ME 366J Final Report

Yifeng Liao, Peter Mathews, Talal Al-Otaibi, Elliott Turner, and Andrew Zhang
Section 19458
Professor Seepersad
TA: Elena Soto Handal
December 5, 2022

Table of Contents

Project Proposal

I. Introduction

In the first phase of our design project, we gathered background information on the existing state of the art technology in the automated instruments sector. Then, to identify the needs of customers who would provide insight on this technology, we interviewed music students, educators, professors, outreach coordinators, and K-12 students. Analyzing these interview transcripts–using a customer needs analysis and a House of Quality–allowed us to determine the exact design parameters we would need to optimize to create a purposeful, impactful automated instrument.

II. Background Research

Studying the state of the art technology in automated instruments revealed a couple of pertinent trends: brass and woodwind instruments would be the least viable. Studying brass instruments further, it would require significant mechanical input in the form of vibrating a mouthpiece in a very specific manner. In a trumpet, for example, sound is produced by "starting a standing wave vibration [with your lips]" (*Trumpet* 2022) where your lips may need to "vibrate at more than 1000 vibrations per second" (*Brass instrument (lip reed) acoustics: an introduction*) for higher notes. There is considerable difficulty in producing waves and vibrational patterns of this nature. Not to mention the elasticity of a material that would be needed to produce thousands of vibrations a second in a precise formation, that mimics human lips, would be difficult to achieve. Further research into woodwind instruments revealed similar engineering challenges. Not only would a mechanism be needed to control which specific notes are played on a flute, for

example, but a steady stream of air would be needed to even produce sound at all. In general, woodwind and brass instruments are also extremely expensive. The average cost "of [a] trombone, trumpet, clarinet, and flute is $200-$1000" (*Meridian Brass Instruments Boise for sale*). These prices are clearly exorbitant and would not be particularly cost-effective for the budget of only $250 for the project. Therefore, by method of elimination from background research, it seems utilizing a percussion or string instrument would be the most viable solution for an automated instrument.

In investigating further, we realized the primary components we would need for an automated instrument regardless of whether it uses a percussion or string instrument. First, we would need a microcontroller for our product to function effectively. A microcontroller would act as the basic computer for the system, allowing (1) potential sensor data to be inputted that would interact with the environment, (2) outputting data to display particular states of the system, (3) interact with a laptop/computer that can transfer code and data to the system, and (4) be integrated with motor drivers that can directly control motors needed for the system. Typical examples of microcontrollers include Arduinos and Raspberry Pis. However, Arduinos are limited as they are "ultimately suited for simple hardware-based tasks," but are cheaper than many other microcontrollers on the market (Harper, 2022). The Raspberry Pi, in contrast, is "more powerful, hardware-wise…" with the downside being that it "requires more [financial] investment alongside the initial kit to get it up and running" such as a "SD card, peripherals, [and] even a screen" (Harper, 2022). Regardless of the microcontroller, it will be integrated with a computer, which will perform the processing for it and use its memory as well. In addition, it will be integrated with sensors and motor drivers.

Another integral component to the operation of an automated instrument would be motors. Typical motors used for an application of this nature are stepper motors and servos. A stepper motor is a “brushless DC motor that rotates in steps” (Dejan, 2022). This is particularly useful as it can be precisely positioned to a specific angle without any need for a sensor. A servo, however, is a “closed loop system that continuously monitors [its] position” (*Tutorial: Stepper Vs Servo*, n.d.). In general, servos offer higher torques at higher speeds and are well-suited to variable load systems. In contrast, stepper motors cost less and require no feedback, making them simpler to control. Its main disadvantage is torque drastically decreases as its speed increases. The downside of servos lie in their cost and their level of complex control. Regardless of the motor, they will be integrated with the rest of the system using motor drivers–where it will be powered and controlled from.

III. Gantt Chart and Task List

We use the Gantt chart and the task list to divide the entire project into small sections and indicate the range of time each section needs to be completed to make sure that we stay on track. The Gantt chart and task list we have will go through changes over time. Currently these two items for phase A, the project proposal part, are finalized; the next two phases will require updates in the future. The current Gantt chart and task list are in Appendix A1-3 and Appendix B, respectively.

IV. Customer Needs Analysis

To obtain more information and gain inspiration for our automated musical instrument, we conducted interviews with 10 people who gave their advice and expectations for an instrument. These ten interviewees include three college students that play musical instruments, a

K-12 educator, four 9th - 12th grade students, an outreach coordinator, and an acoustics expert. The transcripts of the interviews are in Appendix C1 - C4.

Since none of our team members are very skillful in any musical instrument, we interviewed college musicians to learn more about their instruments and hear their opinions about automated musical instruments. From our interviews with them, we found out that the hardest parts about learning musical instruments are reading the music notes and using the correct hand/fingering form. In addition, they emphasized that sound quality is their primary focus when choosing which brand of instrument to buy, and they think an automated instrument needs to be cheap and easy to transport.

Our initial assumption of the function of an automated instrument is that it could help K-12 educators to teach students about engineering and help them focus in class. So we interviewed a K-12 educator and four 9th - 12th grade students to hear what they expect from our future product. The K-12 educator told us that engineering students often struggle with programming, and he hopes that our automated instrument would be intuitive enough to operate so that students can grasp the programming and mechanical knowledge from the product. The responses from 9th - 12th grade students suggest that our instruments should have good sound quality, play a wide range of songs, as well as be loud and durable. Although the students have different opinions about the size of the instrument, we conclude that the final product must be compact and light.

Our interview with an outreach coordinator was to further realize what attributes our instrument needs for an outreach event or a public demonstration. He told us that in order for an outreach with an automated instrument to be engaging, we need to teach the audience how it

works, and playing funny songs would also help. He also told us that he had the worst outreach when there was a robot malfunction, so our automated instrument must be reliable. Finally, he informed us that not only our instrument needs to be compact, but it should take minimum time to set up.

Our last interviewee was an acoustic expert, since we wanted to obtain more technical information and advice about automated instruments. He suggested that the xylophone family of instruments would be the easiest to automate, and instruments that require pneumatics would be the hardest. He also told us that whatever musical instrument we finally choose, we need to find a music note software and learn how to work with midi files. When we asked him about his expectations for an automated instrument, he said that he would like to see it play a diverse range of notes, and the instrument should be able to play higher temp songs, as well as being audible. Finally, he told us that it would be very hard to mimic exactly how the song was played because robotic actuators cannot achieve the motion fluidity of human input, and playing a single note at one time is much more achievable than playing the chord.

Based on our interviews, we interpreted their needs and created a customer needs list with ratings for relative importance in Appendix D. We decided that our design must be cheap, durable, audible, easy to transport, have good sound quality, and be able to play multiple genres of songs.

V. House of Quality

To further build on the customers' needs outlined in the previous section, we constructed a list of customer needs and assigned weights to each of them. This was carried out based on the number of times or frequency at which the need was implied or mentioned. Moreover, care was

taken to consider which need either defines one aspect of the product or if it causally drives everything else as another factor to weighting the needs. A good example of such weighted needs includes the need to fit the instrument on the desk which affects the very selection of the instrument and all other physical attributes. Another example is the customer need to guarantee sufficient musical range was mentioned in an interview and some survey responses but isn't necessarily a core fundamental of the idea behind a proposed solution later. Subsequently, a scale from 1-5 with increasing value denoting higher weight was selected. Therefore, in the previous examples, the need to have the instrument fit a desk was weighed at 5 and musical range at 4.

Overall, 12 customer needs were compiled and weighed, and a house of quality began to be formulated for a more rigorous analysis. After viewing the customer needs, criteria were derived to develop quantitative and qualitative metrics to measure the extent to which one or several customer needs were satisfied. Emphasis was placed on creating metrics that are repeatable in measurement and accessible for the purpose of evaluating and verifying the viability of chosen solution. Quantitative metrics include both physical properties such as the bounding dimension in ft, a useful value mentioned by Dr. Seepersad in lecture, as well as acoustic properties such as bandwidth, the range from the lowest detectable pitch to the highest one measured using frequency in Hz via a microphone. A qualitative measure includes the number of successfully played songs, checked by confirming that a simple majority responds of the audience responds yes to a yes/no survey. All these metrics are assigned up or down arrows to specify the direction of improvement. Referring to the bandwidth metric, a higher bandwidth allows more notes and thus a wider of option of songs to played which helps satisfy customer needs in terms of product performance.

From there, one can see how the metrics can interact with customer needs. The house of quality metrics then mapped to each customer need for the purposes of seeing if each metric positively or negatively correlates with each customer need. This was done by marking an ‘X’ for a negative correlation and ‘O’ for positive (e.g customer need for low cost and bandwidth being negatively correlated as ensuring a higher range of musical notes needs a more expensive system). By seeing which metric may satisfy the fulfillment of a customer need or contradict it, an intuition and further understanding of the metrics is developed.

Finally, target metrics were set based on the understanding of the customer needs, project requirements set by Dr. Seepersad as well as some background research into robotics and acoustic theory. These target metrics aim to encompass multiple aspects to ensure that customer needs are incorporated into future processes on a holistic level. Without a frame of reference created by comparing to different existing products, it is difficult to pinpoint an exact niche or archetype that the product falls into, but the idea is to create a compact and portable educational outreach tool whilst also being versatile enough to play a broad range of songs in terms of tempo and pitch, i.e a lightweight-jack-of-all trades.

VI. Engineering Requirements and Specifications

After analyzing the background research and processing the results of the customer needs analysis, a set of engineering requirements and specifications were produced. These can be seen in appendix F.

The process started by going through the customer needs list and background research to identify desirable attributes. With the help of the house of quality, these were correlated with a

list of specific characteristics. Requirements were then defined for each characteristic to ensure that the final product would meet the customer needs and incorporate successful attributes identified in our background research. Notably, these requirements are defined quantitatively and can therefore be evaluated in a non-subjective manner. Specifically, a method of verification was selected for each requirement that can be used while designing the product and then testing prototypes. These mostly include analysis of the design in CAD and experiments conducted with a prototype.

Each requirement is classified as either a wish (W) or demand (D). Wishes are characteristics that the user would like to have, but would still be interested in the product if it doesn't meet the requirement. On the other hand, demands must be met in order to satisfy the customer. In our case, the only requirements classified as wishes were the bandwidth and maximum rate at which notes can be played. These were deemed to be a wish since failing to meet the requirement would still allow the product to perform as advertised, but may not be able to play as “well” as customers might hope.

In theory, once the design is completed, tested, and determined to have met all requirements, it should exhibit the desired characteristics identified during our customer needs analysis and background research.

VII. Problem Statement

The end result of this product is an automated musical instrument. Specifically, it should be able to serve as a STEM educational tool in a K-12 environment. With this in mind, we interviewed educators as well as K-12 students to gain further insight into what features and

capabilities would make our product well suited for this environment. As with most products, we were told that there was an expectation that it be durable, reliable, and intuitive to use. Beyond this, we learned that one of the most important STEM concepts that students tend to struggle with the most is programming. With that said, our goal is to produce an automated musical instrument that can survive prolonged use in the classroom and can be intuitively programmed by someone with little to know programming experience.

Design Review

I. Introduction

At the end of our project proposal, we specified that our automated musical instrument will serve as a STEM educational tool in a K-12 environment, and it should survive prolonged use in the classroom and can be intuitively programmed by someone with little to none programming experience. In this phase of the project, we first updated the Gantt Chart (Appendix A) and assigned Tasks Lists to each teammate (Appendix B), then we created a blackbox diagram and a function tree for potential concepts. Based on the function tree, we generated potential solutions to sub-problems using mind-mapping, a TRIZ table, and prior art research. The functions and solutions are populated in a morphological matrix, which we used to generate 5 different concepts with sketches. Finally, we performed Pugh Chart analysis on 3 leading concepts before we made a low-resolution prototype for one of the concepts.

II. Functional Modeling

We began our functional modeling by making a black box diagram (Appendix G). The black box diagram defines what our automated instrument needs to do for the customer as well

as what inputs and outputs are necessary. We defined the primary function of the product as "[generating] sound (tune/rhythm)" and focused on keeping inputs and outputs as generalized as possible, since the purpose of the black box is defining the function of the product and not how the function is achieved. We chose electrical energy for energy input with sound and heat as energy output, a hand for both material input and output, and control data for information input with sound frequency as information output. Our choice to use exclusively electrical energy as input does limit our solution, but other options were ruled out for safety and complexity reasons. Chemical energy would be dangerous or require extensive additional measures to prevent user contact and mechanical energy storage would increase complexity due to additional moving parts. We determined that the user would interact with the product with their hand to activate the instrument, and that control data being converted to sound frequency is fundamental in the definition of "automated musical instrument."

In creating our function tree (Appendix G), the focus and main function was to generate sound. This process would involve many different subfunctions, including tracking the energy transfer across the system. First, electrical energy had to be imported into the system, then that electrical energy must be converted into useful mechanical energy, and lastly that mechanical energy would be converted into sound energy. Another sub function vital to the main function of generating sound is importing data. Although this does not involve energy transfer, importing/transferring signals to electrical components would be needed to play an automated instrument. When considering the user experience and customer needs, other auxiliary subfunctions become important to include. For example, ensuring sufficient volume so that the entire classroom can hear the instrument is important and a need communicated by customers. Also, ensuring the setup by the user is simple and time-efficient is important as customers have

expressed a lack of time/skill needed should be taken into account. Lastly, the instrument being secure and preventing harm to anyone is important, although perhaps an implied customer need.

III. Concept Generation for Subproblems

We used a TRIZ table (Appendix H) and made a mindmap (Appendix I) to generate concepts for subproblem solutions.

Our approach to using TRIZ (theory of inventive problem solving) was to identify conflicting engineering requirements and assign corresponding given TRIZ parameters to each of our requirements. The TRIZ concept provides a list of generalized product parameters, a list of design principles, and a matrix with given design principles to solve conflicts between product parameters. For example, we found that the requirements “volume should be lower than 5 ft^3” and “the sound volume should be larger than 50 dB” would likely conflict since power (and consequently sound pressure level) scales with volume. We determined that the most relevant TRIZ parameters were “volume of stationary object - 8” and “power - 21” with power deteriorating as volume improved. The suggested general solution principles were “using flexible membranes and fine membranes - 30” and “principle of universality - 6” which we translated into particular solutions. We decided that for a flexible membrane, we could use a diaphragm to amplify sound. For the principle of universality, a single powerful actuator moved across multiple notes could be used in place of one less powerful actuator per note. We could implement a large solenoid on a belt-driven gantry instead of several smaller solenoids. Our second conflict was between the 140 bpm requirement and product safety which were determined to be “velocity - 9” and “harmful actions generated by the design object - 31.” leading to general solutions involving removal, color, and rushing through. The particular solutions were removal of fast moving parts (i.e. no belt gantry, contradicting the previous

solution), use of brightly colored parts and/or reflective tape to warn of danger, and slow-moving picking motion (RC servos) with faster motion (solenoids) for string fretting. Our third conflict was between setup time and reliability. The only usable general solution was the “principle of preliminary action - 10” which we translated to having the product mostly or fully assembled at all times. Our fourth conflict was between weight and power, where TRIZ suggested the principles of dynamism, periodic action, mechanical vibrations, and turning harm into good. Our particular solutions were an unfolding cone for sound amplification, storing energy in spring with a cam before hitting the xylophone, taking advantage of resonance, and using resonance for sound amplification. Our fifth conflict was between setup time and automation level. The applicable general solutions (principle of segmentation and equipotentiality) were translated into use of modular solenoids with 1-step connection (unified electrical and mechanical connector) and short stroke solenoids for a smaller package and less potential to overcome.

Our approach for the mindmap was to start ideation and concept generation without judgment. Regardless of the idea, it would be written down as part of the mind map. The mind map started at its focus of autonomously making music. In order to complete this process, it first had to import data to gather what song to play. Depending on the song, the instrument would need to import user input and then play the desired song. The method of data transfer would involve either USB, ethernet, bluetooth, wifi, USB drive, etc. All inputted data can then be transferred to a microcontroller, for example. Thus, leading to the second important category: controlling the instrument. It can be controlled digitally through a microcontroller such as an Arduino or a Raspberry Pi. Other methods include analog control such as using resistors to change voltage and move a mechanism electrically. Lastly, a method could be to use a programming wheel that would mechanically control the instrument, such as a music box that

uses levers. Another important category we decided would be to import electrical energy and power the entire system. Options include either DC or AC power and a wireless phone charging pad. Granted, if a large DC/AC power supply was used it would be difficult to meet the customer need of having a compact design. For the process of mindmapping, however, no ideas are to be discounted initially due to their potential viability issues. The last category encompassed converting electrical energy to sound energy. This could be accomplished in many different ways: plucking a string, hitting a wooden block, blowing into a flute, or even rubbing a stringed instrument.

IV. Prior Art

To further generate solutions to our sub-problems, we did research on prior art for converting electrical energy to mechanical energy. The research focused on technologies that create step movement and continuous movement. For step movement, we found that the stepper motor is a great choice for converting electrical energy into step rotational mechanical energy. A stepper motor adjusts the rotational position by sending current to different pairs of solenoids creating a magnetic field, and the magnets will be attracted to the corresponding positions of the solenoids due to magnetic force, thus rotating the shaft. A drawing of the working principle of stepper motors is in Appendix J Figure J1 For continuous movement, we discovered that brushed permanent magnet DC (PMDC) motors (Appendix J Figure J2) and brushless PMDC motors (Appendix J Figure J3) are both great choices. The comparison between the brushed PMDC motors and the brushless PMDC motors are given in Appendix J Table J4; generally speaking, a brushless PMDC motor has better performance than its brushed counterparts, but it is more expensive. As a result, a brushed PMDC motor would be preferred if multiples are needed.

V. Morphological Matrix and Design Concepts

After undergoing the process of functional modeling and idea generation of subsystems, we used this data to form a morphological matrix(Appendix K). Each subsystem/sub function had a particular solution generated for it and this matrix collected all of the solutions together in an organized manner. The list of sub-functions were to: import data, import electrical energy, convert electrical energy to sound energy, convert electrical energy to rotational energy, control instrument, to secure the instrument, ensure setup time is minimal, high velocity strumming, and to ensure sufficient volume. Ideas to meet each of these sub functions were from using a mindmap, a TRIZ table, and prior state of the art technology .

After forming this matrix, each team member chose a specific concept to model by mixing and matching solutions from each section of the subsystem category.

Concept #1: Automated Harmonica

The first idea from the matrix was to include a blowing mechanism to play a harmonica using any motor and a fan(Appendix K). Both the morph matrix design choices and design sketch are in Appendix K. In order to import data, a network ethernet cable would be used. This would leave data transfer a simple process without a need for excessive worrying about signal/transmission loss over a wireless transmission method of data transfer. In order to import electrical energy, an AC power supply would be used. The only potential drawback of this would be the lack of portability and lack of compact design, which are important to customer needs.

To control the instrument, a microcontroller would be used. This has the ability to properly control the blowing mechanism, specifically when to blow certain notes using a fan and control volume. A microcontroller also allows the compact design need to be fulfilled. Due to

there being multiple openings on a harmonica that play different notes, there would need to be a significant number of motors controlled simultaneously, 10 to be precise. This would add increased complexity in programming, significant weight, and take up a significant amount of space. Therefore, if this design would be pursued, a maximum of 2-3 openings would be used for the project. To secure the instrument to a table, clamps(C-clamps, for example) would be placed along each side of the harmonica. This allows the instrument to remain stationary during its entire operation. To amplify the sound for the whole class to sufficiently hear, a diaphragm is used in this design. The only drawback is a loss in sound quality, so we would have to ensure that the quality loss isn't significant when being transmitted through the diaphragm.

Concept #2: Solenoid Xylophone

Our solenoid xylophone concept (Appendix K3) is intended to be simple and high-performance. We prioritized a minimal unique part count and reduced moving mass, which led to our choice to use serial communication over USB, DC power, and short stroke solenoids. Serial communication is native to most microcontrollers with USB (so additional communications boards are not required), and DC power eliminates the need for several AC-to-DC converters. We chose to use one solenoid per note instead of using an actuator on a gantry, so moving mass is limited to the solenoid rods instead of the entire actuator and motion components. One solenoid per note also allows for simultaneous note actuation with zero travel delay between different notes. Minimizing solenoid stroke reduces the risk of a pinch injury and reduces the energy required to overcome spring potential. We decided to use the weight of the instrument for securing (instead of dedicated clamps or glue) to its environment, and chose to have the instrument fully assembled at all times.

Our specific implementation of the solenoid xylophone would likely have a microcontroller driving MOSFET gates to feed power from the DC power supply to each solenoid. Such a setup is scalable up to the number of outputs provided by the microcontroller (or to maximum digital bandwidth and shift register limitations).

Concept #3: Stepper motor Xylophone

A morph matrix selection and a sketch of concept #3 are presented in Appendix K. Our concept #3 is similar to the #2 since they both draw inspiration from Xylophones. Compared to concept #2, concept #3 aims to be more cost saving and easier to produce. This concept uses a USB drive to import data, which would save time on uploading the data into the microcontroller. Instead of hitting the metal plates, this concept chose wood blocks because they are cheaper and easy to make. Several stepper motors with extension arms would be used to hit the wood blocks; stepper motors and controllers will both be powered by DC power supply. The extension arms on the stepper motor need to be painted in bright color to prevent accidental harm to people. Since stepper motors and wood are very light, this design requires a metal base to secure the instrument on the table. Last but not least, the current passing through the stepper motor will need to be large enough to create sufficient volume of sound.

Concept #4: DC Motor Controlled Violin Bow

Appendix K displays the morph matrix selection needed to carry out a sketch of concept #4. Generally, concept 4 focuses on creating an easy to setup system via the use of unfoldable elements and utilizes a readily available AC power source as an example. Concept #4 more closely resembles a violin due to its mechanical energy to sound energy transformation process: rubbing against a string as a perturbation process. As a result, it is somewhat similar to concept #1, but allows for higher frequencies and likely a higher bandwidth . A more software oriented

approach using a bluetooth module is used to input data, bypassing the need for a USB, and allows for direct connection to a power socket by running an AC power source through a bridge rectifier to convert to DC. Moreover in terms of electronics, a microcontroller running PID code and a PWM are used to modulate and control a brushless dc motor, which allows for rapid and highly resolved motion to unlock a wider range of notes. Mechanically, the dc motor moves a lead screw with a servo hung under it. This servo lowers to swing and allows the violin-like bellow to rub against a tensioned string anchored between two points. The overhead platform holding the servo is cushioned to minimize the harm caused by pinch points and is attached to a foldable bracket that can be locked and unlocked with a thumb screw. Finally, to secure components, all components would be seated on a weighted enclosure like an acrylic board. The drawback is the extensive tuning for PID and electrical-mechanical-sound energy conversions to play the right notes quickly, as well as the software-heavy approach of a bluetooth module, and the strong need for complex power and cable management for safety and ease of use.

Concept #5: Roller-Plucker single string system

The fifth concept (Appendix K), uses a plucking and rolling mechanism to manipulate a single string. The roller moves up and down the length of the string, adjusting its effective length. The roller is moved by a belt driven by a brushed DC motor with an encoder. The plucker plucks the string causing it to vibrate and produce sound. It is directly driven by another brushed DC motor with an encoder. Both motors are driven by H-bridge motor drivers via a PWM signal from a Pi Pico (which is also monitoring the encoders). The Pi Pico also connects to a computer via USB and accepts commands via serial port to control the instrument.

VI. Pugh Chart

Upon generating concepts using the morph matrix, it was important to compare the efficacy of each solution candidate in accordance with an established set of criteria. In order to rigorously evaluate these solution options, these criteria were derived from the engineering requirements such that attributes to be compared were quantifiable and easy to approximate without resorting to highly rigorous processes. Therefore, the selected criteria were mass, size, bandwidth, cost, and educational value. Appendix M shows the back-of-the-envelope calculations made and necessary tabulations. It is important to note that educational value was best quantized by surveying 15 high school students to reach each of the 5 concepts from best to worst using digits from 1-5.

From there, the 5 concepts were placed into a total of 3 Pugh charts, setting a datum to reference a different concept each time.

The automated harmonica was selected as the first datum due to it being the only wind instrument, and so significant contrast exists between the harmonica and the remaining concepts. In addition, the team came to the consensus that it is difficult to use pneumatics and/or wind for song production based on an intuitive guess. It is also important to note that the harmonica has the lowest bandwidth of the 5 concepts and the least educational value while also occupying a fairly large volume.

The first Pugh chart reveals that the solenoid xylophone and Roller-Plucker (concepts 2 and 5) with respect to the harmonica datum were the highest scorers (albeit for different reasons). From there, the solenoid xylophone was set as a datum. This leading concept was selected because of the fact that it was the lowest cost solution to build and also for its ease of assembly and setup, whilst also assuring the highest bandwidth by utilizing the high frequencies of the bar.

However, it is the heaviest option as shown in the second pugh chart given that all options are much lighter, despite it being below the engineering requirements' stated 11 lbs. In addition, it is rather large volumetrically and is not the aggregate top choice in terms of educational value.

The final Pugh chart places the roller-plocker as a datum and all other concepts fall behind, meaning that by total criteria points, all other designs scored lower, and thus the Roller-Plucker is holistically the best option. While not the best in terms of bandwidth performance, it is the top scorer of educational value due to its simplistic but unique engineering approach to sound propagation, and is also the smallest volumetrically and the lightest of the 5 concepts. With all these strengths in mind, it seems to be the best option to proceed with. From a perspective beyond the basic pugh chart, the back-of-the-envelope calculations show that the bandwidth of the roller-plucker is heavily dependent on the length of string used, while tension itself affects whether this bandwidth lies in higher or lower frequencies.

In terms of other concepts, concept 4, the dc motor controlled violin bow, qualitatively bears resemblance to concept 5 yet it has higher bandwidth. However, it is also far more complicated in terms of electronics and is the priciest solution by far with the rough-estimate price being $244.

Finally concept 3 bears resemblance to concept 2 yet is slightly lighter. However, the wooden blocks and servos do require it to take up a larger volume and require more in costs at Low-resolution Prototype at $ 150, nearly double that of concept 2 for the same bandwidth.

It is for these reasons that the roller-plucker seems to be the best option for further examination via a low resolution prototype.

VII. Low-Resolution Prototype.

We selected our roller-plucker (concept 5) to develop a low-resolution prototype of (appendix N). The mechanical portion of the prototype was 3D printed and assembled using bearings and M3 hardware. A “knob” was added to the “plucker” to allow us to manually actuate the plucker and the “roller” was also designed to freely move up and down the length of the string. Fishing line was used to simulate a guitar string. Furthermore, cardboard was used to quickly prototype the general size and shape of the stand, including where the control board and motors would be located.

Once assembled, we were able to manually manipulate both the roller and plucker to get a better idea of how the two moving parts of the design actually function and identify challenges with getting the design to perform as desired. The main take-aways from this was that the structure needs to be extremely stiff to withstand the tension of the string, the plucker should be more compliant, and the string should extend further in both directions from the plucker in order to be more compliant at the location it is plucked. Furthermore, the string itself should be longer to support a larger bandwidth and may require electrical amplification in order to produce sound at a comfortable listening volume.

VIII. Further Development

Based on our Pugh chart and low-resolution prototype, it seems the most viable solution moving forward is the roller plucker design(Concept 5). It appears it will be relatively simple to fabricate and assemble, with most parts being 3D printed or purchased off-the-shelf. Initially, it seemed that it would be difficult to buy a guitar string and tension it ourselves without professional equipment, however, through more research and experimentation in the

low-resolution prototype, it seemed achievable. In the future to further verify this leading concept, we will need to develop our code for this design to ensure we can convert audio mp3 files into MIDI files. If significant obstacles are presenting themselves with this, we may have to get MIDI files directly from online sources instead of trying to convert them ourselves. In the low-res prototype, we also noticed quality issues with the surface that the string contacts as it is being pressed down. In order for music to be played reliably from this instrument, we will need to ensure the 3D print is reoriented in the print bed, so that no rough or bumpy surfaces remain. If this issue persists, we can use sandpaper to further smooth it down. In addition, ensuring high enough final bandwidth of the system and well setup for song selection is highly reliant on adequate tensioning of the string and ensuring that the string is long enough to vary the frequency. As such, it is key to understand the trade off between string length and compactness. By further experimenting and creating a mathematical model with other parameters like mass per unit length (through material selection and string diameter), the necessary optimizations can be made to unlock the musical potential of the roller-plucker whilst also being well within the engineering requirements.

Final Report

I. Introduction

From our design review, we discovered that the most viable leading concept was the roller plucker design. Further development from the design review that will need to be addressed in this chapter includes ensuring we have the ability to input MIDI files into the control software and reliably have the roller adjust to the correct pitch when outputting sound. In this chapter, we will discuss the function of the leading concept in detail with corresponding 3D CAD models, perform FMEA in order to eliminate pertinent areas of potential failure in our design, especially high-risk areas, perform experimentation to optimize the roller acceleration performance and temperature of the stepper motor, and optimize rail assembly parameters using simulation to rectify high areas of stress from string tension. Then, we will ensure ease of design for manufacturing and assembly for the customer, including operating and repair instructions for the instrument user, analyze the sustainability of the design, and provide an updated budget, bill of materials, and final drawings for the leading concept. Lastly, we will have final discussions on how our instrument met the needs of our various potential customers and any recommendations to improve our design in the future.

II. Leading Concept

Our selected leading concept is the concept #5 from the design review: the roller-plucker single string system. The 3D CAD model of the concept with labels and dimensions is shown in Appendix O. This concept uses a plucking and rolling mechanism to manipulate a single string. The roller moves up and down the length of the string, adjusting its effective length, thus changing the notes and frequencies. The roller is moved by a stepper motor with an encoder; there are two pulleys at each side connected by a belt to achieve this range of motion. The

plucker, which is connected to a servo motor, plucks the string, causing it to vibrate and produce sound. In support of these functions and customer needs, a tuning peg is used to adjust the tension of the strings, and an electric pickup is used to amplify the sound.

The roller-plucker single string system was selected as the leading concept by comparing it with other concepts in accordance with the engineering specifications. The sketches of other concepts are shown in Appendix K. Pugh charts were used to perform mathematical analysis of each concept. In general, it is projected that the roller-plucker single string system would excel in terms of low physical volume, lightness, and low cost, while still able to have enough bandwidth. It is also the top scorer of educational value due to its simplistic but unique engineering approach to sound propagation.

There are several tasks that need to be performed to move the concept towards a successful prototype. The first one was that since no pre-made instrument would be used for this concept, we needed to test which kind and size of string to choose as it is critical for our desired frequency range and bandwidth. The second important task was to control the roller so that it moves fast enough for songs that have high BPM. One of the problems we worried the most about is that the electric pickup would not only record the motion of the string, but also the motion of the servo motor, thus creating distorted sound.

III. FMEA

We brainstormed several of the most likely and/or severe failure modes for the leading concept, most of which were related to the motion axis and control of the roller plucker. The failure modes of greatest severity (8) were string breakage and stepper overheat. A guitar string snapping has the possibility of permanently harming the user by means of eye damage. We judged that the rigid picker in the initial concept could catch the string and cause excessive

deflection, resulting in wear damage and eventual breakage. Another cause of string breakage we considered was overtension, but the tuning process of the string inherently avoids exceeding the tension limit. We decided to revise the picker design to be more flexible (thinner and longer), such that the picker will deflect and allow the string to slip away before causing excessive deflection. The stepper overheating failure mode solution was to target as low of a current as possible since motor heating is related to P = I^2*R.

Our design had inherently safe architectural features (contributing to low RPN values) as well, such as the tightly toleranced belt and 12 V UL-listed power supply. The belt runs within a channel on the "neck" (roller rail) so a substantial amount of effort is required to get a finger onto the inside of the timing belt loop. As a result, the likelihood of occurrence for a pinch is low. The 12 V UL-listed power supply is the highest voltage in our electrical system, so risk of shock is virtually zero. Additionally, the UL certification means the power supply meets a safety standard which further reduces risk of failure modes like mains voltage isolation breakdown to the low-voltage DC output (*Power supply,* n.d.).

The initial FMEA table (purple section) can be found in Appendix P.

IV. Experimentation

Through preliminary testing and research of our automated instrument, there were certain design parameters that would need to be tuned in order to achieve optimal and smooth pitch adjustment from the roller. In order for the roller to play notes successively and quickly, its acceleration would have to be optimized through the stepper motor controlling its motion. In addition, in order for the stepper motor to play songs for a long period of time, it would need to remain under specific operating temperatures to play not only for a long time, but reliably as well. In order to discover how to optimize these parameters, we wanted to study a couple of

different factors. First, we knew the input voltage and input current into a motor has a significant effect on its performance. We knew how motor speed and input voltage are directly related, in addition, how motor torque and input current are directly related, but needed to investigate how acceleration and temperature fits into these correlations. In addition, we knew that the mass of our moving roller system would have an effect on its performance, but wanted to test to see the exact relationship between mass and the acceleration/temperature of the motor system. In order to study these two phenomena further, we conducted a factorial experiment to answer the following question: how does the motor voltage, motor current, and carriage mass affect the maximum acceleration (without skipping steps) and temperature of the stepper motor?

The experiment studied two responses, the maximum acceleration of the motor (in m/s^2) and the temperature of the motor (in ℃). The factors studied in this experiment were the motor voltage (in V), the motor current (in A), and the mass of the carriage (in g). All of the data captured in this experiment is detailed in Figure Q1. In order to measure the temperature of the motor, a thermal camera was pointed directly at the stepper motor surface to study its temperature after 10 minutes of full-travel cycling at a set acceleration (Figure Q2). In order to measure the maximum acceleration of the stepper motor, an acceleration profile was set and gradually increased until particular steps of the motor were observed being skipped, therefore, achieving a maximum acceleration. The motor current and voltage were set using a DC power supply, using ranges from 0.5 A to 1 A and 6 V to 12 V, respectively. The carriage mass was set by adding blocks of known mass to the system each weighing 10 grams, with a range from 20 g to 50 g.

In order to perform the experiment, we conducted 8 individual experiments and repeated each one 3 times, for a total of 24 experiments. Each experiment studied each factor at its high

and low state of the range mentioned in the previous paragraph, and then studied its response to both of the response variables. We also randomized the order of the experiments to prevent any external factors from giving false relationships between factors and response variables. Some noise factors that would potentially hinder the accuracy of the experiments were any loss in tension from the belt or loss in teeth of the belt which could affect the motion profile and therefore the response variable, acceleration. To keep this consistent across all experiments, a visual inspection of the belt would be performed such that no teeth were stripped or damaged in any way. If the belt had suffered a loss of teeth, it would be immediately replaced with a new belt. Another noise factor that could affect input factors like voltage and current would be power surges. If voltage and current aren't held constant, that could affect the validity of the responses from the experiment. To rectify this, a simple fuse could be placed into the circuit, so that if enough current rushes through, it would prevent any significant disruptions.

From analyzing the results of the cube plots for maximum acceleration (Figure Q3), an important relationship becomes apparent. By minimizing the carriage mass, maximizing voltage, and maximizing current to the range aforementioned, the acceleration is maximized. It seems that the inverse is also true, where maximizing the carriage mass, minimizing the current, and minimizing the voltage results in a minimized acceleration profile. Out of the three factors, it seems current has the highest effect in maximum acceleration as seen in the cube plot. By increasing current from 0.5 A to 1 A, the maximum acceleration increases from 20.83 m/s^2 to 35.00 m/s^2 and from 25.83 m/s^2 to 42.50 m/s^2. This will be an important parameter in the future when we need to maximize the acceleration of our system.

From analyzing the cube plot of temperature (Figure Q4), it becomes clear that by minimizing current, minimizing voltage, and minimizing carriage mass, we can minimize the

temperature of the stepper motor to prevent any overheating. Yet again, it seems that current has the greatest effect on the response variable. By increasing current from 0.5 A to 1 A, the temperature increases from 36.13℃ to 46.20℃ and from 37.50℃ to 48.67℃. This will be an important parameter to focus on minimizing for future design. In addition, it is important to note the minimal effect increasing voltage and increasing carriage mass has on temperature–only increasing temperature by a couple of degrees. Therefore, we may not need to control these parameters for future design because of their minimal effect.

From analyzing the main effects plots, it reinforces the large effect that current has on the studied responses. By increasing current, both maximum acceleration and temperature increase the most out of all the other responses. In addition, the main effect plots further demonstrate that by independently increasing current (Figure Q5), increasing voltage (Figure Q6), and decreasing carriage mass (Figure Q7) will increase maximum acceleration. For minimizing temperature, independently current (Figure Q8), decreasing voltage (Figure Q9), and carriage mass (Figure Q10) will suffice. Similarly, it can be seen from the temperature main effects plot for voltage (Figure Q9) and carriage mass (Figure Q10), that although they have a directly proportional relationship with temperature, they have a very minimal effect on temperature.

From analyzing the maximum acceleration interaction plots (Figure Q11, Q12, and Q13), it seems that all factors are completely independent from each other. The curves for each of the plots were generally all parallel, meaning they had little interaction between each other. For the temperature interaction plots (Figure Q14, Q15, and Q16), they also were all generally parallel, so we can assume the factors were independent for the temperature response as well.

Lastly, from analyzing the regression results (Figure Q17 and Q18), it seems that all three factors have a statistically significant relationship($p\text{-value}<0.05$) with each respective response

variable. Therefore, it is safe to conclude that maximum acceleration and temperature both depend on voltage, current, and the carriage mass. In addition, the Multiple R-Squared values for both responses are 0.985 and 0.998 for maximum acceleration and temperature, respectively, indicating a very strong relationship between the responses and the factors. Regarding the interaction terms, for temperature it seems that these coefficients are all statistically significant as well, meaning there is a relationship between the interactions and the temperature variable. However, the interaction terms for maximum acceleration are all not statistically significant meaning, they will be excluded from the final regression equation.

From the analysis of the experiment results, it seems many parameters should be adjusted to reach our target responses for our final design. If we need to maximize voltage up to 12 V(for higher acceleration) and carriage mass(up to 50 g), the stepper motor temperature will remain generally unaffected. So, voltage can be increased to its maximum of 12 V and the response performance will be optimized. However, increasing carriage mass will have a significant negative effect on maximum acceleration, so we will need to be wary of this tradeoff in the future. Regarding current, increasing it will boost the maximum acceleration but at the cost of higher stepper motor operating temperatures. Due to maximum acceleration being an important engineering requirement in order to play a variety of songs and meet our customer needs, this should be prioritized at the cost of higher temperatures. If higher temperatures become an issue and begin to affect the system's performance(maximum acceleration and reliability), a cooling device can be retrofitted onto the side of the motor to prevent any overheating. This will be contingent on future design only, so may not be critically necessary at this moment.

V. Simulation

While the behavior of the string is relatively intuitive in the sense that a short and/or lighter string will result in a higher pitch and vice-versa, it was important for us to gain a deeper understanding of how the independent variables, string mass (proportional to diameter), tension, and length would influence the pitch of the note. To begin, we found an equation that models this situation seen below.

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

Then, we used the diameter and density of the string to calculate m with the following equation.

$$m = 2\pi\left(\frac{d}{2}\right)^2$$

One of the most critical characteristics of the instrument is its bandwidth which is essentially the range of notes it is capable of playing. We specifically focused on how string diameter (mass) and tension would impact the bandwidth of the instrument for a fixed minimum and maximum string length. To accomplish this, a script calculated the frequency of the string across the set range of lengths for a range of tension values for each size of string available to us. The resulting data can be seen in figure R1 in appendix R. Analysis of this data indicated to us that using a lighter string with greater tension would maximize bandwidth. With this in mind, we optimized our mechanical design to increase the maximum string tension it could support with reasonable deflection and decreased the size of the string.

This model assumes that there is no damping in the system which could result in slightly lower frequencies, however, testing of the actual instrument showed that this method of simulation was extremely accurate (the smartphone app used to measure string frequency produced results which varied greater than any measurable error).

VI. Design for Manufacturing/Assembly

It was very necessary to design our final product for ease of manufacturing and assembly since we targeted low physical volume, lightness, and low cost. In order for our components to be easily manufactured and assembled, we primarily focused on two DFM/DFA rules: minimize part counts by incorporating multiple functions into single parts & standardized to reduce part variety.

There are several parts in our final product that incorporate multiple functions. For example, the pickup holder not only serves as a support for the electric pickup which amplifies the sound, as we needed the electric pickup to be close enough to the string for better sound quality, it is also where the tuning peg that's used to adjust the tension of the string is attached to. This design saved us time and effort to design an additional tuning peg holder for its specific purpose. The motor mount we designed is the center of our assembly. Apart from its intended function, it serves as a base for the servo motor mount and a base for one of the pulleys. The supporting base is attached to the motor mount as well as the pickup holder and the entire roller-belt assembly. This not only greatly reduces the complexity of our physical system, but it also lowers the volume of the final product.

To standardize and reduce part variety, we designed our parts and subsystems so that m3 bolts and nuts are the main way of attachment; this includes the connection between the pickup and the pickup holder, the pickup holder and the motor mount, the motor mount and the servo mount, the servo mount and the servo motor, the motor mount and the supporting base, as well as the motor mount and the roller-belt assembly.

Since almost all the parts are either purchased or 3D printed, it was quite limiting what we could do for ease of fabrication and repair. First, for all the 3D printed parts, we designed them to be symmetrical if possible, not to mention the effort to reduce the potential material needed for printing. For ease of repair, all of the connections between parts use less or equal to 4 attachment points; that means it barely takes time to disassemble the final product to perform repair and replacement.

VII. Sustainability

In order to incorporate sustainable initiatives into our project, we have adhered to several principles and guidelines under the Design for Environment paper as outlined in lecture(Telenko, et al., 2008). First, we have designed for easy repair and upgrading (Principle E-47) through our string tensioner. As the use of the instrument increases, the string will become more and more out of tune. In order to rectify this and not have a need to replace any parts or purchase any new strings, we added a tensioner so that the user can manually adjust the string back to its original sound.

In addition, we have designed for energy-saving feature controls (Principle D-37) by allowing the user to specify a “minimum acceleration” mode. By selecting this feature, the stepper motor controlling the pitch adjustment for the roller will draw minimal current to reach minimum accelerations, and therefore, minimal power consumption as well.

Lastly, we have also designed for minimizing volume specifically for the servo mount that plucks the string through applying methods of structural strengthening (Principle C-19). Through using fillets at points of high stress in the mount, we were able to reduce stress and reduce the need for extra, wasted material to further strengthen it. In addition, during the 3D printing process, strategically placing the layer direction orthogonal to the direction of the build

volume, reduced the need for more, wasted material to strengthen a point of high stress concentration along the slot of the mount.

Currently, our design detracts from sustainability principles by first, not using recyclable resources to 3D print (Principle A-2). The main rail of the assembly is printed with a resin material, which is made using non-renewable fossil fuels. In order to improve this, we could use R-PLA or R-PET which are recyclable 3D printed materials. Another way our design detracts from these principles is by not having a feedback monitor to view energy consumption of the device (Principle D-36). During the demonstration to students, monitoring how stepper motor acceleration can change as a function of the current drawn is vital for responsible energy usage. To rectify this, we could easily add a digital display that monitors voltage, current, and power, so that the user can see how their acceleration curves impact the overall energy usage of the system.

VIII. Final FMEA

We made design changes to the picker and neck to address the failure modes in the initial FMEA before beginning manufacturing and assembly. Specifically, the picker was designed to be 0.8 mm thick and 20 mm long (a good initial guess) to allow for sufficient picker deflection. This deflection successfully eliminated cases in which the string got caught on the picker. Also, we added a central retaining screw to the picker instead of having only a press-fit for retention on the servo spline. The neck thickness was increased from 20 to 25 mm to reduce bending stresses caused by string tension, reducing creep rate and the possibility of string tension loss.

After assembly and testing of our initial prototype (including initial FMEA revisions), we found three more failure modes which were an explosion of a stepper driver, endstop firmware problems, and pulley-shaft interface slippage. We suspect the stepper driver exploded and went

up in smoke due to an LC voltage spike (*Understanding destructive LC voltage spikes,* n.d.) that occurred when the power supply was connected. We followed the advice in the article and placed a high ESR capacitor in parallel with the stepper driver input, and have not had the problem since. The endstop firmware problem occurred when the microcontroller was powered on while the endstop switch was pressed. We found that the switch was pulling down the 3.3 V rail too low during startup (causing a failure to execute the code), and our solution was to change the firmware behavior to move the carriage away from the endstop after homing (bumping into the endstop initially). For the pulley-shaft interface slippage, the cause was an oversized D-shaft hole in the pulley. This caused the pulley to climb up the shaft during movements and eventually contact the string, resulting in wrong notes. Instead of using a set screw like our original suggested remedy, we shimmed the hole with a piece of masking tape as a quick solution. The pulley has not climbed since.

With more time, we would like to develop more reliable solutions for the endstop and pulley problems. On initial startup, the user still has to be careful of keeping the endstop switch unpressed. We could either modify the electrical circuit with a high value resistor in series or add a spring near the endstop. The pulley would ideally be made of metal or fiber-reinforced plastic to allow use of a set screw without a significant rate of creep. Another major concern is overall structure heat deflection temperature, where our current choice of PLA makes leaving the instrument in a hot car dangerous due to the possibility of warping. Otherwise, we were able to reduce our RPN values by at least 40% on average and judge the instrument as generally reliable.

The final FMEA table (purple and green sections) can be found in Appendix P.

IX. Final Drawings, BoM, Budget

The final design seeks to satisfy the engineering requirements and by extension the customer needs, with huge emphasis on minimizing size and mass without mitigating performance as key objectives to be met. In addition, fulfillment of these goals whilst also being within the allocated budget meant that the layout and selection of mechanical and electromechanical components must be kept simple. Appendix S shows that the total price sum of purchased parts totals $176. 12 purchases were made in total and the use of personal printers and the 3D printer at the Makerspace are to be considered free. The most expensive purchase was a brushed DC motor with a gearbox and encoder for $100 and was crucial to plucking the string by the simplest means possible whilst ensuring accurate positioning.

The final design emphasizes ease of operation by tightly packaging the electronics on a side-mounted breadboard with an ‘E-stop’ button added on the side for safety. The ESP microcontroller allows for commands to be sent to the stepper motor and DC motor for rapid concurrent operation. The stepper motor is run in open-loop and is connected to a pulley which has a belt attached to the roller. This roller presses down on the string and the movement of the stepper motor consequently changes the effective length between the two anchor points; thereby quickly changing the frequency. The stepper has a horn acting as the end-effector plucker which peturbs and vibrates the string. The sound is then picked up by the electric pickup which is interfaced to any device by cable to allow music to be played from any device.

The design consists of stabilizing trapezoids as the supporting base best seen in Figure T2. The supporting base is placed between the roller path and pickup, and surrounds the stepper motor and encapsulates it. Nearly above the pulley, a servo mount is used to rigidly secure the servo motor while it plucks from above (Figure T3). This ‘centralized’ approach to housing the

actuators means that the overall volume can be budgeted towards maximizing the length of the neck. Overall, the machine can be held with one hand, carried in a bag, or held in a small container to allow for great portability.

In terms of acoustics, tuning pegs at the back enable the user to scale the pitch of the notes. In terms of bandwidth,the design was configured to allow notes across one octave to be played, ensuring that a wide range of songs can be played without having to overly extend the neck, which would otherwise sacrifice the sound quality. The electric pickup not only amplifies sound, but also allows the user to tune music through software like GarageBand before playing it for additional modularity.

In terms of software, the system relies on concurrent use of Python and C++ syntax (Arduino). With C++ and the ESP8266, servo and stepper header, .ino, and .cpp firmware files were written to accept serial transmitted using Python to pluck or move the roller. Python allows for more seamless high-level functionality such as music to be played rather than directly working with lower-level syntax. In python, timings for plucking and roller positions were configured. Additionally, much of the trajectories were subject to trapezoidal trajectories to optimize speed without reaching acceleration limits and causing the system to slip.

Most importantly, Python code provided the best means to greatly boost user participation by developing a GUI using Tkinter. This GUI keeps a member of the audience engaged by allowing individual notes to be selected and played at different times. From there, the notes are connected in accordance with an optimized trajectory (to prevent note dragging), and position-time,velocity-time, and acceleration-time plots are created. This helps students understand the fundamentals of calculus, which is a key concept in mechanical engineering. One can see Appendix U for a thorough documentation of code and software.

X. Operating and Repair Instructions

Operation of the instrument is designed to be as simple as possible. The physical setup merely consists of connecting three cables. Once connected, a graphical user interface can then be used to “program” and play songs. These instructions can be found in Appendix V. The instructions were kept brief, to the point, and free of technical jargon. As such, nearly anyone comfortable with using a computer and other consumer electronic devices should be comfortable following them. This is aided by the fact that the instrument does not require any mechanical setup or familiarity with any other third party software.

Furthermore, steps to remedy the few issues we did run to throughout development and testing are also included in Appendix V. While these are slightly more involved, requiring the use of a screwdriver, an effort was made to keep them simple and to the point.

XI. Final Discussion and Recommendations

The design process for this project began with extensive background research and customer interviews. From there, it progressed to generating a set of customer needs and criteria to adhere to ensure a successful product. From there, concepts were generated, evaluated, and the most viable one was prototyped, and then assembled. We began by rigorously analyzing information from potential customers, field experts, and secondary research to guide the direction of product development. We then kept the engineering requirements and customer needs as fundamental design philosophies to adhere to when generating these concepts, of which the roller plucker proved to best fulfill these criteria, and so it was prototyped the first time to provide us with some powerful insights in terms of acoustics, mechanical design, etc. By using these insights and analyses, we were able to successfully carry out a highly data-driven and

incrementalist approach to finalizing our design, assembling it and tuning it to cater to the intended target market.

One feature that our team is very proud of is the overall simplicity of the design which resulted in a chassis that requires assembly only once before being used, while also being remarkably simple to tune for good acoustic performance using nothing more than the tuning pegs. More importantly, this simple design where all the actuators and critical components were kept close to one another allows a compact design where the longest dimension can be allocated for roller traversal and vibration propagation without too much interference from electromechanical sound (servo and stepper sounds). This powerful simplicity yielded so many benefits and was highly effective in meeting all of the criteria which related to size and portability. As a further improvement, some excess weight can be removed by trimming the design and optimizing to make it lighter. Also, while possible to wield by hand, holding it at the neck can result in permanent deformation, so implementing a carrying handle can greatly supplement the product ergonomically.

The greatest aspect of the final product is the GUI which serves as a powerful educational tool that can keep any user engaged. Using electromechanical systems and music to introduce calculus to a k-12 target audience member is an idea hitherto implemented, and the roller-plucker system served as the perfect platform to attempt this attempt at creating an educational product. In addition, the GUI is supplemented by the ability to easily connect the prototype to a laptop which can then be used as a speaker to play any music. Simply, by using .yaml files for music and readily available software like GarageBand for an added layer of tuning, the modularity,

versatility, and ease of setup and operation are the strongest selling points that define the prototype.

Even with the excellent performance seen, there are extensive improvements that can be carried out to further increase the effectiveness of the product. A huge glaring issue was the exposed breadboard which left all sensitive electronics exposed to the surrounding environment. As a result, there is a tendency for the electronics to become loose, thus breaking circuits. More importantly, exposed electronics are unsafe to be around, especially with younger audiences from the k-12 audience. By enclosing the electronics in a harness, these issues can be solved, with the added benefit of improving the aesthetics and also allowing the instrument to be transported more easily.

Another improvement that could be done to our final product is selecting and configuring a stepper motor that allows higher voltage input. This will increase the maximum acceleration of the roller and thus allow us to play songs with higher BPM.

Despite these presented issues and suggested improvements , our product fulfills all of the customer needs established at the beginning, with the exception to bandwidth expectations which were deemed rather unrealistic for a product of this small scale . Our product reliably plays multiple songs from digital files, and demonstrates educational principles to K-12 members.

Works Cited

Brass instrument (lip reed) acoustics: an introduction. Brass Instrument (Lip Reed) acoustics: An introduction. (n.d.). Retrieved September 13, 2022, from http://www.phys.unsw.edu.au/jw/brassacoustics.html

Dejan. (2022, February 17). How a Stepper Motor Works. How to Mechatronics. Retrieved September 14, 2022, from https://howtomechatronics.com/how-it-works/electrical-engineering/stepper-motor/

Harper, C. (2022, August 15). Arduino vs Raspberry Pi: The Pros & Cons. Make Tech Easier. Retrieved September 13, 2022, from https://www.maketecheasier.com/arduino-vs-raspberry-pi/

Power Supply Test and certification. UL Solutions. (n.d.). Retrieved December 4, 2022, from https://www.ul.com/services/power-supply-test-and-certification

Telenko, C., Seepersad, C. C., & Webber, M. E. (2008). A COMPILATION OF DESIGN FOR ENVIRONMENT PRINCIPLES AND GUIDELINES. Retrieved December 3, 2022.

Tutorial: Stepper vs Servo. (n.d.). Retrieved September 14, 2022, from https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/stepper-vs-servo/

Understanding destructive LC voltage spikes. Pololu Robotics & Electronics. (n.d.). Retrieved December 4, 2022, from https://www.pololu.com/docs/0J16/all

Welch Music Center. (n.d.). Meridian Brass Instruments Boise for sale. Brass Instruments Boise-Welch Music Store Meridian. Retrieved September 13, 2022, from https://welchmusic.com/brass.html

Wikimedia Foundation. (2022, September 3). Trumpet. Wikipedia. Retrieved September 13, 2022, from https://en.wikipedia.org/wiki/Trumpet

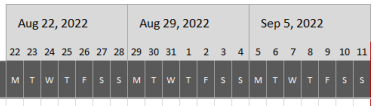
Appendix

Appendix A: Gantt Chart

Linkedin Park
Project Lead

Project Start: 8/26/2022
Display Week: 1

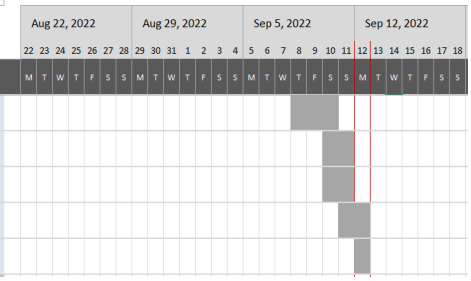
TASK	ASSIGNED TO	PROGRESS	START	END
Project Proposal				
1.Gantt Chart	All	100%	8/26/22	8/29/22
2. Backgroud Research	All	100%	8/26/22	8/29/22
3. Targets & Questions	All	100%	8/29/22	9/1/22
3.1. Questions for K-12 Students	Elliot Turner	100%	8/29/22	9/1/22
3.2. Questions for outreach people	Peter Matthew	100%	8/29/22	9/1/22
3.3. Questions for educators	Elliot Turner	100%	8/29/22	9/1/22
3.4. Questions for musicians	Yifeng Liao	100%	8/29/22	9/1/22
3.5. Questions for robotics experts	Talal Al-Otaibi	100%	8/29/22	9/1/22
4. Go out and Ask	All	100%	9/1/22	9/8/22
4.1. Interview K-12 Students	Elliot Turner	100%	9/1/22	9/8/22
4.2. Interview outreach people	Peter Matthew	100%	9/1/22	9/8/22
4.3. Interview educators	Elliot Turner	100%	9/1/22	9/8/22
4.4. Interview musicians	Yifeng Liao	100%	9/1/22	9/8/22
4.5. Interview robotics experts	Talal Al-Otaibi	100%	9/1/22	9/8/22



Linkedin Park
Project Lead

Project Start: 8/26/2022
Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
5. Formulate House of Quality	All	100%	9/8/22	9/10/22
6. Engineering requirements + spec	All	100%	9/10/22	9/11/22
7. Writing the problem statement	All	100%	9/10/22	9/11/22
8. Writing the proposal	All	100%	9/11/22	9/12/22
9. Review and Submit the Proposal	All	100%	9/12/22	9/12/22



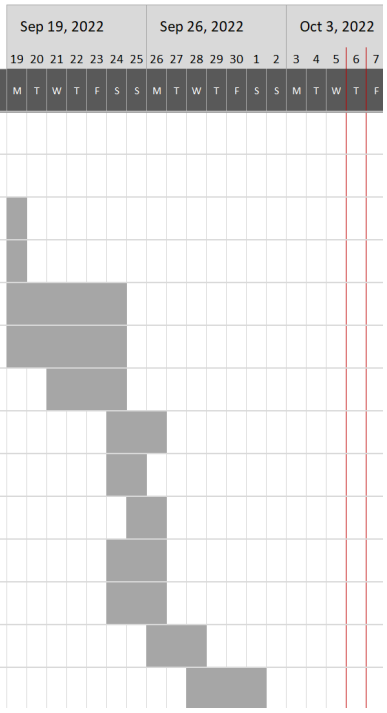
Automated Musical Instrument

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-ch

Linkedin Park
Project Lead

Project Start: 8/26/2022
Display Week: 5

TASK	ASSIGNED TO	PROGRESS	START	END
Design Review				
1. Update Gantt Chart	Yifeng Liao	100%	9/13/22	9/14/22
2. Formulate Concepts and Products	All	100%	9/14/22	9/19/22
2.1. Create the black box	Andrew Zhang	100%	9/17/22	9/19/22
3. Sub-problem and concepts	All	100%	9/19/22	9/24/22
3.1. Discuss about Function Tree	All	100%	9/19/22	9/24/22
3.2. Draw the Function Tree	Peter Matthew	100%	9/21/22	9/24/22
4. Mind-Maping	All	100%	9/24/22	9/26/22
4.1. Discussion on Mind-Mapping	All	100%	9/24/22	9/25/22
4.2. Draw the Mind-Mapping Tree	Elliot Turner	100%	9/25/22	9/26/22
5. TRIZ Table	Andrew Zhang	100%	9/24/22	9/26/22
6. Identify potential solutions to sub-problems	All	100%	9/24/22	9/26/22
7. Make Initial Morph Matrix	Peter Matthew	100%	9/26/22	9/28/22
8. Prior Art Research	Yifeng Liao	100%	9/28/22	10/1/22



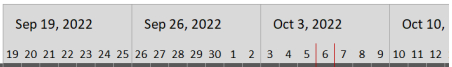
Automated Musical Instrument

Linkedin Park
Project Lead

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Project Start: 8/26/2022
Display Week: 5

TASK	ASSIGNED TO	PROGRESS	START	END
8. Prior Art Research	Yifeng Liao	100%	9/28/22	10/1/22
9. Update Prior Art Research into Morph Matrix	Peter Matthew	100%	10/1/22	10/2/22
10 Decisions of Concepts	All	100%	10/2/22	10/4/22
10.1. Go through the Morph Matrix	All	100%	10/2/22	10/4/22
10.2. Sketch the Concepts	All	100%	10/2/22	10/4/22
11. Create Pugh chart	Talal Al-Otaibi	100%	10/4/22	10/5/22
11.1. Identify key problems	Talal Al-Otaibi	100%	10/4/22	10/5/22
11.2. Justifications / Calculations	Talal Al-Otaibi	100%	10/4/22	10/5/22
11.3. Cost Estimations	Talal Al-Otaibi	100%	10/4/22	10/5/22
12. Low Resolution Prototype	Elliot Turner	100%	10/2/22	10/5/22
12.1. CAD / Sketch A Low Resolution Prototype	Elliot Turner	100%	10/2/22	10/5/22
12.2. Making the Low Resolution Prototype	Elliot Turner	100%	10/2/22	10/5/22
13. Draft Project Design Review	All	100%	9/24/22	10/5/22
14. Review and Submit Design Review	All	100%	10/5/22	10/6/22



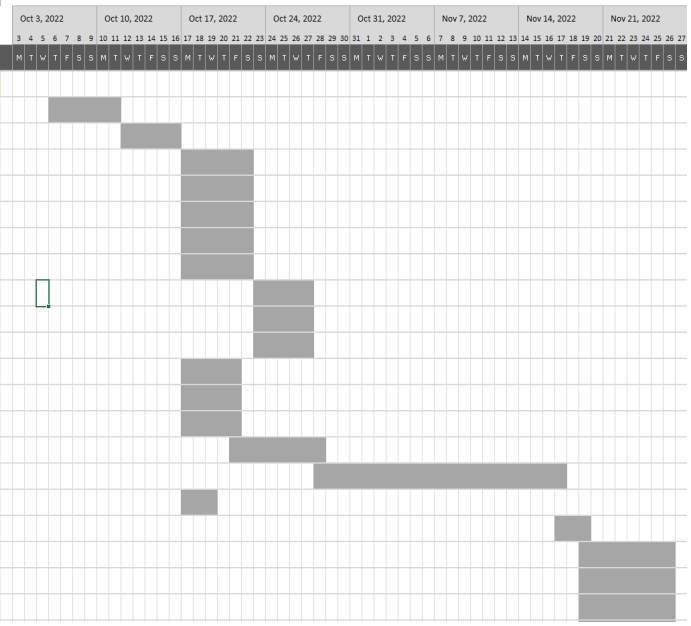
Automated Musical Instrument

Linkedin Park
Project Lead

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Project Start: 8/26/2022
Display Week: 7

TASK	ASSIGNED TO	PROGRESS	START	END
Embody Concepts				
1. Create 3D Sketch	Elliot Turner	100%	10/6/22	10/11/22
2. Bill of Materials	Talal Al-Otaibi	100%	10/12/22	10/16/22
3. CAD Models	All	100%	10/17/22	10/22/22
3.1. Roller, Stepper Mount, and Pully CAD	Elliot Turner	100%	10/17/22	10/22/22
3.2. Tuning Peg, Pickup, Pickup holder CAD	Yifeng Liao & Talal	100%	10/17/22	10/22/22
3.3. Servo & Plucker CAD	Andrew Zhang	100%	10/17/22	10/22/22
3.4. Servo Mount & Supporting Base CAD	Peter Mathews	100%	10/17/22	10/22/22
4. Failure Analysis	All	100%	10/23/22	10/27/22
4.1. Stress FEA Simulation	Elliot Turner	100%	10/23/22	10/27/22
4.2. Equipment Current & Temperature Simulation	Andrew Zhang	100%	10/23/22	10/27/22
5. Building the final prototype	Elliot & Andrew	100%	10/17/22	10/21/22
5.1. Printing Stepper Mount, Pully, Roller	Elliot Turner	100%	10/17/22	10/21/22
5.2. Printing Servo Mount, Pickup Holder	Andrew Zhang	100%	10/17/22	10/21/22
6. Testing the final prototype with GUI	Elliot	100%	10/21/22	10/28/22
7. Adjusting and Fine tuning the prototype	All	100%	10/28/22	11/17/22
8. Deciding the songs to be played	Yifeng Liao	100%	10/17/22	10/19/22
9. Assigning roles of the presentation	All	100%	11/17/22	11/19/22
10. Making the final presentation slides	All	100%	11/19/22	11/26/22
10.1. Concepts and Leading Concept	Andrew Zhang	100%	11/19/22	11/26/22
10.2. Servo Motor Design	Peter Matthew	100%	11/19/22	11/26/22



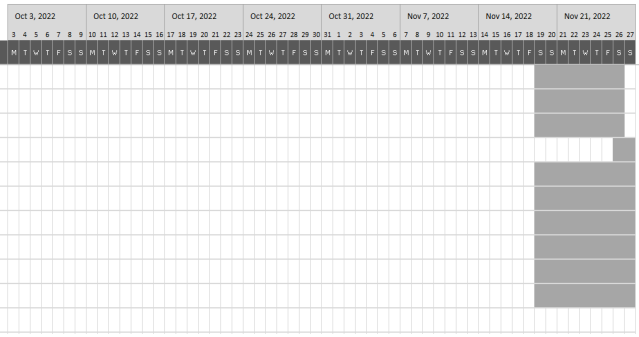
Automated Musical Instrument

Linkedin Park
Project Lead

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Project Start: 8/26/2022
Display Week: 7

TASK	ASSIGNED TO	PROGRESS	START	END
10.3. Strings & Pickup	Yifeng Liao	100%	11/19/22	11/26/22
10.4. Roller, Firmware & Software Design	Elliot Turner	100%	11/19/22	11/26/22
10.5. Final Product	Talal Al-Otaibi	100%	11/19/22	11/26/22
11. Practice final Presentation	All	100%	11/26/22	11/29/22
12. Writing the final report	All	100%	11/19/22	12/3/22
12.1. Intro, experimentation, sustainablity	Peter Matthew	100%	11/19/22	12/3/22
12.2. FMEA, final FMEA	Andrew Zhang	100%	11/19/22	12/3/22
12.3. Gantt Chart & Tasks List, DFM/DFA, Leading Concepts	Yifeng Liao	100%	11/19/22	12/3/22
12.4. Simulation	Elliot Turner	100%	11/19/22	12/3/22
12.5. Final Drawings, BOM, Budget, instruction, final discussion	Talal Al-Otaibi	100%	11/19/22	12/3/22
13. Review and Submit the final report	All	100%	12/3/22	12/5/22



Appendix B: Task List

Project Proposal

1.Gantt Chart	All	100%
2. Background Research	All	100%
3. Targets & Questions	All	100%
3.1. Questions for K-12 Students	Elliot Turner	100%
3.2. Questions for outreach people	Peter Matthew	100%
3.3. Questions for educators	Elliot Turner	100%
3.4. Questions for musicians	Yifeng Liao	100%
3.5. Questions for robotics experts	Talal Al-Otaibi	100%
4. Go out and Ask	All	100%
4.1. Interview K-12 Students	Elliot Turner	100%
4.2. Interview outreach people	Peter Matthew	100%
4.3. Interview educators	Elliot Turner	100%

4.4. Interview musicians	Yifeng Liao	100%
4.5. Interview robotics experts	Talal Al-Otaibi	100%
5. Formulate House of Quality	All	100%
6. Engineering requirements + spec	All	100%
7. Writing the problem statement	All	100%
8. Writing the proposal	All	100%
9. Review and Submit the Proposal	All	100%

Design Review

1. Update Gantt Chart	Yifeng Liao	100%
2. Formulate Concepts and Products	All	100%
2.1. Create the black box	Andrew Zhang	100%
3. Sub-problem and concepts	All	100%
3.1. Discuss about Function Tree	All	100%
3.2. Draw the Function Tree	Peter Matthew	100%
4. Mind-Mapping	All	100%
4.1. Discussion on Mind-Mapping	All	100%

4.2. Draw the Mind-Mapping Tree	Elliot Turner	100%
5. TRIZ Table	Andrew Zhang	100%
6. Identify potential solutions to sub-problems	All	100%
7. Make Initial Morph Matrix	Peter Matthew	100%
8. Prior Art Research	Yifeng Liao	100%
9. Update Prior Art Research into Morph Matrix	Peter Matthew	100%
10 Decisions of Concepts	All	100%
10.1. Go through the Morph Matrix	All	100%
10.2. Sketch the Concepts	All	100%
11. Create Pugh chart	Talal Al-Otaibi	100%
11.1. Identify key problems	Talal Al-Otaibi	100%
11.2. Justifications / Calculations	Talal Al-Otaibi	100%
11.3. Cost Estimations	Talal Al-Otaibi	100%
12. Low Resolution Prototype	Elliot Turner	100%
12.1. CAD / Sketch A Low Resolution Prototype	Elliot Turner	100%
12.2. Making the Low Resolution Prototype	Elliot Turner	100%

13. Draft Project Design Review	All	100%
14. Review and Submit Design Review	All	100%
Embody Concepts		
1. Create 3D Sketch	Elliot Turner	100%
2. Bill of Materials	Talal Al-Otaibi	100%
3. CAD Models	All	100%
3.1. Roller, Stepper Mount, and Pully CAD	Elliot Turner	100%
3.2. Tuning Peg, Pickup, Pickup holder CAD	Yifeng Liao & Talal	100%
3.3. Servo & Plucker CAD	Andrew Zhang	100%
3.4. Servo Mount & Supporting Base CAD	Peter Mathews	100%
4. Failure Analysis	All	100%
4.1. Stress FEA Simulation	Elliot Turner	100%
4.2. Equipment Current & Temperature Simulation	Andrew Zhang	100%
5. Building the final prototype	Elliot & Andrew	100%
5.1. Printing Stepper Mount, Pully, Roller	Elliot Turner	100%
5.2. Printing Servo Mount, Pickup Holder	Andrew Zhang	100%

6. Testing the final prototype with GUI	Elliot	100%
7. Adjusting and Fine tuning the prototype	All	100%
8. Deciding the songs to be played	Yifeng Liao	100%
9. Assigning roles of the presentation	All	100%
10. Making the final presentation slides	All	100%
10.1. Concepts and Leading Concept	Andrew Zhang	100%
10.2. Servo Motor Design	Peter Matthew	100%
10.3. Strings & Pickup	Yifeng Liao	100%
10.4. Roller, Firmware & Software Design	Elliot Turner	100%
10.5. Final Product	Talal Al-Otaibi	100%
11. Practice final Presentation	All	100%
12. Writing the final report	All	100%
12.1. Intro, experimentation, sustainability	Peter Matthew	100%
12.2. FMEA, final FMEA	Andrew Zhang	100%
12.3. Gantt Chart & Tasks List, DFM/DFA, Leading Concepts	Yifeng Liao	100%
12.4. Simulation	Elliot Turner	100%

12.5. Final Drawings, BOM, Budget, instruction, final discussion	Talal Al-Otaibi	100%
13. Review and Submit the final report	All	100%

Appendix C: Interview Responses

Appendix C1: Interview Responses from People that play instruments

Interviewer: Yifeng Liao

Interviewees: Zhenghao Liao (ZL), Yuanmu Li (YL), Haoxuan Mu (HM)

1. **What instruments do you play?**

ZL: Piano

YL: Guqin (an ancient Chinese instrument)

HM: Guitar

2. **How long have you played it?**

ZL: 13 years

YL: 4 years

HM: 4 years

3. **How did you first get started?**

ZL: Introduced by my parents

YL: I started learning online

HM: I started learning from Youtube

4. What are some major differences between your instrument and other major instruments?

ZL: The notes are more complicated

YL: It has a very unique music notes sheet. You have to be able to understand Chinese in order to read it. One of the oldest instruments in human history.

HM: This instrument needs a lot of chords, similar to piano. You must need 2 hands to play it.

5. What was the hardest part of learning your instrument?

ZL: Learning how to read piano notes

YL: Using the correct hand/fingering form

HM: Remembering the chords and using the correct hand/fingering form

6. How would you improve the instrument so that it's easier for a rookie to learn?

ZL: I would not because piano is the king of musical instruments

YL: I don't think this instrument is too hard

HM: I think having something that automatically play chords, new learner usually don't know how to play chords

7. What makes an instrument high quality vs low quality (a good piano vs. a bad piano)?

ZL: Sound warm and less digital, closely mimicking an acoustic piano

YL: It's just how enjoyable the sound is. If it sounds good, then it is a good instrument

HM: Basically good sound quality, comfortable to use, easy to press the string

8. Why do you think people don't learn a musical instrument? (money? Time? The level of difficulty?)

ZL: It is too expansive and time consuming to learn an instrument like piano

YL: People don't have a lot of interest, and it is very time consuming

HM: People don't have interest or not having enough time, but not having enough time is still a part of lacking interest. If the learning process is interesting enough, I'm sure more people would like to learn it.

9. Would you like to see your instrument automated?

ZL: No, I would not want to see that

YL: No!

HM: Yes.

10. What purpose do you think an automated musical instrument could serve?

ZL: Nothing, we have speakers and vinyl record players for a reason

YL: Easier for music producer to make new music

HM: I think it could teach a rookie which string to press to play a chord

11. If you must own an automated instrument, what attributes would you like it to have?

ZL: Cheap, loud, easy to transport, can see the mechanical movement, good sound quality

YL: Has good sound quality, cheap, easy to transport, easy to maintain and fix

HM: Has good sound quality, loud, easy to transport

Appendix C2: Interview Responses from Educator and K-12 Students

Interviewer: Elliott Turner

Interviewee: John Adams (Educator)

1. What are some of the most important skills for STEM students to learn?

Collaboration skills, teamwork, project management, systems engineering, and coding.

2. What particular skills / knowledge do STEM students tend to struggle with the most?

Students initially struggle the most with programming. Some can pick it up quickly, but many others struggle to get used to it and often give up on it.

3. What are some ways you would recommend a mechanism to produce a sound (a "tone" specifically)?

Go buy a solid state tone generator and program a microcontroller to generate sound. Aside from that, "plucking" or "hitting" would probably be easiest.

4. Are there any specific features you as an educator would expect from such a device?

It should be intuitive to use and idiot proof, meaning that no matter what a user does (within reason), they can't "mess it up".

Interviewees: 9th-12th grade students

1. What subject area of STEM do you find most challenging?

I1: I think that coding is really hard to learn

I2: 3D modeling is a lot harder than I thought it would be

I3: I tried learning programming and it was very difficult

I4: Software is very hard to write

2. What stem related activity do you enjoy the most?

I1: I love playing video games. <was asked to pick something else> I really like welding.

I2: I think that 3D printing is really cool.

I3: I also really like playing video games but I also like 3D printers.

I4: I like working on building robots

3. What type of music do you typically listen to?

I1: Rap

I2: Electronic

I3: Hip-Hop

I4: Electronic

4. If you had an automated musical instrument, would you prefer to be able to play existing songs on it or write your own songs for it to play?

I1: Write my own songs

I2: Play existing songs

I3: Write my own songs

I4: I would like to be able to do both

5. What would be the most important factor to consider when evaluating the quality of the instrument?

I1: I think the instrument should be durable enough to be played a lot without breaking or requiring maintenance.

I2: It should sound nice and not make any terrible noises.

I3: It should be able to play popular songs.

I4: It should be loud enough to hear easily or maybe it could let you plug in headphones or something to be able to hear it well.

6. What do you think would be an appropriate size for a musical instrument?

I1: I think it should be small enough to carry it in one hand.

I2: It should be able to fit in the trunk of a car.

I3: It should be small enough to carry and should include a handle to pick it up with.

I4: I would want to be able to comfortably carry it or at least have a case with wheels.

Appendix C3: Interview Responses from Outreach

Interviewer: Elliot Turner

Interviewee: Peter Mathews (Outreach Coordinator)

Have you ever been involved in robotics outreach programs? What was successful here?

Yes, I did Discovery Day. This was a day when kids would come in to see our high school robotics team. They would build a LEGO robot of their own. We would help them build it. Guiding them gently through the build process and asking them questions was super successful. It kept them very engaged. Being hands-on and building the LEGOs themselves were great too. Don't just build the whole thing for them. Let them learn by interacting with it and ask them good questions to help them when stuck. I would recommend making sure to make time to ask them good questions to teach them how it works. Don’t just tell them.

What have the most successful outreach opportunities included?

Participation from the kids was the most successful outreach strategy we had. For example, we had a dancing robot and we would engage the kids by dancing with them as the robot danced. Singing with the instrument would be something very engaging if you did that. If the song you played had lyrics, kids would love that. They also love silly activities. If the song itself is funny, they would love that but still be learning a lot from the activity.

Do you have a budget when doing outreach?

No funding was given to us. We had to use parts that we already owned to present at outreaches. I'd prefer not to use my own money to purchase anything for the outreaches.

Do you have a time constraint when doing outreach?

We were given around 10-20 minutes to present and do our demonstration. Around 1 hour for the whole team(who presented different robots)

Who are the outreaches geared toward? What age range?

K-5th graders. Any age above that would have been a little awkward. Not because of the technology we were presenting, but because of the way we were delivering it. We had a specific tone, script, and activities that were geared toward this age range.

Where did you have outreaches? What equipment did you have access to?

We had our outreaches at the local library in a small room. It’s about the size of a classroom. We have access to outlets, projection systems, tables, and chairs. Everything else we had to bring. It would be ideal if there was no external equipment needed other than that for the product itself.

What have been the least successful outreach opportunities?

I had a robot malfunction during an outreach. Need a reliable system/product that will not fail during operation.

How long would be ideal for setup?

I have very little time beforehand to set up for the outreach event. Not sure of an exact number.

Appendix C4: Interview with Acoustics Expert

(Interviewer Talal Al-Otaibi, Interviewee: Dr Samuel P. Wallen)

1. What is your background in terms of career?

- PhD in mechanical engineering. Thesis: acoustic wave propagation
- Additional: music fan, saxophone player. Played from youth to undergrad.
- Permanent staff researcher at Applied Research Laboratories. Researches acoustics of exotic materials (useful properties for naval applications and generally).

2. Have you ever tutored, taught, or lectured students from k-12. In acoustics?

- Not formally. Involved in outreach events at the University of Washington as well as UT.
- Focused on acoustics demos accessible to younger kids as an audience
- Important deduction based on his experience: content and technicality massively depend on the audience. Demo modified to match the audience's age.

3. Follow up on 2: Do you believe that most k-12 students, specifically STEM focused high schoolers, are taught enough about the basics of acoustics? If not, should there be some way to help guide them?

- 'In general, people are not taught enough about acoustics'. Overall not understanding that hearing in terms of acoustics is a mechanical disturbance that actuates the eardrum.
- Yes! In terms of health reasons: help people understand loudness to gauge levels that hurt hearing.
- Understand acoustics to improve communication through speech (gauge loudness and tone)
- An opportunity to bridge science and art

4. Would a project in action, i.e in demo, help a student understand acoustics as opposed to a standard lecture.

- Hands-on demo, especially for young kids in school (elementary, middle, and high)
- Lecture hard to convey technical info
- Sound can get be powerful sensory/auditory learning technique

- Dr. Wallen's experience: In one of UT's outreach days, inside a lab at the ETC (4th floor), speakers were set up such that they excited the resonance of the room. Room has vibration models/characteristic vibration shapes depending on location. This concept was used for the purposes of a demo.

5. Which instrument would come to mind for automation?

- 'tricky'. All instruments are designed for human input. Automation is tough for that reason. Lack of reverb from electromechanical input
- Xylophone family of instruments as first thought (glockenspiel, marimba)
- Avoid wind instruments and brass instruments as pneumatics is too much work.
- Maybe one string instrument with weight causing tension and a roller to create a closed boundary at different locations.
- Guitars, perhaps using spaces between frets for end effector placement (anchor points).

6. How would you go about inputting music into such a robotic system? Easier to map frequencies directly or try to find discretized notes

- Rudimentary level, maybe using a timed loop for one note at a time
- More viable approach is a midi file. Perhaps use music notation software for sheet music, then convert to midi file (online resources for help or music-savvy/experienced team members can help).

7. Software and hardware resources or recommendations? Any materials or end-effectors to avoid based on your experience with acoustics?

- Look for music notation software and work with midi files. Use Arduino or raspberry pi for input/output depending on intensity.
- End-effectors should bio-mimic human motion if possible and avoid slip over hard surfaces.

8. When it comes to seeing an automated instrument as an acoustician, what features would you like to see? How would you ideally expect it to perform?

- 'It would be important to see it play a diverse range of notes. That way you can open up the possibility of playing whatever songs you might want. It should support the option for higher tempo songs. Should be audible without being too loud.

9. Any limitations, caveats, or drawbacks to using robotics to play an instrument as opposed to manually playing an instrument? (any aspect)

- Inability to mimic human fluidity in motion. Robotics still has a long way to go.
- Limited by song selections (audible frequency ranges affect range of notes and thus electromechanical design greatly). A system must be highly complex based on the selection of more elaborate/complex songs.
- May need to stick to one octave and play one note at a time rather than the chord depending on how the system is built.

Appendix D: Customer Needs List

Customer Statements	Interpreted Need	Rating for Relative Importance
I have very little time beforehand to set up for the outreach event.	Setup needs to be quick and easy for the instrument.	4
I had a robot malfunction during an outreach.	Need a reliable system/product that will not fail during operation.	5
We had our outreaches at the local library in a small room. It's about the size of a classroom.	Instrument and all of its associated components must be able to fit into a classroom.	5
We were given around 10-20 minutes to present and do our demonstration.	The demonstration should be a maximum of 20 minutes	5
No funding was given to us. We had to use parts that we already owned to present at outreaches.	The instrument should not require the user to purchase any additional parts/peripherals.	4
Participation from the kids was the most successful outreach strategy we had. For example, we had a dancing robot and we would engage the kids by dancing with them as the robot danced.	The product must engage the audience in some manner and encourage audience participation.	4
It should have good tone quality and not make any annoying noises.	The product needs to sound pleasant	4

It should play a diverse range of notes	Large bandwidth of frequencies	3
It should support the option for higher tempo songs	Product build should support higher bpm music.	4
I would like the instrument to be cheap to own	The product needs to be cheap to make and maintain	3
I hope the instrument is loud so that it can be easily heard	The instruments should be loud enough	3
I should be able to carry it in one hand.	The instruments should not be too heavy	4

Appendix E: House of Quality

Positive O, Negative X			Setup Time(min)	Sucessfully Played Songs(#)	Bounding Dimension(ft)	Minimum Guranteed Playtime (min)	Additional Cost to User($)	Maximum Weight (lb)	Volume (ft^3) -	Minimum Sound Volume (dB) +	Tempo (BPM)	Bandwidth (Hz)	Cost to Make ($)	
Direction of Improvement			↓	→	↓	↓	↓	↓	↓	→	→	→	↓	
What	Size	Instrument must Fit on a desk (5)			O				O					
		Instrument can be Carried by One (4)						O						
	Reliability	Reliable Operation (5)		O		O								
	Time	Quick, Easy Setup (4)	O											
		Short Demonstration Duration (5)									O			
	Performance	Sound Quality (4)										O	X	
		Loudness (3)								O			X	
		Playing Speed (3)									O		X	
		Musical range (4)										O	X	
		Audience Engagement (4)		O		O						O		
	Cost	No Additional Cost to User (4)					O							
		Low Cost to Make and Maintain (3)									X	X	O	
		Target	3	3+	4	5	0	11	5	50	140	1900	250	

Appendix F: Engineering Requirements and List

Category	Demand / Wish	Customer Needs	Design Requirement	Verification / Validation
Size	D	Instrument and all of its associated components must be able to fit on top of a desk.	Volume should be lower than 5 ft^3. The bounding dimension should be lower than 4 ft.	Check CAD file and carry out measurements after final assembly.
Performance	D	Can play multiple different songs	Must be able to play at least 3 songs	Physical prototype input sweep by matching note frequency and timing with intended input (counting errors) and survey for audience approval.
Performance	D	Needs to demonstrate a mechanical engineering principle	User must be able to directly observe / be involved in the development of a script used to play a song	Survey the audience and ensure that at least half are aware of how the instrument is controlled with acoustics and programming
Performance	D	The instrument should make sound in an interesting and educational way	Sound must be produced via mechanical means	Check CAD file for a mechanism that can be attributed to sound production
Size	D	The instruments should not be too heavy	The instrument should weigh less than 11 lb.	Check CAD file and BOM
Reliability	D	Need a reliable	The instrument	Play alternating

		system/product that will not fail during operation.	should be able to play continuously without failure for 5 minutes	high / low note at maximum rate for 5 minutes
Time	D	Setup needs to be quick and easy for the instrument.	The setup time should be less than 3 minutes	Reset system and load demo song and begin playing in less than 3 minutes
Performance	D	The product needs to sound pleasant	Minimum rating of 4 on sound rating customer survey	Customer survey, sound rating 1-5
Performance	W	Large bandwidth of frequencies	The bandwidth should be 1900 hz	Use tuner to measure frequency of lowest and highest notes and verify that they match or exceed the bandwidth.
Performance	W	Product build should support higher bpm music.	The instrument needs to play at minimum of 140 beats per minute	Play notes at maximum rate and ensure bpm matches or exceeds spec.
Performance	D	The instruments should be loud enough	The sound volume should be larger than 50 dB	Use a decibel meter to measure the volume of the instrument while playing a note and verify it matches or exceeds the spec.
Cost	D	The product needs to be cheap to make	The cost of making should be less than 250	Check BOM

		and maintain	dollars	
Cost	D	The instrument should not require the user to purchase any additional parts/peripherals	0 dollars should come out of pocket from the users	The instrument should work as assembled from design documents with no further modifications.

Appendix G: Functional Models

Figure G1: Black Box Model

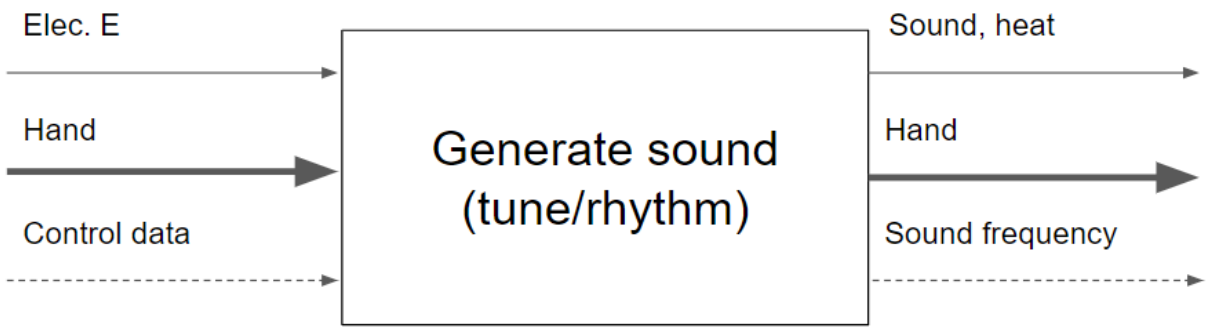
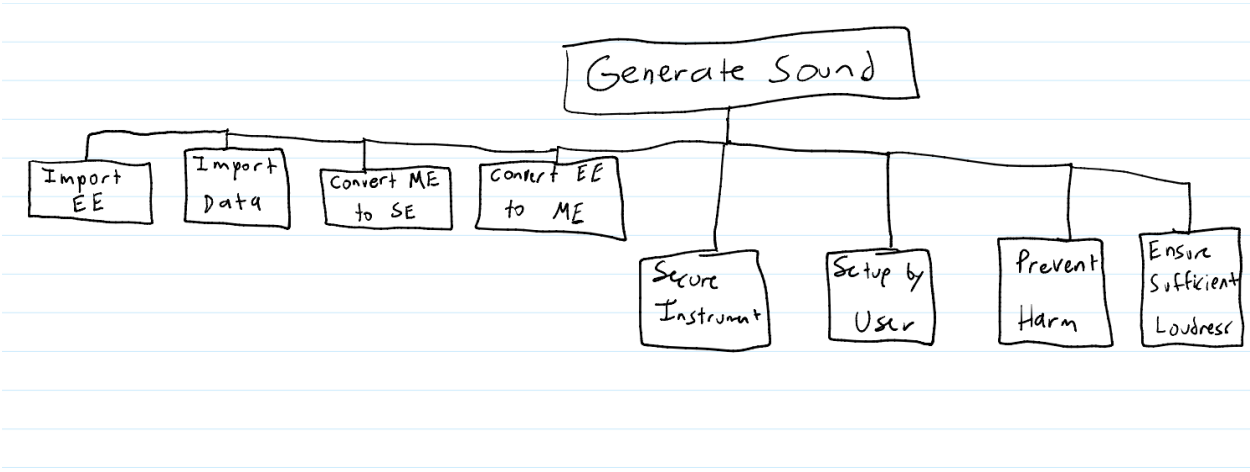


Figure G2: Functional Model

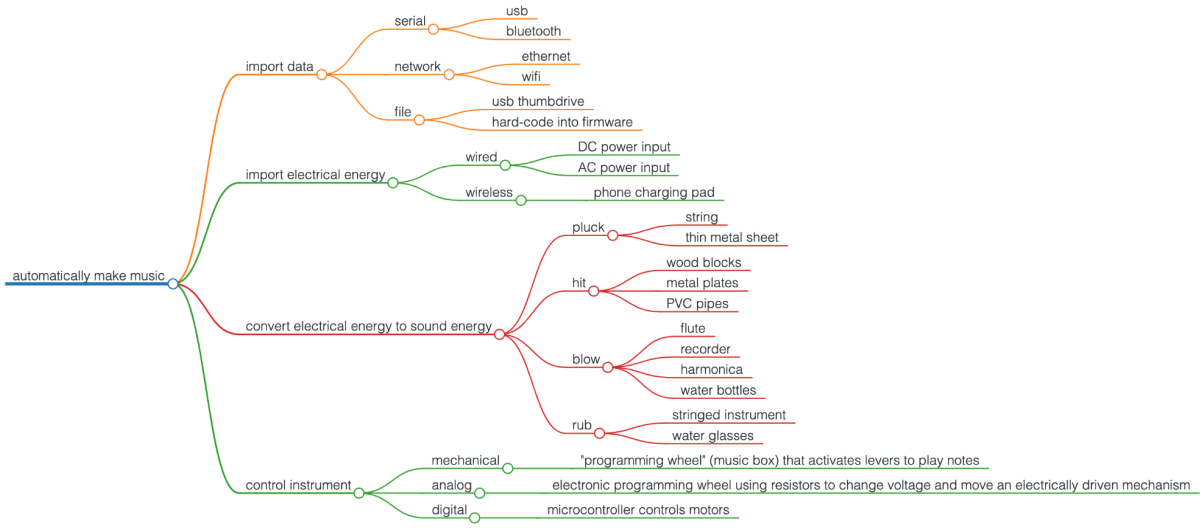


Appendix H: Concept Generation - TRIZ Table

Related Requirements	Want to Improve	What Deteriorates	General Solution	Particular Solution
Volume should be lower than 5 ft^3. Deteriorates: The sound volume should be larger than 50 dB	Volume of stationary object - 8	Power - 21	Using flexible membranes and fine membranes - 30 Principle of universality - 6	30: Use a diaphragm to amplify sound 6: Instead of having multiple actuators for different notes, use a single powerful actuator that can play multiple notes: move around a powerful solenoid/servo for different notes
The instrument needs to play at minimum of 140 beats per minute	Velocity - 9	Harmful actions generated by the design object - 31	Principle of removal - 2 The go between principle - 24 The principle of using color - 32 Principle of rushing through - 21	2: Minimize fast-moving parts and/or pinch points: shields, solenoids and servos (one per note, no motion control) 24: N/A 32: Use bright colors or reflective tape to warn users of possible danger 21: Slow moving pick for string, fast solenoids for fret

The setup time should be less than 3 minutes	Waste of time - 25	Reliability - 27	Principle of preliminary action - 10 Using flexible membranes and fine membranes - 30 Principle of asymmetry - 4	10: Product remains assembled or mostly assembled and ready to use at all times 30: N/A 4: N/A
The instrument should weigh less than 11 lb. Deteriorates: The sound volume should be larger than 50 dB	Weight of stationary object - 2	Power - 21	Principle of dynamism - 15 Principle of periodic action - 19 Use of mechanical vibrations - 18 Principle of turning harm into good - 22	15: Unfoldable sound amplifier cone 19: Store energy in spring with cam before hitting xylophone 18: Take advantage of resonance for string instrument and percussion 22: Use resonance for sound amplification
The setup time should be less than 3 minutes	User friendliness - 33	Level of automation - 38	Principle of segmentation - 1 Principle of discarding and regenerating parts - 34 Principle of equipotentiality - 12 Principle of local quality - 3	1: Modular solenoid housings with 1-step connection 34: N/A 12: Short stroke solenoids - smaller package and less potential to overcome 3: N/A

Appendix I: Concept Generation - Mindmap


automatically make music
import data
serial
usb
bluetooth
network
ethernet
wifi
file
usb thumbdrive
hard-code into firmware
import electrical energy
wired
DC power input
AC power input
wireless
phone charging pad
convert electrical energy to sound energy
pluck
string
thin metal sheet
hit
wood blocks
metal plates
PVC pipes
blow
flute
recorder
harmonica
water bottles
rub
stringed instrument
water glasses
control instrument
mechanical
"programming wheel" (music box) that activates levers to play notes
analog
electronic programming wheel using resistors to change voltage and move an electrically driven mechanism
digital
microcontroller controls motors

Appendix J: Prior Art Research on Converting EE to ME

Figure J1: Stepper Motor Working Principle

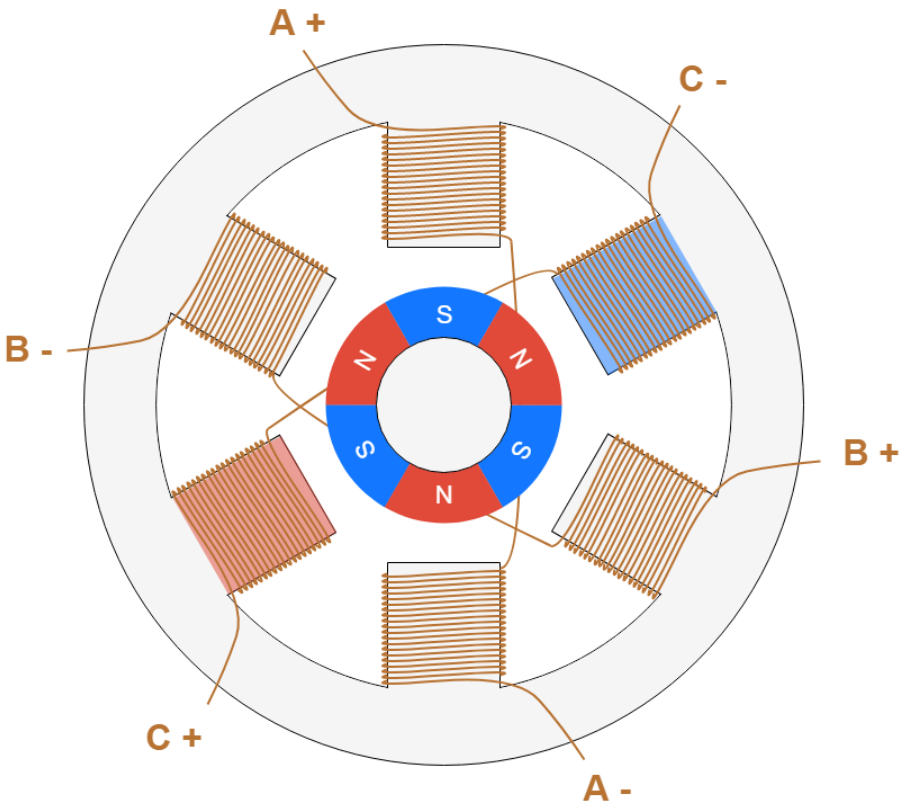


Figure J2: Brushed DC Motor

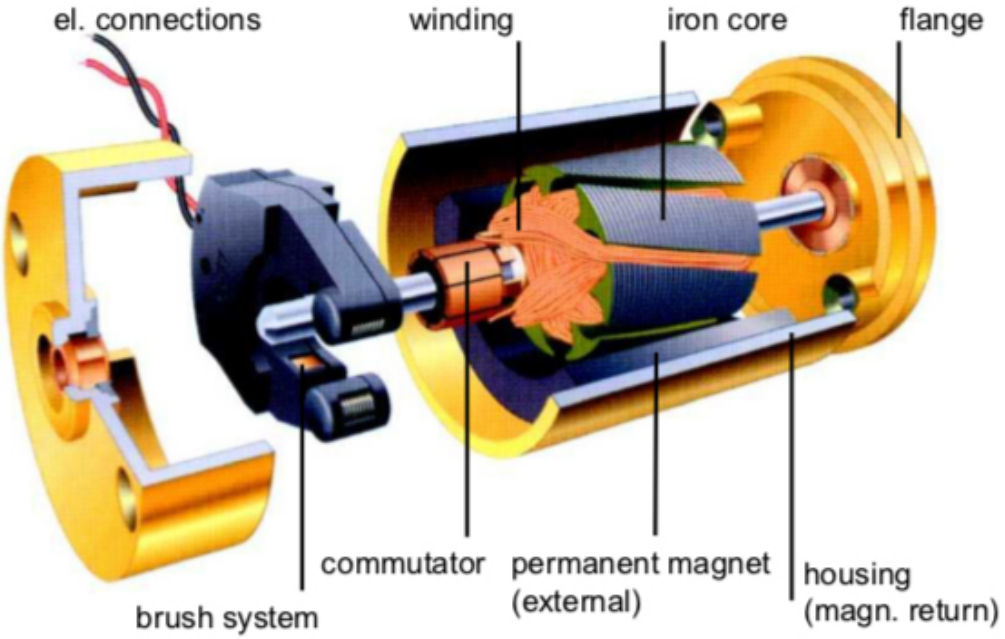


Figure J3: Brushless DC Motor

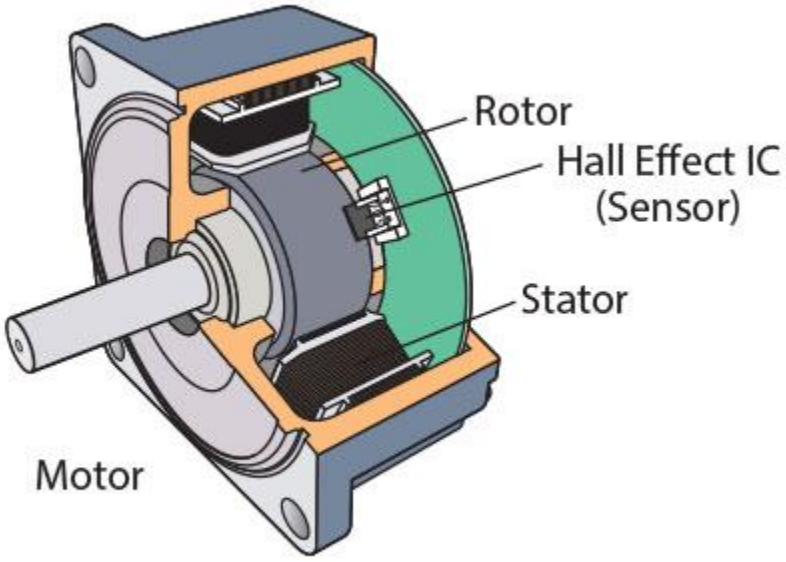


Table J4: Comparison of Brushed and Brushless DC Motor

	Brushed motor	**Brushless motor**
Lifetime	**Short (brushes wear out)**	**Long (no brushes to wear)**
Speed and Acceleration	**Medium**	**High**
Efficiency	**Medium**	**High**
Electrical Noise	**Noisy (bush arcing)**	**Quiet**
Acoustic Noise & Torque Ripple	**Poor**	**Medium (trapezoidal) or good (sine)**
Cost	**Lowest**	**Medium (added electronics)**

Appendix K: Morphological Matrix

Appendix K1: Clean Morph Matrix

Table K1: Morph Matrix

Sub-Functions	**Solutions**		
Import Data	Serial--USB	Serial--Bluetooth	Network--Ethernet
	Network--Wifi	File--USB Drive	File--Hard Code
Import Electrical Energy	Wired--DC Power	Wired--AC Power	Wireless--Phone Charging Pad
Convert Mechanical	Pluck--String	Pluck--Thin Metal	Hit--Wood Blocks

Energy to Sound Energy		Sheet	
	Hit--Metal Plates	Hit--PVC Pipes	Blow--Flute
	Blow--Recorder	Blow--Harmonica	Blow--Water Bottles
	Rub--Stringed Instrument	Rub--Water Glasses	
Convert Electrical Energy to Mechanical Energy	Stepper Motor	Brushed DC Motor	Brushless DC Motor
	Solenoid	(any motor) + Fan	
Convert EE to ME (optional modifiers)	Modular solenoid housings with 1-step connection	Short stroke solenoids - smaller package and less potential to overcome	
Control Instrument	Mechanical--Programming Wheel that Activates Levers to Play Notes		
	Analog--Electronic Programming Wheel using Resistors to Change Voltage and move an Electrically		

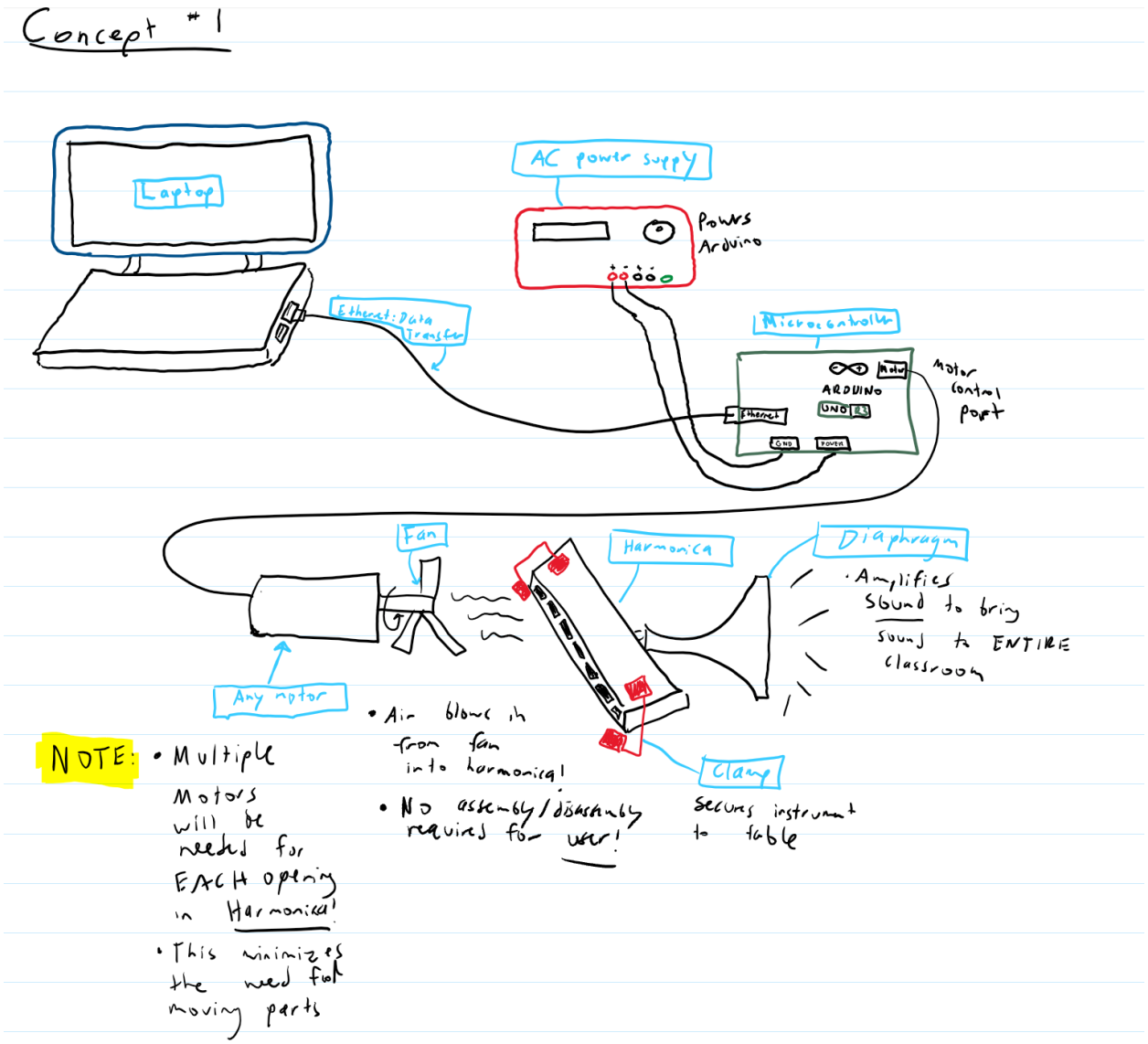
	Driven Mechanism		
	Digital--Microcontroller Controls Motors		
Secure Instrument	Clamp to Table	Weight	Use Glue on Base
Setup by User	Unfoldable elements of product		Product remains assembled or mostly assembled and ready to use at all times
Prevent Harm	Minimize fast-moving parts and/or pinch points: shields, solenoids and servos (one per note, no motion control)	Use bright colors or reflective tape to warn users of possible danger	Slow moving pick for string, fast solenoids for fret
Ensure Sufficient Loudness	Use a diaphragm to amplify sound	Single powerful actuator with motion control to reach multiple notes	Using flexible membranes and fine membranes
	Chosen EE to ME to SE combo sufficient		

Appendix K2: Peter's Concept 1

Table K2: Peter's Automated Harmonica Morph Matrix

Sub-Functions	Solutions		
Import Data	Serial--USB	Serial--Bluetooth	Network--Ethernet
	Network--Wifi	File--USB Drive	File--Hard Code
Import Electrical Energy	Wired--DC Power	Wired--AC Power	Wireless--Phone Charging Pad
Convert Mechanical Energy to Sound Energy	Pluck--String	Pluck--Thin Metal Sheet	Hit--Wood Blocks
	Hit--Metal Plates	Hit--PVC Pipes	Blow--Flute
	Blow--Recorder	Blow--Harmonica	Blow--Water Bottles
	Rub--Stringed Instrument	Rub--Water Glasses	
Convert Electrical Energy to Mechanical Energy	Stepper Motor	Brushed DC Motor	Brushless DC Motor
	Solenoid	(any motor) + Fan	
Convert EE to ME (optional modifiers)	Modular solenoid housings with 1-step connection	Short stroke solenoids - smaller package and less potential to overcome	
Control Instrument	Mechanical--Programming Wheel that Activates Levers to Play Notes		
	Analog--Electronic Programming Wheel using Resistors to Change Voltage and move an Electrically Driven Mechanism		
	Digital--Microcontroller Controls Motors		
Secure Instrument	Clamp to Table	Weight	Use Glue on Base
Setup by User	Unfoldable elements of product		Product remains assembled or mostly assembled and ready to use at all times
Prevent Harm	Minimize fast-moving parts and/or pinch points: shields, solenoids and servos (one per note, no motion control)	Use bright colors or reflective tape to warn users of possible danger	Slow moving pick for string, fast solenoids for fret
Ensure Sufficient Loudness	Use a diaphragm to amplify sound	Single powerful actuator with motion control to reach multiple notes	Using flexible membranes and fine membranes
	Chosen EE to ME to SE combo sufficient		

Figure K2: Peter's Automated Harmonica Design Sketch

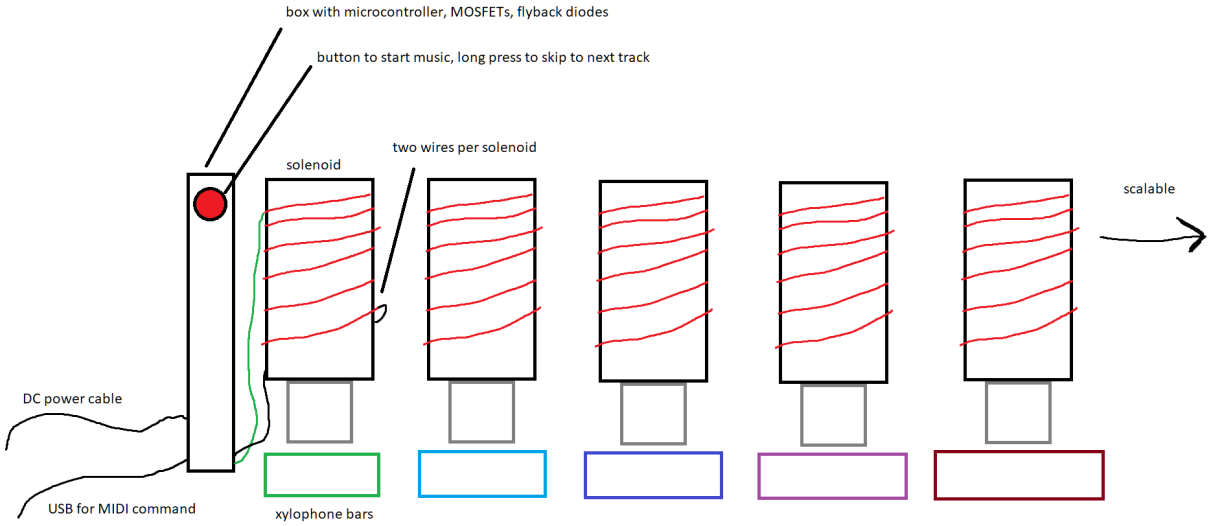


Appendix K3: Andrew's Concept 2

Table K3: Andrew's Solenoid Xylophone Morph Matrix

Sub-Functions	Solutions		
Import Data	Serial--USB	Serial--Bluetooth	Network--Ethernet
	Network--Wifi	File--USB Drive	File--Hard Code
Import Electrical Energy	Wired--DC Power	Wired--AC Power	Wireless--Phone Charging Pad
Convert Mechanical Energy to Sound Energy	Pluck--String	Pluck--Thin Metal Sheet	Hit--Wood Blocks
	Hit--Metal Plates	Hit--PVC Pipes	Blow--Flute
	Blow--Recorder	Blow--Harmonica	Blow--Water Bottles
	Rub--Stringed Instrument	Rub--Water Glasses	
Convert Electrical Energy to Mechanical Energy	Stepper Motor	Brushed DC Motor	Brushless DC Motor
	Solenoid	(any motor) + Fan	
Convert EE to ME (optional modifiers)	Modular solenoid housings with 1-step connection	Short stroke solenoids - smaller package and less potential to overcome	
Control Instrument	Mechanical--Programming Wheel that Activates Levers to Play Notes		
	Analog--Electronic Programming Wheel using Resistors to Change Voltage and move an Electrically Driven Mechanism		
	Digital--Microcontroller Controls Motors		
Secure Instrument	Clamp to Table	Weight	Use Glue on Base
Setup by User	Unfoldable elements of product		Product remains assembled or mostly assembled and ready to use at all times
Prevent Harm	Minimize fast-moving parts and/or pinch points: shields, solenoids and servos (one per note, no motion control)	Use bright colors or reflective tape to warn users of possible danger	Slow moving pick for string, fast solenoids for fret
Ensure Sufficient Loudness	Use a diaphragm to amplify sound	Single powerful actuator with motion control to reach multiple notes	Using flexible membranes and fine membranes
	Chosen EE to ME to SE combo sufficient		

Figure K4: Andrew's Solenoid Xylophone Design Sketch



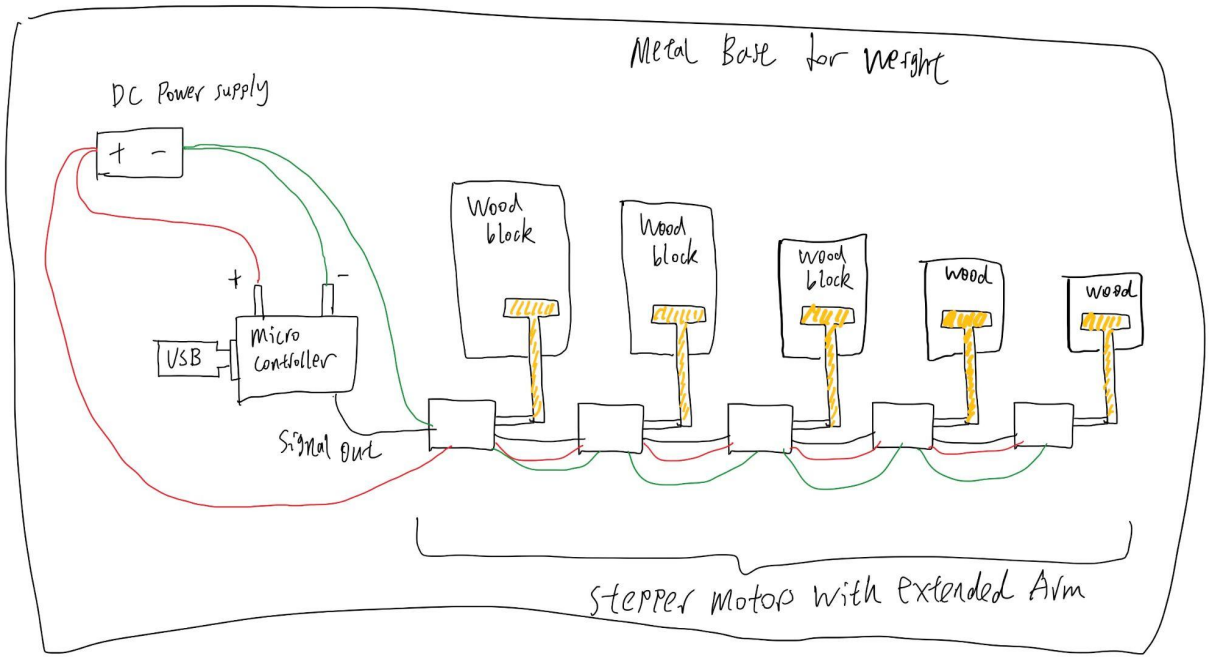
Appendix K4: Yifeng's Concept 3

Table K5: Yifeng's Concept Morph Matrix

Sub-Functions	Solutions		
Import Data	Serial--USB	Serial--Bluetooth	Network--Ethernet
	Network--Wifi	File--USB Drive	File--Hard Code
Import Electrical Energy	Wired--DC Power	Wired--AC Power	Wireless--Phone Charging Pad
Convert Mechanical Energy to Sound Energy	Pluck--String	Pluck--Thin Metal Sheet	Hit--Wood Blocks
	Hit--Metal Plates	Hit--PVC Pipes	Blow--Flute
	Blow--Recorder	Blow--Harmonica	Blow--Water Bottles
	Rub--Stringed Instrument	Rub--Water Glasses	
Convert Electrical Energy to Mechanical Energy	Stepper Motor	Brushed DC Motor	Brushless DC Motor
	Solenoid	(any motor) + Fan	
Convert EE to ME (optional modifiers)	Modular solenoid housings with 1-step connection	Short stroke solenoids - smaller package and less potential to overcome	
Control Instrument	Mechanical--Programming Wheel that Activates Levers to Play Notes		
	Analog--Electronic Programming Wheel using Resistors to Change Voltage and move an Electrically Driven Mechanism		
	Digital--Microcontroller Controls Motors		
Secure Instrument	Clamp to Table	Weight	Use Glue on Base
Setup by User	Unfoldable elements of product		Product remains assembled or mostly assembled and ready to use at all times
Prevent Harm	Minimize fast-moving parts and/or pinch points: shields, solenoids and servos (one per note, no motion control)	Use bright colors or reflective tape to warn users of possible danger	Slow moving pick for string, fast solenoids for fret
Ensure Sufficient	Use a diaphragm to	Single powerful	Using flexible

Loudness	amplify sound	actuator with motion control to reach multiple notes	membranes and fine membranes
	Chosen EE to ME to SE combo sufficient		

Figure K6: Yifeng's Concept Design Sketch



Appendix K5: Talal's Concept 4

Table K7: Talal's Concept Morph Matrix

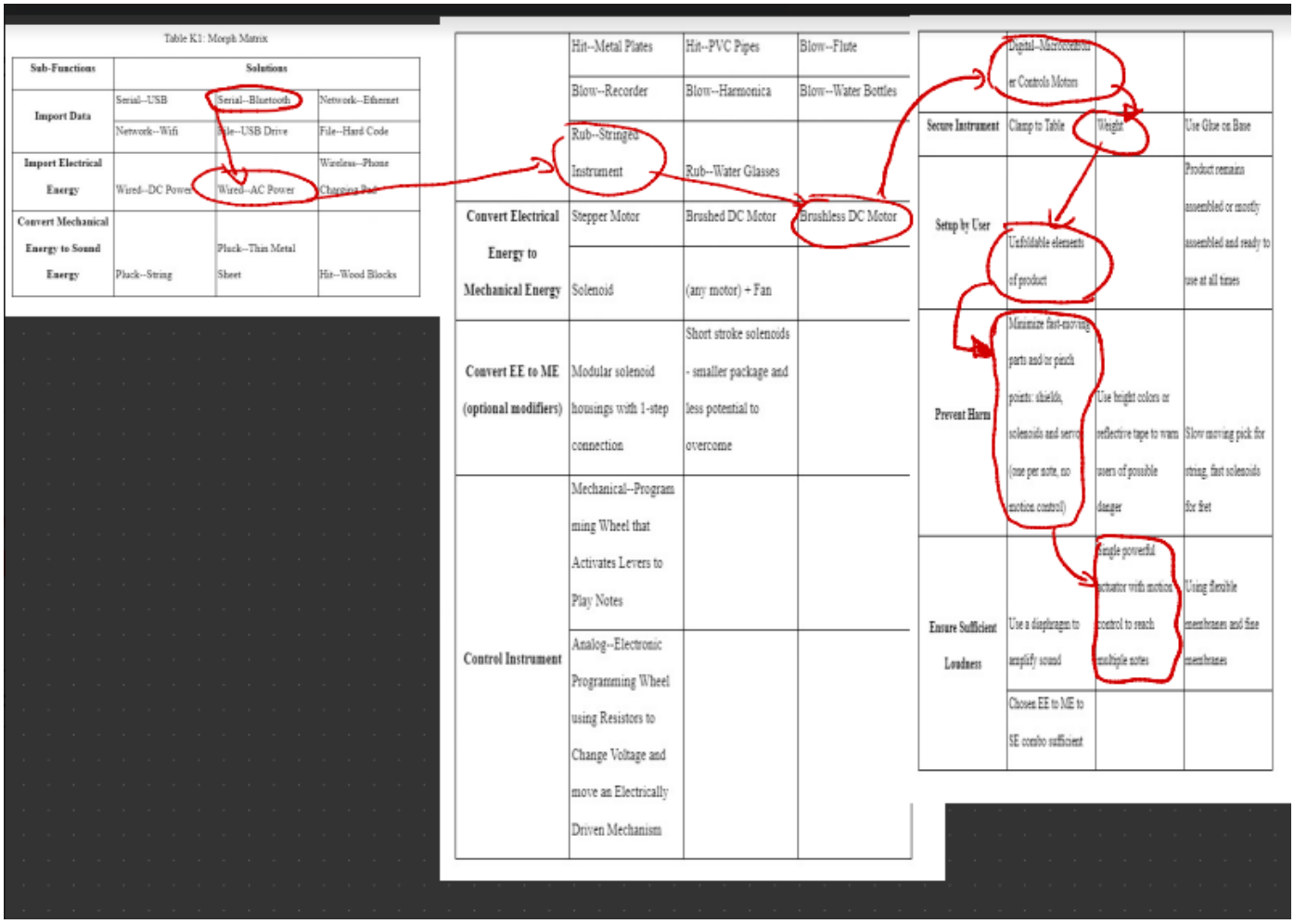


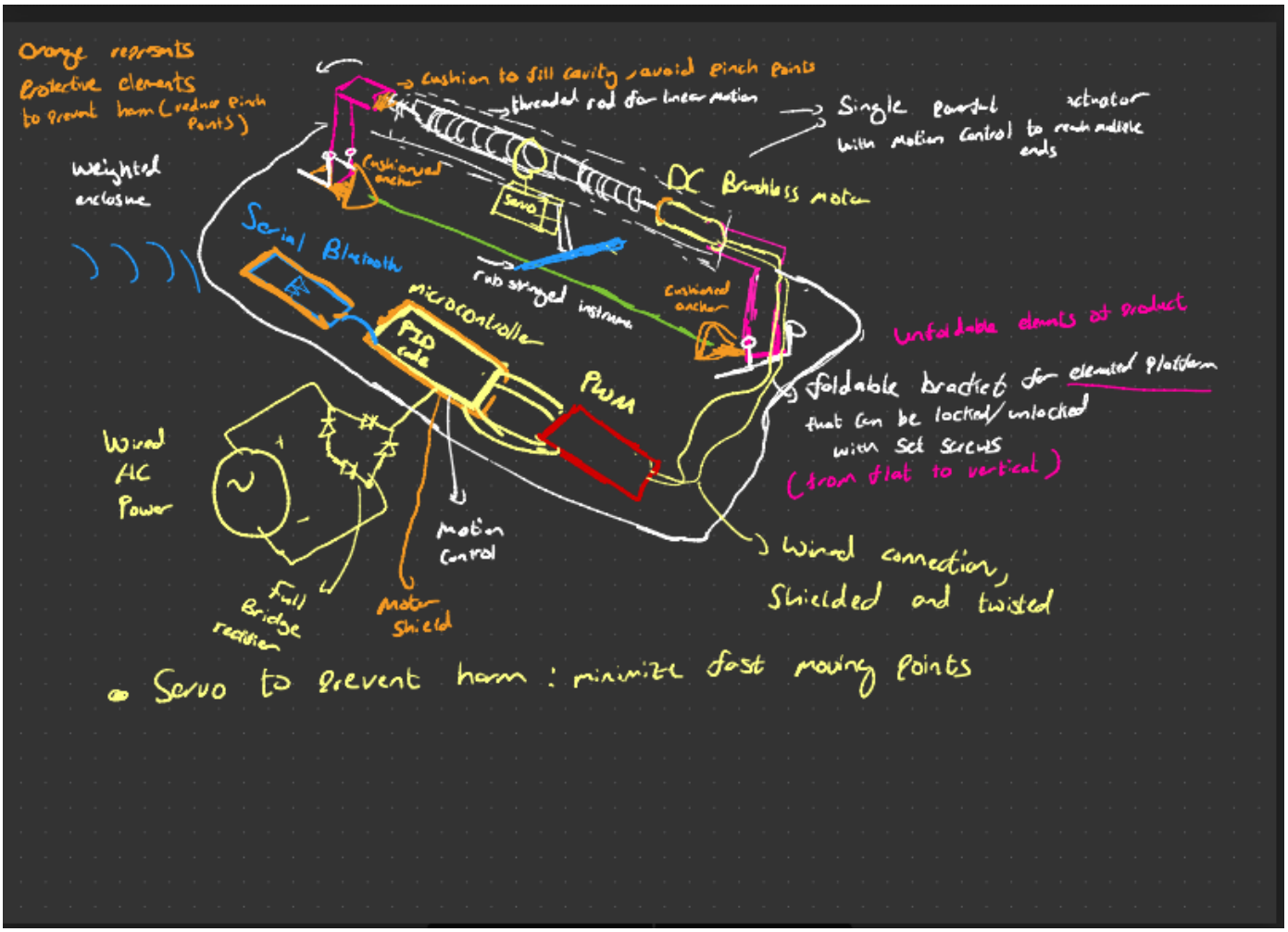
Table K1: Morph Matrix

Sub-Functions	Solutions		
Import Data	Serial--USB	Serial--Bluetooth	Network--Ethernet
	Network--Wifi	File--USB Drive	File--Hard Code
Import Electrical Energy	Wired--DC Power	Wired--AC Power	Wireless--Phone Charging Pad
Convert Mechanical Energy to Sound Energy	Pluck--String	Pluck--Thin Metal Sheet	Hit--Wood Blocks

	Hit--Metal Plates	Hit--PVC Pipes	Blow--Flute
	Blow--Recorder	Blow--Harmonica	Blow--Water Bottles
	Rub--Stringed Instrument	Rub--Water Glasses	
Convert Electrical Energy to Mechanical Energy	Stepper Motor	Brushed DC Motor	Brushless DC Motor
	Solenoid	(any motor) + Fan	
Convert EE to ME (optional modifiers)	Modular solenoid housings with 1-step connection	Short stroke solenoids - smaller package and less potential to overcome	
Control Instrument	Mechanical--Programming Wheel that Activates Levers to Play Notes		
	Analog--Electronic Programming Wheel using Resistors to Change Voltage and move an Electrically Driven Mechanism		

	Digital--Microcontroller Controls Motors		
Secure Instrument	Clamp to Table	Weight	Use Glue on Base
Setup by User	Unfoldable elements of product		Product remains assembled or mostly assembled and ready to use at all times
Prevent Harm	Minimize fast-moving parts and/or pinch points: shields, solenoids and servos (one per note, no motion control)	Use bright colors or reflective tape to warn users of possible danger	Slow moving pick for string, fast solenoids for fret
Ensure Sufficient Loudness	Use a diaphragm to amplify sound	Single powerful actuator with motion control to reach multiple notes	Using flexible membranes and fine membranes
	Choses EE to ME to SE combo sufficient		

Figure K7: Talal's Concept Design Sketch

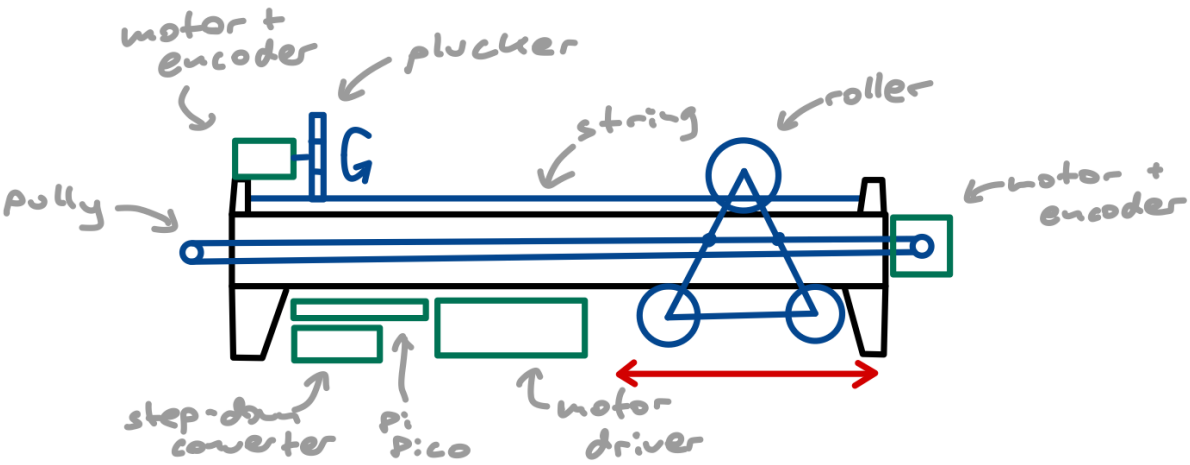


Appendix K6: Elliott's Concept 5

Table K8: Elliott's Concept Morph Matrix

Sub-Functions	Solutions		
Import Data	Serial--USB	Serial--Bluetooth	Network--Ethernet
	Network--Wifi	File--USB Drive	File--Hard Code
Import Electrical Energy	Wired--DC Power	Wired--AC Power	Wireless--Phone Charging Pad
Convert Mechanical Energy to Sound Energy	Pluck--String	Pluck--Thin Metal Sheet	Hit--Wood Blocks
	Hit--Metal Plates	Hit--PVC Pipes	Blow--Flute
	Blow--Recorder	Blow--Harmonica	Blow--Water Bottles
	Rub--Stringed Instrument	Rub--Water Glasses	
Convert Electrical Energy to Mechanical Energy	Stepper Motor	Brushed DC Motor	Brushless DC Motor
	Solenoid	(any motor) + Fan	
Convert EE to ME (optional modifiers)	Modular solenoid housings with 1-step connection	Short stroke solenoids - smaller package and less potential to overcome	
Control Instrument	Mechanical--Programming Wheel that Activates Levers to Play Notes		
	Analog--Electronic Programming Wheel using Resistors to Change Voltage and move an Electrically Driven Mechanism		
	Digital--Microcontroller Controls Motors		
Secure Instrument	Clamp to Table	Weight	Use Glue on Base
Setup by User	Unfoldable elements of product		Product remains assembled or mostly assembled and ready to use at all times
Prevent Harm	Minimize fast-moving parts and/or pinch points: shields, solenoids and servos (one per note, no motion control)	Use bright colors or reflective tape to warn users of possible danger	Slow moving pick for string, fast solenoids for fret
Ensure Sufficient Loudness	Use a diaphragm to amplify sound	Single powerful actuator with motion control to reach multiple notes	Using flexible membranes and fine membranes
	Chosen EE to ME to SE combo sufficient		

Figure K8: Elliott's Concept Design Sketch



Appendix L: Pugh Charts

Table L1: Pugh Chart with Automated Harmonica as datum

Pugh Chart with Concept 1 (Automated Harmonica) as Datum

	Concepts				
	1	2	3	4	5
Criteria	Automated Harmonica	Solenoid Xylophone	Stepper Xylophone	DC Motor Controlled Violin Bow	Roller-Plucker
Cost ($)	0	1	-1	-1	-1
Size (in^3)	0	1	-1	1	1
Mass (lb)	0	-1	-1	-1	1
Bandwidth (Hz)	0	1	1	1	1
Educational Value	0	1	1	1	1
Sum of +	0	4	2	3	4
Sum of -	0	1	3	2	1
Total	0	3	-1	1	3

Table L2: Pugh Chart with Solenoid Xylophone as datum

Pugh Chart with Concept 2 (Solenoid Xylophone) as Datum

	Concepts				
	1	2	3	4	5
Criteria	Automated Harmonica	Solenoid Xylophone	Stepper Xylophone	DC Motor Controlled Violin Bow	Roller-Plucker
Cost ($)	-1	0	-1	-1	-1
Size (in^3)	-1	0	-1	1	1
Mass (lb)	1	0	1	1	1
Bandwidth (Hz)	-1	0	0	-1	-1
Educational Value	-1	0	-1	1	1
		0			
Sum of +	1	0	1	3	3
Sum of -	4	0	3	2	2
Total	-3	0	-2	1	1

Table L3: Pugh Chart with Roller-Plucker as datum

Pugh Chart with Concept 3 (Roller-Plucker) as Datum					
	Concepts				
	1	2	3	4	5
Criteria	Automated Harmonica	Solenoid Xylophone	Stepper Xylophone	DC Motor Controlled Violin Bow	Roller-Plucker
Cost ($)	-1	1	-1	-1	0
Size (in^3)	-1	-1	-1	-1	0
Mass (lb)	-1	-1	-1	-1	0
Bandwidth (Hz)	-1	1	1	1	0
Educational Value	-1	-1	-1	-1	0
					0
Sum of +					0
Sum of -					0
Total	-5	-1	-3	-3	0

Appendix M: Criteria Evaluation & Back-of-the-envelope calculations

Based on standard instrument lengths, one can get a rough estimate of the size/physical volume of the system by neglecting electronics and focusing mostly on mechanical components. All components have lengths rounded up and it is assumed that the volume computation is to be treated like a cuboid-like structure.

Table M1: Cost Analysis for Concept 1

3D printing material costs not accounted for

Part	Cost [USD]
DC Motor x8	32
MOSFET x8	4
Diode x8	2
Arduino Uno or equivalent	30
12V DC power supply	7

Wire, connectors	9
Diaphragm	5
Fan impellers, clamp (3D print)	0
Total:	129

Table M2: Size Analysis for Concept 1

Width [in]:	12
Depth [in]:	18
Height [in]:	8
Total Volume, converted to ft^3:	1

Table M3: Cost Analysis for Concept 2

Part	Cost [USD]
Hand-wound solenoid x8 (magnet wire, steel rod, magnets, 3D printed housings)	25
Plywood for laser cutting	6
MOSFET x8	4
Diode x8	2
Arduino Uno or equivalent	30
12V DC power supply	7
Wire, connectors	9
Total:	83

Table M4: Size Analysis for Concept 2

Width [in]:	24

Depth [in]:	6
Height [in]:	10
Total Volume, converted to ft^3	0.833

Table M5: Cost Analysis for Concept 3

Part	Cost [USD]
Stepper Motor x 5	70
Wood Blocks	20
Arduino Uno Microcontroller	30
Aluminum Extension Arms	20
Misc. Electronics (wires, etc…)	10
Total:	150

Table M6: Size Analysis for Concept 3

Width [in]:	24
Depth [in]:	7
Height [in]:	11
Total Volume, converted to ft^3	1.07

Table M7: Cost Analysis for Concept 4

Information retrieved from the cheapest Amazon listings. Emphasis placed on finding close matches to needed electronics. The violin string price was not obtained on a price per length basis but rather a pre-packaged set. Components related to auxiliary components were not accounted for.

Part	Cost [USD]

Threaded Linear Rail Platform	65
High Torque Metal Gear Servo	25
L298 Motor Drive Controller Board Module	7
Arduino Uno Microcontroller	30
HC-05 Arduino Wireless Bluetooth Receiver	11
Tamiya 54611 brushless RC Motor	60
Steel Folding Bracket Hinge set	15
Violin String set	13
Wires, diodes, & resistors	5
Violin bow	26
Total:	244

Table M8: Size Analysis for Concept 4

Width [in]:	10
Depth [in]:	8
Height [in]:	12
Total Volume , converted to ft^3	0.556

Table M9: Cost Analysis for Concept 5

Part	Cost [USD]
3D Prints	5
M3 Hardware	3
Motors	50
Pi Pico	5

Motor Driver	7
Power Supply	6
GT2 Belt	3
Bearings	4
Guitar String	3
Misc. Guitar Hardware	8
Total:	94

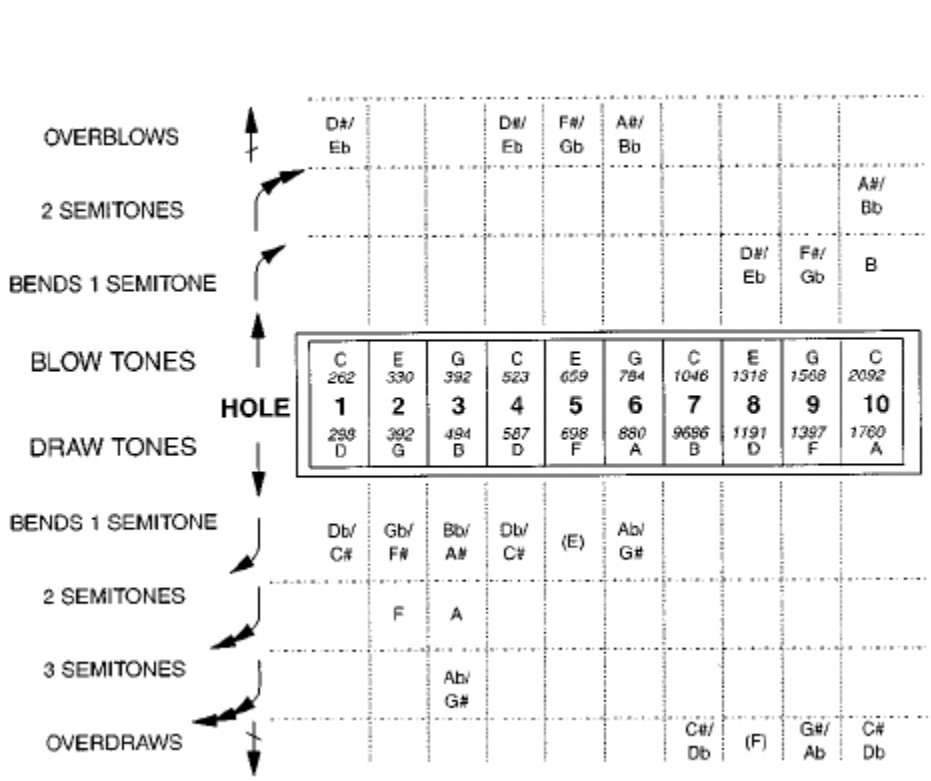
Table M10: Size Analysis for Concept 5

Width [in]:	12
Depth [in]:	4
Height [in]:	4
Total Volume , converted to ft^3	0.11

Bandwidth Estimates

Catalogs and websites were traversed to find statistical averages of acoustic volume and bandwidth ratings. For any documented instrument, we are operating under the assumption that it is human-perturbed to avoid having to account for electric power input.

Figure M1: blues harp diatonic bar (harmonica) with labeled notes and frequencies



The bandwidth is simply the max blow tone frequency minus the lowest blow tone frequency

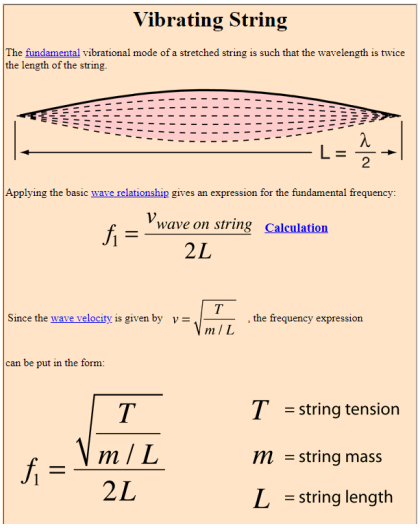
https://www.researchgate.net/figure/Notes-and-approximate-frequencies-of-a-ten-hole-diatonic-har_fig2_13714801

Table M11: Bandwidth (Frequency) estimates of all 5 concepts

Concept	Bandwidth (Hz)
1	1830
2	3837
3	3837

4	1863.39
5	2129

Figure M2: Fundamental vibrational mode frequency of tensioned string (closed-closed anchor points)



http://hyperphysics.phy-astr.gsu.edu/hbase/Waves/string.html

Equation used for concept 5

Figure M3: bandwidth estimation of concept 5 (Roller-Plucker) using guitar string properties

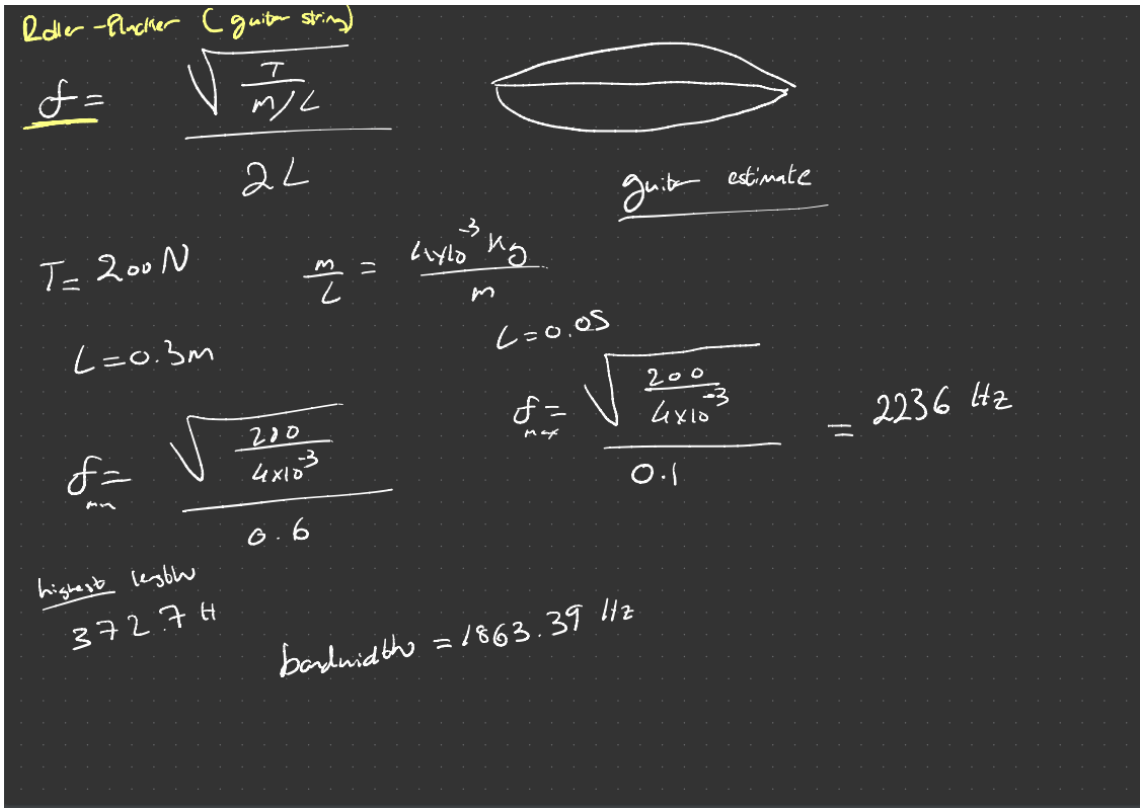
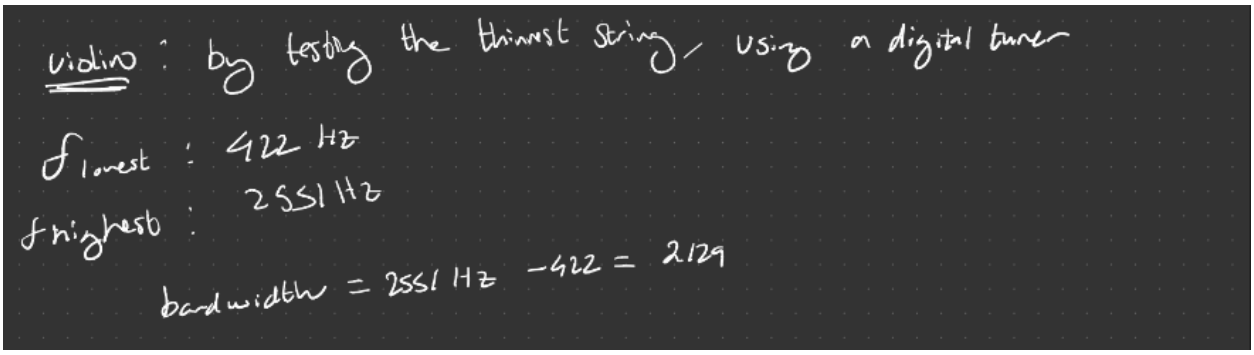


Figure M4: bandwidth estimation of concept 5 (Roller-Plucker) using thinnest violin string. Frequency measured using digital tuner on phone.



Educational Value survey

To quantify the educational value associated with each design, we surveyed 15 highschool students and asked each to rank our five final designs based on how interesting /

informative they found the mechanical mechanism responsible for sound production. The results can be seen in the table below.

Table M12: Survey Results

Student	Concept 1	Concept 2	Concept 3	Concept 4	Concept 5
1	3	2	5	4	1
2	2	4	1	3	5
3	1	5	2	4	3
4	1	4	3	2	5
5	1	3	4	2	5
6	5	4	2	3	1
7	5	1	3	2	4
8	1	2	3	4	5
9	2	4	3	5	1
10	2	1	5	4	3
11	1	3	4	2	5
12	2	3	1	5	4
13	5	2	1	3	4
14	4	1	3	5	2
15	1	4	2	3	5
Total:	36	43	42	51	53

Mass Estimates

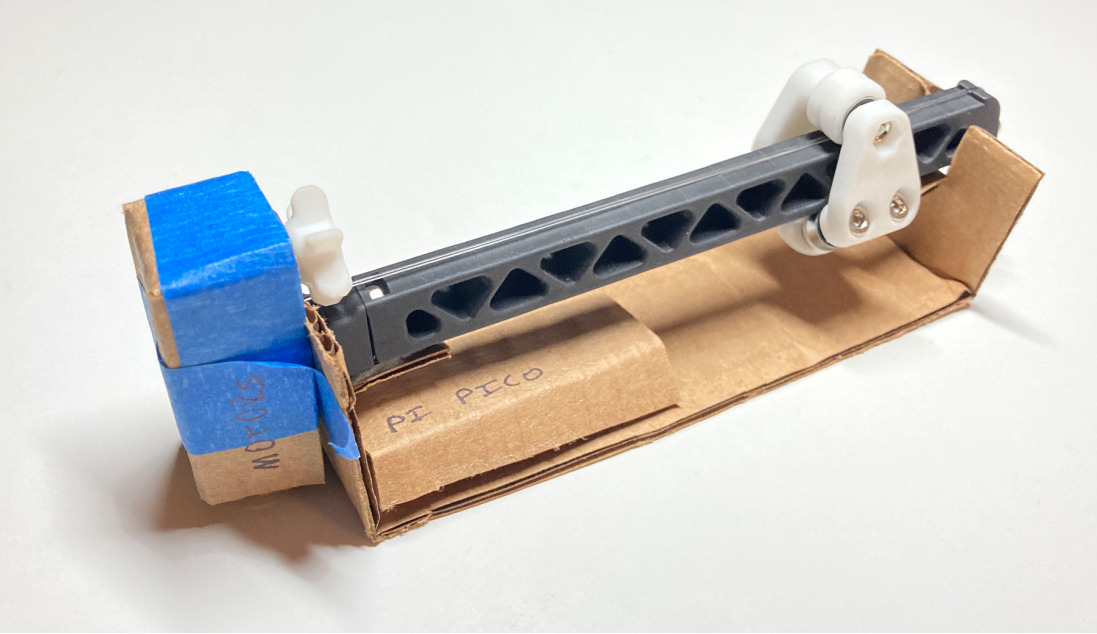
Based on analyzing the mass of the instrument and the heaviest actuators, a rough estimate of the masses is obtained and tabulated in table M13

Table M13: Mass estimates of all 5 concepts

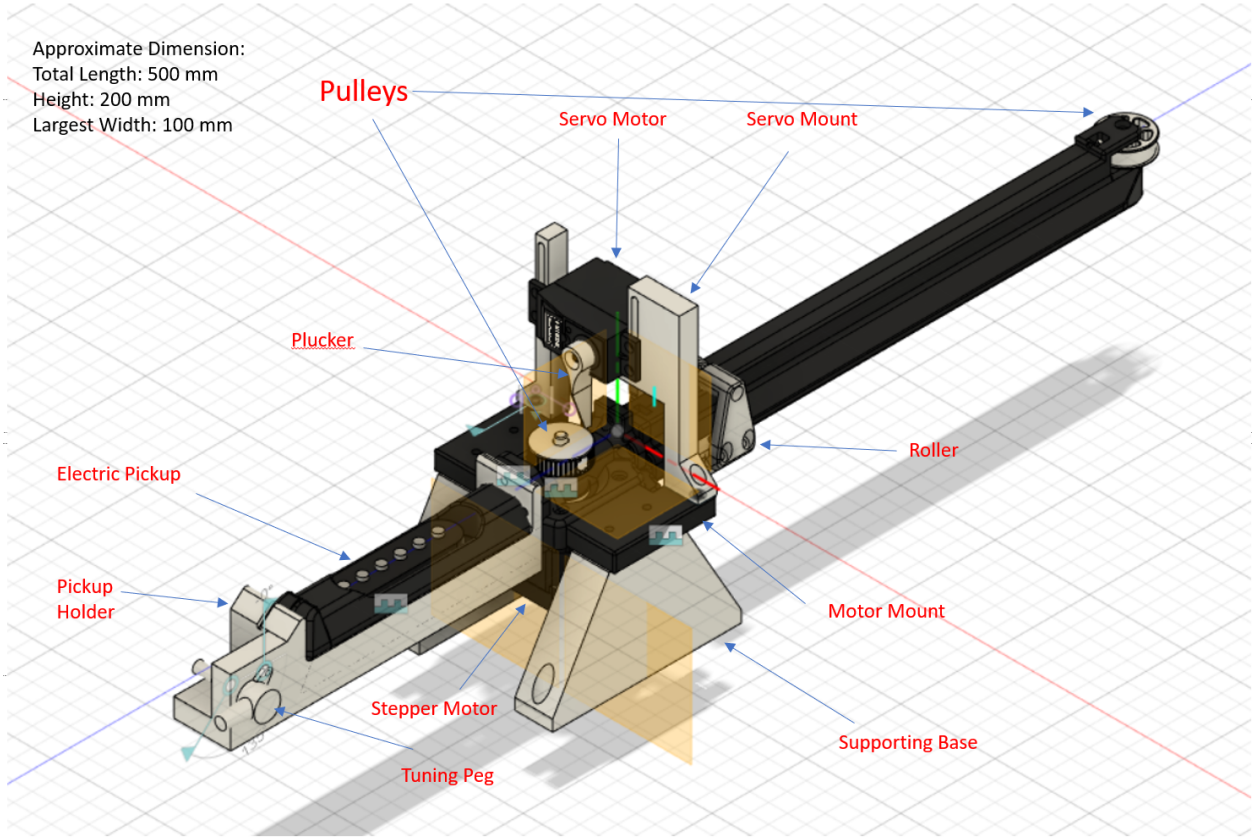
Concept	Mass [lb]
1	4.1
2	10.6
3	9.2
4	7.5
5	3.3

Appendix N: Low-Res Prototype

Figure N1: Low-Res Prototype



Appendix O: Leading Concept



Appendix P: FMEA

FMEA for Roller Plucker

Initial FMEA | FMEA after assembly and testing

				Current Situation						*Improved Situation*				
Failure Location/ Component/Category	**Failure Mode**	**Failure Effect**	**Failure Cause**	**Current Detection Steps**	**S**	**O**	**D**	**RPN**	**Suggested Remedial Measures**	**Revision**	**S**	**O**	**D**	**RPN**
String	String breakage	No more music, possible harm	Overtension, string fatigue, wear due to picker, rigid picker catches string	Tuning using spectrum analyzer estimates proper tension	8	4	4	128	Flexible picker	0.8 mm thickness picker with ~20 mm length	8	2	4	64
Motion axis and control	Overheated stepper	No more music, possible permanent structural deformation, possible harm	Current too high results coil heating motor up near permanent magnets' Curie point, causing lost steps	Thermal camera	8	4	4	128	Minimize stepper current	Stepper current was left as-is, default current did not result in temps above 35°C	8	2	4	64

Motion axis and control	Lost steps	Wrong notes played, possible damage due to roller overrun into end	Exceeded pull-out torque of stepper	Basic F=ma and torque calculations, set appropriate stepper driver current for required torque	7	4	2	56	Increase stepper driver current	Stepper current was left as-is, default current did not lose steps	7	2	2	28
Motion axis and control	Finger jammed between belt and pulley	No more music, possible harm	Intentional misuse by user	See if someone has put their finger in the motion axis	6	2	2	24	Add cover over belt	Left as-is, RPN judged to be low enough	6	2	2	24
Neck	Neck deforms	String loses tension, causing change in tone	Creeping of PLA parts	Check string is parallel with neck	6	6	2	72	Change neck material to PETG	Increased neck thickness to lower stress and reduce creep rate	6	2	2	24
Picker	Picker falls off of servo horn	No more music	Plastic spline failure	Check if string is picked properly	6	2	2	24	Add screw to picker	Screw through picker into M3 threaded servo output	6	1	2	12

										spline				
Stepper driver	Explosion	No more music, smoke	LC voltage spike	Verify stepper movement after startup	7	3	2	42	Add high ESR capacitor in parallel with stepper driver supply	Added 100 uF capacitor in parallel with stepper driver supply	7	1	2	14
Controller	Endstop switch pressed during power-on	Failure to boot (no music)	Microcontroller bug	Check if program works	6	4	2	48	Change behavior in firmware to avoid leaving carriage at endstop	Carriage is now moved away from endstop after homing	6	2	2	24
Motion axis and control	Pulley runs into string	String tuning changes, wrong notes	Pulley slips and climbs up stepper shaft	Listen for correct notes	6	8	2	96	Set screw on pulley	Added tape to shim pulley bore	6	2	2	24

Appendix Q: Experimentation Data and Analysis

Figure Q1:

Data from Factorial Experiment

Trial	X1 -- A (Current)	x2 -- B (Voltage)	x3 -- C (Carriage Mass)	x1-x2	x2-x3	x1-x3	Y1 -- Max acceleration before skipped steps (in m/s^2)	Y2 -- Stepper Motor Surface Temp 10 min. (°C)
1	-1.00	-1.00	-1.00	1.00	1.00	1.00	20.00	35.7
1	1.00	-1.00	-1.00	-1.00	1.00	-1.00	35.00	45.9
1	-1.00	-1.00	1.00	1.00	-1.00	-1.00	15.00	37.9
1	1.00	-1.00	1.00	-1.00	-1.00	1.00	27.50	48.6
1	-1.00	1.00	-1.00	-1.00	-1.00	1.00	25.00	36.9
1	1.00	1.00	-1.00	1.00	-1.00	-1.00	40.00	48.4
1	-1.00	1.00	1.00	-1.00	1.00	-1.00	20.00	38.0
1	1.00	1.00	1.00	1.00	1.00	1.00	32.50	50.4
2	-1.00	-1.00	-1.00	1.00	1.00	1.00	20.00	36.1
2	1.00	-1.00	-1.00	-1.00	1.00	-1.00	32.50	46.5
2	-1.00	-1.00	1.00	1.00	-1.00	-1.00	15.00	38.2
2	1.00	-1.00	1.00	-1.00	-1.00	1.00	30.00	49.2
2	-1.00	1.00	-1.00	-1.00	-1.00	1.00	27.50	37.9
2	1.00	1.00	-1.00	1.00	-1.00	-1.00	42.50	48.5
2	-1.00	1.00	1.00	-1.00	1.00	-1.00	20.00	38.6
2	1.00	1.00	1.00	1.00	1.00	1.00	35.00	51.2
3	-1.00	-1.00	-1.00	1.00	1.00	1.00	22.50	36.6
3	1.00	-1.00	-1.00	-1.00	1.00	-1.00	37.50	46.2
3	-1.00	-1.00	1.00	1.00	-1.00	-1.00	17.50	38.0
3	1.00	-1.00	1.00	-1.00	-1.00	1.00	30.00	49.4
3	-1.00	1.00	-1.00	-1.00	-1.00	1.00	25.00	37.7
3	1.00	1.00	-1.00	1.00	-1.00	-1.00	45.00	49.1
3	-1.00	1.00	1.00	-1.00	1.00	-1.00	17.50	39.0
3	1.00	1.00	1.00	1.00	1.00	1.00	32.50	50.5

	X1 Current Low Value(A)	X1 Current High Value(A)	X2 Voltage Low Value(V)	X2 Voltage High Value(V)	X3 Carriage Mass Low Value(g)	X3 Carriage Mass High Value(g)
	0.50	1.00	6.00	12.00	25.00	50.00

Figure Q2:

Thermal Image of Stepper Motor

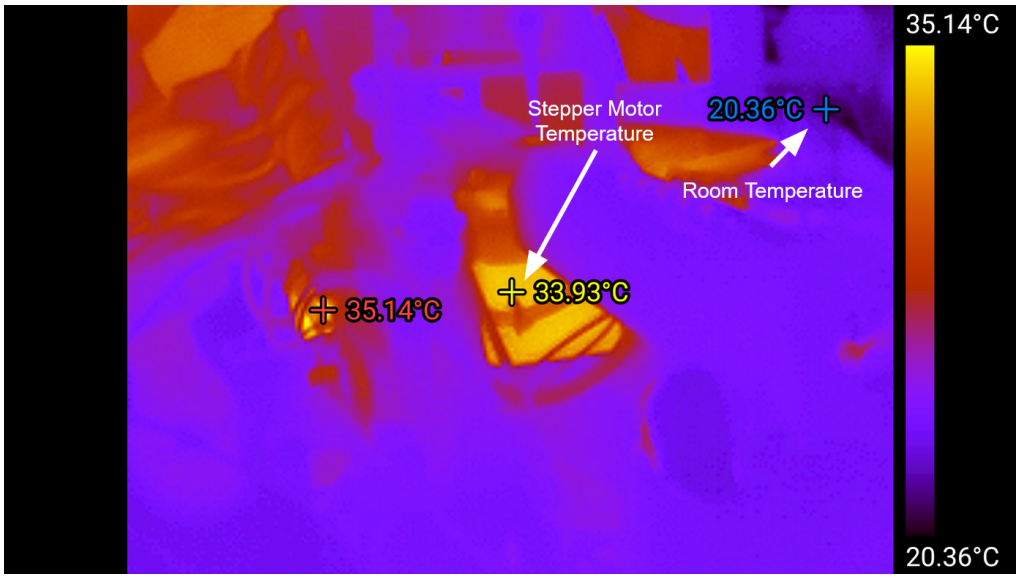


Figure Q3:

Cube Plot for Maximum Acceleration of the Stepper Motor (m/s^2)

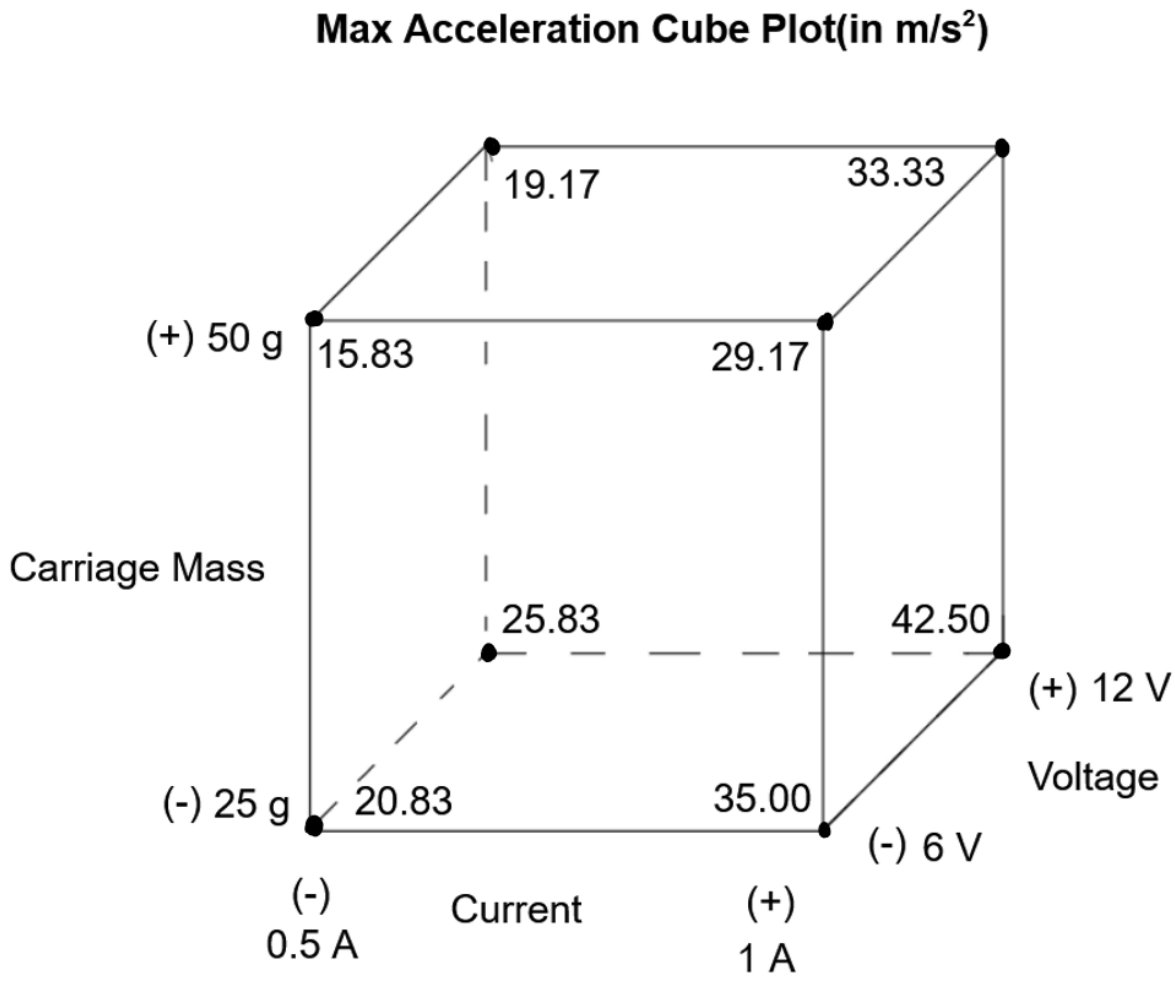


Figure Q4:

Cube Plot for Temperature of the Stepper Motor (℃)

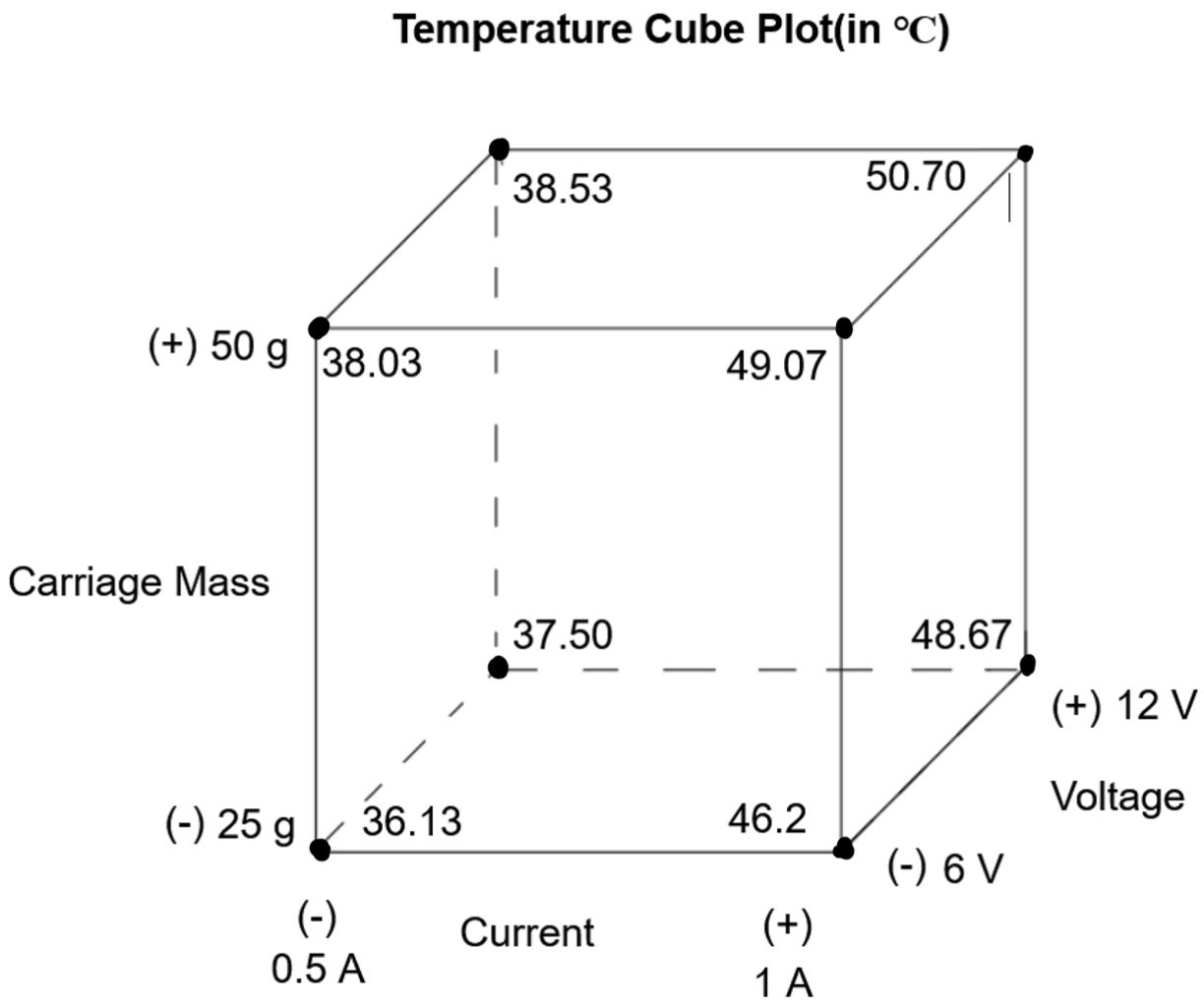


Figure Q5:

Main Effects Plot for Current vs. Max Acceleration

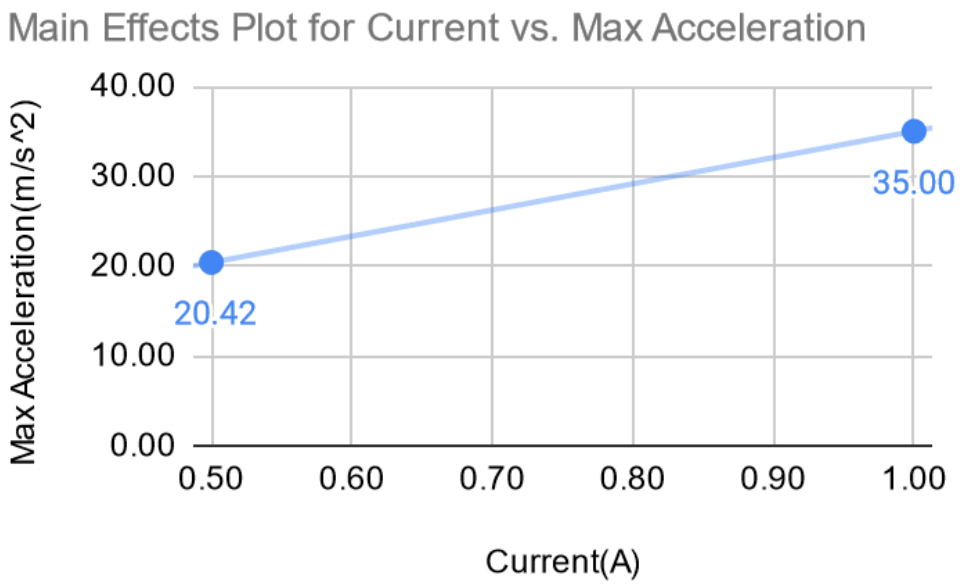


Figure Q6:

Main Effects Plot for Voltage vs. Max Acceleration

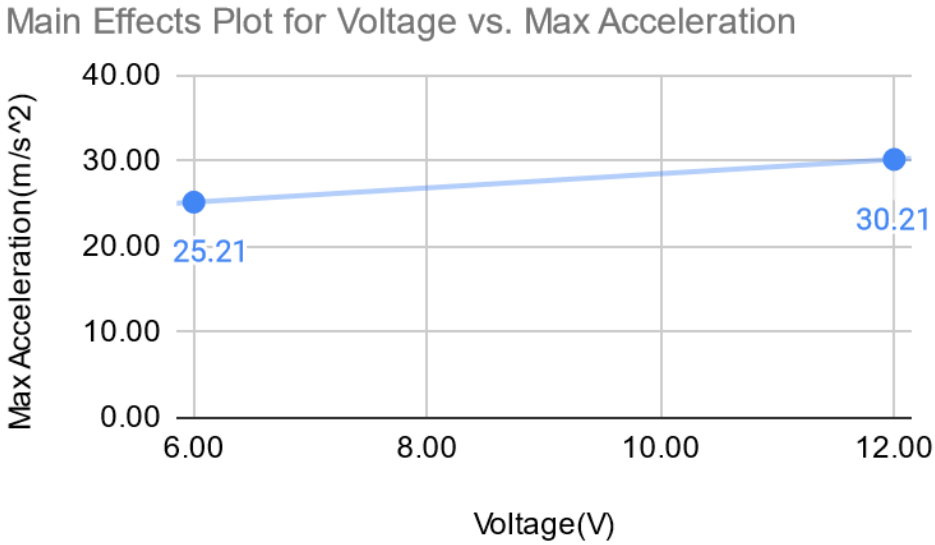


Figure Q7:

Main Effects Plot for Carriage Mass vs. Max Acceleration

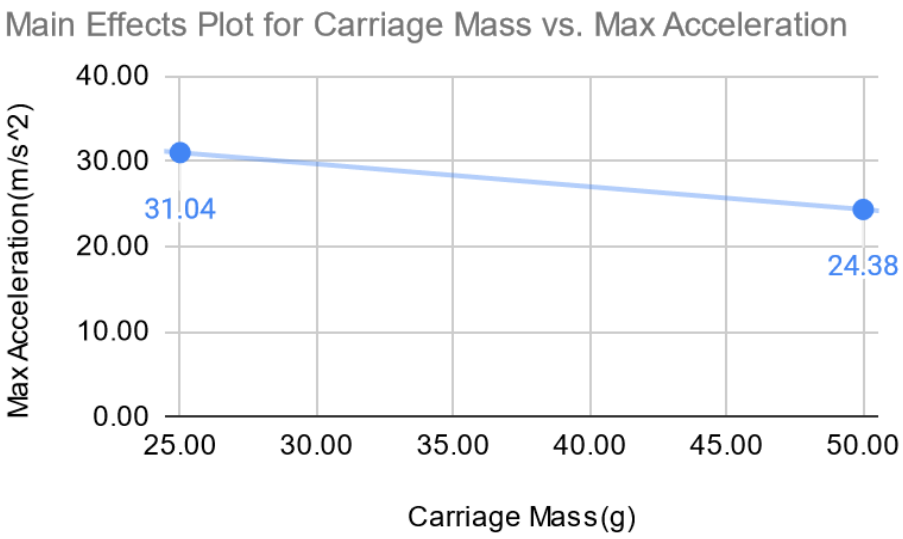


Figure Q8:

Main Effects Plot for Current vs. Temperature

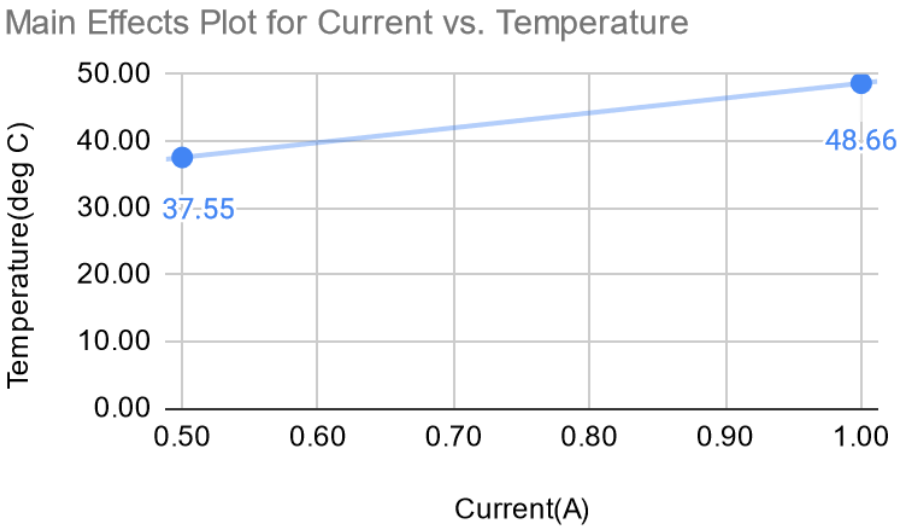


Figure Q9:

Main Effects Plot for Voltage vs. Temperature

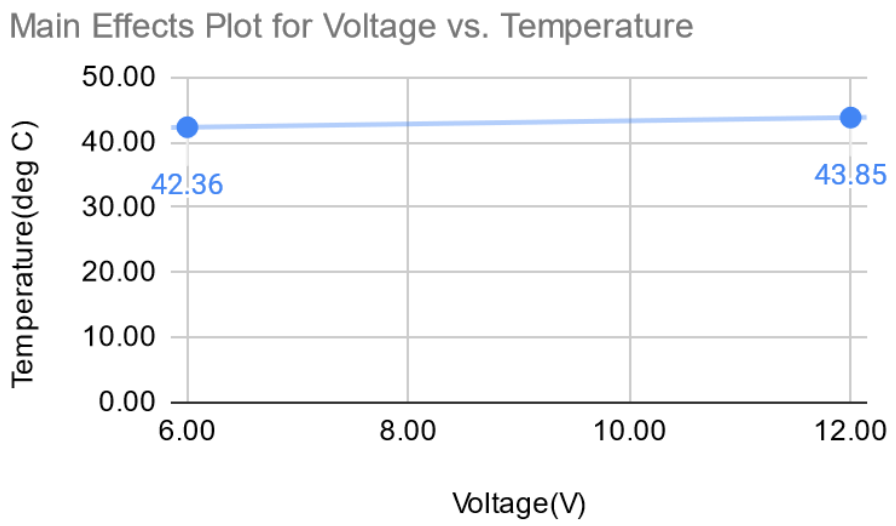


Figure Q10:

Main Effects Plot for Carriage Mass vs. Temperature

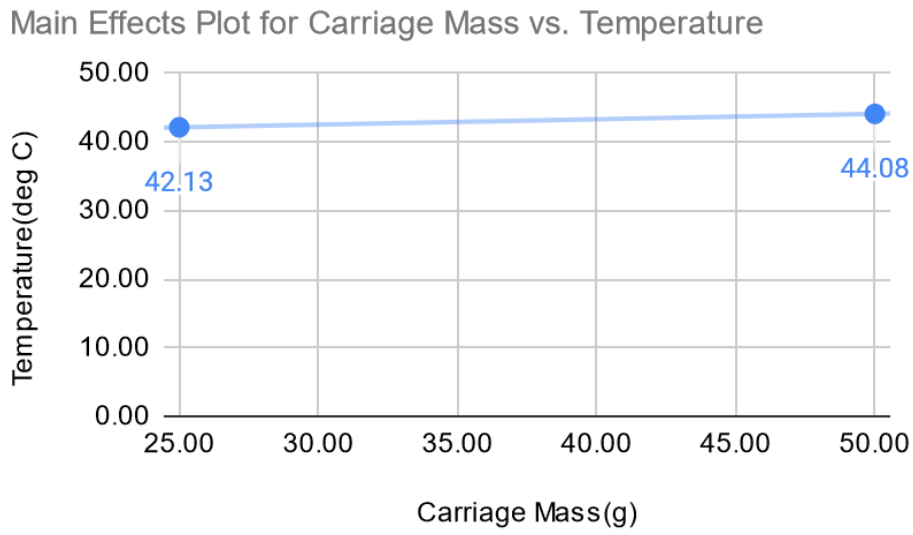


Figure Q11:

Interaction Plot for Current-Voltage Factors for Max Acceleration Response

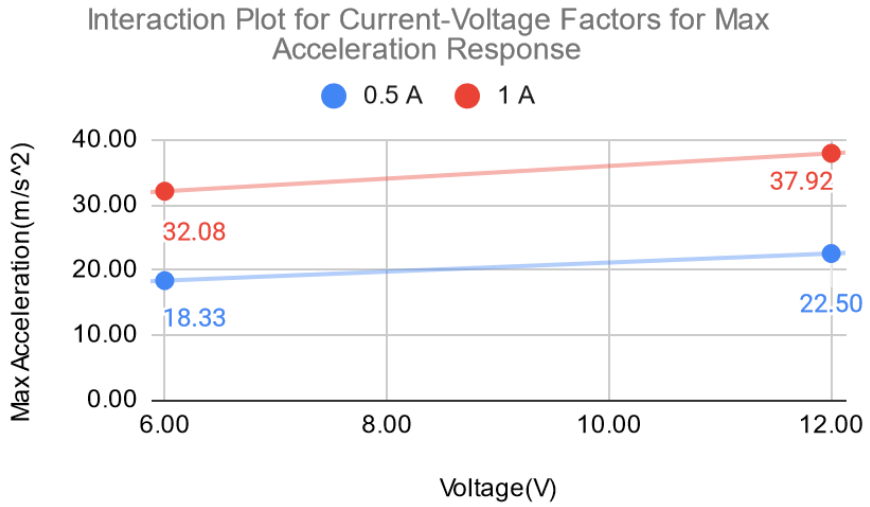


Figure Q12:

Interaction Plot for Voltage-Carriage Mass Factors for Max Acceleration Response

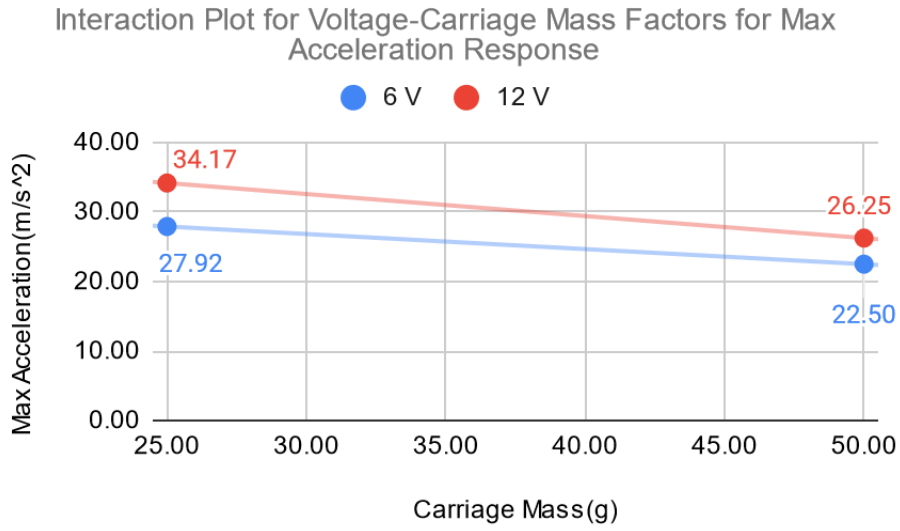


Figure Q13:

Interaction Plot for Current-Carriage Mass Factors for Max Acceleration Response

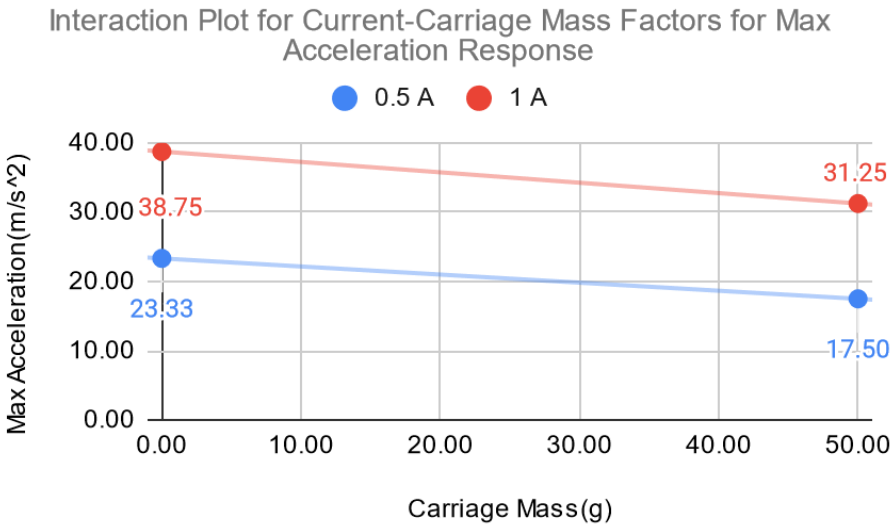


Figure Q14:

Interaction Plot for Current-Voltage Factors for Temperature Response

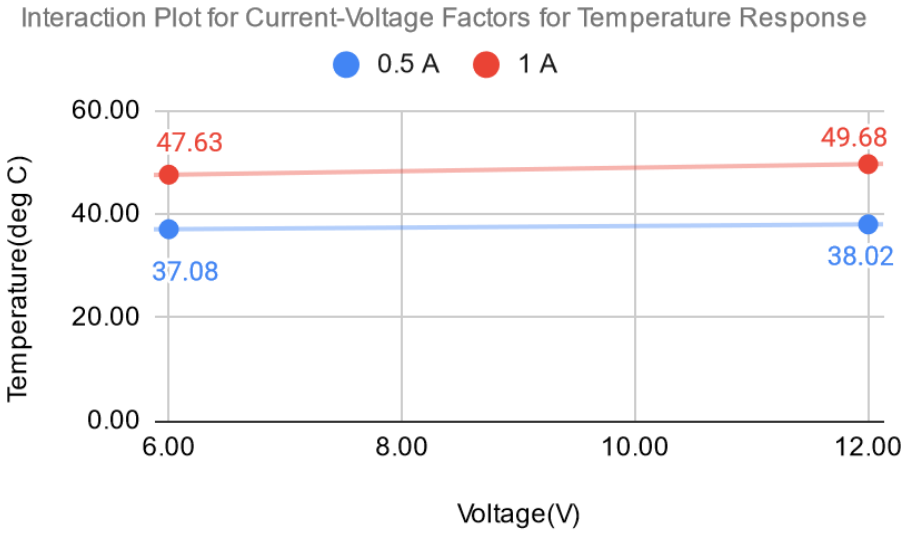


Figure Q15:

Interaction Plot for Voltage-Carriage Mass Factors for Temperature Response

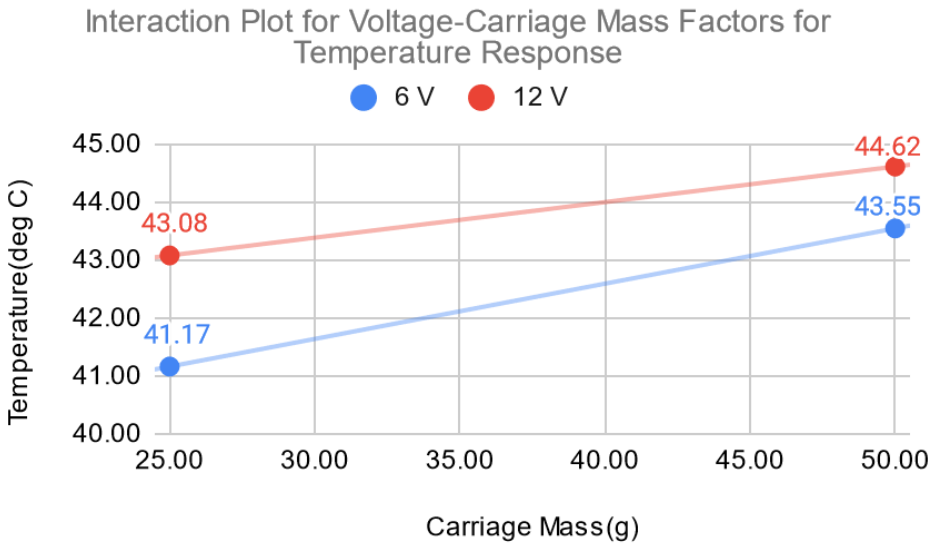


Figure Q16:

Interaction Plot for Current-Carriage Mass Factors for Temperature Response

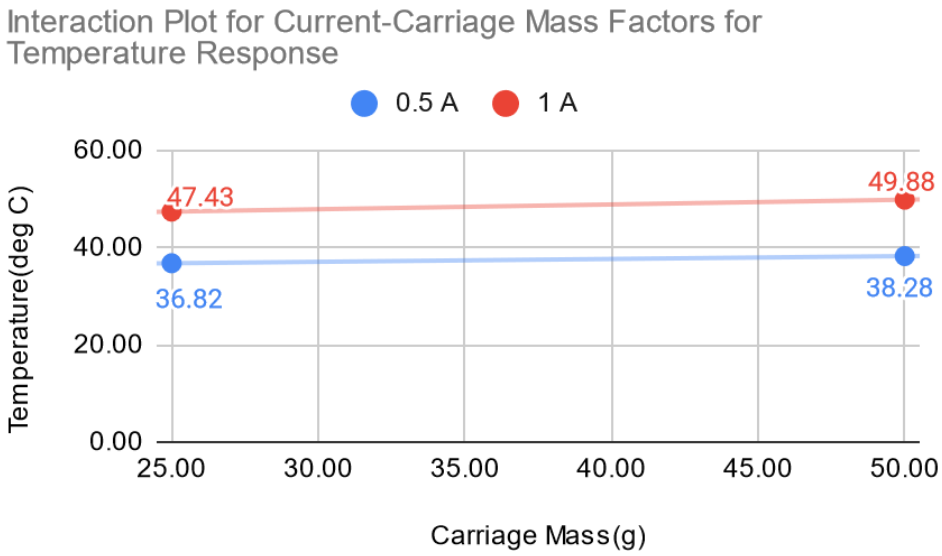


Figure Q17:

Regression Analysis for Maximum Acceleration Response

Regression Statistics	
Multiple R	0.985405025
R Square	0.971023063
Adjusted R Square	0.960795909
Standard Error	1.7327582
Observations	24

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	6	1710.416667	285.0694444	94.94557823	4.0042E-12
Residual	17	51.04166667	3.00245098		
Total	23	1761.458333			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	27.70833333	0.353697787	78.33900689	3.40288E-23	26.96209623	28.45457043	26.96209623	28.45457043
X1 -- A (Current)	7.291666667	0.353697787	20.61552813	1.82249E-13	6.545429567	8.037903766	6.545429567	8.037903766
x2 -- B (Voltage)	2.5	0.353697787	7.068181072	1.88743E-06	1.7537629	3.2462371	1.7537629	3.2462371
x3 -- C (Carriage Mass)	-3.333333333	0.353697787	-9.42424143	3.66606E-08	-4.079570433	-2.587096234	-4.079570433	-2.587096234
x1-x2	0.416666667	0.353697787	1.178030179	0.255012816	-0.329570433	1.162903766	-0.329570433	1.162903766
x2-x3	-0.625	0.353697787	-1.767045268	0.095166645	-1.3712371	0.1212371	-1.3712371	0.1212371
x1-x3	-0.416666667	0.353697787	-1.178030179	0.255012816	-1.162903766	0.329570433	-1.162903766	0.329570433

$$y_1=27.71+7.29x_1+2.50x_2-3.33x_3$$

Figure Q18:

Regression Analysis for Temperature Response

Regression Statistics	
Multiple R	0.998267459
R Square	0.99653792
Adjusted R Square	0.995316009
Standard Error	0.399540177
Observations	24

ANOVA

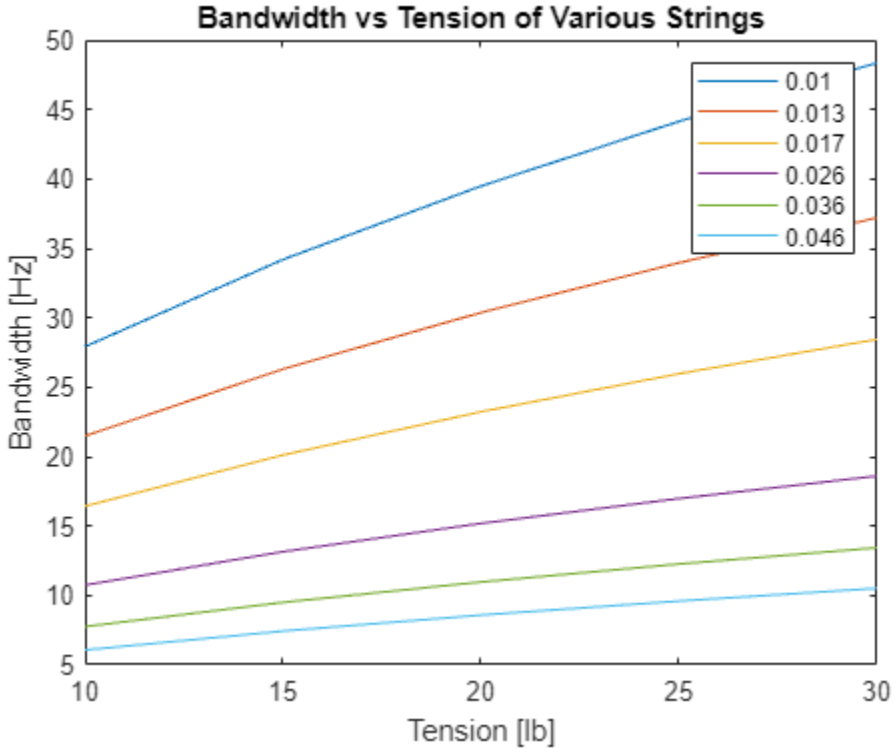
	df	*SS*	*MS*	*F*	*Significance F*
Regression	6	781.1358333	130.1893056	815.5571421	6.01941E-20
Residual	17	2.71375	0.159632353		
Total	23	783.8495833			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	43.10416667	0.081555797	528.5236391	2.79795E-37	42.93209898	43.27623436	42.93209898	43.27623436
X1 -- A (Current)	5.554166667	0.081555797	68.10265934	3.65228E-22	5.382098976	5.726234358	5.382098976	5.726234358
x2 -- B (Voltage)	0.745833333	0.081555797	9.145068284	5.64316E-08	0.573765642	0.917901024	0.573765642	0.917901024
x3 -- C (Carriage Mass)	0.979166667	0.081555797	12.00609523	9.98444E-10	0.807098976	1.151234358	0.807098976	1.151234358
x1-x2	0.279166667	0.081555797	3.423014386	0.003242162	0.107098976	0.451234358	0.107098976	0.451234358
x2-x3	-0.2125	0.081555797	-2.605578114	0.01846573	-0.384567691	-0.040432309	-0.384567691	-0.040432309
x1-x3	0.245833333	0.081555797	3.01429625	0.007812803	0.073765642	0.417901024	0.073765642	0.417901024

$$y_2=43.10+5.55x_1+0.75x_2+0.98x_3+0.28x_1x_2-0.21x_2x_3+0.25x_1x_3$$

Appendix R: Simulation Results

Figure S1: Bandwidth vs Tension of Various Strings



Note that the legend labels are in units of inches specifying string diameter.

Appendix S: Bill of Materials & Budget

D16 fx

	A	B	C	D	E
1	**Materials**	**Price / Unit**	**Quantity**	**Cost ($)**	**Status**
2	GT2 Idler	2	1	2	Received
3	GT2 Timing Belt	6	1	6	Received
4	GT2 Pulley	2	1	2	Received
5	Steel Strings	2	1	2	Received
6	Single String Pickup	8	1	8	Received
7	Brushed DC motor with gear box and encoder	50	2	100	Received
8	Bearings	15	1	15	Received
9	Screws	15	1	15	Received
10	Py Pico Controller	5	1	5	Received
11	L298 Motor Driver	6	1	6	Received
12	M3 Hex Head Fastener Assortment Kit	9	1	9	Received
13	ESP8266	8	1	8	Received
14					
15			**Total**	176	

Appendix T: Final Prototype CAD and Pictures

Note that CAD figures do not have the pulley or string.

Figure T1: Isometric view of final prototype CAD assembly

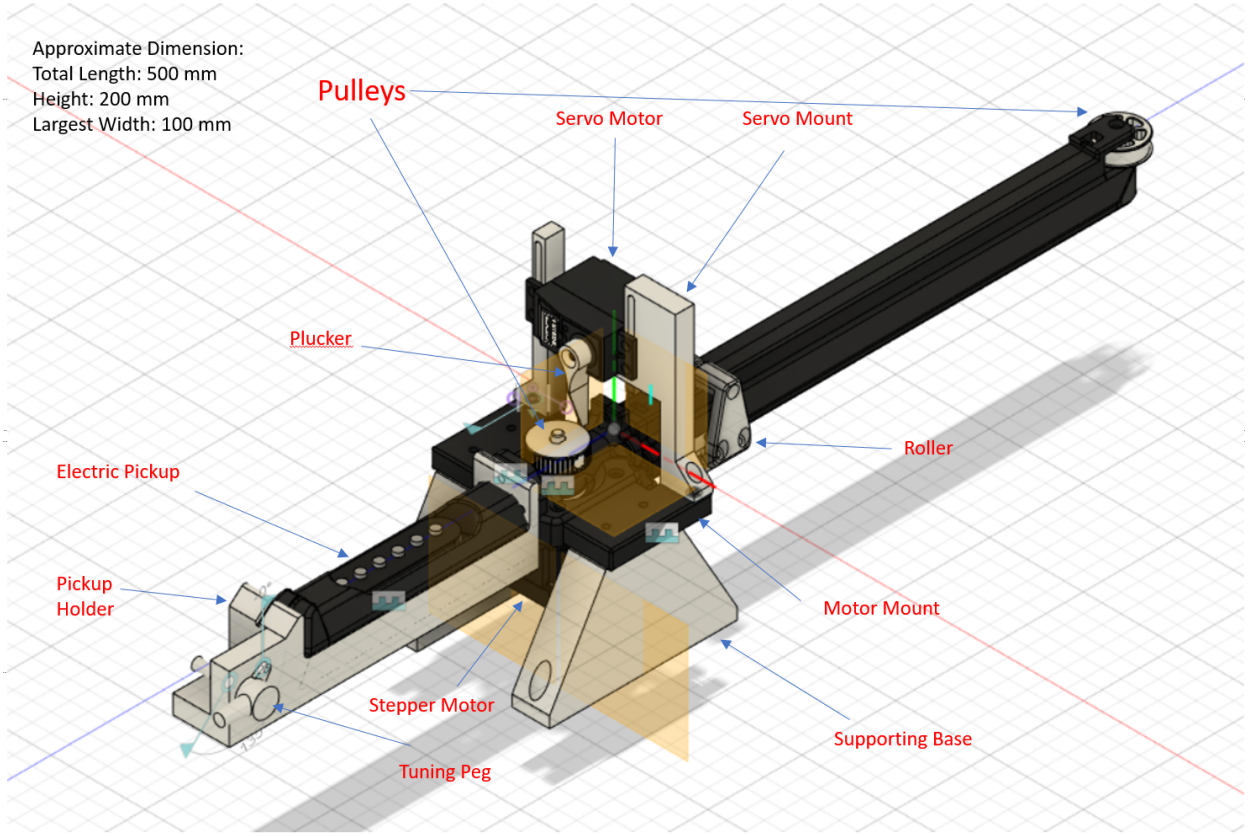


Figure T2: Side view of final prototype CAD assembly

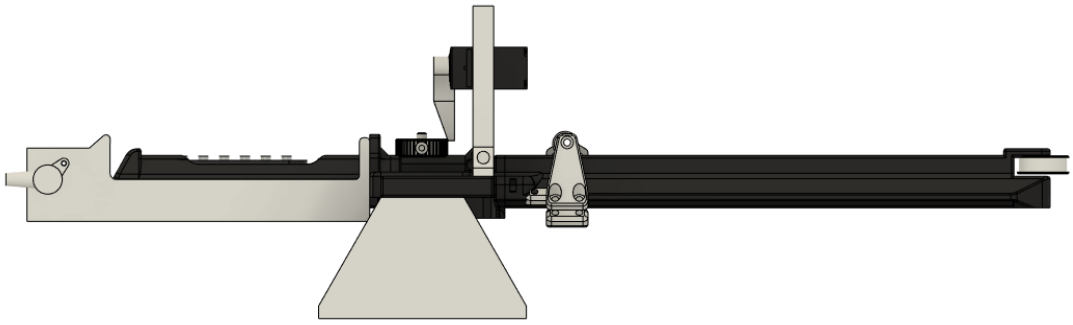


Figure T3: Top view of final prototype CAD assembly

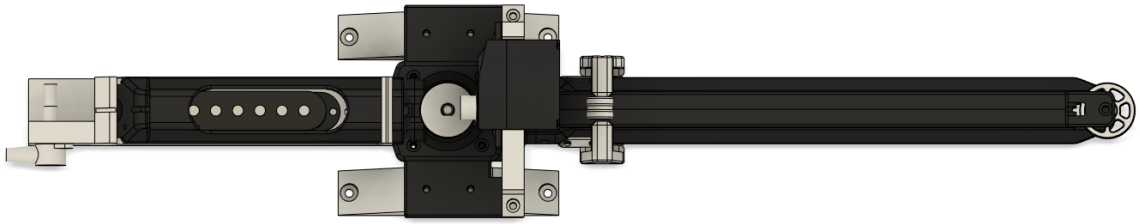


Figure T4: Front view of final prototype CAD assembly

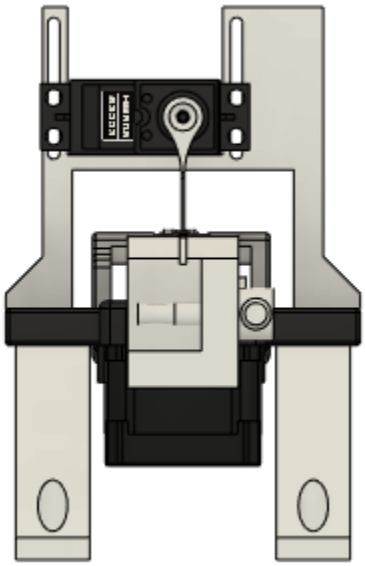


Figure T5: Drawing sheet with assembly projections and dimensions.

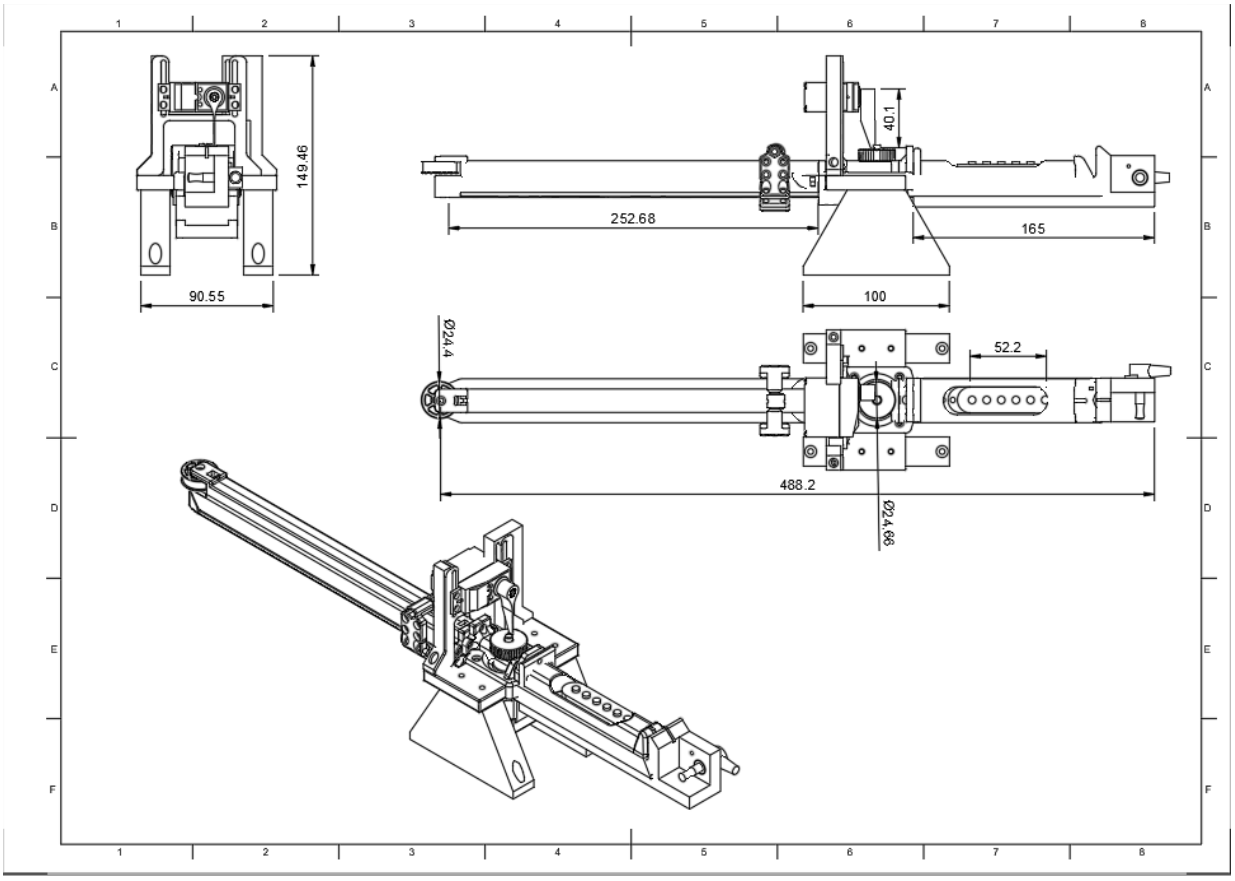


Figure T6: Isometric view of final assembled prototype.

Appendix U: Firmware & Software Documentation

Demo Code:

temp.py demo.py

```
import tkinter as tk

top = tk.Tk()
P = tk.Canvas(top, bg="#EEEEEE", height=250, width=1000)
V = tk.Canvas(top, bg="#EEEEEE", height=250, width=1000)
A = tk.Canvas(top, bg="#EEEEEE", height=250, width=1000)

ps = []

def clear():
    for x in [P,V,A]: x.delete("all")

clear_button = tk.Button(top, command=clear)

max_accel = 100
max_vel = 100

last_time = 0.0
last_position = 0.0
last_velocity = 0.0
last_acceleration = 0.0
time = 0.0
position = 0.0
velocity = 0.0
acceleration = 0.0

def add_point(p1, p2):
    global last_time, time, last_position, last_velocity, last_acceleration, position, velocity, acceleration
    time = last_time + 1
    P.create_line(last_time, last_position+125, time, position+125, fill="blue", width=2)
    V.create_line(last_time, last_velocity*50+125, time, velocity*50+125, fill="purple", width=2)
    A.create_line(last_time, last_acceleration*500+125, time, acceleration*500+125, fill="red", width=2)
    last_time = time
    last_position = position
    last_velocity = velocity
    last_acceleration = acceleration

def callback(event):
```

temp.py demo.py

```
    global last_time, time, last_position, last_velocity, last_acceleration, position, velocity, acceleration
    time = last_time + 1
    P.create_line(last_time, last_position+125, time, position+125, fill="blue", width=2)
    V.create_line(last_time, last_velocity*50+125, time, velocity*50+125, fill="purple", width=2)
    A.create_line(last_time, last_acceleration*500+125, time, acceleration*500+125, fill="red", width=2)
    last_time = time
    last_position = position
    last_velocity = velocity
    last_acceleration = acceleration

def callback(event):
    global last_time, time, last_position, last_velocity, last_acceleration, position, velocity, acceleration
    clear()
    ps.append([event.x, event.y-125])
    ps.sort(key=lambda p: p[0])
    if len(ps) >= 1:
        p2 = ps[0]
        P.create_oval(p2[0]-4, p2[1]+125-4, p2[0]+4, p2[1]+125+4, fill="black")
    if len(ps) >= 2:
        for p1, p2 in zip(ps[:-1], ps[1:]):
            move_distance = p2[1] - p1[1]
            move_time = p2[0] - p1[0]
            coast_time = 0.4 * move_time
            accel_time = 0.3 * move_time
            accel = 100.0*move_distance/21.0/(move_time*move_time)
            coast_velocity = 10.0*move_distance/7.0/move_time
            last_time = p1[0]
            last_position = float(p1[1])
            last_velocity = 0.0
            last_acceleration = 0.0
            position = 0.0
            velocity = 0.0
            acceleration = 0.0
            for t in range(int(accel_time)):
                acceleration = accel
                velocity = last_velocity + acceleration
                position = last_position + velocity
                add_point(p1, p2)
            for t in range(int(accel_time), int(accel_time + coast_time)):
                acceleration = 0
                velocity = coast_velocity
                position = last_position + velocity
                add_point(p1, p2)
            for t in range(int(accel_time + coast_time), int(move_time)+1):
                acceleration = -accel
                velocity = last_velocity + acceleration
                position = last_position + velocity
                add_point(p1, p2)
            P.create_oval(p2[0]-4, p2[1]+125-4, p2[0]+4, p2[1]+125+4, fill="black")

P.bind("<Button-1>", callback)

P.pack()
V.pack()
A.pack()

top.mainloop()
```

GUI Code:

demo.py gui.py

```python
import math
import time
from dataclasses import dataclass
import serial
import yaml
import tkinter as tk
from tkinter import ttk
import customtkinter as ctk
from tkinter import filedialog as fd

ctk.set_appearance_mode('System')
ctk.set_default_color_theme('blue')

class App(ctk.CTk):
    WIDTH = 1000
    HEIGHT = 600

    def __init__(self):
        super().__init__()

        self.title('LinkedIn Park GUI')
        self.geometry(f'{App.WIDTH}x{App.HEIGHT}')
        self.protocol('WM_DELETE_WINDOW', self.on_closing)

        self.grid_columnconfigure(4, weight=1)
        self.grid_rowconfigure(0, weight=1)

        self.frame_editor = ctk.CTkFrame(master=self)
        self.frame_editor.grid(row=0, column=0, columnspan=16, sticky='nsew', padx=20, pady=20)

        self.editor_container = ctk.CTkCanvas(master=self.frame_editor, bg=self.frame_editor['background'], highlight
        self.scrollbar_editor = ctk.CTkScrollbar(master=self.frame_editor, orientation='horizontal', command=self.edi
        self.editor_canvas = ctk.CTkCanvas(master=self.editor_container, bg=self.frame_editor['background'], highligh
        self.editor_canvas.bind('<Configure>', lambda e: self.editor_container.configure(scrollregion=self.editor_co
        self.editor_container.create_window((0,0), window=self.editor_canvas, anchor='nw')
        self.editor_container.configure(xscrollcommand=self.scrollbar_editor.set)
        self.editor_container.pack(side='top', fill='both', expand=True, padx=5, pady=5)
        self.scrollbar_editor.pack(side='bottom', fill='x', padx=5, pady=5)

        self.editor = Editor(self.editor_canvas)

        ctk.CTkLabel(master=self, text='zoom:', width=20).grid(row=1, column=0, sticky='w', padx=20, pady=(0, 20))
        self.zoom_out_button = ctk.CTkButton(master=self, text='-', width=30, command=self.zoom_out)
        self.zoom_out_button.grid(row=1, column=1, pady=(0, 20), sticky='w')
        self.zoom_in_button = ctk.CTkButton(master=self, text='+', width=30, command=self.zoom_in)
        self.zoom_in_button.grid(row=1, column=2, padx=20, pady=(0, 20), sticky='w')
        ctk.CTkLabel(master=self, text='', width=10).grid(row=1, column=5, sticky='ew')
        ctk.CTkLabel(master=self, text='time signature:', width=120).grid(row=1, column=6, pady=(0, 20), sticky='e')
        self.time_signature_entry_1 = ctk.CTkEntry(master=self, width=30)
        self.time_signature_entry_1.insert(0, '4')
        self.time_signature_entry_1.grid(row=1, column=7, pady=(0, 20), sticky='e')
        ctk.CTkLabel(master=self, text='/', width=30).grid(row=1, column=8, pady=(0, 20), sticky='e')
        self.time_signature_entry_2 = ctk.CTkEntry(master=self, width=30)
        self.time_signature_entry_2.insert(0, '4')
        self.time_signature_entry_2.grid(row=1, column=9, pady=(0, 20), sticky='e')
        ctk.CTkLabel(master=self, text='tempo:', width=80).grid(row=1, column=10, pady=(0, 20), sticky='e')
        self.tempo_entry = ctk.CTkEntry(master=self, width=40)
```

demo.py gui.py

```
54          self.time_signature_entry_2.insert(0, '4')
55          self.time_signature_entry_2.grid(row=1, column=9, pady=(0, 20), sticky='e')
56          ctk.CTkLabel(master=self, text='tempo:', width=80).grid(row=1, column=10, pady=(0, 20), sticky='e')
57          self.tempo_entry = ctk.CTkEntry(master=self, width=40)
58          self.tempo_entry.insert(0, '120')
59          self.tempo_entry.grid(row=1, column=11, pady=(0, 20), sticky='e')
60          self.set_button = ctk.CTkButton(master=self, text='set', width=40, command=self.set_timing)
61          self.set_button.grid(row=1, column=12, padx=20, pady=(0, 20), sticky='e')
62          self.add_measure_button = ctk.CTkButton(master=self, text='add measure', width=60, command=self.editor.add_measu
63          self.add_measure_button.grid(row=1, column=13, pady=(0, 20), sticky='e')
64          self.play_button = ctk.CTkButton(master=self, text='play', width=40, command=self.editor.play)
65          self.play_button.grid(row=1, column=14, padx=20, pady=(0, 20), sticky='e')
66
67          self.load_button = ctk.CTkButton(master=self, text='load', width=30, command=self.load)
68          self.load_button.grid(row=2, column=13, columnspan=1, pady=(0, 20), sticky='e')
69          self.save_button = ctk.CTkButton(master=self, text='save', width=30, command=self.save)
70          self.save_button.grid(row=2, column=14, columnspan=1, padx=20, pady=(0, 20), sticky='e')
71          self.mode_switch = ctk.CTkSwitch(master=self, text='min accel', command=self.set_mode)
72          self.mode_switch.grid(row=2, column=6, pady=(0, 20), sticky='e')
73
74          self.set_timing()
75
76      def on_closing(self, event=0):
77          self.destroy()
78
79      def zoom(self, dir):
80          self.editor.zoom(dir)
81          button = self.zoom_in_button if dir=='in' else self.zoom_out_button
82
83      def zoom_out(self):
84          self.zoom('out')
85
86      def zoom_in(self):
87          self.zoom('in')
88
89      def set_mode(self):
90          self.editor.set_mode(self.mode_switch.get())
91
92      def set_timing(self):
93          try:
94              self.editor.set_timing(int(self.time_signature_entry_1.get()), int(self.time_signature_entry_2.get()), int(s
95          except Exception as e:
96              print(str(e))
97
98      def load(self):
99          filename = fd.askopenfilename()
100         ts1, ts2, tempo = self.editor.load(filename)
101         self.time_signature_entry_1.delete(0,tk.END)
102         self.time_signature_entry_1.insert(0,str(ts1))
103         self.time_signature_entry_2.delete(0,tk.END)
104         self.time_signature_entry_2.insert(0,str(ts2))
105         self.tempo_entry.delete(0,tk.END)
106         self.tempo_entry.insert(0,str(tempo))
107
108     def save(self):
109         filename = fd.askopenfilename()
110         self.editor.save(filename)
```

demo.py gui.py

```
108         def save(self):
109             filename = fd.askopenfilename()
110             self.editor.save(filename)
111
112     NOTES = {
113         1: 80,
114         2: 330,
115         3: 590,
116         4: 830,
117         5: 1140,
118         6: 1480,
119         7: 1820,
120         8: 2160,
121         9: 2450,
122         10: 2800,
123         11: 3220,
124         12: 3640,
125         13: 4000,
126     }
127
128     @dataclass
129     class Note:
130         time: float
131         position: int
132
133     @dataclass
134     class Trajectory:
135         time: float
136         position: int
137         velocity: int
138         acceleration: int
139
140         def duration(self, last_position):
141             x_f = abs(self.position - last_position)
142             t_f = (x_f+self.velocity**2/self.acceleration)/self.velocity
143             if 0.5*self.acceleration*(t_f/2)**2 > x_f/2:
144                 t_f = 2*math.sqrt(x_f)/math.sqrt(self.acceleration)
145             return t_f
146
147     class Editor:
148         HEIGHT = 450
149         ZOOM_MIN = 100.0
150         ZOOM_MAX = 1000.0
151         ZOOM_STEP = 100.0
152
153         P_HEIGHT = 150.0
154         P_SCALE = 150.0/4000.0
155         V_HEIGHT = 100.0
156         V_SCALE = 0.005
157         A_HEIGHT = 100.0
158         A_SCALE = 0.00015
159
160         current_zoom = 100.0
161         canvas: ctk.CTkCanvas
162         events: list[Note | Trajectory]
163
164         A_M = 200000.0
165         V_M = 150000.0
```

demo.py gui.py

```
    events: list[Note | Trajectory]

    A_M = 200000.0
    V_M = 150000.0

    mode = 0

    ts1 = 4
    ts2 = 4
    tempo = 120

    def __init__(self, canvas):
        self.canvas = canvas
        self.canvas.config(height=self.HEIGHT)
        self.canvas.bind('<Button-1>', self.left_click_callback)
        self.canvas.bind('<Button-2>', self.right_click_callback) # mac trackpad right click
        self.canvas.bind('<Button-3>', self.right_click_callback)
        self.events = []

        self.song_duration = self.ts1*60/self.tempo
        self.instrument = Instrument('/dev/cu.usbserial-145130')

        self.draw()

    def __draw_axes(self, x, y, h, w, zero=1.0):
        t = 4
        self.canvas.create_line(x, y-t/2, x, y+h+t/2, fill='#C0C0C0', width=4)
        self.canvas.create_line(x-t/2, y+zero*h, x+w+t/2, y+zero*h, fill='#C0C0C0', width=4)

    def generate_trajectory(self, x_f, t_f=None):
        if t_f is not None:
            v_m = 2.0*x_f/t_f
            a_m = 2.0*v_m/t_f
            if v_m > self.V_M:
                v_m = self.V_M
                a_m = v_m**2/(x_f-t_f*v_m)
            return (v_m, a_m)
        else:
            t_f = (x_f+self.V_M**2/self.A_M)/self.V_M
            if 0.5*self.A_M*(t_f/2)**2 > x_f/2:
                t_f = 2*math.sqrt(x_f)/math.sqrt(self.A_M)
            return t_f

    def __evaluate_trajectory(self, num_points, t_f, x_f, v_m, a_m):
        x_vals = []
        v_vals = []
        a_vals = []

        t_a = v_m/a_m
        time_step = t_f/num_points
        t = 0.0
        if t_a > t_f/2:
            t_a = t_f/2
            v_m = a_m*t_a

```

demo.py gui.py

```
204
205     def __evaluate_trajectory(self, num_points, t_f, x_f, v_m, a_m):
206         x_vals = []
207         v_vals = []
208         a_vals = []
209
210         t_a = v_m/a_m
211         time_step = t_f/num_points
212         t = 0.0
213         if t_a > t_f/2:
214             t_a = t_f/2
215             v_m = a_m*t_a
216
217         while t < t_a:
218             x_vals.append(0.5*a_m*t**2)
219             v_vals.append(a_m*t)
220             a_vals.append(a_m)
221             t += time_step
222         while t < t_f - t_a:
223             x_vals.append(0.5*a_m*t_a**2+v_m*(t-t_a))
224             v_vals.append(v_m)
225             a_vals.append(0)
226             t += time_step
227         while t < t_f:
228             x_vals.append(0.5*a_m*t_a**2+v_m*(t_f-2*t_a)+v_m*(t-t_f+t_a)-0.5*a_m*(t-t_f+t_a)**2)
229             v_vals.append(v_m-a_m*(t-t_f+t_a))
230             a_vals.append(-a_m)
231             t += time_step
232
233         return (x_vals, v_vals, a_vals)
234
235     def draw(self):
236         self.canvas.delete('all')
237         song_duration = self.song_duration
238         if len(self.events) > 0:
239             if self.events[-1].time > self.song_duration: song_duration = self.events[-1].time
240         self.canvas.config(width=song_duration*self.current_zoom+50)
241
242         # "constants"
243         WIDTH=song_duration*self.current_zoom
244
245         # draw note reference lines
246         for n in NOTES:
247             self.canvas.create_line(20, 20+NOTES[n]*Editor.P_SCALE, WIDTH+20, 20+NOTES[n]*Editor.P_SCALE, fill='#808080')
248
249         # draw timing reference lines
250         beat_duration = self.ts1*60/self.tempo/self.ts2/2
251         t = 0
252         i = 0
253         while t < song_duration:
254             if i%self.ts1*2 == 0:
255                 self.canvas.create_line(20+t*self.current_zoom, 20, 20+t*self.current_zoom, 20+self.P_HEIGHT, fill='#C0C0
256             elif i%2 == 0:
257                 self.canvas.create_line(20+t*self.current_zoom, 20, 20+t*self.current_zoom, 20+self.P_HEIGHT, fill='#6060
258             else:
259                 self.canvas.create_line(20+t*self.current_zoom, 20, 20+t*self.current_zoom, 20+self.P_HEIGHT, fill='#4040
260             t += beat_duration
261             i += 1
```

demo.py gui.py

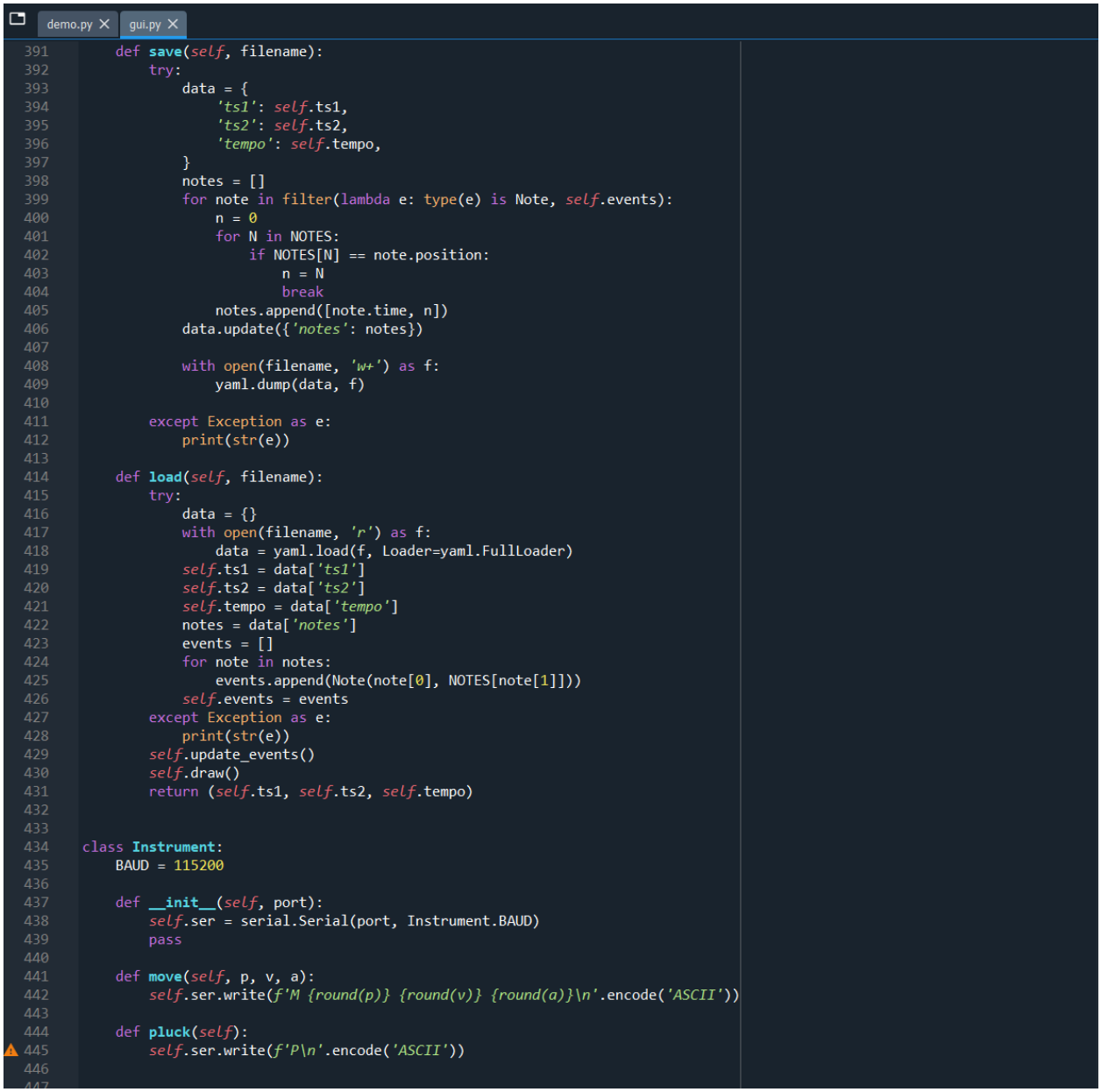
```
246         for n in NOTES:
247             self.canvas.create_line(20, 20+NOTES[n]*Editor.P_SCALE, WIDTH+20, 20+NOTES[n]*Editor.P_SCALE, fill='#808080')
248 
249         # draw timing reference lines
250         beat_duration = self.ts1*60/self.tempo/self.ts2/2
251         t = 0
252         i = 0
253         while t < song_duration:
254             if i%self.ts1*2 == 0:
255                 self.canvas.create_line(20+t*self.current_zoom, 20, 20+t*self.current_zoom, 20+self.P_HEIGHT, fill='#C0C0C
256             elif i%2 == 0:
257                 self.canvas.create_line(20+t*self.current_zoom, 20, 20+t*self.current_zoom, 20+self.P_HEIGHT, fill='#60606
258             else:
259                 self.canvas.create_line(20+t*self.current_zoom, 20, 20+t*self.current_zoom, 20+self.P_HEIGHT, fill='#40404
260             t += beat_duration
261             i += 1
262 
263         # draw graph axes
264         self.__draw_axes(20, 20, Editor.P_HEIGHT, WIDTH, zero =0.0)
265         self.__draw_axes(20, 40+Editor.P_HEIGHT, Editor.V_HEIGHT, WIDTH, zero=0.5)
266         self.__draw_axes(20, 60+Editor.P_HEIGHT+Editor.V_HEIGHT, Editor.A_HEIGHT, WIDTH, zero=0.5)
267 
268         if len(self.events) == 0: return
269 
270         previous_x = 0
271 
272         # draw trajectories
273         for event in self.events:
274             if type(event) is Trajectory:
275                 x_vals, v_vals, a_vals = self.__evaluate_trajectory(event.duration(previous_x)*self.current_zoom, event.du
276                 sign = 1 if event.position >= previous_x else -1
277                 x_vals = [previous_x+sign*x for x in x_vals]
278                 v_vals = [sign*v for v in v_vals]
279                 a_vals = [sign*a for a in a_vals]
280 
281                 last_t = event.time*self.current_zoom
282                 t = last_t
283                 for last_x, x in zip([previous_x] + x_vals, x_vals + [event.position]):
284                     self.canvas.create_line(20+last_t, 20+last_x*Editor.P_SCALE, 20+t, 20+x*Editor.P_SCALE, fill='blue', w
285                     last_t = t
286                     t += 1
287                 previous_x = event.position
288 
289                 last_t = event.time*self.current_zoom
290                 t = last_t
291                 for last_v, v in zip([0] + v_vals, v_vals + [0]):
292                     self.canvas.create_line(20+last_t, 40+Editor.P_HEIGHT+Editor.V_HEIGHT/2+last_v*Editor.V_SCALE, 20+t, 4
293                     last_t = t
294                     t += 1
295 
296                 last_t = event.time*self.current_zoom
297                 t = last_t
298                 for last_a, a in zip([0] + a_vals, a_vals + [0]):
299                     self.canvas.create_line(20+last_t, 60+Editor.P_HEIGHT+Editor.V_HEIGHT+Editor.A_HEIGHT/2+last_a*Editor.
300                     last_t = t
301                     t += 1
```

demo.py gui.py

```
295
296                 last_t = event.time*self.current_zoom
297                 t = last_t
298                 for last_a, a in zip([0] + a_vals, a_vals + [0]):
299                     self.canvas.create_line(20+last_t, 60+Editor.P_HEIGHT+Editor.V_HEIGHT+Editor.A_HEIGHT/2+last_a*Editor.
300                     last_t = t
301                     t += 1
302
303         for event in self.events:
304             if type(event) is Note:
305                 self.canvas.create_oval(20+event.time*self.current_zoom-4, 20+event.position*Editor.P_SCALE-4, 20+event.ti
306
307     def update_events(self):
308         notes = [n for n in filter(lambda f: type(f) == Note, self.events)]
309         notes.sort(key=lambda n: n.time)
310         events = []
311         if len(notes) > 0 and notes[0].position != 0:
312             v, a = self.generate_trajectory(notes[0].position, t_f=notes[0].time - 0.04)
313             events.append(Trajectory(0.02, notes[0].position, int(v), int(a)))
314         if len(notes) >= 1:
315             for note, next_note in zip(notes[:-1], notes[1:]):
316                 events.append(note)
317                 if note.position - next_note.position == 0: continue
318                 if self.mode:
319                     v, a = self.generate_trajectory(abs(next_note.position - note.position), t_f=next_note.time - note.tim
320                     events.append(Trajectory(note.time+0.02, next_note.position, int(v), int(a)))
321                 else:
322                     t_f = self.generate_trajectory(abs(next_note.position - note.position))
323                     events.append(Trajectory(next_note.time - t_f - 0.04, next_note.position, int(self.V_M), int(self.A_M)
324             events.append(next_note)
325         if len(notes) > 0:
326             v, a = self.generate_trajectory(notes[-1].position, t_f=2)
327             events.append(Trajectory(notes[-1].time+3.0, 0, int(v), int(a)))
328         self.events = events
329
330     def left_click_callback(self, event):
331         if event.y > 20+Editor.P_HEIGHT or event.y < 20: return
332         if event.x < 20: return
333         y = min([NOTES[i]*self.P_SCALE+20 for i in NOTES], key=lambda x:abs(x-event.y))
334         self.events.append(Note((event.x-20)/self.current_zoom, (y-20)/self.P_SCALE))
335         self.update_events()
336         self.draw()
337
338     def right_click_callback(self, event):
339         if event.y > 20+Editor.P_HEIGHT or event.y < 20: return
340         if event.x < 20: return
341         closest_note = min(filter(lambda f: type(f) == Note, self.events), key=lambda x:math.sqrt(abs(x.position*self.P_SC
342         self.events.remove(closest_note)
343         self.update_events()
344         self.draw()
345
346     def can_zoom(self, dir):
347         if dir == 'in':
348             return self.current_zoom + Editor.ZOOM_STEP <= Editor.ZOOM_MIN
349         return self.current_zoom - Editor.ZOOM_STEP >= Editor.ZOOM_MAX
350
351     def zoom(self, dir):
352         if dir == 'in': self.current_zoom += Editor.ZOOM_STEP
```

demo.py gui.py

```
351     def zoom(self, dir):
352         if dir == 'in': self.current_zoom += Editor.ZOOM_STEP
353         else: self.current_zoom -= Editor.ZOOM_STEP
354         if self.current_zoom < Editor.ZOOM_MIN: self.current_zoom = Editor.ZOOM_MIN
355         if self.current_zoom > Editor.ZOOM_MAX: self.current_zoom = Editor.ZOOM_MAX
356         self.draw()
357 
358     def set_mode(self, mode):
359         self.mode = mode
360         self.update_events()
361         self.draw()
362 
363     def set_timing(self, ts1, ts2, tempo):
364         for event in self.events:
365             event.time *= self.tempo/tempo
366         self.tempo = tempo
367         self.ts1 = ts1
368         self.ts2 = ts2
369         self.update_events()
370         self.draw()
371 
372     def add_measure(self):
373         self.song_duration += self.ts1*60/self.tempo
374         self.draw()
375 
376     def play(self):
377         if len(self.events) > 0 and type(self.events[0] is Trajectory):
378             self.instrument.move(self.events[0].position, self.events[0].velocity, self.events[0].acceleration)
379         last_time = 0.0
380         for event in self.events:
381             if type(event) is Note:
382                 time.sleep(event.time - last_time - 0.01)
383             else:
384                 time.sleep(event.time - last_time)
385             last_time = event.time
386             if type(event) is Trajectory:
387                 self.instrument.move(event.position, event.velocity, event.acceleration)
388             else:
389                 self.instrument.pluck()
390 
391     def save(self, filename):
392         try:
393             data = {
394                 'ts1': self.ts1,
395                 'ts2': self.ts2,
396                 'tempo': self.tempo,
397             }
398             notes = []
399             for note in filter(lambda e: type(e) is Note, self.events):
400                 n = 0
401                 for N in NOTES:
402                     if NOTES[N] == note.position:
403                         n = N
404                         break
405                 notes.append([note.time, n])
406             data.update({'notes': notes})
407 
408             with open(filename, 'w+') as f:
```



```
demo.py  gui.py
391      def save(self, filename):
392          try:
393              data = {
394                  'ts1': self.ts1,
395                  'ts2': self.ts2,
396                  'tempo': self.tempo,
397              }
398              notes = []
399              for note in filter(lambda e: type(e) is Note, self.events):
400                  n = 0
401                  for N in NOTES:
402                      if NOTES[N] == note.position:
403                          n = N
404                          break
405                  notes.append([note.time, n])
406              data.update({'notes': notes})
407
408              with open(filename, 'w+') as f:
409                  yaml.dump(data, f)
410
411          except Exception as e:
412              print(str(e))
413
414      def load(self, filename):
415          try:
416              data = {}
417              with open(filename, 'r') as f:
418                  data = yaml.load(f, Loader=yaml.FullLoader)
419              self.ts1 = data['ts1']
420              self.ts2 = data['ts2']
421              self.tempo = data['tempo']
422              notes = data['notes']
423              events = []
424              for note in notes:
425                  events.append(Note(note[0], NOTES[note[1]]))
426              self.events = events
427          except Exception as e:
428              print(str(e))
429          self.update_events()
430          self.draw()
431          return (self.ts1, self.ts2, self.tempo)
432
433
434  class Instrument:
435      BAUD = 115200
436
437      def __init__(self, port):
438          self.ser = serial.Serial(port, Instrument.BAUD)
439          pass
440
441      def move(self, p, v, a):
442          self.ser.write(f'M {round(p)} {round(v)} {round(a)}\n'.encode('ASCII'))
443
444      def pluck(self):
445          self.ser.write(f'P\n'.encode('ASCII'))
446
```

Appendix V: Operating and Repair Instructions

Setup

1. Connect the computer that will be used to control the instrument to the microcontroller's micro USB connector. The blue light on the microcontroller should flash once.
2. Use a headphone cable to connect the pickup output to an external amplification device if amplified audio output is desired.

3. Connect the 12VDC power supply to the barrel connector of the instrument. The blue light on the step-down converter should turn on.

Operation

1. Open the graphical user interface.
2. Use the left mouse button to place notes in the position graph to add notes to the current song. Use the right mouse button to remove a selected note.
3. Click the add measure button to increase the width of the graphs.
4. Click the play button to “stream” the notes to the instrument.
5. Click the save button to save the current notes to a song file.
6. Click the load button to load notes from a selected song file. NOTE: this will overwrite any notes currently placed in the graph.

Repair

Broken pick

1. Unscrew and remove the screw that goes through the pick and into the center of the servo output shaft.
2. Take note of the pick’s current orientation.
3. Slide off the broken pick (this will require some force).
4. Slide on a new pick. Make sure it is orientated the same as the broken pick.
5. Replace and tighten the screw removed in step 1.

Loose belt

1. Lossen 2 screws of one of the clams on the roller assembly
2. Move the roller to the end of the neck.
3. Pull the belt taught with one hand and re-tighten the screws of the clamp with the other.

Pick misses the string or gets stuck on the string

1. Loosen the two screws holding the servo in place.
2. Move the roller towards the stepper motor till it hits the limit switch.
3. Adjust the servo height such that the tip of the pick is at the midpoint of the string's diameter.
4. Re-tighten the two screws holding the servo in place. Ensure that the servo does not move while doing this.

Burnt out stepper driver

1. Disconnect the power and USB cords from the instrument.
2. Remove the broken stepper driver from the breadboard.
3. Replace it with a new stepper driver.
4. Re-connect the power and USB cords.