Auto-Tuner with LCD Display

By

John Driscoll

Lee Susara

Nicholas Chan

Final Report for ECE 445, Senior Design, Spring 2026

TA: Eric Tang

May 2026

Project No. 39

Abstract

This project addresses the accessibility gap in musical instrument maintenance for individuals with hearing loss or limited manual dexterity through the development of the Auto-Tuner with LCD Display. Our solution was to develop a device capable of accurately detecting string frequencies and mechanically adjusting tuning pegs to achieve standard pitch. The system utilizes piezoelectric sensors for signal acquisition which removes the need for auditory feedback and a microcontroller driven frequency analysis algorithms for controlling stepper motors. Even though we were able to successfully implement the detection and control logic as well as the physical housing and user interface, mechanical constraints - specifically insufficient torque strength to tune a guitar string - prevented our project from being fully successful. This project validates our original design and provides a clear pathway for a more successful automatic tuner.

Contents

1. Introduction

1.1 Problem

Stringed instruments like guitars are a very important cultural tool. Found in almost every country in the world, music from stringed instruments has united groups and communities for thousands of years. However, many people, for example, those who are hard of hearing, find it very difficult to use these instruments. For these people, tuning a guitar is naturally very difficult, as most of the time they cannot detect minute pitch inaccuracies between notes. This process can then be extremely time-consuming, tedious, and often inaccurate as pegs are turned too far. Additionally, this process is almost impossible for individuals with limited hand strength or missing extremities. To make these important cultural items as accessible as possible, it is necessary to have a device that allows easy tuning for all individuals, regardless of their circumstances, and that is more efficient than traditional tuning methods.

Guitar tuning can be difficult, inconsistent, and time-consuming for beginners. Our goal is to automate this tuning process. Using piezoelectric discs, we can capture the note's frequency and use NEMA 11 stepper motors to automatically turn the peg.

1.2 Solution

Our proposed solution to this problem is a motorized guitar tuner. This tuner would remove all inaccuracy and physical exertion from the tuning process, allowing for a perfectly tuned guitar with a fraction of the effort. The tuner has 3 main pieces: 1) a sensor which captures the vibrations produced by a string being plucked, 2) a microcontroller which interprets the sensor's values, converts them to a pitch value, and sends adjustment values to the peg turners, and 3) peg turners which adjust the position of the pegs to bring the string into tune. All of these components would be held in a plastic case, which would be clipped onto the guitar headstock. Once all pegs are connected to their designated turners, the device will no longer need to be removed until tuning is complete. This allows all strings to be tuned simultaneously, reducing strain and potential damage to the instrument.

The top of the tuner will include two main features: 1) an LCD display to convey information about the string and what adjustments need to be made, and 2) a series of buttons that allow adjustments to the tuning procedures in case a different standard is desired for each string. This tuner, powered by an internal battery, would allow for quick, efficient tuning of a guitar as easily as possible.

1.3 High-Level Requirements

- Allow the user to see in real time what note is being played
- Tune each string to within ± 2 Hz
- Tune the whole guitar in under 60 seconds to maintain efficiency
- Detect frequency within 250 ms of pluck
- Limit our applied torque to reduce the chance of the string snapping.

1.4 Block Diagram

a. Power Supply:
    i. Provides power to all other subsystems
b. User Interface:
    i. Provides the user with the necessary information during the tuning process
c. Position Subsystem:
    i. Records angles and saves them into tuning profiles for later usage
d. Vibration Subsystem:
    i. Responsible for receiving frequencies and putting them through a low-pass filter.
e. Motor Subsystem:
    i. Responsible for turning the tuning pegs
f. Computation Unit:
    i. Responsible for receiving and transferring signals to other subsystems

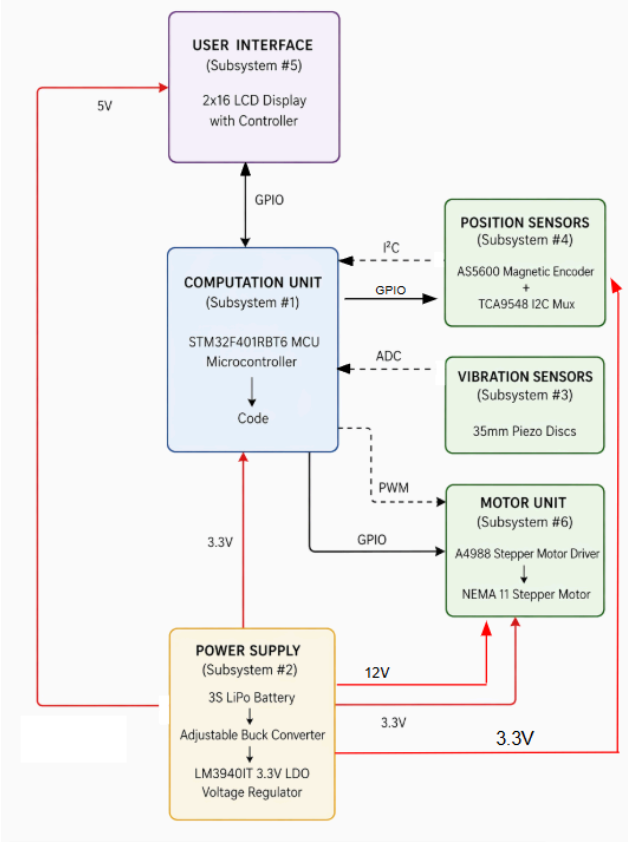


Figure 1: Block Diagram

2. Design

2.1 Design Procedure

Our original design for the guitar auto-tuner featured a box that housed all mechanical components attached to the guitar head and an LCD display that let the user observe the tuning process. The mechanical components consisted of a set of microphones that would record the pitch of the string, six servo motors that would attach to the guitar keys to tune the strings, a controller for the logic of the tuner, an LCD screen to output information about the tuning process, and a battery to power the device. This design focused on being functional and portable, two qualities we thought were necessary for a successful product. Even though our design has undergone numerous changes, we have tried to keep it as close as possible to what we originally envisioned.

Even though we originally planned to use servo motors for our design, we found that this was not an optimal choice. The first hurdle to overcome was introducing more degrees of rotation for the servo motor. Servo motors typically have 180 degrees of rotation, which is not enough to tune a guitar, as multiple 360-degree rotations might be necessary depending on how out of tune a string is. A solution to this was a continuous rotation stepper motor, but these motors often had less torque than was necessary to tune a string and were very imprecise, so other options had to be considered. We ended up choosing a Nema 11 stepper motor as it had the option for extremely small steps, which would aid in precise tuning; its holding torque (0.05884 Nm) was greater than the required tuning torque (0.021 Nm) calculated from the gear ratio of a standard guitar, and was cheap enough that buying six motors would not put us over budget. This meant that our original motor subsystem also had to be updated to include stepper motor drivers, in addition to the stepper motors, rather than the servo motors.

Our original idea of using microphones also had to be refined after learning from our TA that microphones would pick up too much noise, making accurate tuning almost impossible. To address this problem, we switched to piezoelectric discs, an op-amp, and a low-pass filter. The piezoelectric discs would pick up vibrations from the guitar while reducing noise, with the low-pass filter further filtering noise above 600 Hz to enable the precise tuning we desired. The op-amp was used to amplify the signal from the disc, allowing us to accurately determine the string's pitch. This section was further refined after learning that multiple piezo discs placed at different locations on the guitar could provide the desired amplification while also yielding a more accurate reading. This simplified the vibration subsystem, as only the piezo discs and low-pass filter were needed, rather than including the op-amp as well.

After learning about the speed of our motors and the time required to tune a string, we decided to add a new subsystem: the position subsystem. This subsystem was important because it tracked the motor's current position, which could be saved to a user profile. This information is important because it significantly reduces the time required to tune a string, as the motor can turn the peg to the last known “in-tune” position without waiting for the user to pluck the string. This means only small changes need to be made at the end to keep the string in tune. The idea and execution of our code stayed relatively the same, focusing on different tuning preferences and user profiles stored in a dictionary, but the logic for motor rotation and vibration sensing changed as we chose different components than in our original idea.

The final major change to our project was a series of changes to the physical design of our tuner. After acquiring a guitar, we realized the keys were in a different orientation than expected. Instead of keys that stuck out of the head on the sides, they came out of the bottom of the guitar, so we had to shift from a wide box to a deep one. We also realized that our original placement of the LCD screen made it hard to read while the tuner was in use, so we added an arm-like structure to the top of the box to allow repositioning of the screen during use.

Figure 2: Original Design of Guitar Auto-Tuner.

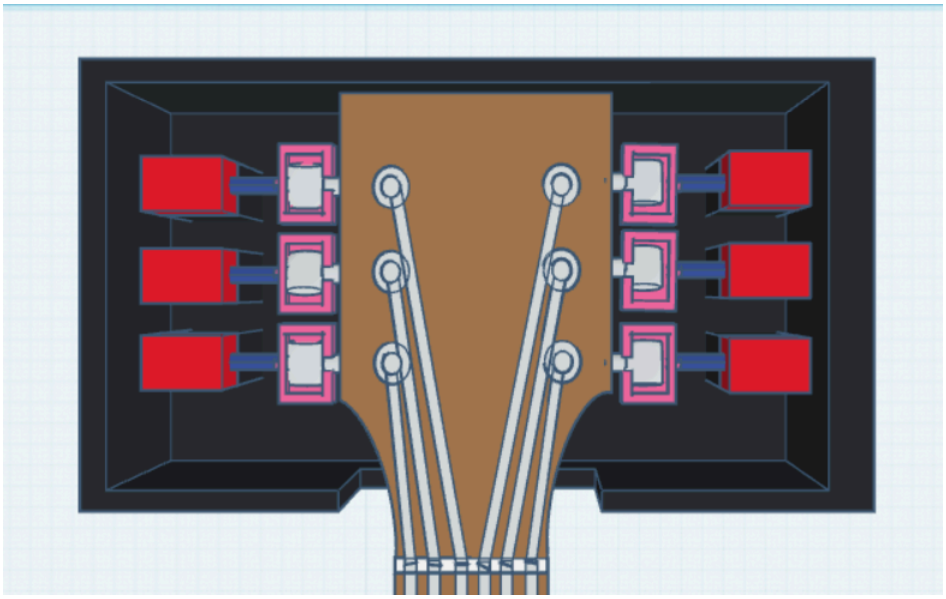


Figure 3: 3d Model of Original Design

Figure 4: Revised Lid to Allow for Repositional LCD Screen

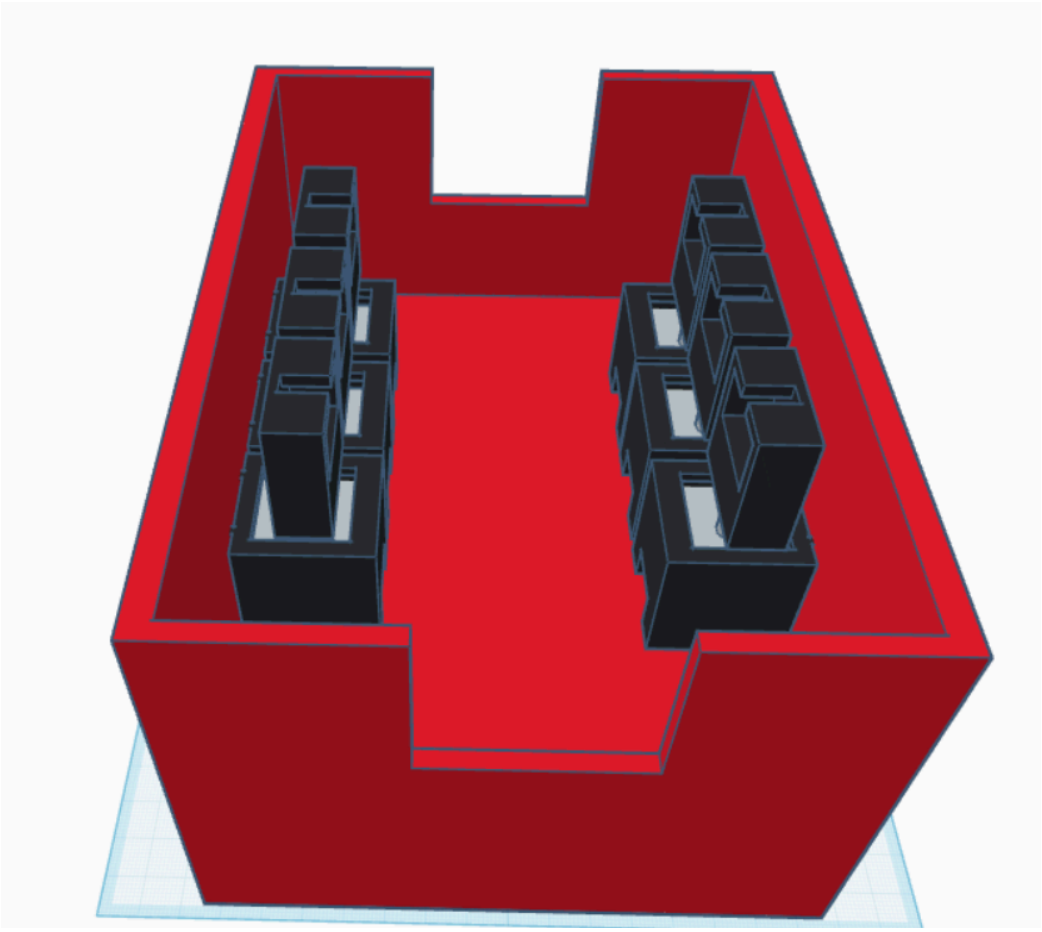


Figure 5: 3d Model of Final Design

2.3 Design Details

We have 6 subsystems: Position Sensors, Vibrations Sensors, User Interface, Power Unit, Motor Unit, and Computation Unit. Starting with our power unit, we will send our 12V through both a 5V buck converter and a 3.3V Linear Voltage Regulator to power all of our components. This includes the motors, MCU, magnetic encoders, and User Interface. Our Motor unit is made up of 2 components: The Nema 11 motor and the A4988 stepper driver. This, along with our 3D-printed coupler, will attach to the guitar's pegs and physically turn the tuning key. Our positions and vibration sensors will simply take input from the guitar, such as the frequency being played and the position of the tuning peg. This will allow us to adjust the pegs so the correct frequency is played and to save the angle of the tuning peg to preserve certain tuning profiles. Our User Interface will allow us to signal our Computation Unit about what we would like to do, such as saving tuning profiles, tuning the guitar to a specific tuning profile, and creating a new tuning profile. Lastly, our Computation Unit will be the “brain” of our project, taking in all of our inputs and using the code we download to the MCU to perform certain instructions, such as determining which way to turn the pegs and telling us what note is currently being played.

2.3.1 Control Subsystem

The control subsystem connects to every other subsystem and consists of an STM32F401RBT6 MCU that receives input from the vibration subsystem (via piezoelectric discs), sends feedback to the motors subsystem (NEMA 11 Stepper Motors), reads position from an AS5600 magnetic encoder, and then sends important information to the user interface (Densitron 162-H1-4C LCD Display Panel). We chose this MCU because it had many GPIO outputs, allowing us to generate many logic signals to send to our other peripherals. It is also very power efficient, so we did not have any worries about runtime being cut short or our MCU getting too hot. Below is our MCU schematic showing our STM32F401RBT6 microcontroller.

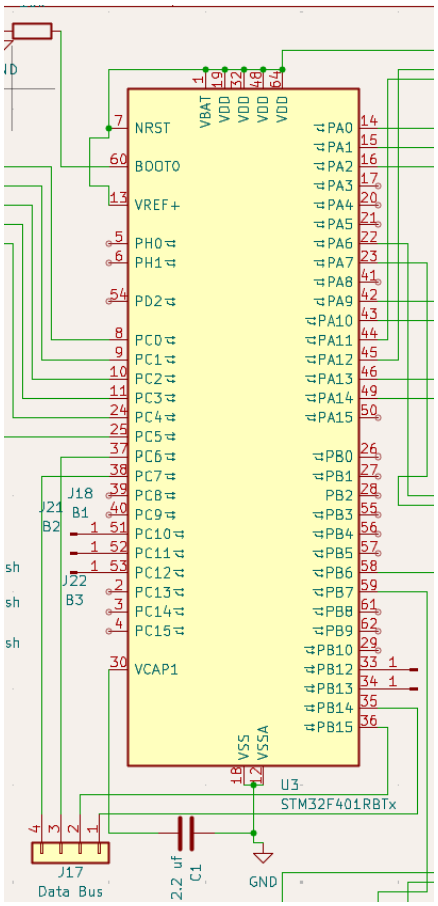


Figure 6: STM32F401RBT6 Microcontroller Schematic

2.3.2 Power Subsystem

For our power subsystem, we opted to use a 12V battery. We did this because typical stepper motor drivers operate at 8-35V, so 12V seemed an obvious choice, as it provided enough voltage for our system while still being a relatively safe power source. Because we also needed 5V and 3.3V in our system, we used an LM2596 buck converter to reduce the input voltage from 12V to 5V. From this 5V output, we used an LM3940IT voltage regulator to bring it down to 3.3V. This gave us all the desired voltage values, allowing us to power all our components correctly.

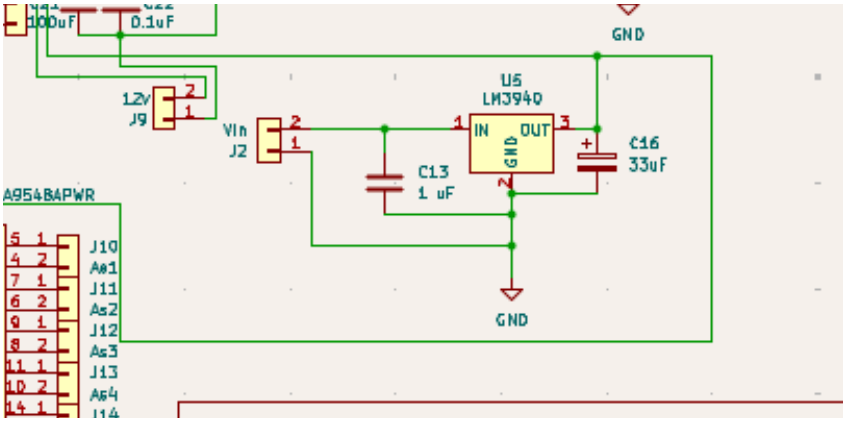


Figure 7: Power Subsystem Schematic (LM2596 Not Pictured, Off-board Component)

The "12V" 1x2 connector represents our 12V input voltage source, and the "Vin" 1x2 connector represents the 5V output voltage of the LM2596 buck converter.

2.3.3 Vibration Subsystem

For our vibration subsystem, we used piezoelectric sensors to detect vibrations from the guitar body. We also implemented a low-pass filter to pass these signals through, which allowed us to set a cutoff frequency to remove potential noise picked up by the sensors. This signal was then fed to our control unit, which processed it and compared the note to a reference frequency to determine whether the note played was sharp or flat. Because we were aiming for a cutoff frequency of 500 Hz, we used the equation below to calculate the resistor and capacitor values for the low-pass filter.

$$f_{cutoff} = 1 / (2*\pi*R*C) \quad (1)$$

Using this equation, we found that the best values for R and C were 3.3kΩ and 100 nF, respectively. We also implemented a 1 MΩ resistor to stabilize the signal and protect against voltage buildup, which would otherwise negatively affect our results. Below is the circuit of the low-pass filter we used:

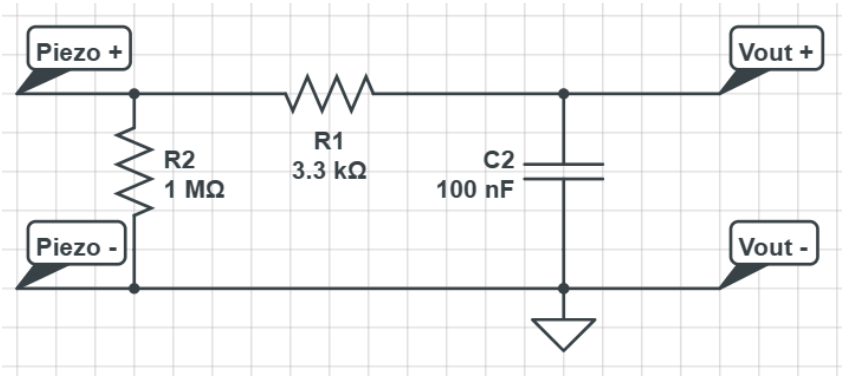


Figure 8: Low-pass Filter

This gave us a cutoff frequency of ≈482 Hz, which was very close to our intended cutoff frequency of 500 Hz. We wanted a cutoff frequency that was around 500 Hz because the high E string yields a frequency of 329 Hz, so to allow for alternate tunings, we wanted to be able to detect frequencies higher than that, so we can either tune down the guitar to an E4 or save a profile that has the high E string higher than an E. Below is our vibration subsystem schematic:

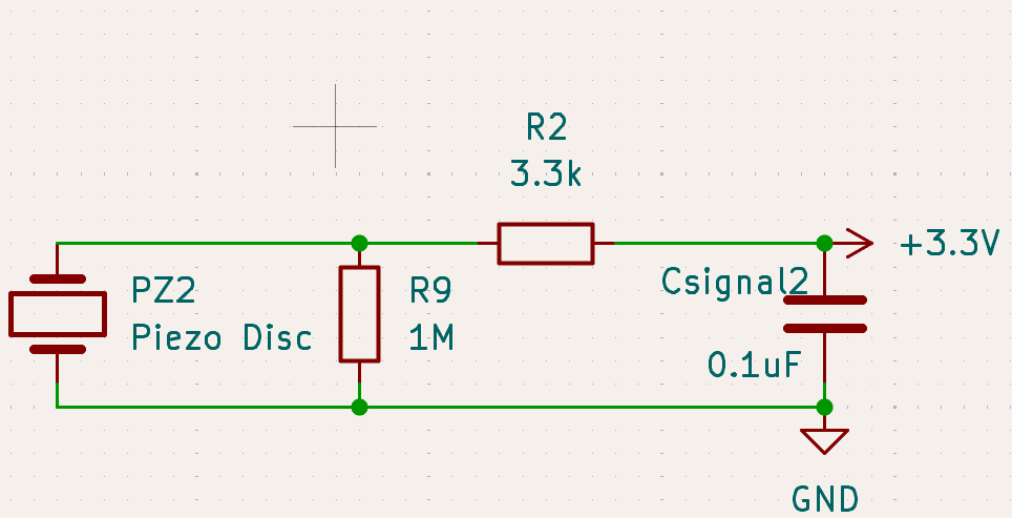


Figure 9: Vibration Subsystem Schematic

2.3.4 Position Subsystem

The purpose of our position subsystem is to record the angles of the guitar's pegs, allowing us to save different tuning profiles and automatically recall them without plucking a string. For this system, we implemented two key components: Six AS5600 magnetic encoders and a TCA9548A I^2C multiplexer. These magnetic encoders would be hooked onto the back of our NEMA 11 stepper motors. The reason they need to be so close is that the magnet attaches to the back of the motor, which spins. For the AS5600 to properly read the stepper motor's angle, it needs to be ~1mm to 3mm away from the magnet; the closer, the better.

The information recorded from the six different AS5600 modules would then be sent to our TCA9548A I^2C mux. Our reasoning for using the mux was simple: Because all AS5600 modules have a fixed I2C address (0x36), we needed a way to separate them into 6 channels. Using our I2C mux, each module sent its SDA (data) and SCL (clock) signals into the mux, allowing us to choose which module we wanted to receive information from. By making each module separate, we eliminated the risk of multiple pieces of information being sent at the same time, preventing overwriting and resulting in a cleaner, more efficient system. Below is our schematic for the position subsystem.

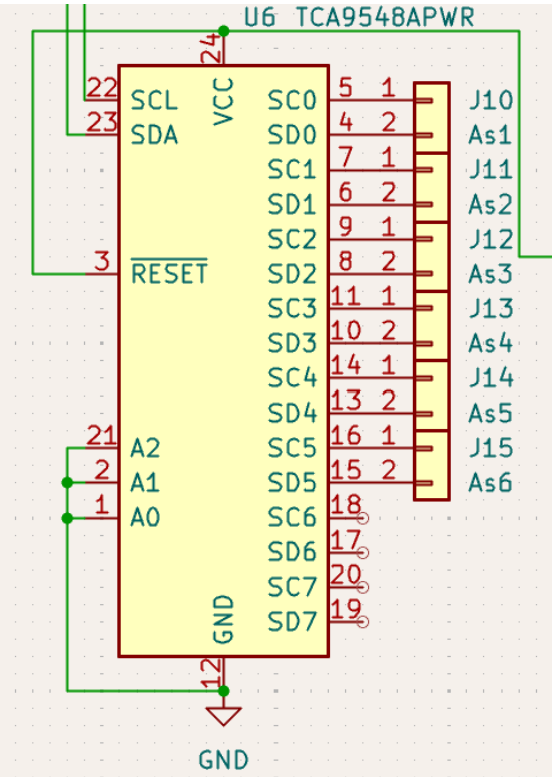


Figure 10: Position Subsystem Schematic (Data Bus On The Right Represents The Six AS5600 Magnetic Encoders)

2.3.5 User Interface Subsystem

The user interface subsystem allows the user to choose how to use the system. For this, we used a Densitron 162-H1-4C LCD display. We opted for this interface because it has 4 on-board buttons, 3 on-board LEDs, and an on-board potentiometer. All of these components gave us a lot of versatility in their use, as each on-board component had a purpose.

Figure 11: Densitron LCD Display With On-Board Components

Because this was a 24-pin display board rather than a typical 12-pin one, we needed many GPIO signals, so our decision to use the STM32F401RBT6 became justified. The four buttons gave us the exact functionality we needed for selecting and saving different tuning profiles. Their functions were up, down, select, and save. The LEDs had their own functions too: the top red LED flashed if the note was sharp, the bottom red LED flashed if the note was flat, and the yellow LED flashed when the note was within 2 Hz of the desired frequency. The potentiometer, like most LCD displays, allowed the user to adjust the screen's contrast. Below is our schematic for this component.

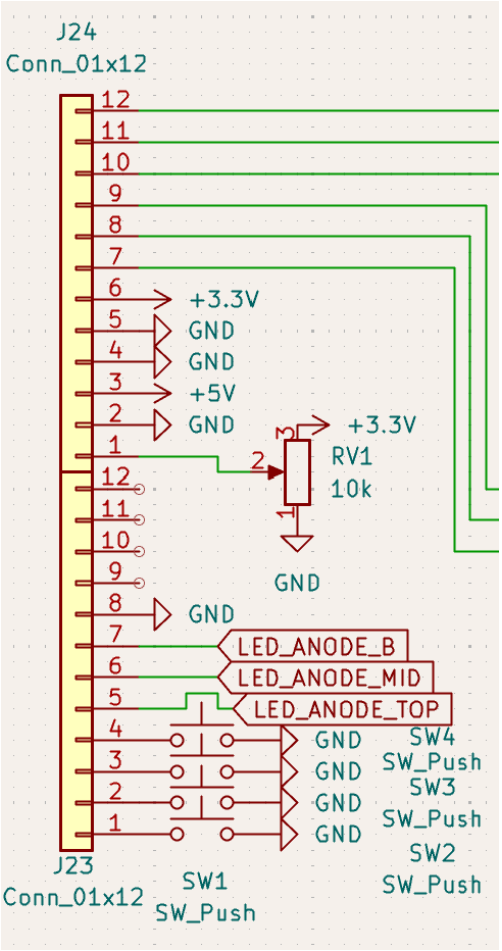


Figure 12: User Interface Subsystem Schematic

2.3.6 Motor Subsystem

Our motor subsystem is responsible for physically turning the guitar's pegs. It consists of the NEMA 11 stepper motor and the TMC2209 stepper motor driver. We chose the NEMA 11 stepper motor because, according to the datasheet, it provided the torque that we needed to efficiently turn the pegs on the guitar.

Constraints	Value
Step Angle	1.8°
Current / Phase	0.67 A
Holding Torque	0.05884 Nm
Operating Voltage	3V - 4V

Table 1: NEMA 11 motor specifications

Because we want 0.9° of rotation instead of 1.8°, we will use half-stepping mode with our stepper motor driver. Although our intended stepper motor drivers were the A4988, because we received incorrect parts, we sourced a single TMC2209 to assist in powering our motor. This allowed us to limit the current being sent to the motor, giving us more freedom to increase the voltage, which in turn increases the torque we receive from the motor. Below is our motor subsystem schematic, showing all 6 stepper motor driver modules to which our motors would be connected.

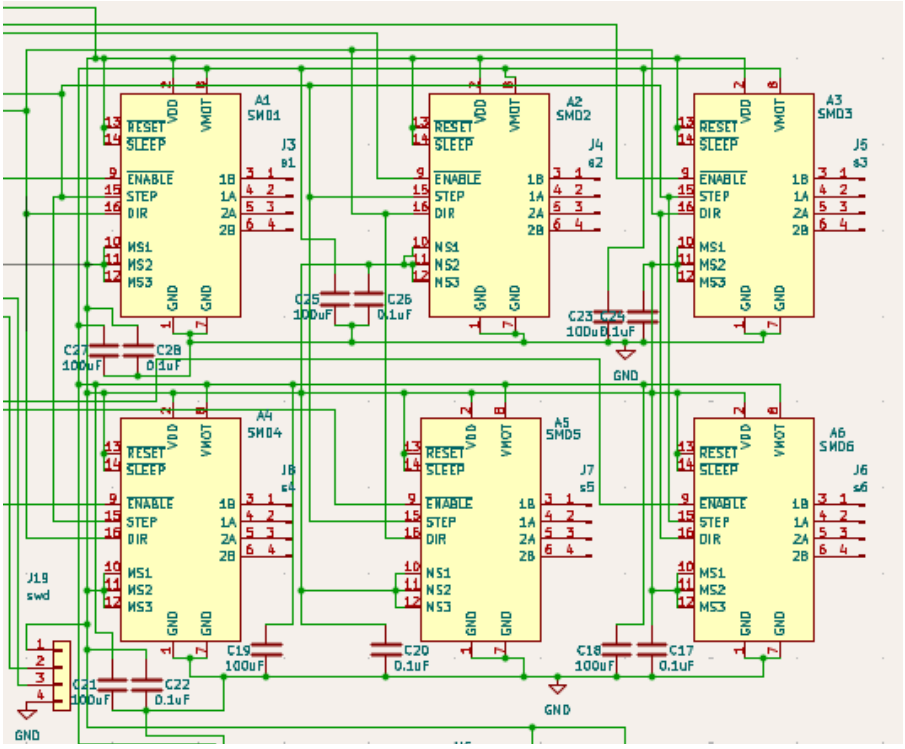


Figure 13: Motor Subsystem Schematic

3. Design Verification

3.1 Power Verification

To ensure proper operation of the LM2596 buck converter, we used a controlled 12V power supply. We wanted to drop this 12V input voltage to 5V, and we did this by turning on the on-board potentiometer on our buck converter.

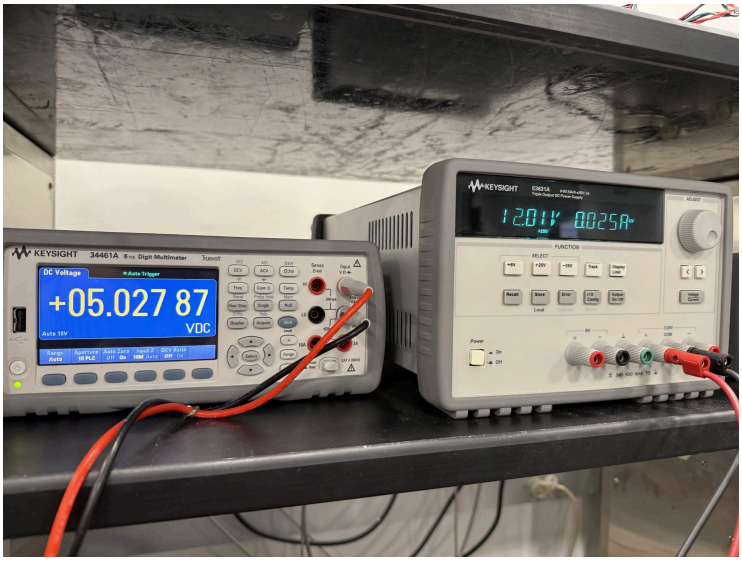


Figure 14: 12.01V Power Supply Input Being Bucked Down To 5V Using A Multimeter

Additionally, we needed 3.3V for logic signals, so we took the 5V output from the buck converter and fed it through our LM3940IT voltage regulator, which provided 3.3V for our microcontroller.

To test the voltage drops across our whole system, we used our multimeter to measure voltage at various points and ensure we still have sufficient voltage. Even if our voltage source is 12V, the power consumption of other components can cause the voltage to drop. Below are our results of said tests.

Action being done	Voltage Value (V)
Raw Voltage	12.01
Raw Voltage w/ LCD on	11.98
Raw Voltage w/ motor on	11.82
Raw Voltage w/ LCD and motor on	11.79

Table 2: Results of voltage drop tests

As we can see, the voltages across the whole system, with all components on, are still around 12V, keeping our design consistent.

3.2 Vibration Verification

We verified that our piezo discs worked by taping two of them onto our guitar, passing the signals through our low-pass filter, and strumming each string. We confirmed that voltage signals were produced for each string. The low-pass filter successfully reduced high-frequency noise, resulting in cleaner signals for processing.

Initially, frequency detection was implemented using a Fast Fourier Transform (FFT) approach. However, while testing across different tuning profiles, the system would produce incorrect frequencies. This behavior was consistent with the FFT locking onto harmonic frequencies instead of the actual one we wanted. This is due to harmonic peaks having higher magnitudes than the true fundamental component.

We decided to switch from an FFT-based approach to an autocorrelation-based approach. Autocorrelation directly estimates the signal's fundamental period rather than relying on peak magnitude. After this modification, the system produced consistent and accurate frequency readings across all tuning profiles.

3.3 Control Verification

For our control-unit verification, we measured the system response latency between string plucking and the corresponding control-action signal. A total of 50 trials were recorded, and the response times were plotted to illustrate the system's timing performance.

The results show that most trials ended with response times below 250 ms. This is good, as our requirement was to reach 250 ms.

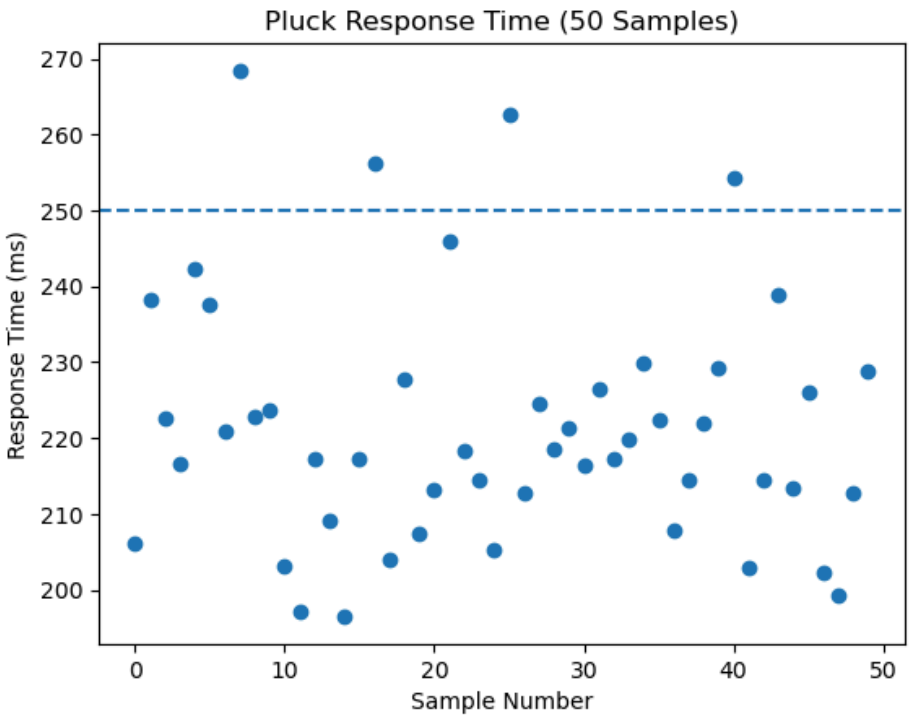


Figure 15: Latency Chart

3.4 Motor Verification

We used a NEMA 11 stepper motor paired with a TMC2209 stepper motor driver. We tested the functionality of our motor by feeding 12V from our voltage source into the stepper motor driver, along with the 4 wires from the stepper motor. We followed the proper instructions to know which coils were connected to each other, shown in Figure 16.

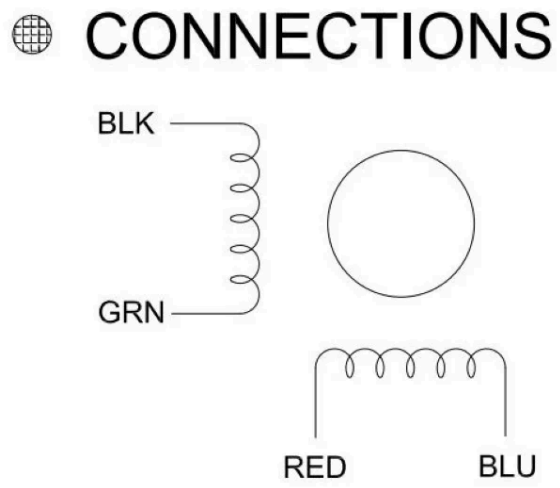


Figure 16: Diagram that shows which color coils are connected to each other

After hooking up our ~EN signal to ground, setting our reset and sleep signals to each other, and feeding 5V into our TMC2209 for the logic signals, we achieved consistent motor movement. After we got our motor to turn effectively, we made changes to our code so that the motor turned clockwise when the note read was too sharp and counter-clockwise when the note read was too flat.

3.5 Position Verification

For our position subsystem, we saved the angle at its normal position, where it would show 0°. We marked this spot and stepped 50 times. After this, we measured the angle and checked whether the code saved the correct angle, which is within 2 degrees of 45 when the step size is 0.9. We did multiple tests like these to verify the position.

3.6 User Interface Verification

We used a Densitron 162-H1-4C as our LCD display with 4 buttons. Verification of this interface focused on display functionality, input responsiveness, and integration with the control system.

To verify this, test messages were printed on the LCD to confirm that the microcontroller and display were connected successfully. The display was also tested under specific conditions (profile selection, tuning initiation, and system restart) to ensure that the correct responses were displayed to the user.

For button input verification, each of the four buttons was tested individually to confirm that the microcontroller detected the correct signal. Buttons were mapped to scrolling through profiles, selecting a profile, and canceling tuning. We verified that each input produced the expected system response. Debouncing behavior was also evaluated to ensure that one press did not register as multiple inputs.

Overall, the user interface reliably displayed system information and accurately responded to user input, providing clear, effective interaction with the device.

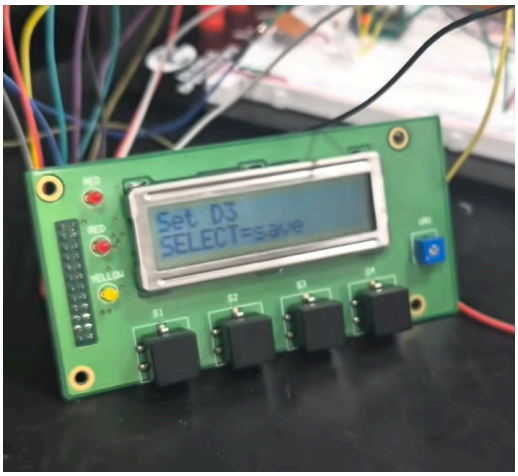


Figure 17: LCD While Saving Profiles

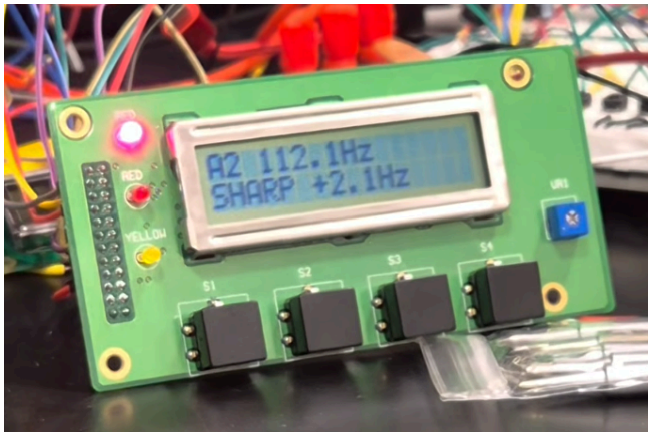


Figure 18: LCD When Out Of Tune Note Is Played

4. Cost and Schedule

4.a Cost Analysis:

See Appendix B for the cost table.

For the Grainger College of Engineering, the average salary is $96,766 per year. Taking into account a 40-hour workweek and 52 weeks worked, the salary is about $ 46.52 per hour.

We are expecting to spend an average of 18 hours per week on this course.

1. Labor:
   $46.52/hour * 16 weeks * 18 hours/week * 3 people = $40,193
2. Parts:
   $174.93 ($24.93 over budget)
3. Sum:
   $40,193 (labor) + $ 174.93 (parts) = $40,367.93

4.b Schedule:

See Appendix C for the schedule table.

5. Conclusion

5.1 Accomplishments

Even though we were not able to make a fully functional auto-tuner, our project was still mostly successful.

Looking back on our project, we succeeded in many areas despite the obstacles we encountered. We were able to get the vibration, control unit, position, power, and user interface completely functional, and most of the motor subsystem. Our project accurately detects frequencies, allowing tuning within 2 Hz and accommodating users.

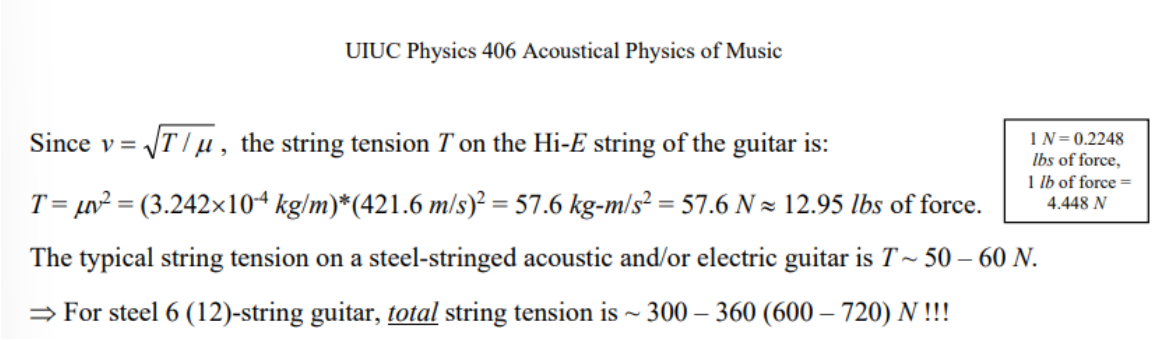
5.2 Uncertainties

When testing the NEMA 11 stepper motor on the tuning peg, it couldn't turn the peg, despite research indicating otherwise. The estimated required torque was calculated using a string tension of approximately 50-60 N multiplied by the effective radius of the tuning peg. However, this has many uncertainties.

First, the guitar's age and condition can change this. The assumed string tension might not be accurate. String tension can vary depending on friction, string gauge, and other things. Additionally, the radius used in the calculations might not be accurate for other guitars.

Second, mechanical forces were not taken into account. Friction within the mechanism, internal resistance in the motor, and inefficiencies between the motor and the peg can increase the torque requirements.

Finally, power delivery limitations could reduce motor performance. If the power supply did not provide sufficient current, the motor might not reach its desired torque.



UIUC Physics 406 Acoustical Physics of Music

Since $v = \sqrt{T/\mu}$, the string tension T on the Hi-E string of the guitar is:

$T = \mu v^2 = (3.242\times10^{-4}\ kg/m)*(421.6\ m/s)^2 = 57.6\ kg\text{-}m/s^2 = 57.6\ N \approx 12.95\ lbs$ of force.

1 N = 0.2248 *lbs* of force, 1 *lb* of force = 4.448 N

The typical string tension on a steel-stringed acoustic and/or electric guitar is $T \sim 50 - 60\ N$.

⇒ For steel 6 (12)-string guitar, *total* string tension is ~ 300 – 360 (600 – 720) N !!!

Figure 19: Physics Lecture Proof

5.3 Ethical considerations

For our handheld automatic guitar tuner, ethical considerations include ensuring the device operates safely and reliably without harming users or damaging the instrument. The system is designed to prevent excessive torque by applying current limits and step-size constraints to the NEMA 11 stepper motors, reducing the risk of snapped strings and injuries. A control threshold will ensure tuning adjustments are incremental rather than abrupt. Additionally, the system will stop the motors if a stall exceeds 2 seconds.

Our project does not involve human or animal research, the collection of personal data, or the handling of sensitive information, so IRB and IACUC approval are not required. Overall, our design will prioritize user safety and responsible engineering practices consistent with IEEE ethical standards [20] and the ACM Code of Ethics and Professional Conduct [21].

5.4 Future Work

For future work, improvements are needed to the PCB design and the motors. The current implementation shows that the selected NEMA 11 stepper motor did not provide sufficient torque to reliably turn the guitar tuning peg. Future iterations should consider higher torque motors or the addition of a gear reduction mechanism to increase effective torque.

The PCB design itself can be refined to improve reliability and signal integrity. We should have used one stepper motor driver. This way, the PCB would not be as clumped up. Also, many things in multiple subsystems were connected to each other. This led to a heavy reliance on individual components of the PCB, and if even one failed, the whole thing would stop working.

References

[19] “Acoustical Physics of Music: Lecture Notes,” Department of Physics, University of Illinois Urbana-Champaign, 2017. Available: https://courses.physics.illinois.edu/phys406/sp2017/Lecture_Notes/P406POM_Lecture_Notes/P406POM_Lect3.pdf

[20] Institute of Electrical and Electronics Engineers, “IEEE Code of Ethics,” IEEE. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[21] Association for Computing Machinery, “ACM Code of Ethics and Professional Conduct,” ACM. [Online]. Available: https://www.acm.org/code-of-ethics

Appendix A: Requirement and Verification Table

A.1 Computation

Requirements	Verification
Frequency detection latency <= 250 ms	Measure how long it takes for autocorrelation to happen by using test code to print latency statements
Button press should be detected within 100 ms	Press button and measure delay to LCD response using print statements
LCD should refresh information every 200 ms	Use print statements that show latency of LCD

Table A1: Control Unit Subsystem Requirements and Verification

A.2 Power

Requirements	Verification
Output voltage 12.0 V ± 0.1V	Measure using multimeter
Supply >= 500 mA	Measure using multimeter
Ensure no significant voltage drops across the system when components are on	Use multimeter at various points of the system while motor and LCD are running

Table A2: Power Supply Subsystem Requirements and Verification

A.3 Vibration

Requirements	Verification
Detect frequencies from 50 Hz - 500 Hz	Inject known sine wave signals through low-pass filter and verify correct detection
Prevent resonant frequency avoidance (other components vibrating)	Ensure components don’t vibrate using rubber feet on breadboard (measure voltage on oscilloscope)
Achieve pitch detection accuracy within ± 2Hz	Tune each string and measure final pitch using reference tuner

Table A3: Vibration Subsystem Requirements and Verification

A.4 Position

Requirements	Verification
Achieve angular measurement accuracy within 2 degrees of actual shaft position	Mark a spot on the motor and step 50 times. Measure that the angle is within 2 degrees of 45^{O} when step size is 0.9^{O}. Repeat 8 times
Allow independent selection of at least 6 encoders using the I^2C multiplexer	Rotate 1 motor and run code to ensure only one input is being read
The subsystem shall measure angular position over a full 360 degree rotation	Rotate the motor shaft through a full revolution and log sensor output values. Verify count range and confirm step size = 0.9^{O}

Table A4: Position Subsystem Requirements and Verification

A.5 User Interface

Requirements	Verification
Display tuning information including detected frequency and target note on a 2×16 character LCD.	Input known sine wave frequencies and verify correct frequency and note are displayed on the LCD and are readable
Allow user input through at least four push buttons to navigate menus and select tuning profiles.	Press each button and verify correct menu navigation and selection
Operate from a 5V supply (LCD) and interface safely with the 3.3V MCU logic.	Measure voltage levels on LCD input pins when driven by MCU

Table A5: User Interface Subsystem Requirements and Verification

A.6 Motors

Requirements	Verification

Must allow functionality for all levels of microstepping for precise tuning	For each level of microstepping, measure how far it rotates using protractor
Stall detection triggers shutdown within 2 s	Test code for stall detection (physically block motor). If the angle doesn't change after 2 seconds of rotation, stop.
It should be able to provide enough torque (0.0245 Nm) to turn the peg	Check NEMA 11 stepper motor datasheet and verify torque capabilities (0.0588 Nm)

Table A6: Motors Subsystem Requirements and Verification

Appendix B: Cost Table

Description	Manufacturer	Quantity	Extended Price (Includes Shipping)	Link
NEMA 11 Stepper Motor	Walmart	6	2*29.22 = 58.44	Link
Peg holders	Us	6	Free	N/A
35mm Piezo Discs	YQBOOM	10	10*0.99=9.99	Link
2x16 LCD Display w/ Controller	N/A	1	Free	E-Shop Self Service
A4988 Stepper Motor Driver	eBay (gy-power)	6	6*0.99 = 5.94	Link
Adjustable Buck Converter	AITRIP	5	7.99	Link
STM32F401RBT6 MCU	N/A	1	Free	E-Shop Order
12V 2400mAh AA NI-MH Battery Pack with Standard Tamiya Connector	N/A	1	0.17	Link
Bottom housing	Us	1	Free	N/A
Lid	Us	1	Free	N/A
Moveable Arm	Us	1	Free	N/A
AS5600 Magnetic Encoder	Walmart	6	6*7.50 = 45	Link
TCA9548 I2C Mux	MusRock	1	7.99	Link
3S LiPo Battery Charger	Tosiicop	1	9.59	Link
LM3940IT 3.3V LDO Voltage Regulator	N/A	1	Free	E-Shop Self Service
MCP23017 IO Expansion Board	Waveshare	1	$12.99	Link

Table B1: Cost Table

Appendix C: Schedule

Week	Task	Person
2/22 - 3/1	Start Schematic for PCB	Everyone
	Complete Design Doc	Everyone
	Sign up for Design Review	Lee
3/1 - 3/8	Design Review with TA and Instructor	Everyone
	Majority of Physical Parts Ordered	Lee
	Finished Schematic and PCB Design and Submitted to TA	Everyone
	Meet with the TA and talked about PCB	Everyone
3/8 - 3/15	Worked on and Presented Breadboard Demo	Everyone
	Fixed issues with PCB and resubmitted in time for Round 3	Nicholas Chan
	Teamwork Evaluation I	Everyone
	Worked on 3d Modeling Key Holders	John
	FFT Code	Lee
3/15 - 3/22	Spring Break	
3/22 - 3/28	Picked up ordered parts	Everyone
	Worked on Modeling Connecting Rods	John

	Successfully Integrated Piezo Discs into previous Design	Everyone
3/28 - 4/4	Individual Progress Reports	Everyone
	Integrate Buck Convertor into updated Design but not working properly	Everyone
	Attempted to Print currently modeled pieces but tolerances too tight so failed	John
	Refined design so Op-Amp is no longer needed for Piezo Discs, instead multiple are used in parallel	Everyone
	Dictionaries for Tuning Preferences	John
4/4 - 4/11	Progress Demo	Everyone
	Team Contract Assessment	Everyone
	Work on Tuning code	Lee
	Work on LCD Profile Code	Nicholas
	Connecting Rod was deemed a failure after multiple failed prints. Deemed it would be best to switch to nuts and bolts as tolerance is more precise	John
	Updated Dictionary File to allow for better integration with tuning code	John
4/11 - 4/18	Worked on Debugging Auto Correlation Code	Lee + Nick
	Start soldering PCB and magnetic encoders	John
	Start modeling motor holders and adjust tolerances after printing	John

	Worked on motor turning code	Nicholas
	Create and start Mock Presentation	Nicholas
4/18 - 4/25	Create Final presentation and start it	Nicholas
	Attempted to Incorporate Stepper Motors into design but unsuccessful	Lee + John
	Model Wall and bottom components of Box and Key Holder Covers. Need to adjust models after openings discovered to be too big	John
	Fixed Auto Correlation Code	Nick + Lee
	Mock Presentation	Everyone
	Mock Demo	Everyone
	Continued Soldering PCB. Attempting to test resulted in no components working. Shifted focus to breadboard based components. Ordered IO-expansion board so motors and LCD can be used at same time.	John
	Get LCD with buttons to work with the design and allowed breadboard to run off battery	Lee
	Updated logic for motors	Nick + Lee
4/25 - 5/2	Work on creating code for saving angles of each string into tuning profile	Nicholas
	Modeled Box top and found files for table mounted arm that could be used in our design. Both printed successfully	John
	Discovered we did not have the actual stepper motor drivers but	Nicholas & Lee

	rather expansion boards. Managed to get a stepper motor driver and coding for them	
	Finish Final Presentation	Everyone
	Final Demo with Instructor and TA	Everyone
	Recorded Final Video and did voice over	Everyone
5/2 - 5/6	Final Presentation with Instructor and TA	Everyone
	Final Report	Everyone
	Lab Notebook turned in during lab checkout	Everyone

Table C1: Week by Week Schedule