimer.begin(runStepper, 10);
  //myTimer.priority(1);
}

void performPolledRoutines(){
  if (PITCH_DETECT_TOGGLE){
    detectPitch();
    runStateMachine();
  }
}

////////////////////////////////////////////////////////////

enum
{
  kAcknowledge,
  kSetTargetString,
  kBeginTuning
};

void attachCommandCallbacks()
{
  cmdMessenger.attach(kSetTargetString, OnSetTargetString);
  cmdMessenger.attach(kBeginTuning, OnBeginTuning);
  cmdMessenger.printLfCr();
}

void OnBeginTuning(){
  tuningState = TUNING;
  cmdMessenger.sendCmd(kAcknowledge, "Tuning Start");
}

void OnSetTargetString()
{
  String target = cmdMessenger.readStringArg();

  if (target == "E"){
    cmdMessenger.sendCmd(kAcknowledge, "Target: E String");
    setTuningTarget(STRING_E5);
  }
  else if(target == "A"){
    cmdMessenger.sendCmd(kAcknowledge, "Target: A String");
    setTuningTarget(STRING_A4);
  }
  else if(target == "D"){
    cmdMessenger.sendCmd(kAcknowledge, "Target: D String");
    setTuningTarget(STRING_D4);
  }
  else if(target == "G"){
    cmdMessenger.sendCmd(kAcknowledge, "Target: G String");
    setTuningTarget(STRING_G3);
  }else{
    cmdMessenger.sendCmd(kAcknowledge, "Unknown String");
  }


}
```

Anthony C. Santoro
acs385@drexel.edu

1013 Surrey Road • Somerdale, NJ 08083 • (856) 745-0473

Education

Drexel University — Philadelphia, PA
Bachelor of Science in Electrical Engineering — Anticipated Graduation: June 2018

Burlington County College — Mount Laurel, NJ
Associate of Science in Engineering — September 2013 to June 2015
Cumulative GPA: 3.04

Honors

- Eagle Scout, Boy Scouts of America, 2013
- First Place, Engineering Week, Camden County College, 2013
- First Place, Tech Challenge, Villanova University, 2012
- Academic Scholarship - Drexel University
- Second Place, Engineering Week, Camden County College, 2012
- First Place, Engineering Week, Camden County College, 2011

Skills

Computer: Auto-CAD, Microsoft Office (Excel, PowerPoint, Word), Python, C++, Matlab, Adobe (illustrator), LTSpice, PSpice
Laboratory: Oscilloscope, Multi-Meter, Function Generator, Soldering, Brazing, Laser Cutting, 3D Printing

Relevant Coursework

Digital Logic Design	Fundamentals of Engineering Design	Fundamentals of Electric Circuits
Programming for Engineers (Python)	Analog Electronics	Digital Electronics

Tiki Obstacle Project

Designed obstacle for RC car consisting of large Tiki with rotating eyes and offset teeth. Read schematic and assembled motors, photo sensor, and sound system to create efficient circuit. Measured parts carefully and cut parts precisely using laser cutter. Collaborated with team of five to evaluate and summarize process in PowerPoint presentation

3D Printer Project

Initiated project to build working 3D printer out of two defunct 3D printers. Tested hot ends of printer and made repairs; ensured printer bed was level. Made test prints to evaluate success of project

Experience

Resolution Management Consultants — Marlton, NJ
Engineering Analysis Technician — March 2017 to September 2017
- Work with a principal partner to develop proposals, reports, and court documents.

Checkpiont Systems — Thorofare, NJ
RF Engineer, CO-OP — March 2016 to September 2016
- Worked under a mentor to detect and troubleshoot problems in different Electronic Article Surveillance systems
- Circuit Simulation using LTSpice

Euro Tans and Spa — Hawthorn, NJ
Maintainance Mechanic — June 2013 to Present
- Maintain 20 tanning beds and 22 tanning booths weekly and repair as needed

Douglas Bartlett — Somerdale, NJ
Construction — June 2013 to Present
- Supervise up to three crew members to ensure high quality of work and adherence to tasks and deadlines

Activities

Member, IEEE, Drexel University, 2015 to Present

Michael Barnes

Phone: (845) 591 0303 | Email: mrb372@drexel.edu | www.linkedin.com/in/MichaelBarnes372
Address: 3175 JFK Boulevard, Philadelphia, PA 19104

EDUCATION

DREXEL UNIVERSITY — Philadelphia, PA
Bachelor of Science Electrical and Computer Engineering — Anticipated Graduation: June 2019

TECHNICAL SKILLS

Programming Languages

- MATLAB | C++ | Visual Basic | AutoHotkey | Java | SQL | Python

Software

- ADS | Cadence Virtuoso | Microsoft Office | Creo Parametric | AutoCAD | Visual Studio | PSS/E (Siemens) | PSpice |

Relevant Skills

- Skilled in use of oscilloscopes, function generators, spectrum analyzers, logic analyzers, and multimeters
- Experience constructing analog circuits and analyzing circuit schematics
- Experience in FPGA design through the use of Altera's Quartus Prime software
- Experience with design, configuration, assembly, performance optimization and maintenance of computers systems

EXPERIENCE

LOCKHEED MARTIN — Moorestown, New Jersey
Electrical Engineer – RFIC group — April 2018 – September 2018

- Assisted in developing test and analysis software for Solid-State Radar
- Facilitated testing, analysis, and presentation of RFIC product for customer
- Communicated with vendors to find root cause for problems seen in products under test
- Developed and maintained software for quick analysis of large datasets collected from products for use in calibration

SRI INTERNATIONAL — Princeton, New Jersey
Electrical Engineer — April 2017 – April 2018

- Assisted construction and design of a new type of retinal imaging device
- Performed root cause analysis and solution implementation of product RMAs
- Wrote up failure analysis documentation on product RMAs for the customer
- Examined PADS layout in conjunction with board schematics to troubleshoot PCB faults
- Assisted senior engineers in researching responses for government issued RFIs

PJM INTERCONNECTION — Audubon, Pennsylvania
Transmission Service Engineer — March 2016 – September 2016

- Facilitated the transfer of energy into, out of, and through the PJM control area
- Oversaw and aided in the purchase and scheduling of transmission service
- Automated tasks via macros and scripts to boost efficiency and productivity
- Leveraged critical thinking skills to solve complex problems while collaborating with other team members

FRESHMAN DESIGN PROJECT

ELECTROLUMINESCENT DIP-COATING DEVICE — Drexel University
Project Designer and Manufacturer — March 2015 – June 2015

- Designed lab equipment in collaboration with Drexel Nanophotonics+ laboratory
- Researched the annealing behavior of DuPont LuxPrint 8153 and 8154L
- Published team progress to blog at du2015-grp940-01.blogspot.com

RELEVANT COURSEWORK

Electronic Devices	Digital Logic Design	Analog Electronics
Multivariate Calculus	Antennas & Radiating Systems	Differential Equations
Transform Methods I & II	Electrical Engineering Lab	RF Components & Techniques

AFFILIATIONS

ASME: Member	2015 - Present
IEEE, DREXEL CHAPTER: Member	2015 - Present
DLAB, DREXEL MAKERSPACE: Member and Project Designer	2014 - Present
ENGINEERING LEARNING COMMUNITY: Selected Member of Lockheed Martin Leadership Program	2014-2015

Steven J. Ngan
3409 Race Street Apartment C
Philadelphia, PA 19104
(240) 478-3257
sjn36@drexel.edu

Education

Drexel University, College of Engineering — Philadelphia, PA
Bachelor of Science in Electrical Engineering — Anticipated Graduation: June 2019

Skills

Computer: Microstation, AutoCAD, Visual Lighting, SKM, Python, Multisim, Distribution Management System (DMS), MATLAB, C++ Arduino Software, Creo Parametric, Microsoft Office (Excel, Word, PowerPoint, Access)
Mechanical: Actuator, solenoid and valve restoration, circuit boards with RGB Arduinos
Language: Khmer (Conversationally fluent)

Relevant Coursework

Digital Logic Design	Programming for Engineers	Transformation Methods and Filtering I, II
Electronic Devices	Analog Devices	Electric Motors and Controls
Electrical Engineering Lab I	Digital Electronics	Renewable Energy
Power Systems I	Computer Networking	

Experience

Gannett Fleming — Marlton, New Jersey
Electrical Design Engineer — March - September 2018
- Designed and drafted various upgrades to facilities, lighting contours, and wiring diagrams using Microstation and CAD
- Analyzed protective relay diagrams to generate trip charts involving breaker failure protection, bus differentials, and over current protection
- Utilized SKM software to calculate short circuit current for panel ratings and to provide Arc flash calculations
- Outlined markups and vendor drawings for medium voltage transformer replacements for PSE&G

Philadelphia Energy Solutions — Philadelphia, Pennsylvania
Electrical and Instrumental Reliability Engineer — March - September 2017
- Cooperated with instrumentation technicians to restore malfunctioning solenoids, valves and programmable logic controllers
- Published and redesigned piping and instrumental diagrams, single line drawings and loop diagrams of machinery in the refinery
- Inspected power transformers for polychlorinated biphenyl (PCB) testing
- Installed various types of flow technologies such as orifice plates, vortex meters and Coriolis meters in obsolete piping

Exelon Corporation (PECO Energy Company) — Philadelphia, Pennslyvania
Transmission and Substation Systems Tester — March - September 2016
- Assisted supervisory control and data acquisition (SCADA) technicians to troubleshoot communication failure in the distribution management system
- Tested levels of volts, watts, and vars on remote terminal units and substations throughout Philadelphia
- Collaborated with field technicians to monitor status, alarms and to conduct point to point tests
- Reported circuit communication failures through Verizon work orders

Unique Fine Jewelry — Rockville, Maryland
Salesperson, Repairman — December 2012 - July 2016
- Communicated with customers regarding sale appraisals and repairs
- Filled customer work orders over phone and resolved concerns
- Soldered necklaces, bracelet bands and mended broken clasps
- Analyzed watch movements and fixed any malfunctions

Activities

Brother, Alpha Phi Omega National Service Fraternity, Drexel University, 2016 to Present
Captain, Intramural Volleyball Team, Drexel University, 2015 to Present
First Violin, Drexel University Orchestra, Drexel University, 2014 to Present
Member, Society of Asian Scientists and Engineers, Drexel University, 2014 to Present

YOSHIN GOVENDER

ELECTRICAL ENGINEER

(630) 605 1519 – yoshin.govender@gmail.com – linkedin.com/in/yoshingovender/

I am an Electrical Engineering student at Drexel University directing my studies to enter the field of controls, automation, and robotics. Throughout my educational career, I have studied and worked in four different cities and three different countries. My travels have given me a perspective that I did not previously have, and after earning my degree, it is my goal to create new technology that will better the lives of humans across the globe.

EXPERIENCE

NASA
2017

APRIL– present

COMPUTER SCIENCE & ROBOTICS CO-OP

Worked at the NASA Goddard Space Flight Center within the Satellite Servicing Projects Division. Designed a piece of software to operate camera positioning units onboard a flight spare tool for the Robotic Refueling Mission 3 (RRM3).

NASA
2017

APRIL– SEPTEMBER

ELECTRICAL ENGINEERING & AVIONICS CO-OP

Worked at the NASA Goddard Space Flight Center in Greenbelt, Maryland on Project PACE and project MUSTANG. Drafted part drawings, and programed Python-based control software with integrated GUIs to test the cards and racks being developed.

DD ROBOT
2016

APRIL - SEPTEMBER

INTERNATIONAL ELECTRICAL ENGINEERING CO-OP

Worked internationally at Ding Dang Automation in Chengdu, China. Designed and constructed robotics that employed computer vision, path following, and PID control. Worked with colleagues to overcome stark language barrier, and helped champion outreach with new international clients.

FRESHMAN DESIGN
2015

APRIL - JUNE

ROBOTIC SURVEILLANCE SCOUT

Designed and constructed a prototype of a CubeSat surveillance robot meant to scan the exterior of spacecraft for defects. Worked within a team of four and researched programming, robotics, and 3D printing to complete the independently-led project.

EDUCATION

Drexel University

2014 – 2019

BACHELOR OF SCIENCE

Bachelor of Science in Electrical Engineering

Anticipated Graduation - June, 2019

Cumulative GPA: 3.63

SKILLS

Programming Languages	Software	Platforms	Other
Python	Qt	Linux	Control System Design
MATLAB	AutoCAD	Windows	I^2C Protocol
OpenCV	Inventor	Arduino	Serial Protocol
C++	Adobe Suite	Raspberry Pi	Digital Circuitry