UCF

COLLEGE OF ENGINEERING & COMPUTER SCIENCE

Pick Pocket Tuner

Senior Design II – Group 42 Fall 2021

Lucas Grayford - Electrical Engineering
Paul Grayford - Electrical Engineering
Luis Vargas – Computer Engineering
Jamie Henry – Computer Engineering

Project advisor: Dr. Samuel Richie

Table of Contents

Table of Figures v

Table of Tables vi

Table of Equations vi

1.0 Executive Summary 1

2.0 Project Narrative 2

2.1 Participant Identification 2

2.2 Project Motivation 2

2.3 Project Description 2

2.4 Features and Options 3

2.5 Stretch Goals 3

2.6 Restrictions and Constraints 3

3.0 COVID-19 Supply Chain Issues 4

3.1 COVID-19 Supply Mitigation Plan 4

4.0 Budget Constraints 5

4.1 Budget Mitigation Plan 5

5.0 Weekly Scheduled Meetings 5

6.0 Miscellaneous Constraints 6

7.0 Design Process 6

7.1 Technical Investigation 6

7.2 Applications of Engineering Fundamentals 7

7.3 Preliminary Testing 7

7.4 Subsystem Testing 8

7.5 Overall Testing 8

8.0 House of Quality Diagram 9

9.0 Competition 10

10.0 Tuning Background 12

11.0 Hardware Block Diagram 15

12.0 Physical Components 15

12.1.1 ¼” Output Jack 16

12.1.2 Selection of Output Jack 16

12.1.3 Testing Process 17

12.1.4 Preliminary Testing Results 17
12.1.5 Implementation 18

12.2 Motor 18
12.2.1 Motor Testing 19
12.2.2 Motor Research 20
12.2.3 Motor Conclusion 22

12.3 Driver/Pulse Width Modulator 23
12.3.1 Driver/Pulse Width Modulator Testing 23
12.3.2 Driver/Pulse Width Modulator Conclusion/Encoder 24

12.4 Peg Winder 25
12.4.1 Peg Winder Testing 25

12.5 Filament and Cura 26

12.6 Battery 27
12.6.1 Battery Theory 28

12.7 Switch and Buttons 30
12.7.1 Switch and Buttons Options 30

12.8 Housing 32
12.8.1 Housing Theory 33

12.9 Buzzer 34
12.9.1 Buzzer Testing 35

12.10 Vibration Sensor 35

12.11 Display 37

12.12 Analog to Digital Converter 38
12.12.1 Selection of ADC 38
12.12.2 Testing Process 38
12.12.3 Testing Results 39
12.12.4 Implementation 39

12.13 Voltage regulators 39
12.13.1 Prototyping Regulation 39
12.13.2 Voltage Regulation Strategy 40
12.13.3 Buck Converters 41
12.13.4 Voltage Regulator Designs, 3.3V 41
12.13.5 Regulator Schematic and Specifications, 3.7-12V 43
12.13.6 Ordering Process for Regulators 44
12.13.7 Testing Process for Regulators 44
12.13.8 Implementation of Voltage Regulators 45

12.14 Processor / Controller 45

12.15 Clock 46

12.16 Charging Circuit 46
12.16.1 Design of Charging Circuit 47
12.16.2 Charging Circuit Using MCP73831/2 48
12.16.3 Requirements and Specifications of MCP73831/2 48
12.16.4 Suppliers for MCP73831/2 49
12.16.5 Testing Process for MCP73831/2 49

12.16.6 Implementation Process for MCP73831/2 50
12.16.7 Prototyping Charging Circuit 50
12.16.8 Testing Charging Circuit 51
12.16.9 Prototype Charging Circuit Implementation 51
12.16.10 Using the Prototype as Plan B 52
12.16.11 Mounting Prototype in Housing 52
12.16.12 Testing Process 53
12.16.13 Implementation Plan 53

13.0 Printed Circuit Board Design 54

13.1 Overall Layout of PCB 54

13.2 Power Distribution 54

13.3 Fusion 360 55

13.4 Collaboration Method 55

13.5 Designing Components Using Fusion 360 56

13.6 Supplier 58
13.6.1 How to Order from JLCPCB 58

13.7 PCB Assembly 59

13.8 Testing the PCB 60

14.0 Prototype Construction 60

14.1 Prototype Construction Beginning Notes 60

14.2 Prototyping Components 60
14.2.1 Processor/Controller 60
14.2.2 Sensor 61
14.2.3 Motor 61
14.2.4 Motor Driver 62
14.2.5 LCD Display 63
14.2.6 USB-C Charging Circuit 63
14.2.7 Battery 63

14.3 Additional Components in Prototype Construction 63
14.3.1 Piezo Sensor Filter 63
14.3.2 DC-IN Power for Breadboard 64
14.3.3 3D Printed Guitar Peg-Winder 64

14.4 Next Prototype Construction Steps and Ending Notes 64

15.0 Inside of the Jowoom 65

16.0 PCB Layout 68

16.1 Ultra-Librarian 69

16.2 PCB Parts Placement 69

16.3 PCB Lead Wired Part Placements 71

16.4 PCB Prototype 73

16.5 PCB Wiring 74

17.0 Project Summary – Hardware *77*

18.0 Prototype Testing *78*

18.1 Prototype Testing Beginning Notes **78**

18.2 Prototype Component Testing **78**

18.2.1 ESP8266 Testing 78

18.2.2 Piezo Sensor and Filter Testing 79

18.2.3 Motor Testing 84

18.2.4 LCD Display Testing 85

18.2.5 FFT Input Testing 86

18.2.6 USB-C Charging Circuit and Battery Testing 86

18.3 Next Prototype Testing Steps and Ending Notes **87**

19.0 Software Block Diagram *87*

20.0 Algorithm *88*

21.0 Software Features/Functions *89*

21.1 Fast Fourier Transform **89**

21.2 Tuning a New String **89**

21.3 Display Interface **90**

21.4 Tuning Libraries (Stretch Goal) **91**

21.5 Tuning Other Instruments (Stretch Goal) **91**

21.6 Programming Language(s) **92**

21.7 Programming/Software Libraries **92**

22.0 UI/UX Software Hierarchy and Design *94*

22.1 Software Hierarchy Description **94**

22.2 Software Tuning **95**

22.3 Software Display **96**

22.4 Handling Flat and Sharp Notes **97**

22.5 Motor communication with FFT **99**

22.6 New String Mode **99**

22.7 Autotune Mode **101**

22.8 Free Move Mode **102**

23.0 Software Summary *102*

24.0 Project Budget Estimates *103*

24.1 Final Prototype Cost (Estimated Range) **103**

24.2 Project Development Costs **103**

24.3 Development Cost (Estimated Range) **104**

25.0 Project Milestones 105

26.0 Hardware & Software Requirement Specifications Tables 106

26.1 Hardware Requirement Specifications 106

26.2 Software Requirement Specifications 110

27.0 Final Testing Plan 113

27.1 Objective for Final Testing 113

27.2 Description of Final Testing Environment 113

27.3 Stopping Criteria 115

27.4 Description of Individual Test Cases 115

27.5 Trace of Individual Test Cases to Requirements 121

28.0 Conclusion 122

Bibliography .... A

Table of Figures

Figure 1, House of Quality Diagram 10
Figure 2, Representation of Physical Components on an Electric Guitar 14
Figure 3, Representation of Physical Components on a Violin 14
Figure 4, Hardware Block Diagram 15
Figure 5, Comparison of Pure Tone output jack to a standard stereo output jack 17
Figure 6, 5V Stepper Motor (Model: 28BYJ-48) 19
Figure 7, Brushless DC Motor with Encoder 12V 159rpm 21
Figure 8, Line Connections for Arduino (will vary from ESP8266 connections slightly) 22
Figure 9, 12V DC Motor Dimensions in mm 22
Figure 10, UL2003 Driver Board (left), L298N Motor Driver Control Board (right) 24
Figure 11, Universal Peg Winder Head 25
Figure 12, 3.7V Lithium Ion Polymer 1200 mAh Battery 29
Figure 13, Switch Mounted 31
Figure 14, Full Button Layout for Connections 31
Figure 15, Multicolor Changeable Buttons 32
Figure 16, Pick Pocket Tuner Housing Model 34
Figure 17, HYDZ 3-5V Buzzer 35
Figure 18, Rounded IPS-TFT LCD Display 37
Figure 19, Regulation Diagram 40
Figure 20, Desired Voltage Regulator Design 41
Figure 21, Schematic Design Choice 42
Figure 22, 12V Regulator Design 43
Figure 23, ESP8266 on NodeMCU Development Board 46
Figure 24, Charging Circuit Design with LM317 (Intellectual Design Provided by Texas Instruments) 47
Figure 25, Simple Application Circuit by Microchip 49
Figure 26, LiPo charging circuit to be used in prototyping and testing phase of project 50
Figure 27, Image of Adafruit Charger Showing Mounting Holes 53

Figure 28, Student-Made Piezoelectric Sensor (Symbol on Left and Footprint on Right) 57
Figure 29, Artistic Rendition of the Initial Prototype Mock-Up 65
Figure 30, Jowoom internals 67
Figure 31, Jowoom internals front side 68
Figure 32, Final Internal Housing Layout 71
Figure 33, Final External Housing 72
Figure 34, PCB prototype v1.0 73
Figure 35, PCB Prototype V3.0 74
Figure 36, Jowoom Automatic Tuner (deconstructed) showing two piezoelectric sensors in parallel 81
Figure 37, Breadboard Prototype Demonstrating iPhone Speaker Vibration/Tone Test 82
Figure 38, Oscilloscope Frequency From Piezo Sensor, iPhone Test 83
Figure 39, Oscilloscope Frequency from Piezo Sensor, iPhone Test, with Filter 83
Figure 40, Software Block Diagram 88
Figure 41, UI/UX Software Hierarchy/Design 94
Figure 42, TFT display GUI Layout During Tuning 96
Figure 43, Natural Notes Written on the Keys of a Keyboard 98

Table of Tables

Table 1, Requirement Specifications 3
Table 2, Frequencies for a Guitar Standard Tuning 13
Table 3, Critical Specifications for Efficient Regulators 42
Table 4, Critical Specifications for Chosen Regulator 42
Table 5, Critical Specifications For 12V Regulator 43
Table 6, Power Distribution Calculations (Incomplete Table) 55
Table 7, Motor Pin Connections 75
Table 8, Display Pin Connection 76
Table 9, Push Button Pin Connections 76
Table 10, Frequency Device Pin Connects 76
Table 13, Flat/Natural/Sharp Frequency Example starting at Middle C 98
Table 14, Frequency Examples of One Higher and Lower Octave For Each Frequency 100
Table 15, Demonstrating One and Two Step Offset Tuning from E Standard Tuning 101
Table 16, Estimated Final Cost for Device 103
Table 17, Estimated Cost for Development of Project 104
Table 18, Project Milestones Chart 106

Table of Equations

No table of figures entries found.

1.0 Executive Summary

To tune a stringed instrument, a peg is used for tightening or loosening the string threaded into a cylinder. This peg usually consists of a worm drive and the peg head or key. The worm drive consists of a screw-like gear and a spur gear to create a smaller volume object with similar gear ratios. Some players also use peg locks to help ensure that the peg will stay in place to help reduce loss in tuning on the string.

The purpose of tuning a stringed instrument is to create sounds that illicit emotions in those who hear and play it. Tuning the strings to the correct pitches ensures that all notes that can be played on the instrument will make the correct sounds and will create captivating music. To ascertain that the tuning of a stringed instrument is correct, the user would match the frequency of the plucked string to a known frequency, such as a tuning fork or another audible device, by ear. However, tuning stringed instruments by ear could lead to incorrect tuning and incorrect pitches across the strings. This has led to the creation of automatic guitar tuners to eliminate human error and make the tuning process faster.

Sensors are used by these automatic guitar tuners to help eliminate the human error. These include vibrational and microphone sensors that pick up the frequency of the plucked string. This gives an accurate reading of the string's frequency that can then be compared to the tuning specifications. Our group's Pick Pocket guitar tuner uses two piezo vibrational sensors that will sit in the housing of the product to pick up that string's frequency. This frequency, read in as analog voltage, then is transformed into the frequency domain to read its value. To achieve this, the Fast Fourier Transform was used. The Fast Fourier Transform was chosen as opposed to the Discrete Fourier Transform solely on computation time. Since time is valuable to our automatic tuner and is one of the deciding factors among competitors.

The main purpose of this transform is to have the sensor's input value as a value in hertz to compare to the desired string frequency the user picked. Our tuner then figures out the ratio difference between the two frequency values and send voltage to the brushless DC motor accordingly. The motor then spins the peg winder that is designed to fit a wide array of pegs to allow for the tuning of multiple instruments. When the desired frequency is met, visual feedback will let the user know to move on to the next string.

To compete with the existing automatic guitar tuners, our group's focus was to capitalize on cost efficiency, computation/tuning time, and the ability to tune multiple string instruments. These aspects were researched heavily along with dismantling of some competitors' tuners to meet these requirements. Another way to compete with the market was to have our tuner have multiple modes for everyday string instrument use. This would include tuning new strings and freely tightening and loosening the peg. With the creation of the prototype, further testing was conducted on the algorithm/software created to tune the instruments within 5 cents accuracy to further develop the tuner and within the desired time frame.

2.0 Project Narrative

2.1 Participant Identification

Our Senior Design project is called the Pick Pocket Tuner, an automatic guitar tuner that facilitates quick tuning a guitar within a small timeframe. Group number 42 consists of the following members: Lucas Grayford, Paul Grayford, Jamie Henry, and Luis Vargas. The composition of the group by academic major consists of two Electrical Engineering majors, Lucas, and Paul, and two Computer Engineering majors, Jamie, and Luis.

For the Pick Pocket Tuner, there are no defined sponsors, or financiers, to the project beyond the members of the group.

2.2 Project Motivation

The main motivation is to assist musicians, specifically guitar players, in tuning their instruments quickly, and accurately in any environment. Usually tuning a guitar accurately, is a tedious process that can take quite some time away from a musician. When on stage, recording, or just at a jam session, environmental noise is a thing that needs to be accounted for, and a tuner with a ¼" input isn't always readily available, or the musician needs to tune as they're playing. However, our project has the option of tuning with a ¼" cable for guitarists who can use this accurate form of measurement.

When starting out to learn the guitar, one of the most intimidating roadblocks is tuning the guitar correctly. Anyone should be able to learn an instrument, and this is a way in which our product could help people feel more confident in picking the guitar up. Whether the guitarist is just a beginner, or an expert playing in front of fans, we want to create an accurate, fast way of tuning that is comfortable and easy to use.

2.3 Project Description

The Pick Pocket Tuner includes a motor, a screen, buttons, a power switch, a controller, piezoelectric sensors, a ¼" input jack, and a basic housing to keep it all enclosed. As a method of powering the device, the group decided on a lithium-polymer battery that is rechargeable via a USB-C cable. As far as determining the method to find the frequency at which the strings on the guitar are reverberating, two vibration sensors were used. Meanwhile, the ¼" input jack is used for frequency readings from electric guitars and ¼" cable inputs.

The method of operation is as follows: the user takes this portable device, and it has a screen and button interface that allows the user to interact with the software that gives feedback to the user on whether the string is in tune once it is in use. The device then is placed with the tuning peg device on the tuning peg itself, and the user strums the desired string to be tuned. The device then uses both the sensors within to determine the vibrational frequency and then directs the motor to either adjust the string tension by turning the tuning peg clockwise or counterclockwise. The microcontroller will communicate with the display via an SPI connection and will communicate to the motor via a PWM connection.

Requirement Specifications	Measurement	Units	Interval
Tune a guitar in a maximum time	3	minutes	±1
String a new guitar in a maximum time	5	minutes	±1
Tune strings to the correct frequency	1	cents	±5
Battery life with respect to strings tuned	100	strings	±15
Max weight	2	pounds	±0.5
Ease-of-use interface with respect to time to select and tune a string	45	seconds	±10

Table 1, Requirement Specifications

2.4 Features and Options

After discussions within the group, and technical research, we have decided that the following list would add good functionality as a standard set of features and options to the Pick Pocket Tuner:

- Comfortable, ergonomic, and able to fit in one hand
- Ability to tune a guitar to alternative tunings (Drop-D, Drop-C, DADGAD, etc..)
- Tune new strings quickly from start to finish
- Input jack for tuning with a 1/4” cable

Our team believed that including these features and options, would give us a quality device we can be proud of with the final product. We were able to accomplish all of these while staying on time and keeping costs low.

2.5 Stretch Goals

After discussions within the group, and technical research, we have decided that the following list would constitute as a good set of stretch goals to be implemented into the Pick Pocket Tuner:

- Ability to tune other stringed instruments like a ukulele or a banjo
- Tune the instrument faster and more accurately

2.6 Restrictions and Constraints

After initial discussion and considerations, our group has determined that this is a list of the most crucial restrictions and constraints that we would face while developing, prototyping, and testing the Pick Pocket Tuner:

- Budget of the project
- Allotting enough time for the project
- Steep learning curves
- Supply chain issues (Difficulty in part acquisition)
- Competition in the market

There are no federal restrictions to be aware of for the project. We feel these requirements would demonstrate satisfactory engineering qualities on both the software and hardware side of things. Below are flowcharts that show how these requirements were achieved on both ends of the spectrum.

3.0 COVID-19 Supply Chain Issues

One of the most constraining issues that our group attempted to overcome was the supply chain issues that was happening during both semesters due to the COVID-19 pandemic. Electronics specifically are suffering a lot due to the pandemic, and an increase in demand does not help the situation. As our group was trying to order different components, we ran across issues of large delays, specifically, when trying to order a voltage regulator for the device. We have seen delays for almost two months, so our team has decided to come up with a mitigation plan.

3.1 COVID-19 Supply Mitigation Plan

One of the last components that we had to implement in our device was our voltage regulator, which had the most concern of arriving on time. Setting a plan in place to prepare for this was crucial to being able to deliver a product that was functional by the end of the semester.

The first crucial step in this mitigation plan was to order almost all the other components that we were planning on using. This did not mean resistors and capacitors necessarily, but components like the display, sensor, input jack, motor/driver, etc. Once these parts were determined, we were able to find the power consumed by all the components and we were able to choose a voltage supply that met all our design requirements. Once this was determined, we were able to pick a regulator for the system.

Even though this was a difficult issue for our team to overcome, it provided us with great experience on how to deal with these issues once we get jobs in the engineering industry. It also made sure we did not procrastinate and got most of our work done on a strict schedule, which allowed us a stress-free integration process. Some may find it easy to look at all the negative ramifications of the supply chain issues happening, but our team tried to look at the positive side, by using it as a tool to help us stay way ahead of schedule.

4.0 Budget Constraints

Our group was unable to secure a sponsor for our project, so the budget was split up between all the group members evenly. Since our expenses are coming from out of our own pockets, the team had to be careful not to make any unnecessary purchases, and we had to be extra careful when prototyping our project, to not burn or damage any components we planned on using in our final design.

4.1 Budget Mitigation Plan

Even though our team footed the bill for the entire project, it gave us an opportunity to learn how to improve our money handling skills. Our plan was quite simple, yet effective. We recorded the costs of everything anyone purchased in an excel document. At the end of senior design 2, we split the cost in four even amounts, and we paid each other back once all was said and done. Since we were all aware of the fact that we would be shelling out our own money for this project, it allowed us to pay attention to detail and keep waste to a minimum and efficiency to the maximum. Also, it provided us an opportunity to research more about all our components before we decided on buying certain products.

Using our own budget for this project could be looked at as a huge constraint, however, our team responded well, and everyone was on the same page about spending each other's money. We had to be extra focused and stayed on top of schedule to avoid paying any extra fees due to expedited shipping or ordering things last minute that could cost more money than we wanted to. Staying ahead of schedule not only gave us peace of mind for our project itself, but it also allowed us to not have to spend so much money.

5.0 Weekly Scheduled Meetings

Every week we had group meetings to discuss the progress of what has been completed and what needs to be completed. Some meetings were arranged in an online means, too, if there were incidents that kept members from having to be remote for reason that may be deemed reasonable. These regularly scheduled meetings were an important requirement that was needed to keep the team focused as a whole. They helped with keeping track of what each member needed to do and get taken care of. They also allowed team members to work together when maybe one team member was stuck and needed help completing a task so that they could keep up with the group. These meetings were typically held in the library every Tuesday and Thursday from usually 9am-12pm during senior design 1. During senior design 2, these meetings were held in the senior design lab usually between 9am and 5pm. Most meetings were productive but have most of the time been for check ins or updates on the tasks at hand. Most of the members had a full load of classes so it made it harder to accommodate for each member. We also used ClickUp which allowed easier planning and dividing of tasks for the main parts that needed to be done each time.

A good amount of timing issues needed to be taken care of mainly due to other class loads and outside jobs that members had. All members were understanding towards the daily struggles that each member has outside lives and other important tasks or issues that may arise. There have been a few missed meeting days that have happened but not a substantial amount that caused any major issues with progress. Some meetings have ended in a stuck scenario on what to do next or how to go about the next task that needs to get done. There are different ways to solve each issue, it just comes down to determining which is the right method to go about getting it done.

6.0 Miscellaneous Constraints

There are many constraints one can run into while designing and producing a project. Time constraints are very common, hence why keeping a strict schedule is so important. Our team was overall a very busy group, so it was essential to complete all our schoolwork on time or early. This especially included classes outside of senior design. If we were not performing well in any of the classes we were taking, then senior design would suffer significantly, and this needed to be avoided at all costs.

Another constraint was the lack of knowledge we might have had in certain areas of our project. Working on a project can be discouraging if we get stuck or just have a hard time understanding certain concepts. This gave us an opportunity for growth in skill and knowledge of the projects we were working on. This does not only include our senior design project, but on any project, we might have been working on outside of school. Also remembering that we are not alone is important when questions like these arise. We have resources on the internet, but if we are not experiencing any progress, then we have so many resources that the college can offer us.

There is a wealth of knowledge at our disposal, and we took advantage of that. This constraint can greatly affect our project by reducing the amount of time we are able to spend on building and designing other parts of our device. The longer we were stuck on something, the shorter amount of time we had to make a working useable project.

7.0 Design Process

An effective design process was essential to meet all deliverables for the year. Our design process consisted of technical investigation, application of engineering fundamentals, and testing (preliminary testing, subsystem testing and overall testing). These kept us on the correct track and allowed us to learn a lot about each of the components we chose for our project. It allowed us to manipulate the components in a way that gave us the best results.

7.1 Technical Investigation

Our project started with a great amount of technical investigation. All the investigation our team conducted acted as the backbone of our project. Without going through datasheets, surfing websites, watching essential videos, we would not have been able to begin working on our project. As the team became more familiar with the components we planned on ordering, the more likely we were to make them work together quickly and easily.

The team would compare at least three different products we could use for one component and go through all the datasheets and reviews and see how easy they are to implement with our other products and narrow them down to one final product we ordered. However, if for some reason the product we ordered did not work, we would have two viable back up options we could use as a second option. Technical investigation was another key element to this project as we conducted research effectively and thoroughly to not suffer when the time came to assemble our product.

7.2 Applications of Engineering Fundamentals

Throughout our time at college, we have learned how to think critically and attack different problems in all sorts of fashions. This project gave us students an opportunity to apply what we have learned to be able to demonstrate to prospective employers, and our professors, that we are capable of being engineers. On the hardware side of things, we applied what we have learned about Kirchhoff's Law to calculate the current in and out of all the components to make sure that nothing is being overloaded, or if something is not receiving enough current.

Additionally, we also calculated power consumption, as that was very important to our project specifically since there are many components that are drawing power. We also wanted to implement what we have learned in embedded systems such as different lines of communication, writing code, and enabling different pins of the device. Other software components were implemented as well like creating an algorithm and making a graphical user interface for the display.

We as students have been given all the foundational tools, we need to complete this project, however, as bourgeoning engineers we needed to become more familiar with new methodologies, practices, and technologies to be able to specialize in all the different areas involved to create a final working product. Our team is very competent, and through applying the knowledge we have gained throughout our academic careers, we are confident we performed well when necessary to deliver our final working product, through the utilization of all the Engineering Fundamentals.

7.3 Preliminary Testing

We started to develop parts of our project by conducting preliminary tests. This gave us a solid foundation to not only start developing subsystems, but this also allowed us to begin more in-depth research on how pieces will integrate into the whole system, and with one another. Preliminary testing essentially was composed of one "generalized" step, getting a response for each component as it pertains to our project. For example, we implemented buttons on our project, so the button we decided to use was connected to our development board through a breadboard, and we tried to get a response from pressing the button.

An example of something we have already performed would be us turning a prototype servo motor. This was successful as it taught us how to manipulate the speed and rotation parameters of the motor. This process has been incredibly insightful as this was a crucial step when our finalized design became entirely autonomous. The purpose of this Preliminary Testing was to bring the components to their basic operational capacities, so that the group can gain an in-depth understanding of each component that was needed, and/or utilized, to deliver a finalized working product.

7.4 Subsystem Testing

Subsystem testing was also a crucial next step to our development process. This process consisted of trying to get two or three different related components to interact with one another. This was based off responses from those different components that have been part of the Preliminary Testing. An example of this Subsystem Testing would be using the buttons to scroll through the graphical user interface (GUI) we had on our screen. On a higher level, this demonstrated to us that the buttons can be integrated with the display, the GUI works, the software detailing the menus works, and that the processor/controller was able to execute the necessary code to realize the function we were testing, scrolling through a menu using the buttons as a User Interface.

A step further was to utilize said buttons to then select an option on the menu being shown on the display, then be able to visually see the correct output for whatever the option chosen was. Subsystem testing was like putting a complex jigsaw puzzle together. This process assisted the group to see the overall, broad-spectrum/picture of the whole project. The successes of the Subsystem Testing were a crucial milestone in our implementation process, and successes in this allowed us to move our project into the overall testing phase.

7.5 Overall Testing

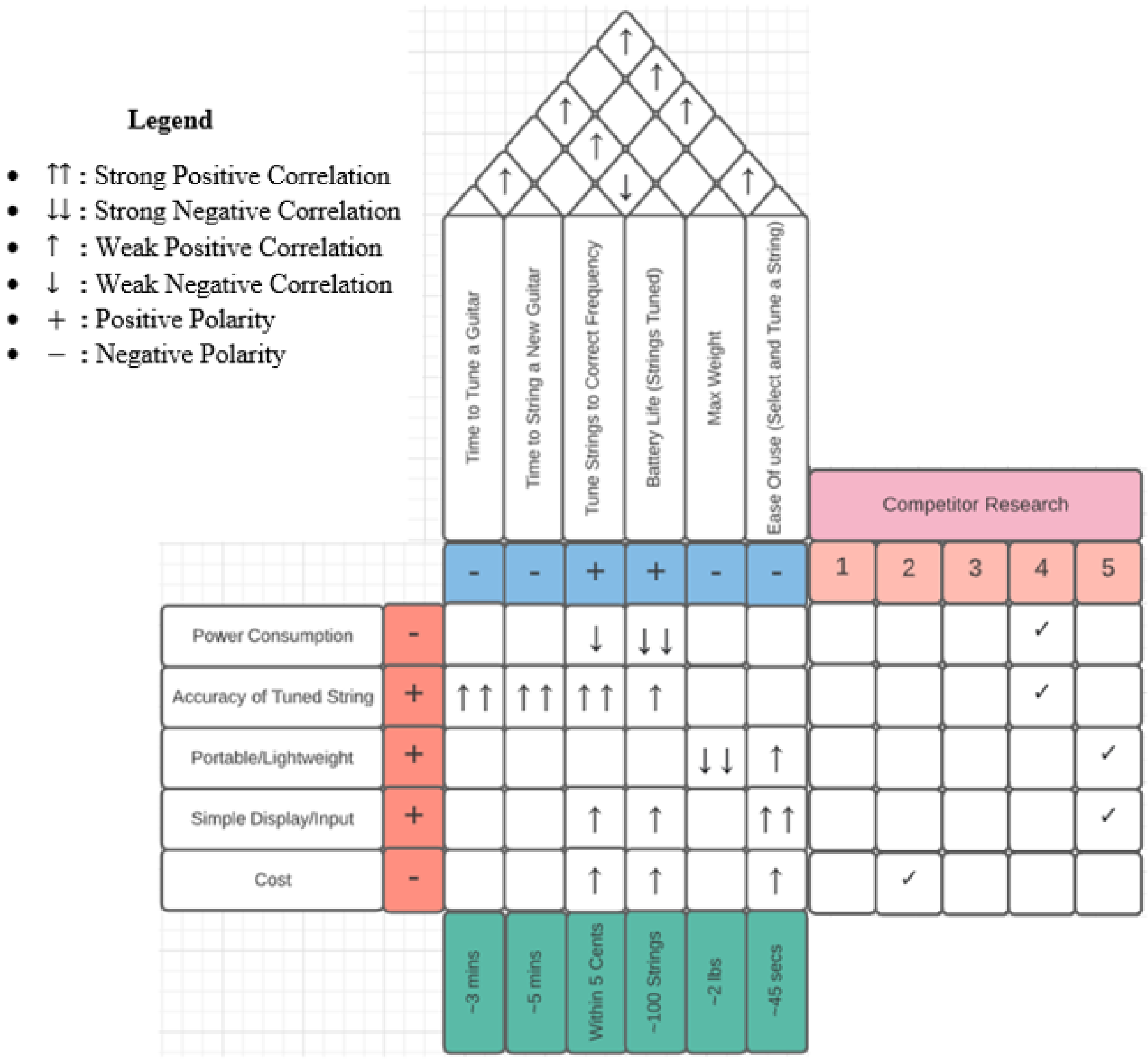
Once we successfully determined that the different components involved were interacting with one another in the correct manner, we then went into what our team referred to as overall testing. This process was when the prototyping process started coming to fruition. By this point we have gotten most, if not all the necessary components to communicate, relay the necessary/correct information between one another efficiently, effectively, and as error-free as possible. This demonstrated that we were now in a place to begin the implementation of all the features and functions we outlined in the Requirements Specifications section that we determined our project would include. An example of this sort of testing would be selecting an option on the interface, plucking the string we want to have tuned, the sensor picking up the vibrations, the algorithm running and relaying the information necessary to the motor controller to then have the motor turn to try and tune the guitar string, and having this process repeating until the string is completely in the correct tuning.

Overall testing was the process that consumed the most amount of time. Once we determined that the project was comfortable at this point in our development process, we began the ordering process of the necessary PCB’s. The reason for this is because when we got the device working on a breadboard prototype, we did not want to waste any time, especially due to the logistic issues that were plaguing the technology-based creator space due to shortages and/or production issues world-wide. We were aware that the ordering of components would be tricky both semesters, so we needed to start perfecting our project as early in the semester as possible. This resulted in testing being an essential step for creating any kind of product, and our Senior Design project is no exception. Being proactive in these steps helped us tremendously in our endeavor and made our lives easier, and ideally lead to the successful delivery of our final project.

8.0 House of Quality Diagram

From our house of quality diagram, we see that most of the client requirements have a positive correlation to our technical requirements. With most of our requirements being related to time. Time is a valuable aspect with our product as our main goal was to tune instruments automatically and efficiently. Since this is a valuable aspect, our requirements were built around and positively correlate with it.

This is seen in the correlation matrix at the top of our diagram. Our competition is the Roadie 3 automatic guitar tuner and performs quite well with the customer requirements show in our diagram. However, the cost is where the Roadie 3 does not perform well as it is quite expensive. We tried to outperform the competition in this customer requirement to put our product ahead.


Legend

- ↑↑ : Strong Positive Correlation
- ↓↓ : Strong Negative Correlation
- ↑ : Weak Positive Correlation
- ↓ : Weak Negative Correlation
- + : Positive Polarity
- – : Negative Polarity

		Time to Tune a Guitar	Time to String a New Guitar	Tune Strings to Correct Frequency	Battery Life (Strings Tuned)	Max Weight	Ease Of use (Select and Tune a String)	Competitor Research				
		-	-	+	+	-	-	1	2	3	4	5
Power Consumption	-			↓	↓↓						✓	
Accuracy of Tuned String	+	↑↑	↑↑	↑↑	↑						✓	
Portable/Lightweight	+					↓↓	↑					✓
Simple Display/Input	+			↑	↑		↑↑					✓
Cost	-			↑	↑		↑		✓			
		~3 mins	~5 mins	Within 5 Cents	~100 Strings	~2 lbs	~45 secs					

Figure 1, House of Quality Diagram

9.0 Competition

Initially, to begin analyzing the competition one must look at specifically vibrational sensor based chromatic/strobe tuners to fully be able to understand how the necessity for an automatic guitar tuner came about. As a result, one of the most popular, and accurate, tuners on the market that utilize vibration sensing technology is the Snark ST-8 Super Tight Clip On Tuner. This chromatic tuner is a simple device with a LCD screen that informs the user whether the current string being plucked is sharp or flat, based on the chromatic tuning system that it employs.

The way the Snark ST-8 operates is that it has a plastic clip, that has a silicone/rubber padding that allow the vibration waves to traverse from the point of origin on the headstock, through the clip and the attached stem, and into the main housing of the ST-

8 device. Once the waves reach the housing, there is a piezoelectric disc within the housing that would reverberate indicating that here is some sort of vibration happening at a certain frequency that then is relayed onto the user based on what the tuning is desired. The ST-8 is a very popular tuner because it is capable of being used while in loud environments, however the tuning itself is still manual. Based on all this, our group decided that using this device as a ‘base-point’ to begin our research was a solid foundation.

A device like this already exists in market, the Roadie3 designed and sold by Band Industries, Inc. This product by Band Industries has been in market since about 2017 and has gone through 3 iterations, and they have a specific device for bass guitars. One of the limitations that this device has is that there is no additional input to verify the accuracy of the string tuning. To bypass this issue, our project was implemented so that the accuracy is within a ± of 5 cents. Additionally, the choice of motor was one that allows us to make fine adjustments of the tuning peg to obtain that desired accuracy. We also wanted to find a way to combine their two products into one.

Our project was going to attempt to find a motor that can turn a bass guitar peg, which allowed for our tuner to tune both instruments. We also implemented an extra jack that allowed an artist to tune very accurately to the pickups that are already placed on the guitar, by the company that made the guitar. Another potential competitive edge we gained was by adding an option to tune a new string quickly.

The Roadie3 does not have an option for tuning a new string from start to finish, however it does have a manual driver, that turns the peg whenever a button is pushed by the user. We implemented a software setting that allows the user to select an option that tunes a string from having zero tension to the correct tension and frequency. Finally, one last small software improvement was to add a requirement for tuning accuracy when the string has too high of a frequency.

When tuning a guitar, if the string has too sharp of a pitch, the person will tune below the frequency, then back up, to ensure that the string is under tension and at the correct frequency. This seems like a small feature; however, all musicians have a standard of tuning the string low if it is high and then back up. The Roadie3 does not have a feature like this, therefore, the guitar goes out of tune more quickly.

Another product that already exists on the market that is like the Roadie3, is called the Jowoom T2 Smart Tuner. The way it operates, assumedly so, is like that of the Roadie 3 as it utilizes sensors embedded within the device housing to sense the vibrational response from strumming a string to then activate a motor into turning at a variable rate to turn the peg winder so as to tune the guitar.

The main components are similar between the T2 and the Roadie3. They both analyze the frequency and tune the guitar using a motor based on the frequency response and the difference from what the current pitch of the string is and what the desired pitch is. In both cases the devices have already entered market and are therefore primary competitors, however the main drawbacks of the T2, based on reviews, is that the motor used on the device is not very fast, and is prone to slippage when trying to wind a string fast. Similarly, to the Roadie3, the T2 is unable to tune a Bass Guitar as the motor inside the T2 does not produce enough torque to spin the geared tuning peg on the heads of Bass Guitars. With that said, based on reviews, the consistent feature that the T2 does have over the Roadie3 is the cost. With the Roadie3 costing upwards of $130 per unit, it doesn't make full sense why someone would choose the Roadie3 over the T2 when they basically both have the same feature set and similar limitations.

Our group believed that these features were enough to compete with the Roadie3 and deliver a product that goes above and beyond. The hope was to be able to implement all these changes, but we are confident that we achieved most of these.

10.0 Tuning Background

A guitar, as well as majority of all other stringed instruments, have simple operational concepts. Through history, there have been different stringed instruments that have been utilized to make some sort of tonal sound, whether for ritual or entertainment we can't really be sure of. However, the operational concept is that there is a string, some material, usually wood, that provides tension, and a body of said instrument. The methods of playing the instrument would vary from plucking to using another tensioned string, both of which had the goal to create vibrations that would then produce sound, at the appropriate pitch.

Some of the oldest stringed instruments were bowed, and closely resemble what is today a violin, and as a result the basic design for stringed instruments hasn't really changed drastically and can be translated across different instruments. In the design, there is typically a body, a bridge, a neck, a head which can be referred to as the pegbox, and tuning pegs. The idea is that the body neck and head are connected in a linear fashion (see Figure 2 and Figure 3 below for reference). From there the strings that are to be used are fed through the appropriate orientation to be anchored at the bridge, and then they are fed to the tuning peg. The tuning peg, through a series of gears, provides torque that then translates to tension on the string. This change in tension is effectively what changes the pitch/intonation of the string, and by relation the vibrational frequency. As in a violin, a lute, mandolin, or guitar, the operational concept remains the same, however in further discussion, as it pertains to our project, we are
referring to the guitar explicitly.

Like a violin, with a guitar there is a bridge, neck, body, head, and pegs, the difference is the size of the instrument, the shape, and number of strings, but fundamentally the tuning operates the same. The tension for a guitar string varies through each string manufacturer, metal used, coatings, gauge, whether it's meant for electric or acoustic guitars, and a slew of other variables that are yet to be determined for a baseline for our project, however, tension is directly representative of the tuning and vice versa once all other variables have been accounted for, so there are ways to mathematically represent the tunings in relation to their tension, and frequency giving us a way to quickly adjust if necessary for other instruments.

As such, a guitar string tuning process begins once the string is anchored at the bridge and fed through the tuning peg. The peg is then turned and depending on the orientation of the peg and head this turning would either be clockwise or counterclockwise, this turning of the peg is what provides the tension. More tension means that when the string is played the higher the frequency the string vibrates at and thus the higher the pitch, conversely lower tension means lower pitch, and frequency.

For a guitar, the tuning standards have been established for some time, and many people will argue about which frequency to use. In our project, and going forward, when we refer to standard tuning it is to mean the EADGBe tuning, which is based on the pitch standard of A440. Breaking this down, this means that if you play guitar and are trying to tune it, the thickest and lowest string is the E, followed by A, all the way until you get to the thinnest which is the e string, on a traditional, right-handed guitar it would be that the strings go from left to right, from thickest to thinnest.

The A440 standard is referred to as the Stuttgart Pitch, and it is corresponding to the 440Hz frequency for the A note above the middle C in a Piano. This A440 pitch is standardized by the International Organization for Standardization as ISO 16, and as such it is used as a reference frequency to calibrate the equipment and instruments to be used in tuning. EADGBe was chosen as the ‘standard’ tuning for a guitar, mostly due to playability across keys and scales, making the arrangement of music to be the most pragmatic in this tuning. With that said, since we are operating under the generally accepted standardization of EADGBe tuning, the corresponding frequency values for each of the strings under the correct tuning is characterized by the following table, with accompanying Scientific Musical Notation.

Tuning	Frequency	Units	Scientific Musical Notation
E	82	Hz	E2
A	110	Hz	A2
D	147	Hz	D3
G	196	Hz	G3
B	247	Hz	B3
e	330	Hz	E4

Table 2, Frequencies for a Guitar Standard Tuning

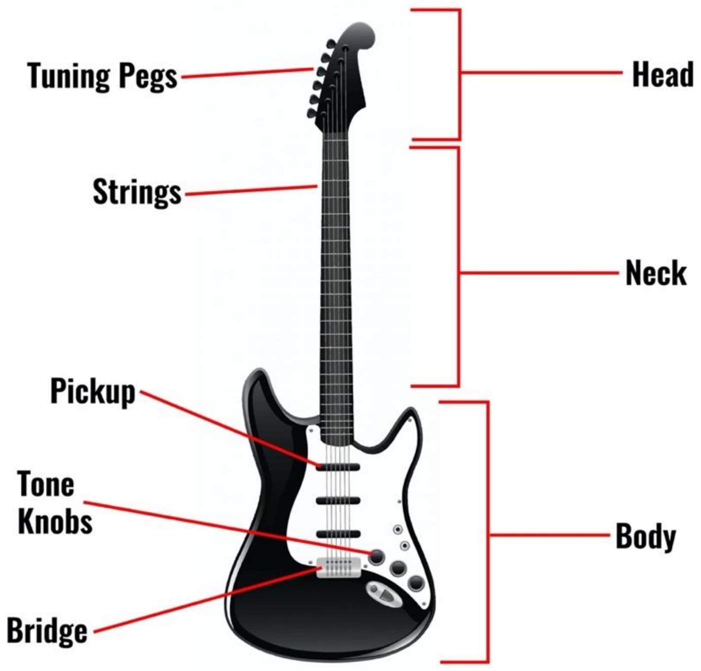


Figure 2, Representation of Physical Components on an Electric Guitar

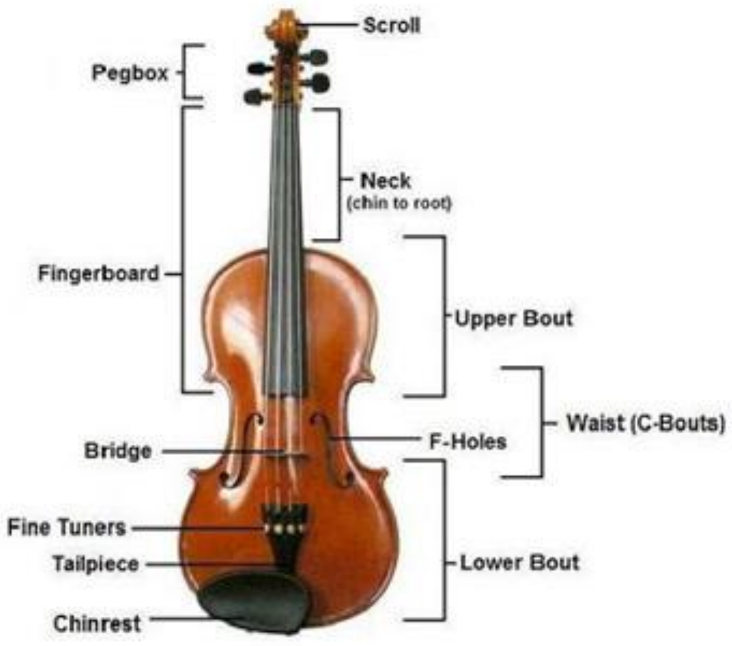


Figure 3, Representation of Physical Components on a Violin

11.0 Hardware Block Diagram

The main setup for the hardware diagram consists of the battery tied to an on/off switch to keep the system from running all the time. Next dropping down in voltage to reach the correct voltage level for the processor to the display and the regulator to the motor. The processor takes readings from the vibration sensor and tells the driver what to do which controls the motor to tighten or loosen the pegs on the instrument. The processor also displays values on the LCD display which has inputs and buttons that are read on the processor to change the settings on the display and select on a small screen interface.

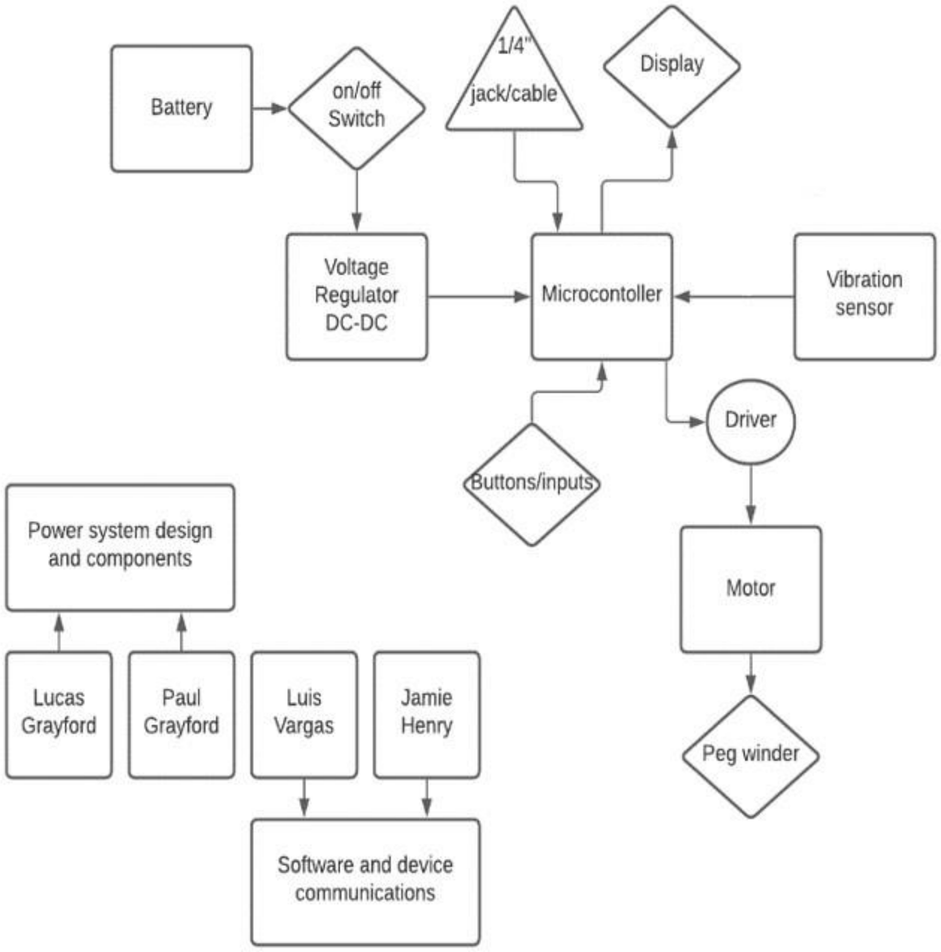


Figure 4, Hardware Block Diagram

12.0 Physical Components

12.1.1 ¼” Output Jack

One of the electrical components that was used in our project was the ¼” output jack that a cable will be plugged into to read the frequency from the pickup of the actual guitar. There are four commonly used output jacks. These being mono, stereo and TRS (tip-ring-sleeve). Each type of jack has its own advantage, however for the sake of the project, we believed that the mono jack and the stereo jack were the most useful jacks for this project.

The mono jack only has two connections, a live/hot connection, and a ground pin. The live pin picks up the signal produced by the pickups in the acoustic or electric guitar and sends them through to the output on our device. This means that we can read exactly what the guitar is producing, making a more accurate tuning experience. The way our project will implement these will be by connecting the device to our PCB and processing the analog signals into digital signals. This will aid in communicating these digital values to our device.

The stereo jack has an extra ground pin that completes the circuit in the active electronics on the guitar themselves. This could be useful to implement into our project, to complete a circuit for the device to detect the jack immediately, thus bypassing the vibration sensor all together. More research was conducted through testing both jacks, and their usefulness. This component will add hardware complexity to the project.

12.1.2 Selection of Output Jack

Since one of our requirements for this project was accuracy, we did not want to sacrifice on components that will gather data. There are many products that could work well in our project; however, we tested the accuracy of two different output jacks. One is a simple stereo jack, and the other is a Pure Tone Mono Multi-Contact ¼” output jack. The stereo jack has three pins as stated before, and the Pure Tone jack has 5 pins, two positive pins and three ground pins.

The stereo jack is used in many guitars and bass guitars and works very well to complete the job that is required of it. The pins create a contact with the cable that is plugged into it and reads an analog signal from the instrument. This analog signal can then be amplified through an amplifier or go into a tuning pedal for the guitarist to fine tune his instrument without the hinderance of any unwanted noise or interference.

One of the common issues with this jack, is over time, the connection pins can become less sensitive and become looser over normal use. This can cause unwanted crackling or interference, which is the sign of a poor electrical connection. We want to deliver a quality product that can be reliable for a long time after it is purchased. This degradation over time will be accounted for in our project.

The Pure Tone output jack has two live connection pins and two ground connection pins that connect to the cable coming from the guitar. It also has an extra ground pin to complete the connection in the circuit it plugs into. The advantage of having these two extra pins is reliability, accuracy, and longevity of the product. This jack will be put to the test against the stereo jack to see that if the accuracy and reliability gained from this product will offset the cost of a normal stereo jack.

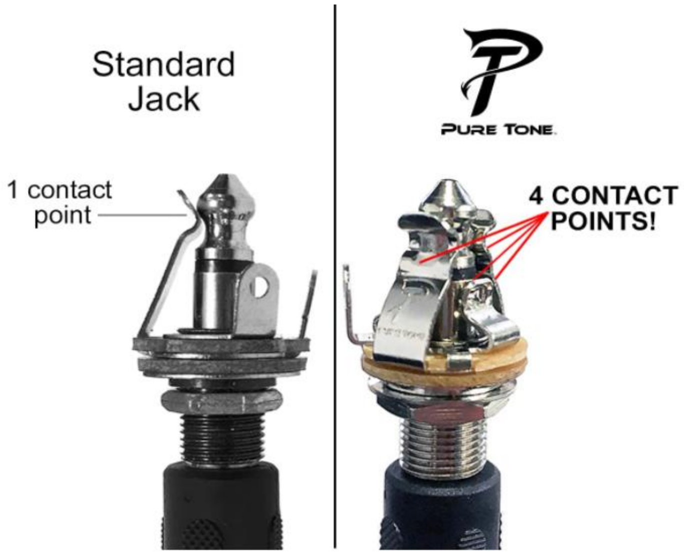


Figure 5, Comparison of Pure Tone output jack to a standard stereo output jack

12.1.3 Testing Process

The plan for testing these jacks is laid out in a few simple steps:

1. Tune a guitar with a device that is already known to be accurate and reliable. (Guitar must have an output jack already.)
2. Connect an oscilloscope probe to each jack, and ensure the connections are sturdy
3. Use a ¼" cable to connect the output jack from the guitar to the output jack that is being tested. (Ensure all connections made are still solid and on the correct pins.)
4. Pluck the desired guitar string and watch the oscilloscope measure the frequency. This process will be repeated thoroughly to ensure accurate readings.
5. The frequency that is clearest and closest to the frequency attained when the guitar was tuned originally, will determine which output jack will be used in our final project.

The team's prediction was that the Pure Tone output jack will read a signal that has less noise and would be more accurate to what the guitar is producing. This was due to the extra pins, and the quality of product.

12.1.4 Preliminary Testing Results

The testing of the Pure Tone output jack was limited to being able to detect a signal altogether. Further testing was conducted to determine which jack was better or worse for our purposes.

12.1.5 Implementation

The output jack was mounted firmly with hardware, on an accessible location on the final device. The pins on the inside will be connected to the PCB through wires that are soldered onto the output jack and connecting to pins on the circuit board. These connections were connected to an analog to digital converter (ADC) to do the obvious and convert an analog signal to a digital signal that is read by the processor.

If the user decided to use this form of input, then the piezoelectric sensor readings will be bypassed. This came from the expectation that the jack will be a more exact reading than the sensor, and we did not want to confuse the processor, on which data to read. The idea for this bypass was implemented in the hardware. The extra ground pin on the jack that completes the circuit bypasses and cuts off the sensor signal to only have the output jack being read.

12.2 Motor

For the overall motor there were multiple options to use. The main two options were between using a stepper motor or a servo motor. The system could have ended up using either one, more research and testing was required to see which one will exactly be the best to use. The whole plan varied on how we wanted to apply the motor in the end. With the battery being a DC power source there was a preference to attain a DC type motor and not have to deal with a larger power drawing AC motor. The stepper motor was the first option that was being considered since they are accurate, precise, high torque, efficient, and low speed. It is an open loop system, but it can be made a closed loop system if it needs to be. The vibration sensor is going to be telling the processor the values which then the processor will tell the driver which will tell them motor which way it needs to turn and by how many steps it will be.

Stepper motors are relatively cheap and have a longer life span than most servo motors. Servo motors typically have a good to average torque yield and usually have a better torque yield at high rpm from the 1000+ rpm range. For this case on restringing a guitar it is a little excessive to go past 500 rpm. Most stepper motors start losing most of their torque past the 500-1000 rpm marker, which again there is no need to go past that point. Also, if choosing a stepper motor the choice of phase would be additional factor to consider whether to use a 2-phase or a 5-phase stepper motor. The best option between those two would be using a 5-phase if one can be found that fits in the budget. 5-phase would provide more torque, less noise, and more accuracy by lowering the rotation for each step it takes. Servo motors would just waste more power not only that the servo motors usually cost around double what stepper motors costs, but servo motors life span is about half as much as a stepper motor.

If there is no correctional feedback, and the stepper motor can give enough torque while maintaining accuracy that is the current method of choice. Most stepper motors can have a relatively fast rpm rate around 1000 rpm which would allow for extremely fast restring speeds. The Roadie 3 does 110 rpm where ours could be almost 10 times faster than the leading brand of the Roadie. Comparisons were tested between both motors to determine which one will be the best use. Roadie does not state the motor type that they used for their design, so since the Roadie 3 was attained and dismantled, a brushless DC motor was seen and used in the project. Once that is done the calculations for the torque and be found and used to help determine the motor and if the continued use for the stepper motor will be used.

12.2.1 Motor Testing

There were varying motors to choose from and the final decision for which motor type was a brushless DC motor for the system. One of the main motors that was used for testing is a 5V Stepper motor 28BYJ-48. It is a relatively small stepper motor that worked with the current software that was being used and developed in the project.
The first test was to see if the software was even compliant with the motor's controls. We were able to get the code to function with the vibration sensor plate and at least turn when a vibration was picked up and it was successful in turning the motor head. The only issue was breaking down the calculations of the actual vibration that was being received from the sensor and processed by the processor and calculating whether the motor needs to turn left or right and by how much. It sounded simpler than what it is made out to be. There are other methods that some musicians use to tune their instruments by tuning it to the correct frequency then tuning up and back down a hair to tension the string better. Which this method is an additional preset to add when getting the final reading from the sensor after the string is plucked and reads the correct relative frequency for the string.

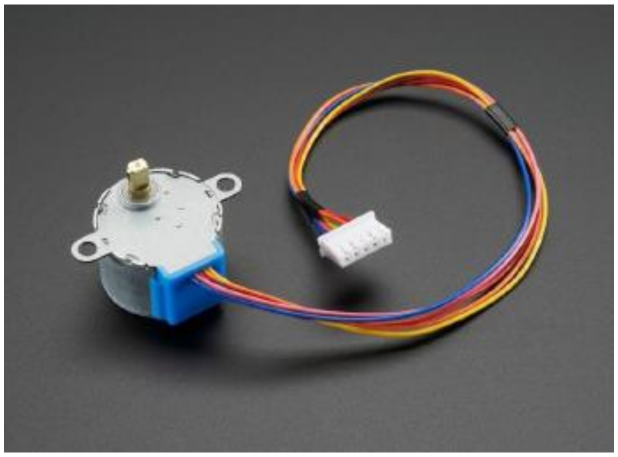


Figure 6, 5V Stepper Motor (Model: 28BYJ-48)

This motor was more for experiment testing than being the actual motor that was used in the final design. This motor was meant more for learning purposes rather than much useability purposes. It is a unipolar stepper motor and does 32 step with a 1/64 gear reduction giving it 2052 steps per revolution which was more than what we really needed. The motor also falls very short with its 6rpm which is a tenth of the speed of the Roadie. It does not have a high enough torque rating for what is needed to turn a tuning knob from what has been researched for what is needed to turn a knob is around .2 Nm of force, this motor only produces about .015 Nm which is not practical and is less than a tenth of what is needed. Again, it was a test motor, which another result was found that these motors are relatively easy to break.

The motor head could not withstand a good amount of force applied to the peg head before it has an internal break which caused it to no longer work. Its plausible breaking point was from trying to apply force to the peg head of the motor trying to put the tuner head piece on the end of the motor to conduct further testing. After the tuner head was attached to the motor there was a test to see if the motor could even turn a guitar peg, but before that could happen the motor would not turn at all. There was either a failure in the connection lines and hooking up the testing setup or a failure in the motor. It turned out to be a failure in the motor since the software would run and the motor would make noise, so it was trying to turn but most likely one of the gears got stripped when putting the tuner head on with a good amount of force. This only caused a slight delay of needing to get more test motors since they are ample in supply.

12.2.2 Motor Research

There was also vested money and research in a Nema 17 Bi-polar stepper motor which we did not put to test for two main reasons, it is heavy and does not meet the handheld friendly requirement since no one wants to stick an almost 1 lb. block in their pocket and with all the other components that need to fit in the housing it would not make it practical for a handheld device, also the driver requires at least 10-28 VDC for the driver even though the actual stepper motor is rated for 2.9 V it would take a power supply to test. The motor exceeds in the torque rating that is needed where this motor was rated .46 Nm which was a bit excessive but would allow for the stretch goals tuning. Looking at the competitors Roadie 3 we were uncertain which type of motor they decided to use for their system until a Roadie 3 was attained, tested, and dismantled to see how they went about picking components for the build. Most of the motors that are supplied are either too much or too little and there is no nice in between. The thought was that the Roadie uses a planetary styles gearbox motor that would fit the way it is styled in the housing since they are typically longer and thinner type motor and give a decent amount of torque behind them to turn most knob pegs but a bass guitar’s tuning knob.

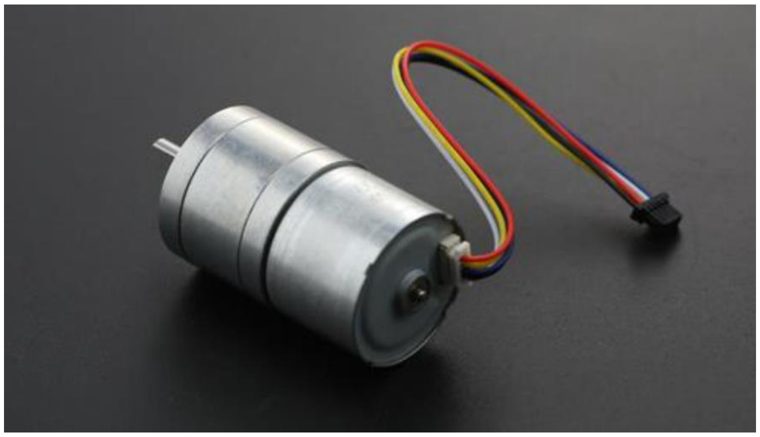


Figure 7, Brushless DC Motor with Encoder 12V 159rpm

The motor chosen was another Nema 16 stepper motor which was a step down we went for more of an exact rating motor for torque, where this motor has .21 Nm of torque, and has a lower current draw of only .4 A compared to the Nema 17 that was rated for 2 A. It also weights about half as much as the other motor. The only problem with the motor was that it is rated for 12 V which is a little higher than what was wanted so varying on the power supply that was used for the system which led to the implementation of a step-up voltage to supply power to the motor to run it at max efficiency.

After further consideration for a finalized motor, we decided to go with a brushless DC that functions at 12V that can function up to 159 rpm. We decided to go with this motor for multiple reasons. The Jowoom used a very similar motor, and we were unable to find a motor that met the qualifications that their motor was. This motor came with an encoder so there was no extra need for a on board driver which helped save room and added costs on to the final design for the tuner. The motor is slightly smaller than the Jowoom and is only about 20 rpms less than the competition of the Jowoom, but the main competition is going to be more of the Roadie 3 than the Jowoom. The Roadie 3 motor is 110 rpm, so our project was still overall faster at restringing the instrument and tuning. The motor also reaches the proper torque that was needed to turn the tuning pegs. It has a reduction ratio of 45:1, when we were thinking that we needed to be super accurate with the tuning we were a little wrong when thinking about the amount that the motor needs to turn. When we used the Jowoom to tune a standard electric guitar with the Jowoom and it did a good job for a quick tune, but its interface was lacking, but besides that point their motor was good enough in accuracy. The holding torque is .231 Nm which falls into the range that is needed. The motor was compatible with the Arduino software that was used. The motor is a 5-pin motor for wire 1:PWM, 2:Power-, 3:direction, 4:FG, 5:Power+.

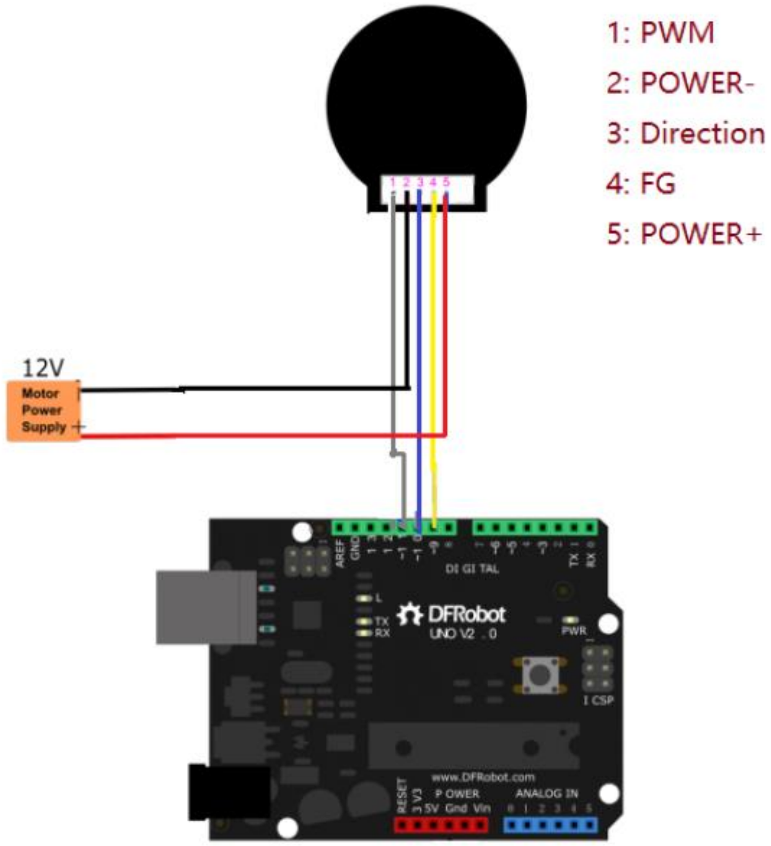


Figure 8, Line Connections for Arduino (will vary from ESP8266 connections slightly)

The supplier for the motor is DigiKey for $19.90 with 20 in stock. It is rated for 12 volts and does 159 rpm, it has a stall current of .7 amps meaning no matter how much torque the motor is fighting that the current will not exceed. Technically the motor has a base rpm of 7100-7300 rpm internally then has a gear reduction ratio of 45:1 dropping it down to the 159 rpm, where it sacrifices speed for torque, allowing this small motor to produce the torque that is needed. There was no given value for what the regular amperage will run at but estimations are about half of that.

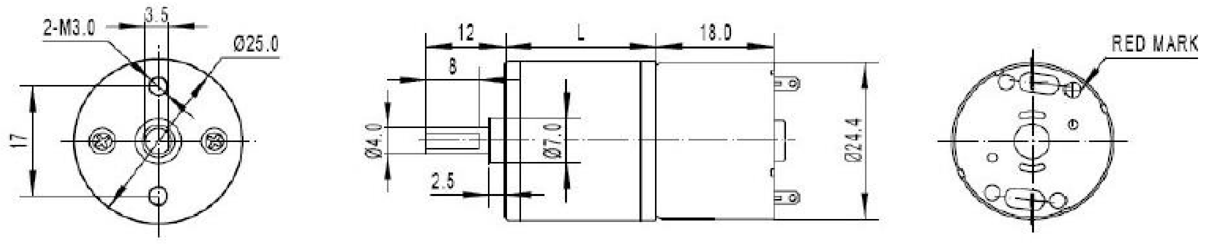


Figure 9, 12V DC Motor Dimensions in mm

12.2.3 Motor Conclusion

The motor was a key part for this device to work and since stepper motors are relatively accurate and can have a relatively high holding torque it should be good enough for the design even aside from its blocky size, but in our case the later change to an even lighter weight motor was a better method while keeping the same torque although it was no longer a stepper motor. There are a lot of motors to choose from but whether they would be the right fit for the system was questionable since there was already a fair amount of head way that was made on the software side for programming the test motor to work and being along the lines of a stepper motor even though the testing has been done with a unipolar stepper motor, so small changes were needed to compensate for the change in the end system. The most difficult part was trying to find a motor that would meet all system requirements without copying the competitors motor even though the motor that they used was unknown and may be more simplified, also the torque rating was unknown that they used.

12.3 Driver/Pulse Width Modulator

Along with using the stepper motor, a stepper motor driver would be required so the pulse width modulations that are sent to the driver can tell the motor how many turns will be needed. There may have been a need for a pulse generator to send a signal to the driver to then tell the motor what it needs to do. Correct drivers must correspond with the proper phase motor since a 2-phase motor might not work with a 5-phase driver and vice versa. If a stepper motor was used it would have also made it easier to have preset restringing options for the tuner rather than having a manual wind-up. There are also other integrated drivers that would be heavily considered.

Using an integrated driver would have helped save a lot of space compared to using most block drivers that most of the time are used for controlling multiple motors, and would take up the space of someone’s hand, which would have ruined the whole design of being handheld in one hand. Since only one motor was used for the tuner, the integrated driver would only focus on that singular stepper motor. The integrated driver is substantially smaller than most drivers, they are small and fit on the back side of the motor which makes it convenient for saving on the housing space overall. A more accurate driver can be accounted for once the proper motor is selected for use.

12.3.1 Driver/Pulse Width Modulator Testing

Most of the testing that was done for the driver was the ULN2003 driver board although it was mainly used for the software testing purposes with the current 28BYJ-48 5 V stepper motor this driver is only compatible with unipolar stepper motors. When the actual swap was made to the bipolar stepper motor the driver was no longer useful for testing purposes and the SN754410NE was used for testing the Nema 16 bipolar stepper motor when it was acquired.

Figure 10, UL2003 Driver Board (left), L298N Motor Driver Control Board (right)

This board was only used in testing purposes for the software and was not included in the final components of the tuner. The driver was compatible with the ESP8266 processor that the main programming was being implemented with. When testing for the bipolar stepper motor the L298N motor driver control board supplied by Amazon for $10 for a pack of 4 was used for the testing of the Nema 16 motor, but it was not implemented in the final design.

Due to changing the motor to an even smaller motor that uses and encoder. A driver was still able to be used for the motor, but with the simpler motor that was used in the final design. After testing, the driver was not used in the final design. The previous driver that was going to be used was said to be a wrong choice for the final design and was removed from the final system design.

12.3.2 Driver/Pulse Width Modulator Conclusion/Encoder

There was no need for a pulse width modulator since the servo motor was a throw away option, but the driver was necessary for the motor to function the way we want it to for the tuner. There were encoder brushless motors, but they are harder to control and use pulse width modulation, rather than for the stepper motor and driver can accurately be told how much to turn and knowing how much it will turn. It makes life a little easier for being able to tell the driver to tell the motor how many steps need to be done for a quick restring so it can already be close to being tuned and would only need smaller fine tuning afterwards. There are ways to run the motor without the driver but there would have been a need for more testing with the new motor with the encoder. Usually there are usually four types of encoders being an optical, mechanical, electromagnetic, magnetic encoder type. The Jowoom uses a magnetic encoder due to the magnet that juts out on the back of the whole motor encoder section. All that is known for the encoder is that it is incremental motor type, but between the different types of methods that it uses to read was not stated in the data sheet between the four different types and it is just known that it is not a magnetic type at least. The only thing that is known is that it uses pulse width modulation to control the speed and rotation of the motor, if the encoder is completely taken apart to determine what type it exactly is it will be stated later otherwise there is no risk wanted in breaking the current motor just the find the exact type.

12.4 Peg Winder

The peg winder is the head attachment at the end of the tuner that grabs the peg heads to be tightened or loosened. It may be a simple piece, but it needed to be able to function with enough strength so that it wouldn't break after a couple of windings. The thought was to design it out of a strong thick plastic that allowed the vibration sensor to make proper readings through the material of choice. There were two potential design heads that were tested, one of hard plastic and another made of metal that needed to be coated with clear epoxy so not to scratch the actual pegs of the instruments being tuned.

There were additional thoughts on adding an interchangeable peg winder head for the tip of the motor since the pegs on a bass guitar are much larger than average guitars. If the winder head was too small, it might break or damage the motor since it would not allow the proper torque to be applied towards the tuning pegs of the guitar. There was also the possibility to make a peg winder head that would be ergonomic and usable for both plain guitars and bass guitars.

12.4.1 Peg Winder Testing

For the peg winder there were a variety of peg winding heads to choose from. The competitor chose to use a 4 prong styled head piece for their tuner and it being the third iterations there are not many more improvements that they could make to the peg head design to make it more robust. Their peg head seems to be either made of aluminum or a dense plastic and is rather low profile. Being in a lower budget range and with the friendly use of 3D printers it saves a bit of time and money to 3D print a peg head for the for the tuner. There was a file that was made public and required small modifications to the back side of the tuner piece to make a proper keyhole for it to fit on the end of the motor.

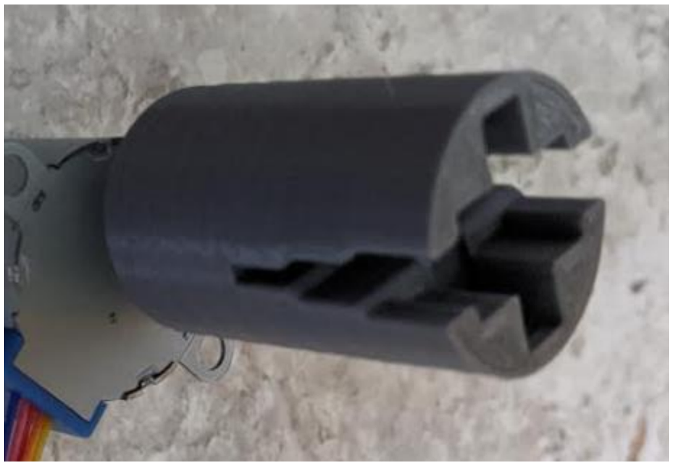


Figure 11, Universal Peg Winder Head

The dimensions for the peg winder head’s outer shape is 25mm in diameter and 42mm in height, the internal peg notch perpendicular to crevasse is 21mm long by 6mm wide by 7mm deep. The whole notch is crevasse is also 21mm deep and tiers down in 3 widths all the way down from 6mm to 4.5mm to 2mm. The larger middle notch in line with the crevasse is 17mm long by 10mm wide by 8mm deep and the current peg head for the design is a universal tuning head since the aim is to try and get a wider variety of stringed instruments. It has a longer peg head than the Roadie 3, but it has deeper notches to make up for all the varying tuning knob types there are on varying stringed instruments. It is also not a four-prong design and is a split 2 prong head which allows it to run deeper. There was a thought to modify it to be four prongs, but there was a risk of the prongs breaking since the shaft is relatively long the tuner knobs would possibly break due to the torque at the tip of the peg head. Even if the peg head for the tuner was 100% infill plastic it would still possibly break. If the peg head was made from metal, then it would have been more possible to make that adjustment. There were more modifications that needed to be made to the file since the current modifications were made for testing the 5 V motor. Once the bipolar Nema 16 was ready for testing, the motor shaft might have been too long for the peg head and would need an extension made in the file. Also, the keyhole for the back would need to be changed to match the shaft.

The final form of the peg winder was edited to have a 15mm long screw to thread in the lower side of the peg winder and meet the flat side of the motor time, so it is well secure to the tuner. There was an issue making the hole large enough in the Freecad software and making the STL and porting the final version as a STL file to Cura. Varying at what layer height and how big of a flat end screw that was used it should still make a tight fit, this helped keep the universal tuning peg on the end of the motor tip without the need of gluing the piece to stay in place. It needed to have a solid enough contact so that the vibrations could be picked up better by the tuning device. The back peg winder keyhole needed to be changed to allow a 4mm diameter, but the round head of the motor tip was cut and flattened out 3mm and it needed to be relatively accurate because if the motor head was slightly off it wouldn’t feel smooth for fast windings when restringing a guitar or any musical instrument.

12.5 Filament and Cura

Most of the part making for housing and the peg winder was crafted using a 3D printer from one of the team members. The filament being used was supplied by Amazon and is around $20 from when last purchased. It is 1.75mm grey PLA+ filament, we used PLA+ because it holds better than regular PLA and ABS, ABS filament was not chosen to be used for two main reasons, the filament tends to warp and bend due to changing temperatures and humidity in the air, where being in Florida there is a lot of humidity in the air which would cause problems for clean and level prints. The second reason ABS is not being used is because it gives off toxic fumes which are not healthy for humans to be inhaling. An additional note is that most printers do not like swapping between the two different filaments and the nozzles must be changed more frequent if that is the case.

Only perks of ABS are a lower print temperature, but the after product in the end is still usually not as strong as PLA or PLA+ is.

PLA+ melts at varying temperatures if not being heated at the proper temperature could cause the filament to not feed at a proper rate and the stepper motor of the printer will start skipping which is bad. Heat the filament too high and it will no longer melt and go straight to burning. The melting point for the filament is anywhere from 180 - 230 °C, but for most printing the housing and the peg winder are going to be printed at 218 °C with a layer height being .2mm which gives a better layer adhesion which should help with the vibrations carried through the housing to be a little better than other layer height options. PLA+ is strong and durable enough for containing all the components in a compact fashion. PLA+ is mostly made from recycled corn starch so it does not burn any toxic fumes and smells sweeter than burning plastic.

When using Cura, it is the main software allowed the file that is going to be made in FreeCad and put in a STL file. Cura runs taking STL and other 3D file types and makes a pathing for the 3D printer to follow by making a gcode file time where the printer can understand what is meant to be made and make a real-life version of the 3D model designed. For the settings made in Cura for the prints the infill will be 100% meaning it will be a solid plastic fill for the best rigidity structure and best way for the vibrations to be picked up by the sensors. Setting the printer to do 100% infill also allows some other parameters to be ignored like wall thickness and wall count since it will just be a solid piece of plastic, other fields that are ignored are the infill pattern because the printer is at max infill percentage it will only follow one specific pattern doing each print layer's print lines 90° from each other so it maximizes the layer adhesion.

For the bed of the printer, it was heated to the temperature of 55 °C. However, the print bed usually can almost be skipped because it can still allow prints to stick to the bed even at room temperature or a little warm, varies on how much power you want the printer to be drawing, but that was the temperature of the print bed for the housing and peg winder head. For the print speed, it was 60 mm/s which is a decent speed for running on max infill to keep the layers warm while printing. Each print was printed on a breakaway raft, so it had better contact to the print bed and kept from prints failing and snapping off the print bed. Supports were used for both prints whether it needed them or not, but the motor end of the shaft needed support since that end was circular and the peg winder head only needed a small amount of support on the base just for the keyhole on the back. That is mostly all the settings that were used by Cura for printing with an addition of a light layer of glue put down by a glue stick on the print bed for the initial layer to stick the best. It was run on a slightly older version of Cura on 4.4.1 version since it is an open-source type of slicer and constantly gets added features to it, but with small variations from each software version.

12.6 Battery

The Pick Pocket Tuner is operated with a battery. The options that we had were making the project rechargeable battery with USB power charger, or simply using a 9V battery. A

9V battery was the original thought for a version where the battery replacement would be quick and easy. The only downside was determining how long it would be able to power everything before it runs out of power. Currently the main competitor, the Roadie 3, is powered by a 500 mAh LiPo rechargeable battery however they did not give a voltage range for their battery but most of the batteries that match that criterion are 3.7 V LiPo. Having a rechargeable battery was a selling point, due to the sustainability and common place of having a chargeable device. Replacing batteries for devices has been phased out in most common appliances, therefore implementing a rechargeable battery makes our product more marketable and competitive.

The battery choice was determined by the voltage ratings for all our components. If the components for the device only required a maximum of 5V, then getting a battery as close to that output would be great for dealing with the regulation of the voltage. Another key feature that was important for the battery was the current output. We did not want the output to be too low to where our project will not function properly, but also not too high in case anything was burned up.

12.6.1 Battery Theory

Once all main components were accounted for, the battery was the final decision for how much power will be needed to run the system. For the final project, the group stuck with a 3.7 Volt Lithium-ion polymer 1200 mAh battery supplied by Amazon for $12, where it weighs about 23 grams, it is a larger battery than the competitor of the Roadie 3 but is smaller than the Jowoom, but where the Roadie 3 did not state the what exact voltage that they used but they did use a battery rated at 500 mAh and is a Lithium-ion type battery most of those batteries with those 2 parts going together are rated at 3.7 V. It could be higher but based on the size it is most likely that the battery used by Roadie 3. The battery did not have enough voltage to power mainly the motor and driver since the two together require a little more than 12 Volts. The overall system needs to power an LCD screen, processor, driver, and motor. There was further investment in a voltage step up or booster for when running power for the motor and driver. The reason of using a 3.7 Volt battery was because most of the competitors ended up using that as their voltage amount. The thought was to do a voltage divider followed by a necessary regulator to give a more exact voltage when going into the processor and LCD screen.

Since the battery is a DC power supply there was going to be a way to recharge the battery after it is depleted. A USB-C type charging cable was chosen for recharging the battery. This was the choice because it is commonly found along most phone charger types so if the charger for the device is ever lost a phone charging cable will make a good substitute for it. The battery pack was a bit larger than what we were planning for, but there was a decent amount of room in the housing that could be accommodated, it also allowed for balancing of weight distribution in the housing to counter the weight of the motor, although it put a damper on the overall weight of the tuner, but it is reliable and gives it a more solid feel in the user’s hand. The dimensions for the battery are 60mm long by 35mm wide by 5mm height with a 100mm lead wire length.

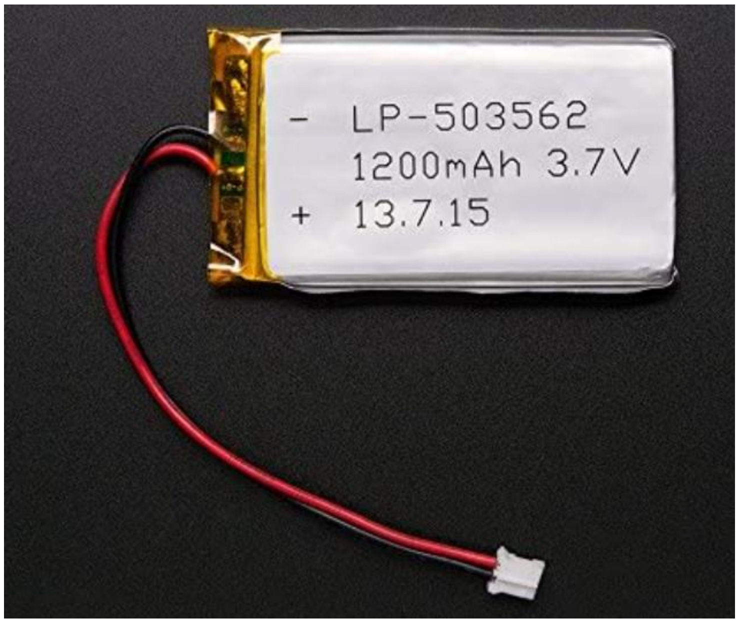


Figure 12, 3.7V Lithium Ion Polymer 1200 mAh Battery

The thought was to use Lithium-ion battery like the competitors, we thought there would be an issue in finding them in a higher voltage, but we decided to use more regulators to deal with that issue. There was a thought of getting around with a higher voltage at first and thought that is combining two or three of them in series giving either 6.4V – 11.1V however, the method may have sounded nice but would have led to hazards and was not recommended. There was a possibility of one of the Lithium-ion battery packs to not charge properly, it would be like putting water into two cups, an overflowing cup, and an empty cup where the now overflowing cup must leak water into the empty cup. If that were to happen a battery could be overloaded when charging and potentially explode. It did not offer a balanced way for recharging the batteries to have a proper functioning system. It would ruin the consistent voltage rating for each of the components that is needed for the design. Another battery option was the standard 9V Alkaline battery there were slight plus sides and down sides to the option.

The main upsides to them was that they are commonly found in almost any store, but they can be a little costly if they are constantly getting burned through, also most musicians use them for some of their other devices so it would add a slight convenience if they are already on hand. The downside is how long they would last in tuner which would be less than hour if not even 30 minutes if the motor was constantly running. They only have about 500 mAh which is the same as the Roadie 3 battery rating but higher voltage rating than the Roadie 3.

It was not the best choice so we decided to stay away from using that option since it would not be as efficient, so rather than being wasteful, the decision for a rechargeable battery is the better option and it made it more user friendly. Power efficiency was not the main concern for the system, but it was a concern to be focused on when trying not to waste power and drain the battery quicker than what was wanted. The plan was to optimize the system at the end with tweaking components and parts as we kept moving forward. The main part was getting the closest parts that would comply with the system and allowed it to at least run. Once getting past that, the fine tuning could happen, and more accurate parts could be attained after proper testing.

12.7 Switch and Buttons

Most of the interacting with the interface was done through the buttons. There were 2 types of button layouts that were considered. First layout was to have a five button Directional pad layout to make it easy for the user to interact with the interface on the display screen. The basic D pad would cover the options to move around on the selection screen with up, down, left, right, and a center select option. There was the thought of adding two extra side buttons that will be a back button and or a home button varying on how complex the interface was going to be.

The other option was more of a limited option where it would minimize the number of buttons needed to be pressed which would be an either a left, right, and select, or it would be a up, down and select. The button choices would mainly be determined by the interface and how many features that were made available for the user. Lastly, the switch for the tuner so the device can be turned on and off. There were thoughts about either adding and additional button or switch for a turbo mode on the device allowing for quick tuning from the device to where it will still be accurate enough for the instrument to sound proper and in tune.

12.7.1 Switch and Buttons Options

For the main power option, we used a simple on off switch. There are a few other options that were considered but rather than having a switch that is tied down to the PCB layout where it is physically on the board. The idea was to run wires to connect and solder them in so the switch can be used in a more practical spot rather than being limited where the PCB layout must be at. It gave a little more freedom when it came down to the finalized housing of the tuner, where the option to put the switch could go almost anywhere on the outside surface of the housing. The switch was supplied by Radio Shack and comes with two just in case and switches break they can be replaced easily.

The switch is rated 30 VDC 0.5A which was not a big worry since we were not going any higher than 12V in the system. The switch is 23mm in length and 7.5 mm in width and a 12mm in depth, it then runs with wires that connect to the main power supply before the regulators. The switch has a lifetime of 20,000-40,000, which is good enough to last the user a long while since they won't be tuning their instrument every day. The switch comes with metal flap with port screw hole more M1 rated screws at 6mm so drilling with a 1/64" drill bit allows it to thread nicely.

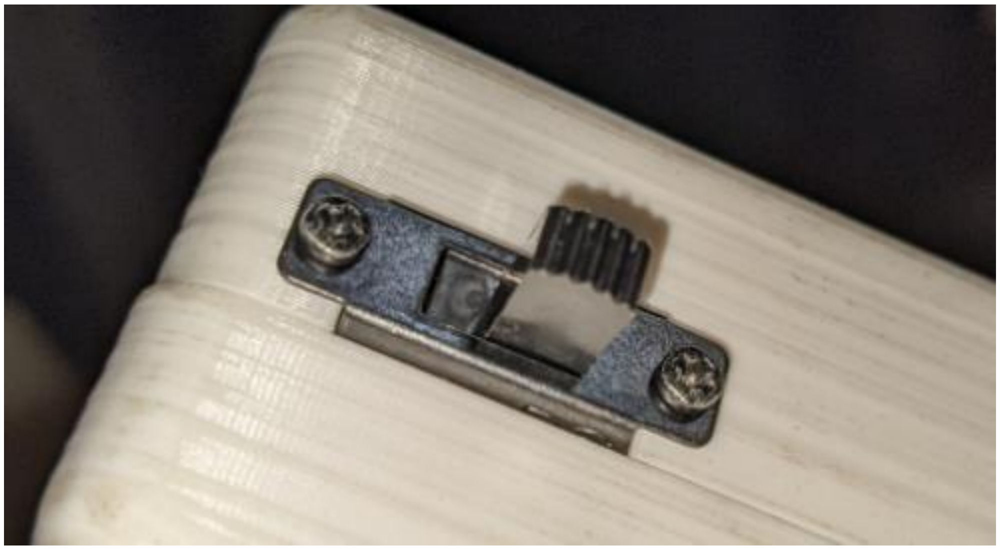


Figure 13, Switch Mounted

As for the buttons there are 3 buttons for interacting with the interface being left, right, and select. The buttons used were color coded with the schools coloring being black and gold so the left and right select were made black with the center select be yellow which was as close as we could get to it being gold. There was a question for adding a back button so for going back a menu the left button was opted, also the select center button is an option for some of the menus when it offers the prompt of done or not.

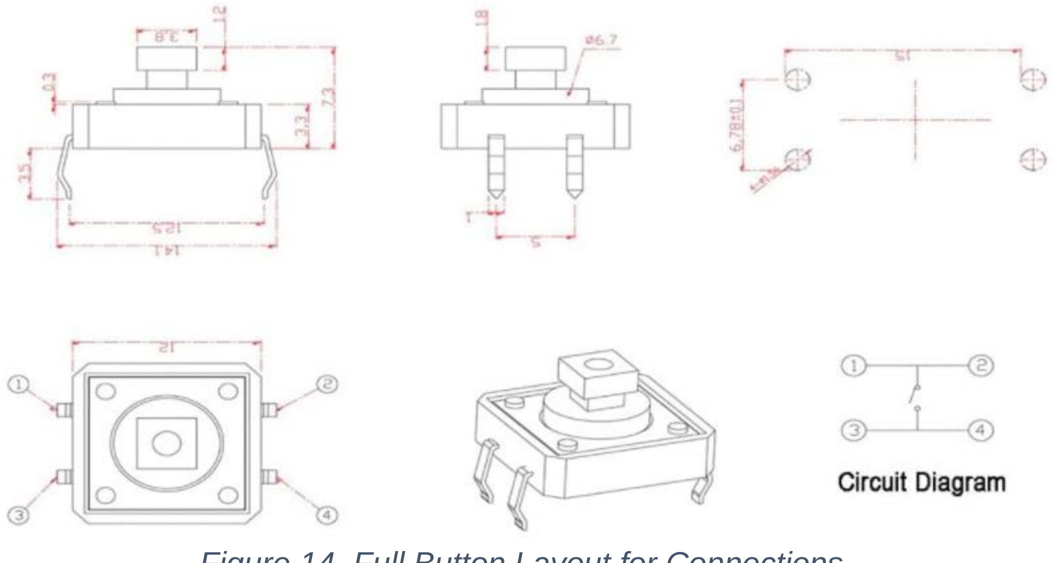


Figure 14, Full Button Layout for Connections

The thought is to have a minimal number of buttons of the device, so it is simplistic for the user and not too many buttons to where is starts becoming confusing to use the device. The dimensions for the buttons are 12mm by 12mm by 7.2mm height, but that is without the top caps of the colored buttons on the actual push buttons. These are momentary push buttons for inputs with the interface and are not toggle inputs. They work up to a 12V DC environment which works for our tuner design. The buttons also have a lifetime of around 30,000 pressed before possible failure. The buttons are 4 pin buttons and will be soldered onto the PCB layout. The kit comes with 25 push buttons and 5 varying color caps for the buttons making for a total of 25 push caps too.

The buttons are going to be wired to the 3V output from the ESP8266 then out to the three buttons that will have their own lines tied to their corresponding GPIO pins to be read as an input by the processor. Each button will have a 10kΩ resistor then run off tied to ground. For example, how one of the buttons is wired up using the schematic pin numbers from above in figure 14. The 3V line would go out form the ESP8266 and connect to either pin one or two then on pin three the line would run into the corresponding GPIO pin for that button's action on the interface. Lastly pin four on the button would connect to a 10kΩ resistor to then ground.

Figure 15, Multicolor Changeable Buttons

12.8 Housing

The housing of all the components was the last part of the process once all the proper parts are attained. The overall quality that needed to be met with the housing is that all the components are concealed except for the few parts that need to extrude from the housing being the: push buttons, power switch, motor head, ¼" output jack, and the toggle switch to swap between piezo sensors and ¼" setting. The whole design is made to fit in the comfort of one of the user's hands. There is a consideration to make the design ambidextrous so if the user were to play the guitar lefthanded or righthanded they could still see the display screen properly, however if one were to use the tuner in the right hand it is a little more difficult to see the screen but it is viewable if you angle it a bit more.

The Roadie 3 has a flaw with this, and it is not friendly for the user to tune a guitar peg head with the right hand since the display screen for their design would be on the underside facing away from the user. There are a few design concepts that were though of on how to fix the issue but was not used because it did not look right as a design option and made the housing even larger and awkward to hold if we did. As for the housing design a Fusion360 software was used to make prototype builds for all the parts to fit while keeping an ergonomic and comfortable feeling for the user. The design material is PLA+ and 3D printed to keep costs down for the budget.

12.8.1 Housing Theory

The final determination for the housing for all the components was the very last part of the project, most components were not finalized since there were changes and fixes to the PCBs that were being made while testing. Once the PCB layout was created and designed the housing began to be designed. Its design was going to have to be a tight and compact design with most of the bulky components being on the outside perimeter, the design looked closer to the Roadie Bass which is bit bigger than the Roadie 3. It has a bulkier design to it, there is an obviously larger motor in it.

The housing for the Pick Pocket tuner was split into two half case designs. The front facing case side holds the motor, display, power switch, two 10mm piezo sensors epoxied down, and main PCB unit that has the MCU. In the back side case housing, it held the 12V regulator, charging circuit, and the ¼" output jack. The other component that was not tied down was the battery, but it was sandwiched between all the components free floating but secured from the wires pressing on both sides when sealed. The housing can be shaken and there are no rattling components. Since the motor is a cylinder-shaped design most of the components were put below it and out of the way. From the front face of the design the motor is directly behind the screen while the power switch being just off to the left of the motor, and the PCB just below that. The PCB and charging are both held down by four 4mm M2 screws. The two case panels when put together are held by five 8mm M2 screws around the perimeter of the housing, two on the left, two on the right, and one on the bottom side. There was an added pick holder on the front face for an easy access ready pick whether for using it to aid in the tuning process or just need an extra pick. The overall housing is designed to be simple and easy to hold in one hand when using the tuner.

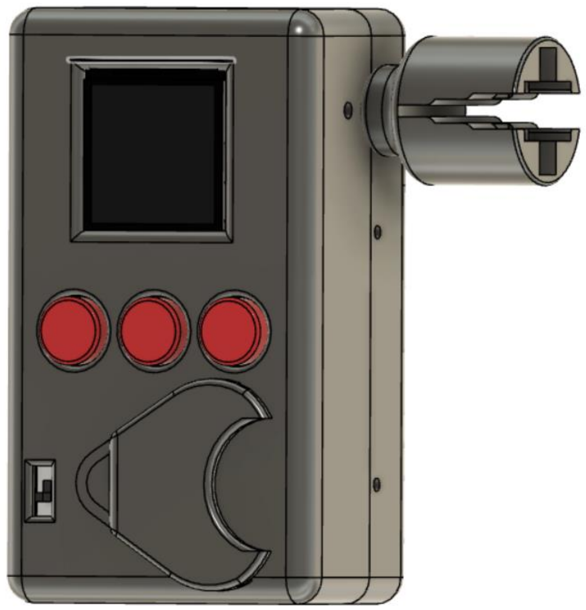


Figure 16, Pick Pocket Tuner Housing Model

There is one main issue with most of the designs and that they are not ambidextrous friendly that allow the user to still view the screen when tuning the guitar. If the screen is on the left side of the device, it could still be viewable in either hand it would just take a little proper angling from the user's side. Since the screen is a nice square shape, it would fit nicely on the opposite side of the motor. There are still some issues with tuning some instruments where some pegs are on the opposite side of the guitar head like most classical guitar heads.

When tuning those types of guitars, it is in the hand of the user at that point whether they want to spin the guitar around and tune it backwards or reach their hand around to the other side and lose visibility of the screen. Since there is a high chance that the user will lose visibility of the screen in this way, there is a thought to either add an audio beep that will notify the user to go to the next peg or whether the tuning is complete. There are a couple ideas on how to model it to be comfortable in the hand and provide functionality for the user. The design of the housing will be done in Fusion360 which if a user-friendly provided access given from the school. The material used will be standard white PLA + 1kg spool supplied from Amazon. When using the material on hand the walls thickness the final design will use 100% infill for when printing the housing so we can maximize on the vibrations carrying through the unit to be read from the peg winder head that is in contact with the instrument. The main housing design is going to be a split in half case design like the Jowoom but in a different shape or to do a lid shape design that will allow all the components to be easily accessed the lid would be held down by five M2 8mm screws. The motor will follow a similar front facing screw mounting like the Jowoom.

12.9 Buzzer

As an added in thought the has been a question of how the user will know when the tuner is done tuning the peg of the instrument. There is an added buzzer in the system as an extra notification feature for the user. The main reason this arose was because of a housing issue where if the user was tuning their instrument and have the tuner in a position where they are unable to see the screen and know whether to go on to the next peg or not. Most people would not notice a small detail be there as an extra aid, but it would be beneficial for the user in the end than getting frustrated not knowing whether the tuning is done, and the device needs to be moved to the next peg. The main feature of the buzzer is a simple notification to the user where once the vibration sensor meets the proper tuned note and the motor no longer needs to make a turn, the buzzer will give off a beep to let the user that the tuning is done for that peg and to move on to the next peg. There is also the option to have a different beep for the final beep to let the user know they are fully done with their tune.

12.9.1 Buzzer Testing

The buzzer does not require an alternating current signal and just needs a 3V – 5V to allow it to give off a 2 kHz beep. Another good point about the buzzer it does not need a control signal sent to it, just a voltage input to give off sound which is completely fine for the system. The only downside to the buzzer it does not give off any other frequency besides the 2 kHz frequency, which still works for our system. For when tuning is done for one peg and needs to go to the next peg the buzzer can give off a simple quarter second long beep to let the user know. When the tuning is fully done for the instrument, we can have the buzzer give off 2 beeps so do quarter long beep and wait another quarter second and beep again for another quarter second duration or it could a whole one second beep. The specs for the buzzer's dimensions are 12mm in diameter and 9.7mm in height. For the pins they are 6mm long wires separated by 7.6mm. Supplied by Amazon and salvaged from the Piezo sensor, also available for individual ordering.

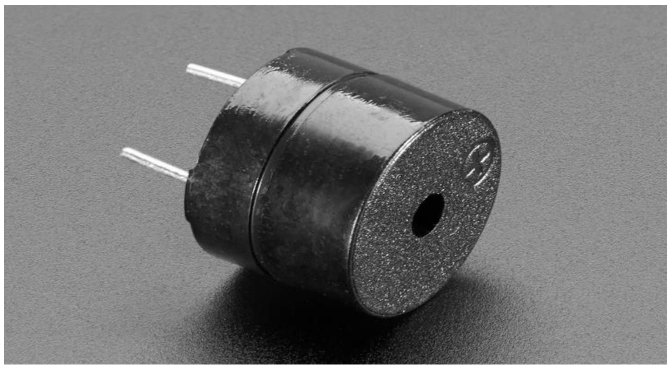


Figure 17, HYDZ 3-5V Buzzer

12.10 Vibration Sensor

The vibration sensor will be the pseudo-heart of the project, since this is the piece that will enable the tuner to function in just about all environments (i.e., loud, quiet, on stage, etc.). The sensor to use is still under determination, due to testing and parts acquisition. As it stands the vibration sensor is due to operate by taking the vibrational input from the guitar, through the wood, and through the guitar tuning peg, and interface with the processor to determine what frequency the string is reverberating at. Once that frequency is determined the processor does the rest and 'tells' the motor to be used to turn to tune. The choice of the vibration sensor has been hampered through determining what kind to use, MEMS, Piezoelectric, etc. Through research and direct comparison with other products already in market that accomplish a vibration-based tuning approach, it was determined that the most cost-effective sensor to go with seemed to be a piezoelectric device (PES for short). We opted to use two 10mm diameter piezoelectric sensors that were epoxied down just below the main PCB that holds the MCU.

The operational concept of this device, and size/shape of a PES determined that due to cost and ease of integration that it would give a device that is very sensitive and relatively accurate to be able to accomplish what our project is set out to do. Since the PES can effectively 'generate' the voltage due to the vibration being sensed through the guitar strumming, we then should be able to use those voltage peaks to determine the frequency at which the string is vibrating at, and thus relay through the processor to turn the stepper motor accordingly.

One aspect of the vibration sensor that might often get overlooked is that the choosing of a PES sensor makes sense due to its innate resistance to electromagnetic radiation, and electric fields. Take for example an end-user utilizing the product in a live performance setting. In such a setting there are typically multiple electric devices around, and with a guitar you also have magnets, high currents and other electromagnetic field generating devices, such as amplifiers, effects cabinets/racks, circuit breaker panels, computers, etc., that could interfere with the vibration sensors.

For the PES as our vibration sensor, anticipated that the device wouldn't have this issue and would then be an issue to not worry about. As previously mentioned, the integration of the PES sensor was simple, and straightforward. The PES device takes two leads, or two contact points, to then be able to be interfaced with. Those two leads, one a 'signal/voltage' lead and the other a ground, is what we directly interface with to be able to determine the voltage being generated by the vibrational sensations, then use some mathematical formulas to convert those voltages generated to the appropriate frequencies to determine what note to tune the guitar to.

Another reason we have chosen the PES is because most tuners have a device like this in them already. Therefore, we would be using something that is already accurate and used in the market. We also decided against choosing a microphone because of the inaccuracies that can be factored in such as background noise or electromagnetic frequencies. Many phone applications for tuning a guitar use a microphone, and there have been common complaints of this feature not working very well because of people trying to tune in a noisy environment. Since we want to stick to the accuracy of our project, the PES made a lot more sense. Overall, the PES is much more impressive on an electrical standard because of the simple, yet complex way the device performs. Finding the correct PES will be a challenge because there are many options in the market. Further testing concluded between the two piezoelectric sensors that were going to be used the two options that were being used was the 20mm diameter or the 10mm diameter. After epoxying to two different housing cases with each case piece having two of the same diameter type piezoelectric sensors. One case having 2 x 20mm diameter and the other having 2 x 10 mm diameter sensors. The case with the 10mm diameter sensor ended up giving more consistent and fluctuating results so we opted for using them.

12.11 Display

Our device will include a small 1.3" TFT display to make the user experience more intuitive and enjoyable. Our screen is compact since we want the whole project to fit in one hand comfortably. The screen displays instructions, frequency values, waveforms, menu options, and other useful information. The screen connects to the processor through an SPI configuration. The screen that is used is a thin-film-transistor liquid-crystal display and has a driver. The display also has an Arduino library which works with our chosen ESP8266 processor as it is an Arduino processor. This allows us to spend more time on creating an esthetic display that is user friendly and that will complement our software design.

We also thought of making a phone application for this device, however, the decision was made that having a physical display on our device would be a better fit for our project. Making a phone app for our project did not make sense for a few reasons. The goal for the tuner is to be useable by anyone from any age. If we were to force the user to use a smartphone, we would be missing out on anyone who does not own one like young children or adults who do not own one. Also, it will decrease our overall budget because there is no need to buy a license for an application. Having an app available for this device one day might make sense as an additional feature, but not as a necessary tool. Therefore, we stuck with having a streamlined high-quality screen on the device that will make the experience a pleasurable one.

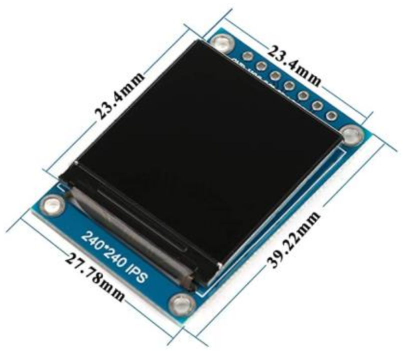


Figure 18, Rounded IPS-TFT LCD Display

12.12 Analog to Digital Converter

After some more design efforts, we concluded that we wouldn't need to implement an analog to digital converter (ADC) because the node MCU already comes with it as one of its pin ports. The ADC is used to translate the analog signal from the guitar and convert them into digital signals, which the algorithm can read. This means that when the frequency reading from the vibration sensor comes in, it will pass through the ADC and be converted to a decimal value. The value that will be read by the processor and read the feedback from the device. We can be accurate with a simple 10-bit converter, as the values are divided by 1024 accuracy division. Since the frequency range of a guitar has only 4 significant figures maximum, this is a sufficient degree of accuracy, while allowing us to maintain the cost low on this specific component. A fair amount of research went in how to determine the best way of utilizing the ADC since it is a crucial part for being used to determine the accuracy of the tuning.

12.12.1 Selection of ADC

After performing some technical investigation, our team has considered ordering and testing a 10-bit ADC from Texas Instruments (Part number: ADS7888SDBVT).

But the main option we went with was the analog input pin (A0) on the ESP8266 which also includes a 10-bit ADC. This ADC can receive an input voltage of 0-3.3 Volts. This would be perfect to use for the piezo sensor which does not in generate nearly enough voltage for this to be destroy the ADC on the ESP8266 itself. If the voltage is greater than 3.3V it could fry the A0 pin and render our tuner obsolete. We ended up adding a voltage divider for the ¼" output jack so the electric guitar would not fry the A0 pin. This was tested and was successful so there was no need to have an external ADC connect to the MCU and the A0 pin was able to be utilized which helped reduce the cost of our project, and allow our PCB to be smaller.

12.12.2 Testing Process

Once the output jack goes under the correct testing procedure, we were able to see if we needed to implement the external ADC. From there, we moved forward with one of two options:

1. Using the internal ADC alone:
   - Testing the software, and getting correct digital values to what voltage is being produced
   - Find the maximum and minimum digital values that are attained
   - Reach desired results

2. Using the internal ADC along with external ADC:
   - Find out how to transmit the digital value to the ESP8266 from the external ADC

- Use the software to extract the data from the ADCs and use them in conjunction to give information
- Find the maximum and minimum digital values that are attained
- Reach Desired results

12.12.3 Testing Results

The team has concluded that the ADC on the ESP8266 itself is going to be sufficient for the purposes of our project.

12.12.4 Implementation

The ADC is between the output jack, and the processor. The job of this component will be to take the analog inputs from the sensor/jack and make a digital signal for the processor to use in the calculation algorithms. The data is translated and directed to the correct results. The Vin terminal will have to positive branches coming from the jack and the sensor, and the negative terminals will be tied to ground.

12.13 Voltage regulators

For the voltage regulation for our system, we need inputs of 3.3V and one at 12V. The current plan for the regulators is to regulate a voltage from a battery, down to one of those expected levels. A linear regulator will be used for the 3.7V to 3.3V since it is easier to drop down in a small voltage amount without losing too much power and potentially frying the input. We expect to use a buck boost regulator to regulate to the correct 3.7V to 12V. This should not overheat our product and should allow for our project to run efficiently.

The regulator that is already on the NodeMCU-ESP8266, is the AMS1117. It was a optional regulator but it was unavailable for trying to get the regulator and components separate from the node MCU. We ended up going with a 3.3V regulator MCP1700T-33002E/TT. It was a relatively easier fix for at least the first regulator and it was relatively efficient being up to 91% efficient. An issue with these regulators is that they are linear regulators. The team believes that it will still be useful to test these regulators our to see if we can manage the heat and power dissipation properly. These regulators worked properly, and it saved a lot of time and headaches when figuring out how to regulate the voltage correctly throughout the circuit.

If these linear regulators did not work as intended, then we would have resorted to the buck boost converters that were used for the motor. These are prototyping regulators that electronic distributors sell, which were helpful for us test with.

12.13.1 Prototyping Regulation

To get the design correct for our project, we needed to prototype the voltage regulators to see if our calculations were correct. To do this, we used the XL6009 buck boost converters. These converters can take in 1.25-35V. this is plenty of DC regulation for our project and work for what we need it to do. It could be said as well that the regulators would not run at an above average efficiency, due to their wide input and output voltage ranges, however, they are still efficient enough for our device. This board is larger than we want for our project, but only need 1 of these boards to accomplish a complete prototype. They allow our project to function and are a good way to test and understand how different voltage levels are used in the system. This whole process helped us tremendously in showing how we need to incorporate all the correct components and regulators we implemented.

12.13.2 Voltage Regulation Strategy

Since our voltage source is going to output 3.7V we need to be strategic about the different regulators we decide to use. The three components that require different voltage levels are the motor (12V), the display (3.3-5V), and the ESP8266 (3.3V). The strategy for satisfying all these requirements is to design one step down regulator and one step up regulator. Below is a flowchart that shows the progression of all these regulators.

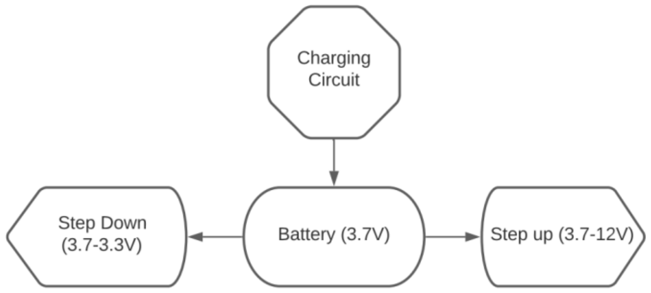


Figure 19, Regulation Diagram

From the diagram above, one can tell that this is the most effective regulation solution to our problem. It allows every component to receive its correct voltage, while allowing for optimal regulation. The display that we are using, has a 3.3V Ultra low-dropout voltage regulator, which means it can take an input of 3.3V and use that to run whatever logic it needs. This is very convenient for use, because it allows us to use the 3.3V output pin from the ESP8266 to power our display, and we do not need an extra voltage regulator to get us to 5V because it is already included on this module. This will not only save us money, but it will also save us on power consumption for the circuit, which means our battery life will be extended. Using the least number of regulators possible is crucial for being able to physically build this product since so many regulators were and still are on backorder.

Also, worth noting is the charging circuit we are using, the plan for the charging circuit is to implement a USB C Charging port. The voltage level that the battery will be charging at will be slightly higher than the battery voltage, to reach the correct charging value. USB charging usually stands at around 5 volts.

12.13.3 Buck Converters

The ideal design for our PCB would be to include buck-boost or buck converters. However, the problem our team was having, as well as every other team, was actually being able to order these converters. Voltage regulators especially have insane backlog times (39 weeks or greater). This obviously did not work for us since our project had to be built in about 20 weeks. Most of the regulator designs we wanted to use were efficient and size ratings were small but were all out of stock. This did not leave the team with many options for making our design.

The next section will discuss the type of voltage regulators we would have used for our project if it had not been for the global supply chain issues. The design below is a very efficient, compact, and useful designs for what could have been implemented into our project. We also used the TI WEBENCH tool to design each specific regulator. WEBENCH is a very useful tool and provides ample information for each design. After the very good design option is shown, for the schematics that has the parts in stock so we could actually build our project. They are not the most efficient or impressive designs, but we are hoped that they would work for our project.

12.13.4 Voltage Regulator Designs, 3.3V

The example for an improved design compared to the one used in our final project, is shown in the figures below. The specifications of this schematics are shown in the tables right below the figures. This shows a good comparison of what we wanted to use versus what we used for our project.

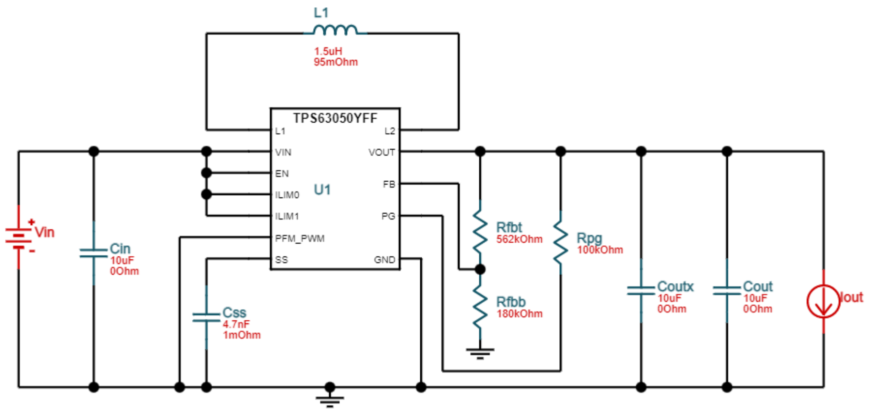


Figure 20, Desired Voltage Regulator Design

Vin	3.0 - 3.7V
Vout	3.3V

Iout	200mA
BOM Cost	$1.09
BOM Count	9 items
Power Draw	0.04W

Table 3, Critical Specifications for Efficient Regulators

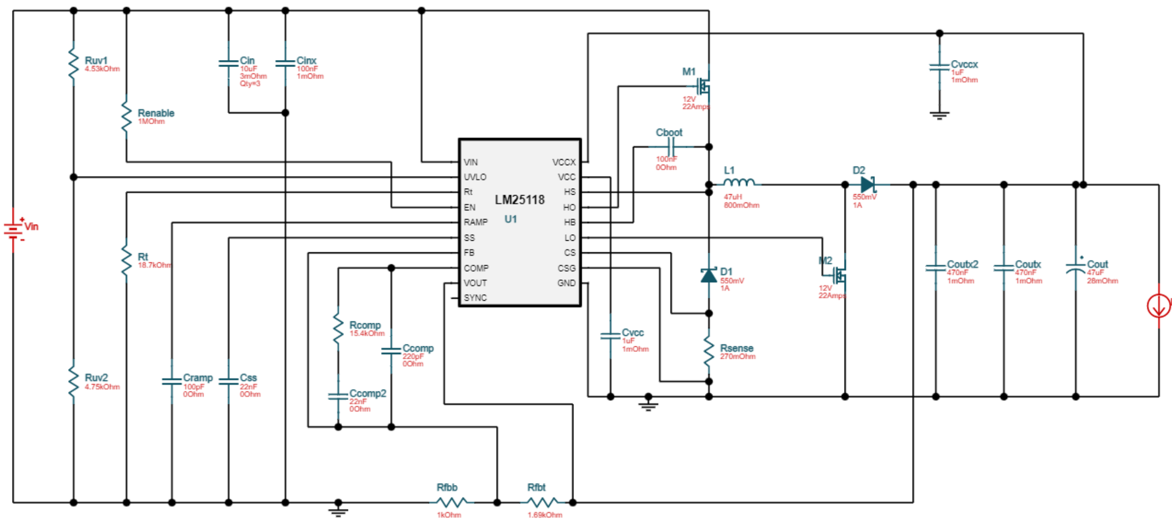


Figure 21, Schematic Design Choice

Vin	3.0 – 4.2V
Vout	3.3V
Iout	200mA
BOM Cost	$5.87
BOM Count	29 items
Power Draw	0.28W

Table 4, Critical Specifications for Chosen Regulator

Unfortunately, these comparisons aren't even close as to the what the better design is. The battery used has a total of 4.4Wh, and one of these regulators is already using up 0.21W to run. With one more inefficient regulator, our battery life is suffering. However, this was still be the best way of dealing with the supply chain issues.

Since we did not want to waste space in this document, we used efficient back up regulators along with one we attempted to use. However, the other efficient schematic will be shown in the appendix for reference, and the regulators that were chosen will be shown along with all specifications for each circuit.

12.13.5 Regulator Schematic and Specifications, 3.7-12V

To design the next regulator, we again used the WEBENCH tool on TI, and put in the parameters we wanted our circuit to have. As expected, we ran into some supply chain issues, and ended up using a similar circuit to our last design. Both circuits are going to use an LM25118 IC to achieve the correct voltage regulation. This integrated circuit has a very wide input range (3 - 42V). There are regulators out there that have a much smaller range, and a much smaller output range, however, this regulator should serve us well in this project. There was an option of using a different regulator that was also available, however that regulator had an even wider voltage range, which would cause our circuit to be less efficient, and it would take up more of a footprint on our PCB. The output voltage range is 1.23 – 38V, which is way more than our project would need. However, this circuit is available which means we were able to receive these ICs for when we went to order our printed circuit board. This was one of the few circuits that suited us well for this device and was available at the same time.

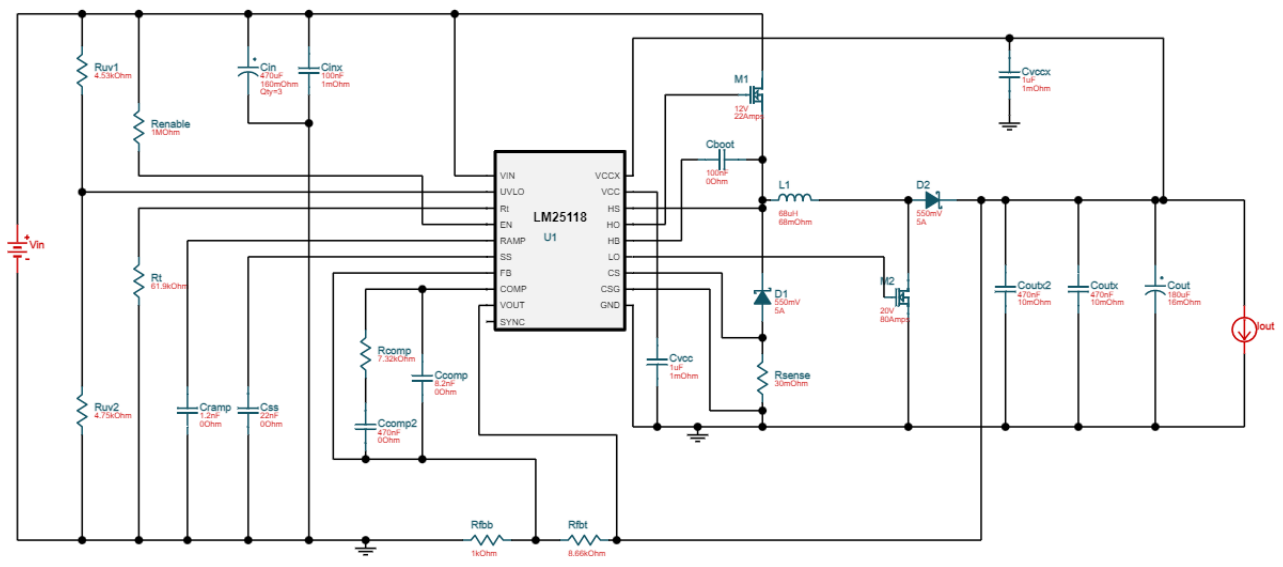


Figure 22, 12V Regulator Design

Vin	3.0 – 4.2V
Vout	12V
Iout	700mA
BOM Cost	$6.87
BOM Count	28 items
Power Draw	1.27W

Table 5, Critical Specifications For 12V Regulator

The circuit above seems like a complex circuit, just to get us to 12V, however it runs at a decent efficiency and was attempted to be implemented into our design. The footprint for this circuit took up some real estate, however we able to minimize on its layout to make everything fit on our board in an efficient way.

12.13.6 Ordering Process for Regulators

Since voltage regulators were in such high demand, we could not waste any time in ordering these components. After doing many thorough technological investigations online, there were enough parts for us to order for the regulators we need. This does not necessarily mean it was the case for the second part of the semester. Therefore, it was only wise to start ordering these regulators as soon as possible. A great way to achieve this was to have a completed design of our PCB before the next semester came, to simply implement the circuits and get our PCB built all at one time. However, we thought there would not be many opportunities to improve the regulator design or build another PCB if there was something else wrong with the board itself. Which is why another option was on the table.

Another plan for our design was to order separate smaller PCBs. This means we could order a few regulators of each kind and test them. The main 12V regulator was not functioning properly. This made us make a major adjustment towards the end of the semester since it was too late to fix the troubleshooting issue we were having with the regulator.

12.13.7 Testing Process for Regulators

Since we went with separating the regulators from each other to test. Below are the steps that were taken, to ensure a proper testing to be conducted on our products.

1. Connect the regulator to a breadboard using lead wires.
2. Turn on DC power supply and set to 3.7V first.
3. Connect the supply to the breadboard
4. Connect a load on the other end of the voltage regulator and connect a multimeter in parallel with the load to measure the voltage travelling across the load.
5. After correct voltage is being measured across the load, increase the voltage up to 4.2V to see if the same output is being generated.
6. Record data and results.

This test can be conducted for both regulators and was an effective and safe way for us to test the voltage regulation. If the regulators did not function as they should, then we would have had to go back to the drawing board and start over on our regulation designs. If the regulators were not working after the first or second time it would have been detrimental to the project as a whole, because of the supply chain issues could prevent us from improving on our design. However, we were hopeful that the designs we were going to implement would be effective and work.

12.13.8 Implementation of Voltage Regulators

Implementing the regulators into our design would be straightforward. The regulators will both be connected to the terminals of the battery in parallel. The end of the 3.3V regulator will go straight into the ESP8266 to give it the voltage required to power that design. The 12V regulator will have its input connected to the battery and its output connected to the positive and negative terminals of the motor we are using. This provided the correct voltage levels needed for both the MCU and motor to operate at peak performance. The motor however also has three wires going to the MCU to be controlled by pulse width modulation. The implementation of these regulators again were fairly simple and allowed all components to operate as they should.

12.14 Processor / Controller

The processor we have chosen for this project is the ESP8266 which is a CPU with low power consumption and Wi-Fi capabilities. The processor is 32-bit with a maximum clock speed of 160 MHz which is more than capable of reading our input sensor, powering our motor, and drawing to the display. The processor also has low power consumption is also ideal for our project as one of our technical requirements is to have around 100 strings tuned in respect to battery life. For memory, the processor has 80 KB of RAM and flash storage. With the standard for flash memory being 4 MB. This amount of storage is plenty for our software to run and for our user-saved string frequencies to be stored for faster selection and ease-of-use.

As for communication, the ESP8266 has pins for SPI, UART, and I^2 C. This is ideal as we have choices for which communication method, we implemented for interacting with the other devices of the project. These multiple communications are also able to be used for concurrent communication among multiple connected devices. This is ideal for our technical requirement of timing as we wanted to be able to tune all the strings on a guitar in around 3 minutes and around 5 minutes for a newly stringed guitar.

Our group researched and tested different ways to implement our project in terms of reading the frequency from the plucked guitar string. One implementation that we researched was having the sensor attached to the bridge communicate with an ESP8266 and have that processor communicate with the handheld device that has the same ESP8266 to wind the motor for maximum accuracy. This processor can be ideal for that implementation as they have Wi-Fi capabilities and could communicate with each other. With the sensor on the bridge of the guitar, we can achieve a greater accuracy in terms of reading frequency which would satisfy our technical requirement of being within 5 cents of the user desired frequency.

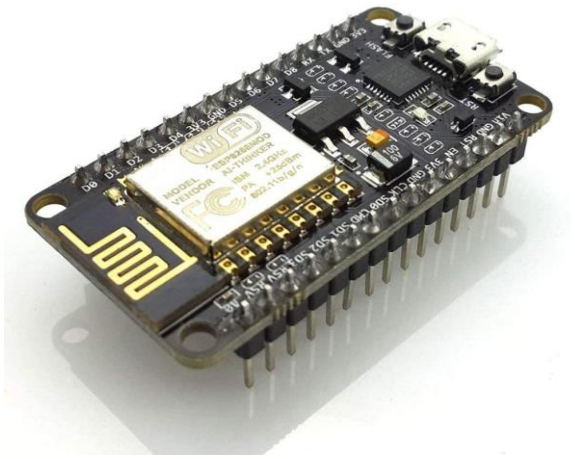


Figure 23, ESP8266 on NodeMCU Development Board

12.15 Clock

With the ESP8266, compatible clocks were further researched. From what was gathered it was better off to use the internal clock that already came standard in the ESP8266, being the 80-160 MHz adjustable clock. There were thoughts of adding a external crystal clock mainly because we were opposed to internal RC clocks which tend to be inaccurate and are influenced by temperature changes. For external clocks, we had the choices of a quartz or ceramic crystal, silicon, or an external RC circuit. Our Pick Pocket Tuner will require the utmost accuracy when communicating. The processor will be communicating with the sensor, the display, and the motor driver all to perform precise measurements and calculations. Since our project has the engineering requirement of having the frequency accuracy be within 5 cents, we thought we were likely going with the silicon or quartz crystal choice, as these clocks have the best accuracy. But after going through the research and hassle of adding a new on-board clock the one that came standard with the ESP 12E was good enough for us to test and implement.

12.16 Charging Circuit

Since the team has decided on using battery, we wanted to implement a charging circuit. This is an enticing feature since it allows our project to have a sustainable way of being used. Replacing batteries is a hassle, and it creates waste by going through so many batteries assuming this product is used every time a musician wants to tune his guitar. Having replaceable batteries is more sustainable on the environment and just makes it more convenient if you want to reuse this product regularly. This product will be draining quite a bit of power as well because of the motor that will be implemented in the project. This means that the battery life can deplete quickly if the motor is consistently running, thus making it more inconvenient to go through many 2AA batteries all of the time, rather than being able to just quickly recharge.

12.16.1 Design of Charging Circuit

When performing some technical investigation, it can be verified that we needed to find a circuit that can charge a lithium Polymer (LiPo) battery. Luckily LiPo Batteries charge the same way, by requiring a constant voltage to charge the circuit. One important parameter to keep in mind is the charging current rate. The original charging circuit was designed but due to supply chain issues we were unable to use or implement the design. We ended up having to resort to our plan B for the charging circuit and used a premade Adafruit charger that was also designed to work with the battery we already got and used for testing.

Integrated circuits have been created by companies such as TI to make help rechargeable circuits easier to design. One of the Integrated circuits we planned on using for our charging circuit is the LM317 from TI. Below is the basic schematic we decided not to use for our design. The output voltage to the battery should usually be about 17-18% higher than the voltage listed on the battery, according to online resources. Therefore, since we plan on using the 3.7V battery, this means that the output voltage from the charging circuit should be around 4.3-4.4V. Also, we only wanted a current of 0.5A to be entering the battery, so this parameter is known as well. Since we know these two parameters, we used simple equations to solve for what the resistor values should be in this circuit. The equations we used are shown below.

$$(1)\ V_{out} = 1.25\left(1 + \frac{R_3}{R_2}\right)$$

$$(2)\ Z_{out} = R_1\left(1 + \frac{R_3}{R_2}\right)$$

$$(3)\ I_{out} = \frac{V_{out}}{Z_{out}}$$

Final Values:

$R_1 = 2.5\ \Omega$, $R_2 = 1\ K\Omega$, $R_3 = 2.5\ K\Omega$, $Z_{out} = 8.8\ \Omega$, $V_{out} = 4.4\ V$, $I_{out} = 0.5\ A$

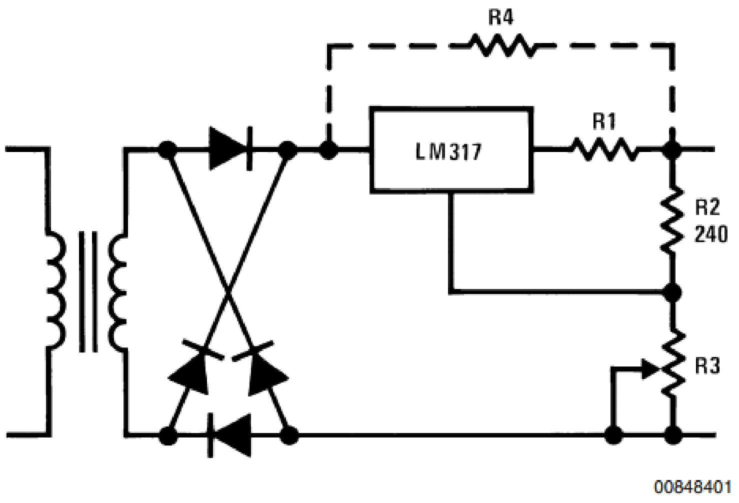


Figure 24, Charging Circuit Design with LM317 (Intellectual Design Provided by Texas Instruments)

As one can see, this circuit may not be the best to charge our battery, because the LM3117 is not very efficient and finding 2Ohm resistors is not easy and would not fit well into our circuit. Also, since the devices in our project either run off 12V or 3.3V, having a 3.7V battery may be a more efficient battery for us to use. Since the voltage regulation should be much easier, and more efficient, our team will move forward with the 3.7V LiPo battery.

12.16.2 Charging Circuit Using MCP73831/2

After doing some further research on circuits for charging batteries, we found that there are many other options to use, that are more efficient, smaller, and cheaper. One of these circuits is the MCP73831/2. The integrated circuit has all the requirements we need for our project. The following paragraphs will discuss the circuit to be built around it, suppliers, implementation and plans to test this circuit.

12.16.3 Requirements and Specifications of MCP73831/2

The integrated circuit has five pins, each with specific labels. These labels are STAT, VSS, VBAT, VDD, and PROG. The VDD label is the input voltage that is being provided by either a wall outlet or a USB-C cable. In our design used a cable that comes from a premade wall outlet that worked for our conditions. The VSS label is the designator for ground that the company who made this circuit chose. VBAT is where the positive end of the battery connects to. This is where our red lead wire coming from the battery will be connecting. That STAT pin is used to determine the charging status of the battery. In most diagrams, LEDs will be connected to this pin to indicate that the battery is charging and will turn green to indicate that the battery has been fully charged. Finally, the PROG pin is used to determine the charging current rate. The datasheet for the IC shows different resistor values that one can use to regulate the current rate that goes into the battery to charge it.

The output voltage for this product is 4.4V which is perfect for what is needed for our project to be charged by. Since the LiPo batteries use the constant voltage method to be charged, the voltage charging the battery needs to be slightly higher than the rated voltage for the battery. However, most of the charging comes from the constant current method. The battery is mainly charged by receiving a constant current which will eventually raise the voltage of the battery until the rated voltage is reached. The circuit shown below is the layouts that are using in our own design.

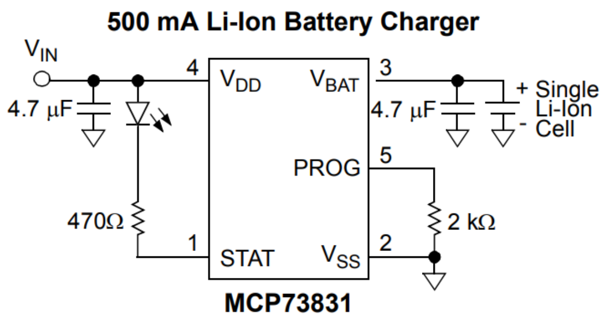


Figure 25, Simple Application Circuit by Microchip

The 4.7uF capacitors are the values recommended for the capacitances because it compensates for when there is no load.

One more component that will be added into out schematic for this circuit is the USB-C Charging port. This I where Vin will pass through so our circuit is able to be interacted with by the user. There are data lines on the USB-C port, but these not be used, since we are just trying to charge a battery. Adding this USB-C Port will also us to be competitive in a global market since the world is trending mostly towards the desired USB-C interface.

Something that our team was aware that the circuit can get hot and heat dissipation that might occur in this circuit. It seems that this IC can run warm, but through designing and testing different components used in this circuit, out team knows that the heat will not be a problem.

12.16.4 Suppliers for MCP73831/2

Our team plans on ordering this circuit from Mouser electronics. They seem to have a very reliable track record, and they have plenty of these chips in stock. Microchip themselves seem to distribute to a warehouse in Atlanta, which means that the shipping time for this product should not be too much of an issue. It also will help since this seems to be a common circuit to come by.

12.16.5 Testing Process for MCP73831/2

Testing the design for our circuit should be simple. We shall build the circuit using the components stated above, using a breadboard and electronic equipment to simulate and test our circuit. This IC can be ordered on a breakout board which will make it very easy to test. Once the circuit has been checked and all the connections have been made correctly, we moved forward with the simulation process. After the simulation was ran correctly, we finally connected a battery to the charging side so we can ensure that the circuit is functioning correctly. After some charging is done to the battery, we tested the voltage level and make sure that the battery is charging properly. We also took into account any heat that may come from the battery or from the IC and made sure nothing was overheating or damaging any components. Once this method was perfected, we created the final schematic layout in Fusion 360/EAGLE and saved the schematic.

12.16.6 Implementation Process for MCP73831/2

Implementing the charging circuit design will first start with implementing it into our PCB design. Once the schematic is created, it will be imported into our final PCB schematic. Once the PCB Design is finished, we then design the board layout to have the charging circuit near an edge of the board so we can build our housing around that spot to allow a place for the port to stick out and be plugged into. This process is simple, however designing the 3D model of the housing around this component should be the same as the other PCBs, since none of us have a great deal of experience working with CAD modeling, however, we believe that after some trial and error we were able to incorporate it well.

12.16.7 Prototyping Charging Circuit

For efficiencies sake, and the sake of gathering more data on the battery consistently, we needed a reliable way to charge the battery multiple times to give a detail specifications sheet. Therefore, having a charging circuit prototype benefitted the process of design and implementation. There are many premade circuits on the market that help with charging batteries, but there is one made by Adafruit, that gave us exactly what we were looking for in our project. The device is very small and compact and uses 5V USB Charging to charge the circuit. We chose the option to use USB-C which would be an even better implementation since most electronics are starting to use USB-C more commonly. The circuit is shown below in comparison to a small LiPo battery and a laptop. I can also be charged with any phone charger that uses a USB C port

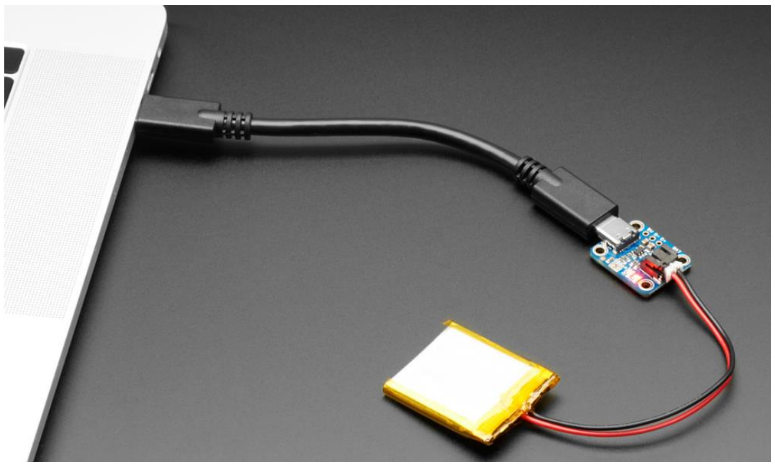


Figure 26, LiPo charging circuit to be used in prototyping and testing phase of project

This means that our project can be distributed anywhere in the world, and it should be able to be charged without a hitch. And as one can see, the circuit is very compact, and fits well within our design requirements and specifications. Below discusses how to implement and test the charger for our project.

12.16.8 Testing Charging Circuit

There are multiple criteria that were be tested on this circuit, so that we could become very comfortable in utilizing this circuit. The different questions were attempting to find answers for are as follows:

- Does it work?
- How long does it take to charge a fully depleted, half depleted or quarter depleted battery?
- Can the circuit damage the battery?

Our team believes that these questions gave us all we need to know about how to use this prototype for our device.

The testing process to answer the first question is simple. All that needed to happen was to connect the charger through a reliable USB-C port that already works. Once the connection was made, and the indicator light turns on, we knew that the circuit was receiving power. After it starts working, we disconnected the source and plug the terminals of the battery into the correct terminals on the charging circuit. It is worth noting that the voltage level of the battery was tested before the circuit started charging it. The voltage that was measured before the charge was recorded and compared with the voltage levels. The circuit plugged back into a source and left on for about 30 minutes. After the 30 minutes were up, we measured the voltage level of the battery and noted if it increased. We knew the circuit was effective when we noticed the voltage of the battery increase.

The next step in testing the charging circuit was to see how long it takes to charge the battery at different battery lives. To test these ranges, we depleted the battery to each of the levels stated in the question above. After each level was reached, we measured the time it took to charge to full capacity at each charge. Once we found all our times, we saw consistent rate that the charging circuit supplies was 500mAh.

Finally, we tested overcharging, and saw it can either damage the life of the battery or damage the battery in any other way such as heat. To test this, we connected a fully charged battery to the charging circuit for a certain period of time, and then discharged the battery completely. This process was be repeated several times to track the life of the battery and see if the battery life becomes weaker, or if there are any signs of damage. This gave us a good understanding of how our customers should go about charging their product safely.

12.16.9 Prototype Charging Circuit Implementation

The reason for prototyping a charging circuit, would be to learn how to implement a design of our own, or more specifically condense or improve designs that already exist on the market today. It also allows us to make sure other parts of our circuit are operating correctly and how we expect them to.

Firstly, we ordered one of the USC-C charging circuits from Adafruit and tested it, with our 3.7V battery that was going to be tied to it. Most of the teammates on the team have the capability of plugging this circuit into their laptops or walls with different devices that take the USB-C connection. This will make the process of testing very easy and convenient for us. Also, it is worth noting that the circuit comes with a standard charging rate of 100mA, which is going to be very slow if our battery has a capacity of 1200mAh. However, there was a jumper on the board that we soldered and changed the charging rate to 500mA. This means that if our battery becomes fully depleted, it will reach a full charge in about two and a half hours. This may seem like a large charge time; however, we are expecting the battery to last a long time on a single charge. The Roadie3 only has a 500mA LiPo battery, which means it charges quickly, but can also deplete quickly as well. Therefore, we expect the decision to include the longer battery life will be worth it in the end.

The implementation process for testing purposes was simple. There are positive and negative terminals on the battery charger that connect to the positive and negative terminals on the battery. We must ensure that the connections are at the right terminals, or we could damage the battery or even the charging circuit. Once everything was connected properly, we started to connect other parts of the circuit that would be powered with the battery to see how well everything is functioning.

12.16.10 Using the Prototype as Plan B

The goal for prototyping this circuit is so that we can learn how to design and implement our own. However, our plan B for this prototype, is using it in our final device. The team did not plan on using this circuit; however, we were able to get a working product the next semester and using this circuit was a huge benefit we were able to implement into our design. All measures were taken to ensure our best effort was put forth in designing a charging circuit that works, however, the supply chain issues were ultimately led us to the decision to use the prototyping circuit in our complete project.

12.16.11 Mounting Prototype in Housing

It is necessary to use the Adafruit LiPo charger in our final project, and had to figure out a plan to mount the circuit to our housing design. The board comes with four through holes and each hole has a diameter of 2.5mm. The image to depict these holes are shown below.

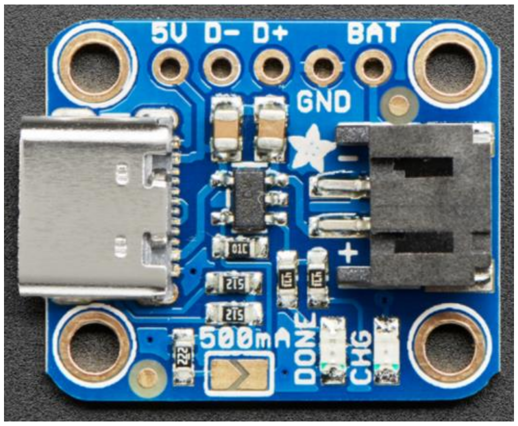


Figure 27, Image of Adafruit Charger Showing Mounting Holes

The plan was to mount the circuit using the four holes and developing the mounting holes on the housing itself. We used four very small M2 4mm screws. The screws also are Philips heads type screws, since it is the most common design to find for screws this small. This circuit is very well secured when placed in the housing because it must be able to withstand people unplugging and plugging in the charging cable. This can be a difficult process for some people, as some people struggle with depth perception, and plugging the cable into the small port with too much force could cause the mounting screws to shift or come out, rendering the mount useless. Therefore, it will be essential to ensure the circuit is secured tightly to be able to withstand uncommon forces.

12.16.12 Testing Process

The first step will be to deplete the battery at a decent rate. This can be done by making a circuit that will drain the battery at a steady rate, without overheating or damaging any of the components. Once the battery is depleted to about half of its battery life, it will be disconnected from the circuit that it was powering and set aside. This gives us a benchmark as to how much batter life is left and allows us to proceed with this knowledge.

The next step is to build the charging circuit on a breadboard and test the output voltage and current we are getting without connecting this circuit to the battery. We wanted to ensure that it is operating at a safe voltage and current for the battery before we could connect the two together. Once the team was satisfied with the results, then we use the battery. If the next step was to proceed with connecting the battery. After it is connected, we tested to see if the battery was charging like it should. Once the circuit was working properly, we would start a timer to measure how long the battery takes to charge once it is half depleted. All things went well, and the battery becomes fully charged and on how to implement it into our design.

12.16.13 Implementation Plan

The implementation process for this circuit was somewhat simple once the design was complete. The circuit will be connected to the positive and negative terminals of the battery, a switch will also be tied in between the battery and the regulators to control the overall power to be able to flow. The terminals of the battery will also be connected in parallel with the voltage regulators that will be used to distribute the correct power throughout the system.

13.0 Printed Circuit Board Design

Designing a PCB was no small task. This part alone had the ability to make or break our entire project. There are also many steep learning curves when learning how to design and build a working PCB. This is something that comes with practice; however, our team planned on breaking the process down into smaller pieces to make the process go over smoothly. The following subjects will be discussed thoroughly as the regard to the design of the PCB:

- Overall layout of how we think the components will connect
- Power distribution and ratings
- Fusion 360
- Suppliers
- Prototyping and testing

13.1 Overall Layout of PCB

The layout of our PCB shall be a very important quality in our overall project. We planned on making it a compact as possible and have a good shape that is affordable and will fit in an ergonomic housing. Most of the components we shall have on the circuit board have small footprints, so this will aid in keeping our PCB relatively small. The ESP8266 has a very small footprint, however it should have plenty of computing power for what we need to accomplish.

13.2 Power Distribution

Determining how power will be distributed throughout our project is essential to make sure all components are operating at the correct ratings. The table below shows all our devices that draw power, and at what rate they draw electricity. There are some components that still need to be put under load and tested to see what kind of current they will draw. So far, this table gives us most of the measurements and estimates that we know of.

Component	**Voltage**	**Current**	**Power**
Motor/Encoder	12V	200ma (expected Load current)	2400mW
TFT Display	3.3V – 5V	100mA	330-500mW
ESP8266	3.0V - 3.6V	200mA	600-720.mW
3.3V Regulator	3.3V	200mA	280mW
12V Regulator	12V	0.7A	1270mW
Total			4,880 – 5170mW

Table 6, Power Distribution Calculations (Incomplete Table)

Finding out these measurements is crucial for when we started designing our PCB. Using what we measured will allow us to make sure every component gets the correct voltage and current they need to receive. Otherwise, we could run into problems of components being overloaded, or not running at the load they need to be running at, which could cause major issues for our final PCB. Knowing these calculations also aided us in estimating our battery life. Knowing how many watts or amps everything draws is crucial in determining battery longevity, and it can also show us some areas where we could improve upon.

13.3 Fusion 360

The software that will be used to design our PCB is Fusion 360. Fusion 360 is a versatile tool and allows us to intuitively design our PCB from scratch. This was also the software that everyone on the team seemed to be most familiar with, so it made sense to choose this as our design tool.

Another contributing factor as to why we chose to use Fusion 360 as our development software was because it was made available to us through an education license. The license to use the software costs $400 and since we have four group members, the total we would have to pay would be around $1600 not including taxes. This was a huge savings on our development costs and will allow us to be able to spend more of our budget on the physical components for our project.

Fusion 360 also has a very attractive user interface. Other Programs like EAGLE seem to be a little more rigid and outdated on the interface tools, however Fusion 360 has very good labels and icons that are easy to differentiate between. This feature does not seem super important but enjoying and thriving on the design software should translate directly to the efficiency and ability to get work done on our design in a positive trend. So far, this trend has shown to be true, and the team is able to work comfortably and on schedule. All the listed reasons above show why our team chose Fusion 360 to complete all of our design for the printed circuit board we are developing.

13.4 Collaboration Method

Something else that was discovered throughout the development of our PCB, was the collaboration method that can be utilized in Fusion 360. This collaboration method works similarly to how Google Drive works, except it is used for the development and design for a printed circuit board. The students were able to create one big project for all senior design and share the emails used for their accounts and allowed the folder to be shared between all of the students. The two electrical engineering students Paul Grayford and Lucas Grayford are the main developers of the PCB and can work efficiently together through this process of collaboration. The folder containing all the schematics that were designed are shared between everyone, and both students can work on the schematics together in real time. The most efficient way that the two students are utilizing the tool that Fusion 360 implemented is to work on different circuits of the PCB to and once they are finished, copy, and paste them into our main design file. This allows the circuit to be built in individual parts, which will aid us in our overall design by being able to adjust wherever we see fit. It also allowed each student to become more familiar with the subject matter for each part of the circuit by allowing each team member to have a hand in something that they were able to design and research themselves.

An example of utilizing this feature would be when Paul Grayford was designing the voltage regulator Circuit, and Lucas Grayford was figuring out the connections from the ESP8266 to the LCD display within our project. As Paul was developing the regulators, in the same folder Lucas was able to work on figuring out and troubleshooting the connections for the LCD display in real time. Both were available to ask each other questions and see what the other was working on real time. Both students were able to assist each other, and in the end complete the design for each circuit and paste them into the main PCB file making the process run very smoothly. This was only one example of many where we were effectively able to communicate and collaborate on different parts of the circuit together.

13.5 Designing Components Using Fusion 360

In a perfect world Ultra Librarian would have all the footprints and symbols we needed for our project, however that is not the case. There are some components that we had to design in our project on Fusion 360 itself. An example of using the design tools would be designing a footprint for our piezoelectric component. Also, one of the footprints was improper for the 12V regulator and caused a lot of problems and troubleshooting to deal with until the actual problem was found being a footprint error with one of the files.

There was another component that was also not in the library and needed to be designed. Paul Grayford headed up this task by first creating a new library that would include all the student made components for our project. After the library was created, he attempted to create a new part from scratch. After a substantial amount of time learning how to create a symbol, he still was running into issues on how to design the component needed. Since he did not want to waste time, after doing some research, he tried importing a similar design and modifying it to the specifications required for the component. This seemed to work better, as he found a library that had a buzzer with a similar footprint. Since buzzers use the piezoelectric disks like the sensor we are trying to implement into our project, it ended up being a good choice to modify. After he modified the symbol, for the schematic diagram, he was able to modify what it would look tike for the footprint to be on the PCB. The symbol and footprint are shown below.

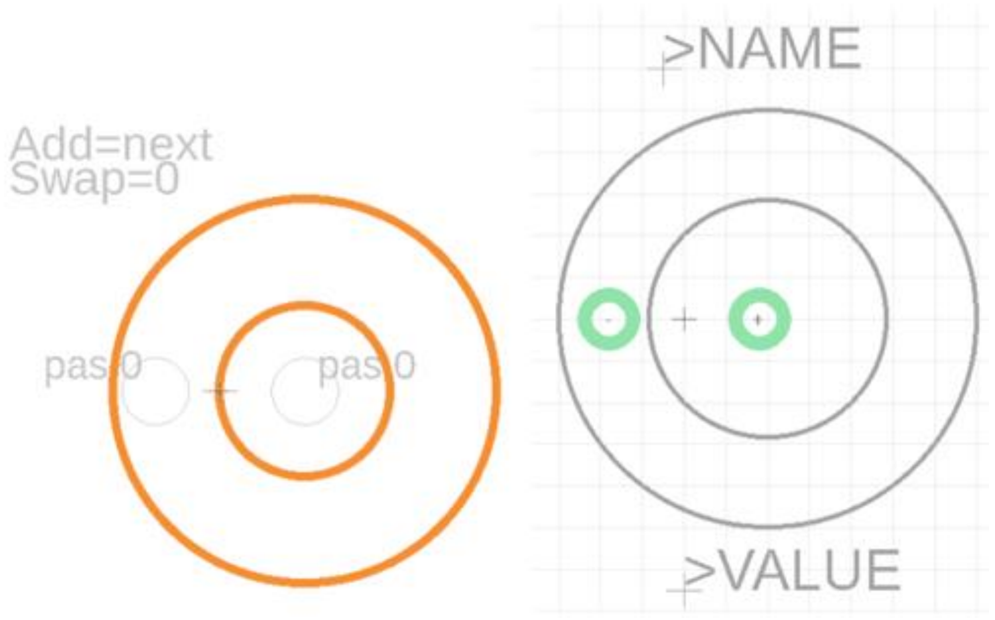


Figure 28, Student-Made Piezoelectric Sensor (Symbol on Left and Footprint on Right)

The actual modification process was learned very quickly. Once the old part was imported, one could select the lines, pins, and other important features of the symbol and modify them so that they fit the description of the part needed. The creation tool also lets you delete segments that will not be used and add segments that will be used. The circular features on the footprint were added in with the draw tool and were modified with accuracy using the dialogue box that changes the width and height of the shapes. The pins that connect the wires also had to be adjusted with their position They can be seen as the very faint gray circles that have the labels on them. For the footprint itself, the modification was very simple. All that had to be done was adjust the radiuses of the circles so that they would reflect more of what the component looks like. Once this was accomplished both parts of the component were complete and ready to be used.

After the footprint and symbol were made, it was saved in the library that was mentioned before. This allowed our new component to be added to the PCB design by importing the library with the library manager into the main design file. After it was added to the project, the part can now be added with ease by anyone who has access to the main printed circuit board file. Other components needed to be adjusted as well like the Zener diodes, because they were taking up a lot of real estate in the schematic diagram.

Learning how to use the creation tool was very useful and will continue to become a convenient way for us to implement components into our design, even if we could not find a file online that has everything we need. It will also be valuable experience for us to learn how to make different components if this ever comes up in our career path.

13.6 Supplier

The team plans on using the company JLCPCB to supply us with our printed circuit boards. After doing some technical investigation, they seem to be the best company for our needs. JLC provides boards for an affordable price, and they allow the buyer to buy in bulk, which will come in handy once the PCB design is perfected and we are able to have backup boards in case of sustained damaged or unexpected roadblocks. Overall, the company seems to have a great track record and will serve us best once the team is ready to start ordering PCBs.

13.6.1 How to Order from JLCPCB

Ordering from this company is simple. Once the board layout and schematic designs are complete, then a Gerber file will need to be generated. The file is used to show all the traces that are used on the board and will allow them to print the device clearly. Once the file is uploaded, there are many options to choose from such as: How many layers, area of the board, color, thickness of the PCB, how many boards to be ordered etc. Once all the selections and preferences have been made, the billing addresses and the checkout is then filled out. JLCPCB claims that the build time for most of the PCB's they build only takes up to 24 hours and then will only take 3-5 Business days to ship. This is a very quick turn-around time and will be beneficial if some of our boards do not function properly because it will allow us to order replacements in a timely manner.

JLCPCB also gives the option to assemble the components on your board for you. This could be a helpful tool for us to use since we have so many small resistors and capacitors to keep track of, which means that our team could spend a substantial amount of time soldering all the small components and could make us lose time on testing and design fixes. Therefore, this could be a good option for us, even if we can have JLCPCB assemble certain parts of our PCB, it would save us a lot of time. Also, soldering on the ESP8266 will be a challenge because the distance between the pins on the MCU itself is very small and would be very difficult to solder as amateur engineers. If this piece can also be assembled on the PCB before it ships to us, it would save us a lot of time as well. As the PCBs starts to get ordered, and the process becomes more familiar, we can then be able to adjust and order everything we need for the completion of this project. The only downside to this method would be the price. Having it assembled for you is about 3.5x times the price in some cases. This may be something we look towards down the road if we are in a pinch and do not have the time to spend on the task of soldering every little piece as previously mentioned. Overall, this method is something to keep in mind if we are

Finally, the pricing of the boards is very affordable, and can be as low as $2.00. This is not case for our board; the shipping was most of the cost for the boards rather than the cost of the boards themselves. Overall, this was not outside of our budget when it came to finally ordering multiple PCBs.

13.7 PCB Assembly

If the team decides to assemble the PCB on our own without the help of any supplier, there needed to be a process to assemble the PCB smoothly. Something that made this process go well was the labeling of the location where each specific component will go. This labeling will have to be placed on the PCB by the fabricator of the board. Something else to consider will be how to keep all the parts separated to not get any resistors or capacitors mixed up. Since we have so many semiconductor devices on our PCB that are of different value, it was a challenge to keep everything organized to not solder on the wrong component in the wrong spot of the board. One way to mitigate this risk is to keep the resistor or capacitor in the package until it is ready to be placed. Once it is ready to be placed, the semiconductor will be taken out of the package, and immediately soldered onto the board in the correct spot. This ensured that we do not have any random semiconductors laying around that could be lost. This process was time consuming, however it reduced the chance of losing certain components or placing them in the wrong place. Another way to keep track of all the pieces would be to have the bill of materials open as we're placing materials. This allowed us to cross components off as we work and reduce the risk of missing components. It will also help us to stay organized as to what was put on the board already and what was not.

If the above theory somehow goes wrong and a semiconductor is taken out of the package before it is marked where it needs to go, then there needs to be a plan to make sure that the component is taken care of accordingly. The following steps should be taken to ensure the product is placed where it belongs:

1. Once the semiconductor is found outside of the package and the package is nowhere to be found, stop all production.
2. Measure the semiconductor to see what value it carries, and take one of two steps below
3. Either:
    a. Find the location on the PCB and immediately solder the component on
    b. Or mark the value and label it as such, stash it in a safe, organized place and continue working on the task at hand before placing the part down
4. Continue work once all steps above are complete.

Taking these steps should ensure that every component is accounted for properly and in an organized manner. Not only will these steps reduce the amount of headache during the process of assembling, but it also allowed us to reduce any unnecessary waste when it comes to cost and time. Cost because if we lose components or damage them, had to buy more time because the ordering time of these components could set us back a substantial amount of time. Overall, we need to be careful about how we approach the whole assembly process of the PCB.

The actual order that the components be soldered to the PCB could be significant in the amount time spend soldering on the components. Not only could it be time saving, but it could reduce confusion on where components should go. The plan to solder the PCB components on to the board is to start with the largest components and work our way down. This means soldering on components like the regulators, power transistors, and the ESP8266. This will allow the rest of the components wall into place more easily. Because the larger components will be places, the small semiconductors will start to line up, and it will start to make more sense where stuff goes.

13.8 Testing the PCB

Once our first PCB is ordered, it will be smart to assume that the design will not work the very first time it is created. Testing our PCB effectively will be crucial on how we proceed with the project. If the PCB has a design issue and is not working for whatever reason, efficient trouble shooting and critically thinking effectively will be imperative to improving the design and acquiring the desired results for the circuit. It will also be important to test if ALL functions of the PCB are operating properly. The team will not be satisfied if the PCB design is just “good enough.” We want to ensure that the PCB is doing what was intended of it, and to make sure that every single function works on the board.

The process for testing the PCB is self-explanatory. We connected all the components of our project to the board and generate a signal with a DC power supply to the project to see how everything behaves. Once we have diagnosed and fixed all the bugs that we shall most likely have, then we shall start trying to implement our design with the battery we choose and see if we can generate a similar response. Once all of this is accomplished, all that was left was to design the housing for the whole project to fit into and putting everything together in a way that allowed everything to operate at maximum efficiency.

14.0 Prototype Construction

14.1 Prototype Construction Beginning Notes

This prototyping section is considered to be complete, however it is noted that due to shipping delays, parts availability, cost, and/or defects during construction, components used or specific parts used may have been adjusted in order to meet final deadlines and/or project requirement specifications. This is important to distinguish as some components which were initially to be designed and implemented by our team were instead swapped out for off-the-shelf components due to the inherent inability to continue re-designing and re-ordering PCB’s to be reprinted for testing due to the increasing shipping costs, and further parts costs and availability. With that said, we described the changes made, as they were necessary or as they were done in previous sections, and in subsequent sections.

14.2 Prototyping Components

14.2.1 Processor/Controller

The processor/controller used for prototyping the Pick Pocket Tuner, is based around the ESP8266 microcontroller. The board used during breadboard testing is the NodeMCU Amica-ESP8266 prototyping board which contains the ESP8266 microcontroller as well as soldered on-board pins for prototyping ease through a traditional breadboard or direct connection to the soldered-on pins, with a micro-USB connection for ease of use when necessary to implement software to use for testing. This board was procured by the Team Member Luis Vargas, as he was previously familiar with the board and had spares to be able to provide for testing. This board is a simple open-source IoT platform and is connected/interfaced with through the Arduino IDE and the CP2102 Driver and the appropriate ESP8266 board manager library for the Arduino IDE platform. For readability this board will be referred to as ESP8266, the board, or simply MCU.

The set up for the testing environment for the Arduino IDE assumes that the Arduino IDE is already installed and is working correctly prior to proceeding further. The interfacing between the Arduino IDE and the MCU requires that the board manager for the ESP8266 family of boards be added, manually, to the Arduino IDE to then add the correct board explicitly to be able to upload the software appropriately. In doing so, this enables the Arduino IDE to communicate through a serial connection to the ESP8266 board allowing for the total use of the available pins and functionality of the board. Once this is accomplished, the Arduino IDE must be configured to recognize the board at the correct COM port to be able to gain full functionality at 9600 Baud, which is especially important when using the Serial Plotter or the Serial Monitor in the Arduino IDE. With the assumption that this works correctly, we are now ready to begin utilizing the ESP8266 for prototyping the project.

14.2.2 Sensor

The sensor utilized in the prototyping of the Pick Pocket Tuner remains to be the Piezoelectric sensor defined under the previous Vibration Sensor section, where it is explained what function this sensor is to serve. This Piezoelectric sensor will be directly connected to the MCU to be able to utilize it as the main point of contact at which we shall begin to test the capabilities of vibrational sensing and thus begin our signal processing. Given the importance of this piece to the Pick Pocket Tuner, there were two sensors initially purchased to be able to submit to testing from SparkFun Electronics, the first a circular ceramic disk Piezo sensor (part no. SEN-10293), and the second being a flexible PVDF polymer-film Piezo vibration sensor (part no. SEN-09196).

For both devices they are connected to the MCU through the analog input pin that the board has, as these are devices that generate an analog ‘signal’, vibration, to be processed into what we ultimately need to be able to accomplish what Pick Pocket Tuner set out to do.

14.2.3 Motor

The motor utilized in the prototyping of the Pick Pocket Tuner will start with a 5V DC Stepper Motor, model 2BYJ-48. This motor has a 4-coil unipolar arrangement with each coil rated for +5V, a torque of 0.03 Newton-Meters, and has 64 steps per revolution, thus making it a relatively simple motor to begin prototyping with since it is not difficult to control with microcontrollers, like our ESP8266. This motor was procured through a kit that Team Member Jamie Henry already had in his possession. This motor is interfaced directly with the motor driver, which is identified in the Motor Driver section below. This motor/driver combination is then directly interfaced with the GPIO Pins on the MCU corresponding with the appropriate pins on the soldered-on board. Additionally, this motor requires an additional and/or externally supplied 5V DC power to operate appropriately.

After determining that the previously mentioned motor did not produce sufficient torque to appropriately turn the tuning pegs on a guitar, much less so for a bass guitar, and because of this we decided to move towards using the DFRobot FIT0441 Brushless DC Motor with an operating voltage of 12V that has a torque rating of 2.4 kg*cm. This motor is able to provide enough torque to turn the tuning pegs of guitars, and in comparison, to other motors on competitor's products, would be enough to turn the tuning pegs of other instruments.

14.2.4 Motor Driver

The motor driver used in the prototyping of the Pick Pocket Tuner will start with a Texas Instruments ULN2003AN Stepper Motor Driver, which is already connected to a pre-made PCB. This PCB is to be used as the primary interface/driver for the 2BYJ-48 motor mentioned previously. For the remainder of the Prototyping Components, the Prototype Testing, and Integration of Parts sections, when referencing 'the motor', the reader is to assume that the nomenclature is inclusive of the motor AND the driver simultaneously for simplification and ease-of-readability by the reader.

This 16-pin driver from Texas Instruments is chosen for initial prototyping since it came with the 2BYJ-48 motor in the parts kit that Team Member Jamie Henry already had procured previously. This ULN2003AN driver is made up of seven high-voltage, high-current, NPN Darlington transistor pairs that utilize suppression diodes for inductive loads with a base resistor to each Darlington pair. This combination of devices allows for the driver to maximize the effectiveness allowing operation directly with supply voltages of 5V, over a wide temperature range of 40°C to 105°C. This driver will be directly connected to the appropriate GPIO/Digital pins on the board so that we are able to control the stepper motor appropriately.

Similar to the progress of the initial motor went, the ULN2003AN Stepper Motor Driver was be changed for the encoder that comes integrated with the DFRobot FIT0441 DC Brushless Motor described previously. This integrated controller comes with the leads necessary to be able to interface with, and integrate into our prototype, while being directly attached to the rear portion of the housing on the motor itself.

14.2.5 LCD Display

The LCD Display that will be used in the prototyping of the Pick Pocket Tuner is the Adafruit 1.69” Round Rectangle Color IPS TFT Display. For simplicity, and ease-of-readability-and-reference, for the remainder of this document this LCD Display will be referred to as simply as the ‘display’, ‘LCD’, or as the ‘screen’. The display is an IPS-TFT, which means that this is an In-Plane Switching Thin-Film-Transistor display. This IPS-TFT model is an LED based LCD that due to the IPS allows for some of the best color reproduction and viewing angles among other types of common display panels, and the TFT allows for improvements in image contrast as well as addressability.

This device was procured by Team Member Paul Grayford from Adafruit (Model No. ST7789, Product ID: 5206). The interfacing and integration of the display was determined through technical research such that the display can be integrated using the SPI communication interface on the ESP8266 based on the datasheets of the display. The interface for the display through the MCU is through the SDMOSI, SDCK, SDMISO lines which the MCU can utilize to communicate the necessary data to and from the display’s controller.

14.2.6 USB-C Charging Circuit

The USB-C Charging circuit utilized in the prototyping of the Pick Pocket Tuner, is the Adafruit Micro-LiPo Charger for LiPoly Batt with USB Type C Jack from Adafruit. This charging circuit was procured by team member Lucas Grayford. This charging circuit has breakout pads that allow for 5V, two data lines, D+ and D-, a GND, and VBAT. This charging circuit already has a USB-C jack soldered on, has a 100mA charging current with the capabilities of increasing that charging current to 500mA by shorting a jumper to facilitate fast charging in the device once integrated. Attempts were made to access the data lines on this charging circuit to try and include a battery status readout on the display, however due to a lack of publicly available data sheets on this specific charging circuit, we were unable to include this feature.

14.2.7 Battery

The battery utilized in the prototyping of the Pick Pocket Tuner, at the time of writing this document will start with the Lithium-Ion Polymer Battery – 3.7V 1200mAh from Adafruit. This battery was procured by team member Lucas Grayford. This battery, as described above, is 3.7V output with a 1200mAh runtime. It is currently our choice for prototyping, and further integrating into our design choice.

14.3 Additional Components in Prototype Construction

14.3.1 Piezo Sensor Filter

The filter that was considered being used in the prototyping of the Pick Pocket Tuner was a Resistor-Capacitor based circuit. The purpose this filter serves is to take the input signal from the piezo sensor and then 'clean-up' the signal so that it is more easily read by the analog input pin on the MCU. This filter takes two resistors at 1MΩ, R1 and R2, and two capacitors. One capacitor, C1, is 0.1µF and the other, C2, is 0.01µF. They are all connected in parallel in the following order: In – R1 – C1 – R2 – C2 – Out. On the input side, we connect the Piezo electric sensor on the positive terminal and the negative to ground, on the output side we connect the positive terminal to the analog input pin and then the negative to ground. This is what we shall initially use to clean-up the input analog signal from additional signal noise to the ESP8266 so that we may progress in our prototyping.

Ultimately this filter was removed altogether as once the piezo sensors were affixed to the housing of the device, the group decided that there was no additional input filtering necessary. In future iterations, it may become necessary to revisit, as there may be signal noise emitting from the piezo sensors at very low frequencies.

14.3.2 DC-IN Power for Breadboard

This piece, although seemingly inconsequential, is necessary since it will be used during breadboard prototyping to provide power to our devices when they require additional external power that the MCU cannot provide. This piece is unlabeled; however, it was procured by Team Member Jamie Henry in the same parts kit that was used to procure the initial motor used for prototyping. This DC-IN board takes power from a 9V/1000mA power supply and converts it to either 5V or 3V for use on breadboards. For initial prototyping purposes, it was primarily utilized as a 5V output to provide the necessary power for the motor to be used.

14.3.3 3D Printed Guitar Peg-Winder

The peg-winder that will be used in the prototyping of the Pick Pocket Tuner is a 3D printed peg-winder procured by Team Member Lucas Grayford. This device is attached through friction onto the moving stem on the FIT0441 motor so that we can affix the winder to the tuning peg on a guitar to test if the motor produces sufficient torque to be able to move the tuning peg during initial testing, and ultimately to be able to determine how much to turn the peg-winder to be able to tune the guitar.

14.4 Next Prototype Construction Steps and Ending Notes

At this point we have finalized our first 'official' version of the device prototype. Additionally, the preliminary artist rendition of the mock-up of our system prototype is in Figure 20, below. Fig. 20 contains a breakdown of all the included parts in this initial breadboard prototyping phase with the wiring diagrams of the tested devices. Those devices which are part of the Prototyping Construction section but are still under technical research, i.e. the display, will be included in the mock-up, however specific wiring will not be included in this diagram.

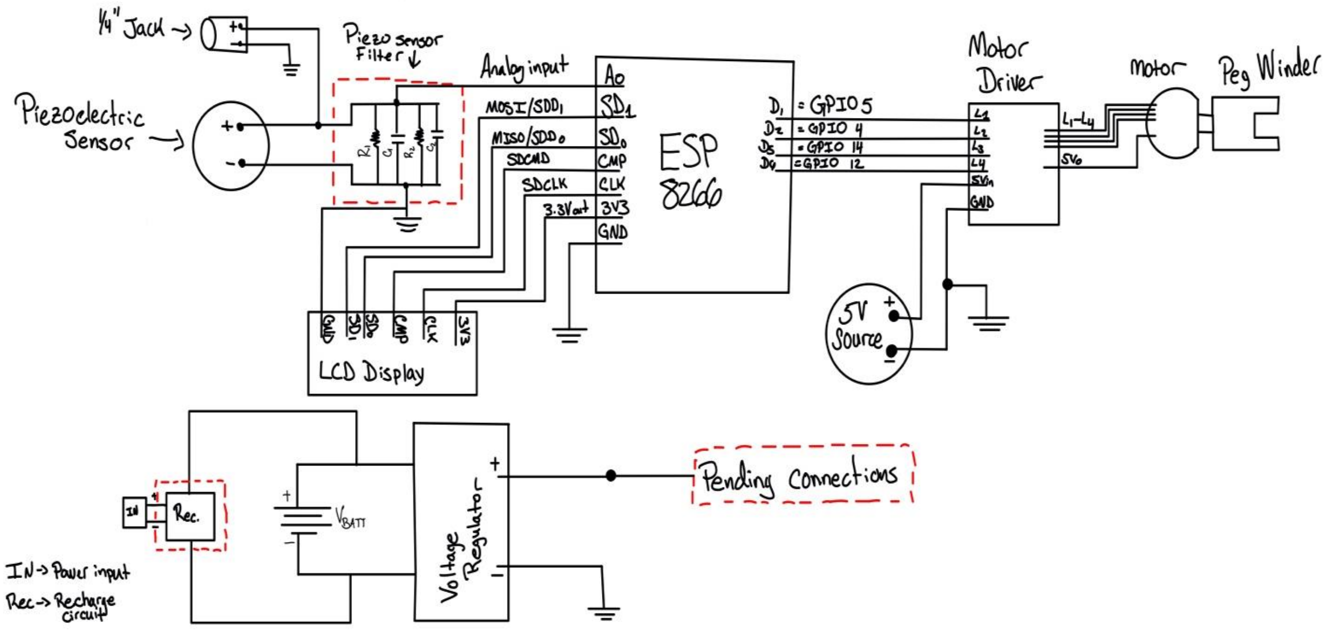


Figure 29, Artistic Rendition of the Initial Prototype Mock-Up

15.0 Inside of the Jowoom

We ended up finding a similar style of a tuning device as the Roadie but a bit more of a primitive type of system that follows most of the same principles that we are trying to reverse engineer. When taking apart the Jowoom it was held together by 5 screws mainly, three 20mm x 2mm screws and two 8mm x 2.5mm screws on the front face behind the peg head where there is also a screw holding the peg head in place too. After prying the case aside there were a quite a few things to learn how the Jowoom was manufactured.

The first thing that was noticed was the brushless DC motor that they decided to use for their tuner which inspired us to make the change to a similar type of motor for our design. The motor we decided to use we is a little slower in RPMs in comparison to the Jowoom, but comparing it to the Roadie 3, our motor has a higher RPM. The motor used inside the Jowoom is about 25% longer than the motor that is being used in our design. The other thing that was noticed about the Jowoom's motor was that it had a 6-pin interface rather than our 5-pin motor. We do not know what their extra pin line was for, but we consider that it could be an extra control line.

There were a couple other things that we noticed that could help us improve upon our design, and one of the things was an extra vibration sensor that was connected in parallel to their system. The first of the two sensors were connected and mounted to the plastic housing of the Jowoom itself, on the inside, so it would be able to pick up the vibrations better when the housing is in contact with the instruments peg head. Then, the second sensor was connected to the circuit board directly so it could transmit the vibration signal to their embedded system. It seems like a smart decision to do that for our housing and properly mount the secondary vibration sensor on the inside of the housing like the Jowoom.

The reason in stating that their system is a bit primitive is that they did not have a processor for determining most of their calculations and it was more of an embedded type of system. Their system has three different select modes for the styles of tuning, which are a little hard to navigate through without the use of the user manual to get the exact setting type that the user is looking for, thus leading to a clunky outdated User Interface and User Experience. The first standard startup mode is the auto mode, which is the easiest mode to understand and works well overall. Just put it on the tuning peg and pluck the corresponding string for that peg and it quickly reads the frequency range that the string is close to and make the adjustment to the peg and does a double beep and lights the green led to tell the user that the string is done and to go on to the next string.

The other two modes are custom, and semi and the user can press the “s”, button on the Jowoom to select the string type customization, which can be a little confusing for a novice to use its interface. The nice thing it has is a string number and the corresponding letter pitch for the string. It also allows the user to select through three different selections being guitar, “uke”, and “chrom.” Uke being short for ukulele setup and chrom for chromatic selection. However, it is important to note that the interface was being stubborn going to those selections after taking the device apart and putting it back together, even following the user manual to get to the settings.

The Jowoom has five buttons but only two of those buttons are really used for the Jowoom, technically there are six buttons if including the reset button, but to access the reset button it requires a needle or pin to press it as it is recessed into the housing. The two main buttons used are the power and mode, simply to just turn on the device and mode button to change between the three available modes of operation when pressed and held. When just simply pressing the mode button it allows the user to manually select between the strings. Two of the other buttons are a up and down for manual winding mainly for quick manual restringing, but our tuner will not have that and rather a restring option in the interface that will get the string to a close tune. Then the last button is the select button which allows the user to change the mode more for guitar, uke, or the chromatic settings.

There is only a plan to use three buttons in total for the Pick Pocket tuner. The buttons will be mainly as stated before in the buttons and switches section to have a left, right and select option, and hold select feature to go back or back to the main menu. Another feature the device has that was noticed later after using it was that it has an auto shut off feature when the device is not used for around over a minute. For our device that won’t be possible if we use a switch method to delegate whether the device is on or off, but there is a thought to add a chirp or beep to notify the user to turn off the tuner if it is no longer being used, so that the power can be saved in the system.

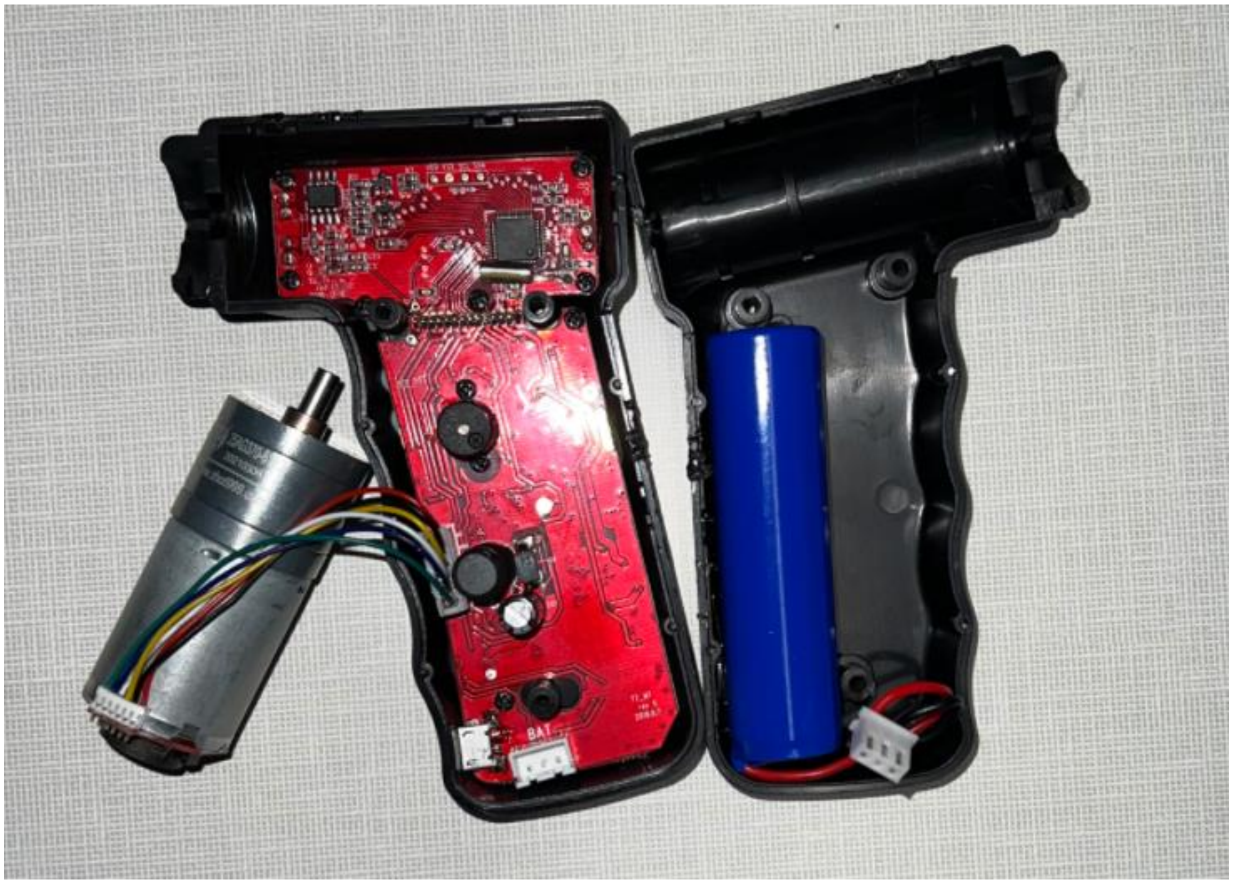


Figure 30, Jowoom internals

They had it set to be very simplified to where it had more of thresholds to adjust the tuning to match the frequency it needed. So, most of the outcomes are already hard set than calculated which can make the system quick and simple to use. For the system that is being designed by us, our intent is going to be more of a mix of the Roadie and the Jowoom so there can be best of both worlds type of scenario. Our device will be more like the Roadie in terms of capabilities, but a cheaper option, however it will be lighter than the Jowoom because their system is not heavy in the hand weighing in at 212 grams, but it still has a solid feel to it, where most of the weight is the motor and battery.

For the Jowoom, their PCB design was a little odd, since it did not actually use a processor. The board fills up most of the upper side of the housing, and there is a lot of empty space not being utilized in the housing next to the battery. We believe that the design of the Jowoom could have been much slimmer, but there are 3 large components on the back of the PCB that caused the void space, these components being a large inductor, capacitor, and a buzzer. We believe that there could have been saved space based on the battery used, if they used a LiPo battery like the Roadie, but our group determined that the idea behind this design decision was that for the Jowoom, they were using this battery choice more for longevity and not needing to make as many recharges.

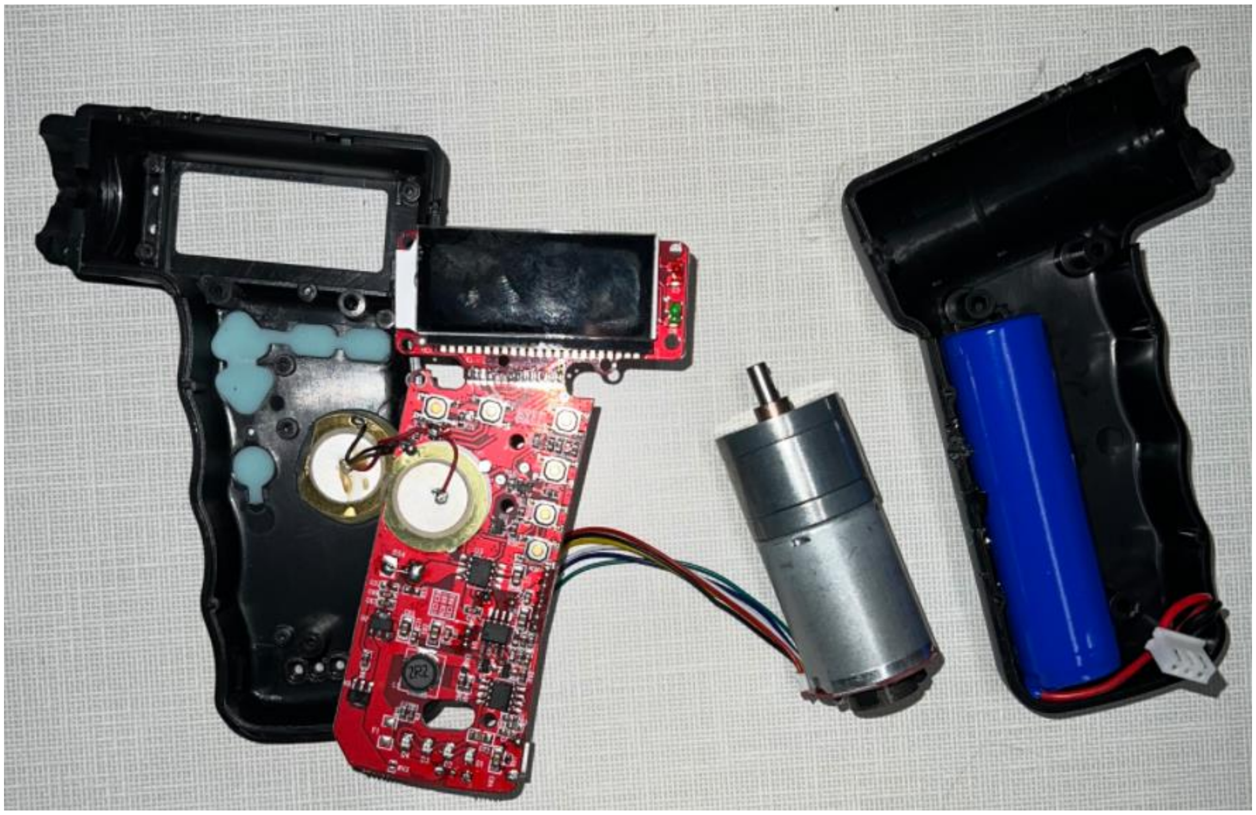


Figure 31, Jowoom internals front side

Overall, we are not going to reverse engineer an exact version of the Jowoom. However, dismantling a competitors product gave our group a significant insight on which direction to begin, and ultimately how to implement some small changes on our system that we may have missed in the first place, or not thought of whatsoever. Additionally, and more importantly, this process allowed us to reevaluate over certain changes in our original design like the battery and motor change.

16.0 PCB Layout

Once all parts were finally picked, and procured, the final PCB design can commence, and we are to begin determining a final layout of all the components on the PCB. First thing is determining what specific components are going to end up being on the PCB, since most of the components are going to have lead wires running to the actual board. The components that are going to have pin headers and run lead wires to the board are going to be the battery, switch, motor, and screen. The main components that are going to be on the actual PCB are the two main regulators, the two regulators stepping our 3.7V battery output to 3.3V for our main PCB and a second step-up regulator going to 12V for our motor. The last component attached to the board is going to be the ESP8266. The rest of the components will have varying pin headers placed on the board.

There are some problems with a significant amount of the components, for example a few of the component's wire-pin headers do not come with matching connector types, so the problem is trying to find the end connectors and making sure they match. The other problem is that the libraries for most of the pin connectors are not available and either need to be downloaded from a potential unknown source or custom design. Instead of potentially wasting money on buying wrong connectors and trying to waste time troubleshooting, our group ultimately decided to cut the lead wires and attach these wires directly onto the pin headers. As we are aware, currently there is also a problem in supply chain issues with many of the components which brought us back to the cost saving idea. A large amount of the connectors that were available, were only available in a large bulk orders amount, and that's if they were even the right connector. Beyond that, most of the vendors had supply chain issues and were quoting a 30-week lead time on their components. The idea for our project's PCB is more for ease of use than looking pretty, also the housing design is going to be determined after the finalized version of the PCB is done so the wires will be out of sight, and outside possible interference from the user. There will be some cable management done when this is all put together within the housing so it will look more presentable, and serviceable, if someone were to open the housing. This was the main idea for getting around some basic problems and keeping the PCB design method to be practical, and relatively simple.

16.1 Ultra-Librarian

Ultra-Librarian was what was mainly used to get most of the schematic and footprint designs for the PCB layout. There were problems with most of the products that were from the Adafruit library, where they would have lookup information for the components on file but not have any actual footprint and schematic files to go with them. It does not make much sense to have products on file then have zero things to provide under the file type for the product. Ultra-Librarian gave footprints and schematics for regulators, ESP8266 and the power MOSFETs for the regulators. For the regulators, even though we were limited to the regulator types that were available for use, due to supply and demand issues that were present for most of the more common and efficient types of regulators. Ultimately, we ended up using the largest regulators, which weren't the most efficient ones because there is not much that could be done about the situation. The ESP8266 gave a nice schematic and the only changes that needed to be made to it was some channel labeling for the input and output pins for all the possible lines that need to be run to interface all the proper components for the system. Even though Ultra-Librarian had its limited uses for what we were able to find and use, it was still a helpful tool with key parts that were needed in the overall PCB layout and design.

16.2 PCB Parts Placement

The first component on the far left of the schematic is going to be the USB-C type charger for the battery, which will connect to the power line of the battery to allow for recharging. The battery and the switch are going to be tied to a 4-pin header. The idea is to cut the lead wires for both the battery and the and the switch and attach female end adaptors so that they can plug into the male pin headers. Ground will run to pin 1 and power from the battery will run to pin 2. Pin 2 and 3 will be wired together on the PCB board design and the power switch lines will be connected to 3 and 4, where 4 will be the main line that runs off to the two regulators that will run in parallel to the rest of the system. The 3.3V regulator will then run into the ESP 8266 input pin. From there are mostly interconnections with all the GPIO pins and other data lines running to each set components. There are a total of eight main components that are going to be interconnected with the ESP 8266, which are the four different pin header runoffs, two 1x2, one 1x5, and 1x7 pin headers. The other 4 components are going to be the three buttons that will be connected to three different GPIO lines and having a data line voltage sent through them to know which buttons are going to be pushed. Each button requires a 10kΩ resistor runoff to ground. The last component tied to the ESP 8266 is the buzzer that is going to be used for indicating when the tuning is complete for the peg being tuned and to go to the next peg or when fully done tuning.

Most of the components are going to be laid out surrounding the ESP8266 to make routing lines as short as possible and keep it as efficient in layout planning as we can. Since most of the components plugging in are going to be done through pin headers it allows us to line up most of the lines to the right places without tangling lines up too much. The two largest pieces that are going to be in the way on the PCB are the two rather large inductors which will take up most of the room on the PCB layout design. The target design is to have the board be nice and compact but not too compact where the inductors might give off some interference to the other electrical components. The layout is going to roughly follow a similar layout to how it is setup in the schematic and for how it is arranged too. The battery and switch will be far off to the left-hand side of the setup where the charging section will be too. Next the regulators will follow and will be formed rather close together and will sit parallel to each other, but inductors will be slightly off stacked diagonally from each other so there will be some room for them. Next to follow is the ESP8266, pin headers, buttons, and buzzer which will off to the right-hand side of the regulators.

The locations for most of the components are going to lead to a more elongated type of breadboard design which is okay for what we want since the housing and PCB are going to be going hand and hand with each other. There are other options for changing the variation on the PCB since there may be problems that could arise on the first board that determine whether it will work correctly or not. This in turn would take a little more time to finalize our design since corrections would need to be made and any adjustments in the layout placement for the board would lead to reprinting the PCB. The board is going to have a flush fit to sit at the base of the housing or to the top of the housing. The main cause for it being on the top side is because of the buttons need to be able to get pressed or else it makes the useless if they are not accessible for the user. The layout could have a change to make them run lead wires too if they are not able to extrude from the housing to be pressed by the user. The goal is to have the board mounted with 4 through holes on the edge of the board with backside surface mounts on the housing that will line up with the holes to hold it in place securely. The through hole size will be for 2.5 mm screws whose length may vary with the final housing design.

16.3 PCB Lead Wired Part Placements

There are some other components that need to be considered for how they are going to be placed around the PCB. This specifically applies to our battery, motor, display, audio jack and the power switch. The battery will be tucked on the lid of the housing with a plastic bracket holding it in place with a small dab of glue to make sure it stays in place better in case the device gets dropped or knocked around. The motor was already described for how it is going to be mounted in the housing, and it will sit just above the ESP 8266 and out of the way of the lead wire. It may protrude out slightly depending how much room is going to be given. The display screen is going to be as close to the buttons as it can possibly be so it is more user friendly because it will not make sense to have the buttons on the opposite side of the screen, original idea for the screen was to put it on the butt-end of the device opposite of the motor head, if that were to be the case then the buttons would need lead wires run to them. The audio jack and switch are the two most flexible pieces for the design because they can be mounted to the housing wherever there is room left over for them to sit, both also are having lead wires run to the board, the only thing to focus on is not tangling up all the lead wires that are running inside of the housing.

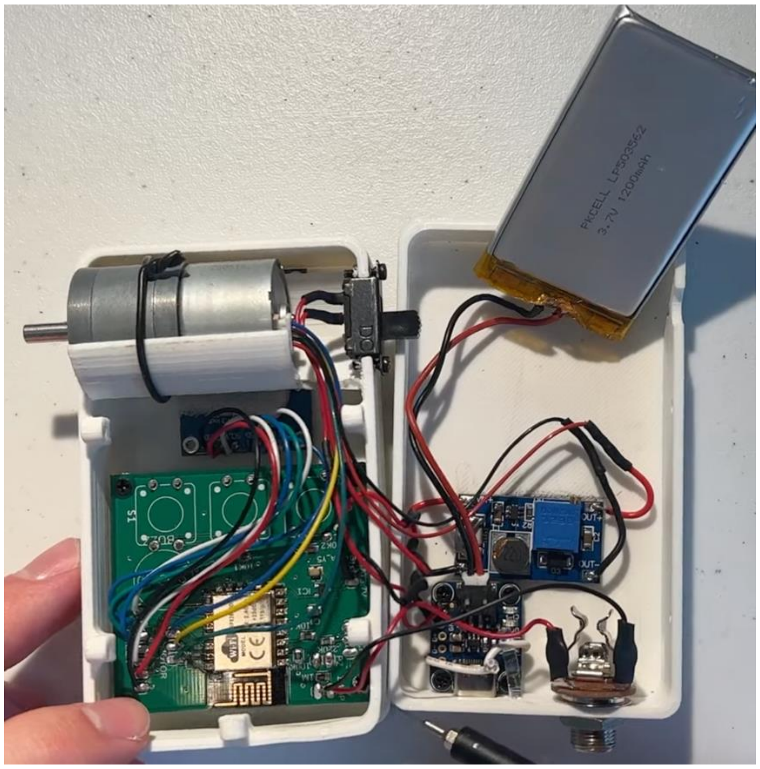


Figure 32, Final Internal Housing Layout

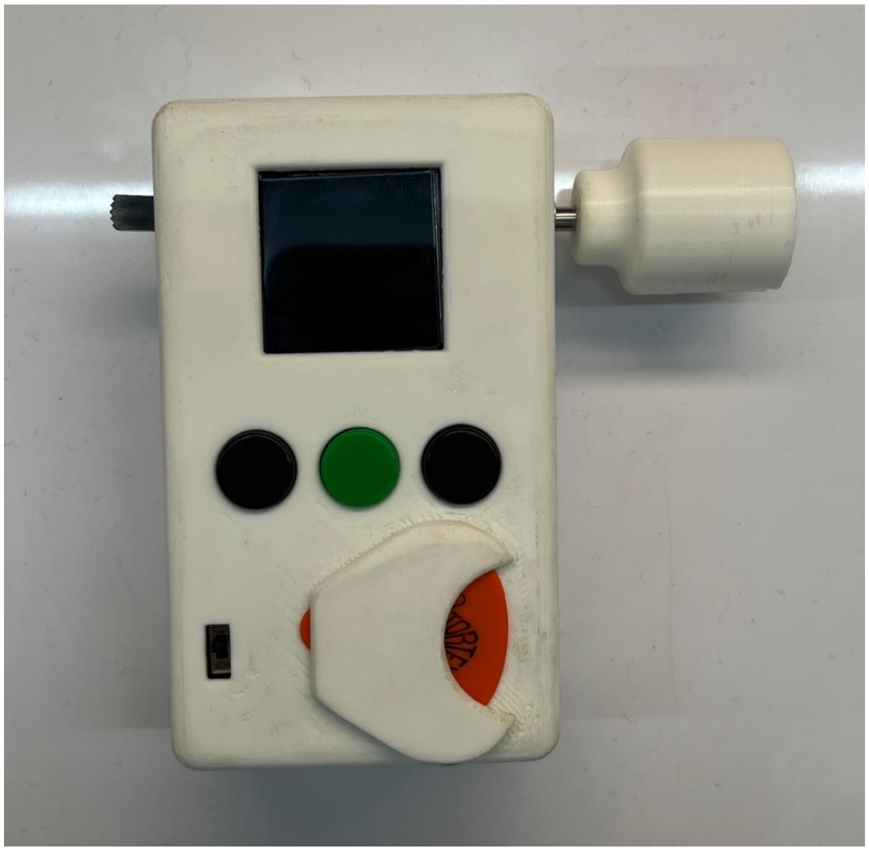


Figure 33, Final External Housing

As can be seen in Figures 32 and 33 above, ultimately, we decided that the motor, and power switch would be set on the top of the housing, with a bracket holding on the motor to the housing, with screws to make sure it didn't spin in place. Secondly, the screen, buttons, selector switch for the ¼-inch input jack and Piezo sensor were affixed to the front face of the housing, where they would allow the user to interact with easily, the buttons and the selector switch were the only things that were attached to the "back" of the PCB so that the housing design would make sense, and on the PCB's "front" is where the remaining components were attached. Additionally, the ¼-inch input jack was attached on the bottom of the front of the device housing, with the 12V regulator, USB-C charging circuit and battery being on the bottom of the back side of the housing. This led us to a finalized compact design without any visible, or extraneous wires, that can be seen in Figure 33 above.

16.4 PCB Prototype

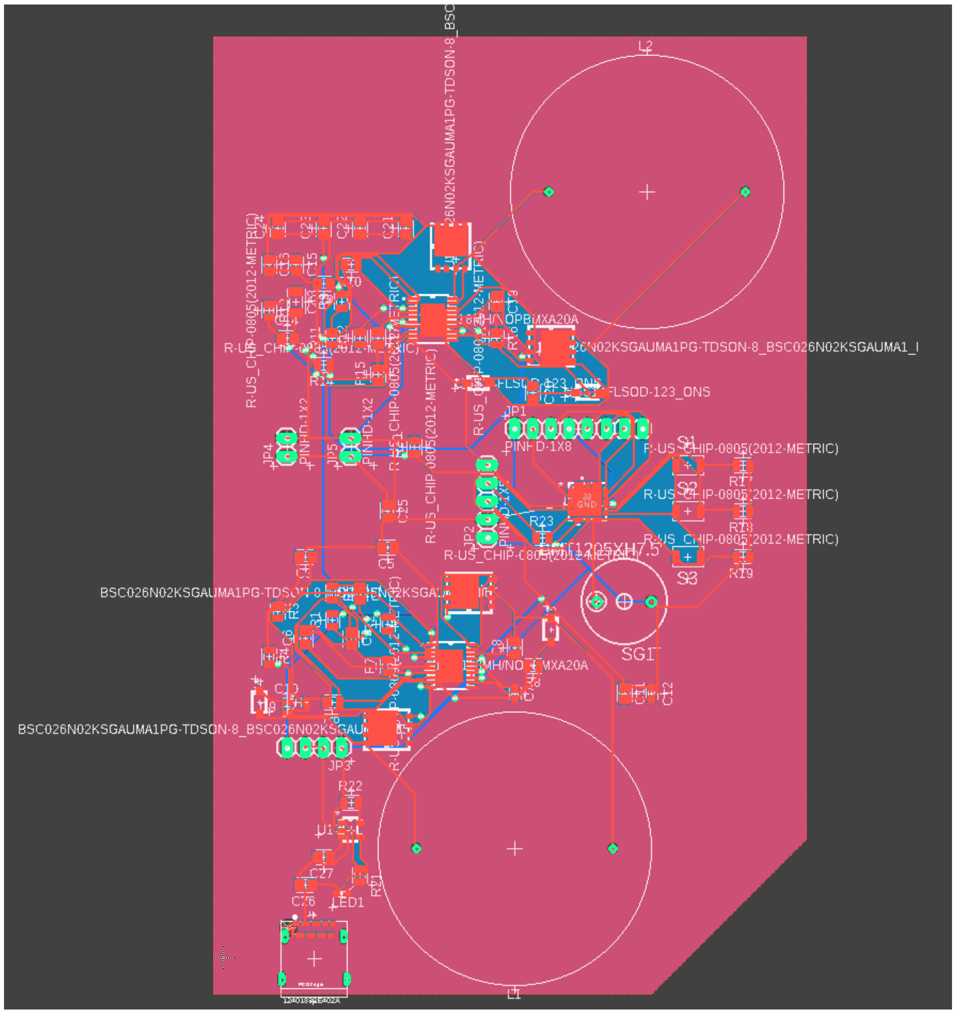


Figure 34, PCB prototype v1.0

There are some finishing touches that still need to be made to it mainly being the through holes for the drilled location, and routing the vias in. There is a difference in the version being used which does not have a rat's nest option to put everything together. There is also an issue where converting the whole piece to the 3D design would not work and put the layers in and it would just put the component pieces in only.

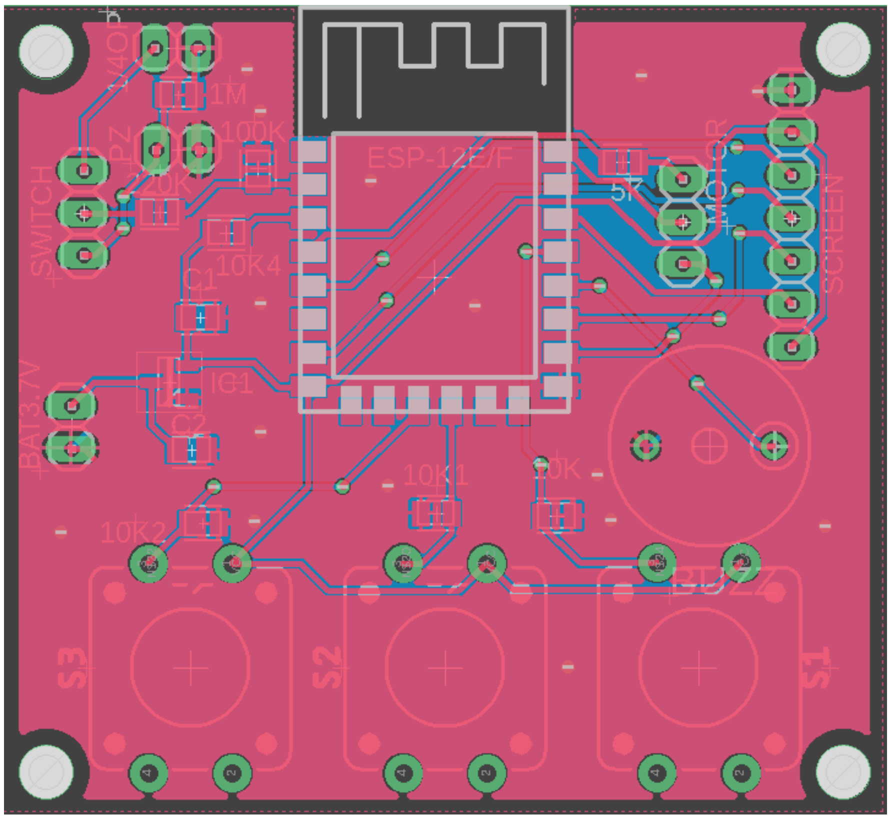


Figure 35, PCB Prototype V3.0

Ultimately after finding some grounding pad errors, and component wire errors that wouldn't allow our device boot, we simplified the design and made our third and final revision of our PCB, seen in Figure 35 above. This board only includes a step-down regulator from our battery supplied voltage of 3.7V to 3.3V, the USB-C charging circuit, and 12V step-up regulator were removed from this PCB, and due to budgetary restraints and supply chain issues we ultimately decided to use off-the-shelf components for our USB-C charging circuit and 12V regulator, the USB-C circuit and 12V regulator can be seen in Figure 33 above.

16.5 PCB Wiring

We decided to keep track in of what wires had which pins of the ESP8266 were associated with to avoid any confusion. This will help keep track of what pins were used and save time from second guessing for what wire went where, while also serve as reference for later if changes in the design were made where, for example, if there was a wire that was previously sent to the wrong spot and can easily be fixed with a simple rerouting. It is not mentioned in any of the tables but the input line for the ESP 8266 for voltage in coming from the regulator is pin 7- CHIP_EN, which enables the chip high for on and low for off and small current consumption.

ESP 8266 Pin # / Type / Other	Motor pin # / Type
10 – MTDI, GPIO12	1 – PWM
33 – GND	2 – Power (GND)
13 – MTDO, GPIO15	3 – Direction
9 – MTMS, GPIO14	4 – FG signal (requires 5kΩ resistor)
12V Regulator	5 – Power (12V+)

Table 7, Motor Pin Connections

Most of the pin connections of the motor were a little more complicated that what they should be. For the first part the motor lines come with a nice ribbon with partial incorrect labeling which is bad and already led to confusing some of the wiring. The ribbon does not mention that is goes with any proper method for the final joint end connector. Some of the lines trade places so it made it harder to determine what each line was connected to and what each line's function was on top of that.

The motor datasheets said it could be used to plug straight into an Arduino board which was incorrect, because even in the diagram that they provide they do not use the straight plug-in connector and rather send out lead wires to the corresponding pin ports that are also poorly shown. Beyond that, the documentation was poorly written, and we had to determine which pins on the Arduino itself were used, why, and how to "convert" them for use with our ESP8266. For the side that is supposed to go straight to Arduino, we snipped that end off completely, so there is no hassle needed to go around and find the proper end connection that will work and trying to configure the lines that were mixed up on the wire ribbon.

To make matters worse, the lead wires do not match the diagrams color coding sequence. The only lines that do match the two power lines for line two being the black power line for negative/ground and line five being red for positive power. The wire coloring coming out from the motor from line one to five was: blue, black, yellow, green, and red. In the diagram provided on the pdf for the layout of the wires from one to five was labeled: grey, black, blue, yellow, and red. This already leads to basic confusion if the lines are even the same, but it's not all bad in the end because the three wires that are the main source of confusion are all going to different GPIO pins. The actual truth will come from the GPIO pins when testing the software through each pin. The lines took more time with having to reconfigure the motor commands in the code.

ESP 8266 Pin # / Type	Display pin # / Type
33 – GND	1 – GND
3 – VDD3P3 3V	2 – 3v3
21 – SDIO_CLK, GPIO6	3 – CLK
20 – SDIO_CMD, GPIO11	4 – CMD
23 – SDIO_DATA_1, GPIO8	5 – SD1
24 – GPIO5	6 – D1

25 – GPIO3	7 – D2

Table 8, Display Pin Connection

The display connection pins are more straight forward when connecting the lines to the ESP 8266. Most of the lines corresponded between the datasheets, but the displays data sheet was not every clear with some of the pin run offs. The first five lines are using the main descriptions and not the GPIO pins, but they were added for labeling just in case, however, the last 2 pin line for D1 and D2 are going to be using GPIO pins for the digital display. There was a debate to whether the display should have been directly mounted to the PCB using a 1x11 pin header so all the pin slots would be filled on the display but not all of them would be used. Instead, it was a better idea to use only the pins that are going to be used on the display and run the necessary lines only. It allows room to be saved on the actual PCB for the final run.

Since the regulators are taking up the most room on the actual board there needs to be anyways that can be used to save room in the end. If the screen were to be placed on the actual PCB too it would also take up unnecessary room where other components could be instead. Running lead lines for the display fixes this main issue, the only other issue is making sure a pin line is not missed for the display to work properly.

ESP 8266 Pin # / Type	Button #
16 – GPIO4	1
18 – GPIO9	2
19 – GPIO10	3
17 – VDDPST, Digital Power Supply 1.8-3.6V	All buttons

Table 9, Push Button Pin Connections

All the buttons will be tied to the set GPIO pin numbers. When a button is pressed it will draw a voltage from the ESP 8266 pin 17 and take the voltage in tied through the connected GPIO line which will allow the system to read the set input given to interact with the interface for the tuner. The buttons will mainly be used for the user to do directional interactions with the interface in this method. Each button will also have a 10kΩ resistor tied to them.

ESP 8266 Pin # / Type	Frequency in Pin #
6 – TOUT, ADC	1
33 – GND	2
14 – GPIO2	Buzzer

Table 10, Frequency Device Pin Connects

The last pin headers are the 2 pins that are going to connect the Piezo vibration sensor and the audio jack. Even though they are two different 1x2 pin headers both pin lines number one are going to run into the ESP 8266 pin 6 which is used for all frequency measurements that will be read from the instrument when the tuner is pressed up against the tuning knobs picking up the string's vibration. The audio jack acts as a more precise way for measuring the vibrations due to picking the direct signals from the electric guitar's pick-ups. It has the same approach where the tuner will be pressed up against the tuning peg but will take the readings through the ¼" jack to tell the tuner how much it needs to tune instead.

17.0 Project Summary – Hardware

A large amount of the time for the hardware side was researching proper parts and part acquisition for the project. It came down to what was mainly needed to achieve the end objective goals and determining what was going to be the best approach to accomplish this. There were a few ways for going about making the correct component choice and it was more of a group effort to figure out what should be used and what worked well for other members in the group. Most of the tasks were divided off into split group objectives. The hardware side oversaw most of the physical components that were going to be used in the system and integrating them together to have a finished form. Only a few parts were managed by the software side which was mainly the ESP 8266 since this choice directly affected the software implementations.

There was a fair amount of second guessing within our group, and where somethings needed further research before moving on to the next step. If both the software side and hardware side were out of sync with each other certain processes started to slow down and would take more unnecessary time to resolve when there could be an easier solution to the answer than what was already being thought of.

However, this is not to say that, for example, the software side did not need the hardware components to proceed because they needed a good amount of the components to do the actual software testing on them to make sure they would work properly to move on. Some of the main parts were mainly the screen and motor for the system testing. There were other side components tested too like the vibration sensor to see if it would pick up the necessary frequencies for instrument testing. After most part acquisition was completed and out of the way it came down to tying everything together in a schematic and forming a PCB layout.

There were significant supply chain issues and ordering certain parts for testing purposes, this also limited what was either originally going to be used for some components, like for example two of these changes were the regulators and the motor choices being changed. Towards the end, even with making some of these component level decisions we were able to accomplish most of the things that needed, or what was at least wanted. The plan was to end with a prototype PCB by the end of our Senior Design I semester so there would be less time lost when it comes to finishing the final design in the Senior Design II semester. The only thing that we fell behind on was having a pre-running breadboard prototype or a more makeshift version to see all the parts working together by the end of Senior Design I, but that ended up being what was worked on first thing of the Senior

Design II semester. We know that our components work but have not been fully integrated with all the pieces together to have at least a partial working system. That was one of the big goals that we wanted to end Senior Design I with, but we were at least able to get relatively close to that end goal and have a PCB design to start with for Senior Design II which led us to being able to have a more refined completed design earlier in the final semester.

18.0 Prototype Testing

18.1 Prototype Testing Beginning Notes

The testing procedures were not specifically written beforehand, and as such, are based on a trial-and-error methodology that is primarily used to determine compatibility and proof-of-concept prototyping. For the sections beyond the ESP8266 section below, it is assumed that the device has been properly set up with use on the Arduino IDE platform and can successfully communicate via the USB connection to be able to upload the testing software onto the board for use, therefore no additional setup beyond testing will be explained.

18.2 Prototype Component Testing

18.2.1 ESP8266 Testing

The MCU that was chosen has been researched and tested by all team members, across various meetings to determine whether or not this MCU would be the choice for our project. It was determined and agreed upon by all of the members in Group 42 that the ESP8266 would be utilized as the center of operations for our Pick Pocket Tuner. The testing methods employed for this device, due to parts constraints, availability, and time, were simple.

We determined that the core components of our project (i.e., motor, LCD, and piezo sensor) were able to be interfaced through the available number of pins, and communication methods (SPI, UART, etc.) available on the MCU. Through further technical research we additionally determined that an external Analog to Digital Converter is likely to not be necessary and was determined to be correct through the testing done of the piezoelectric sensor and ¼-inch input jack. Initially, for this device, since it was already connected to a prototyping board, all we did was connect the pins used as needed, and/or connected the board/pins to a breadboard to then use Dupont jumper cables as needed for the necessary wiring.

Additionally, we tested the ESP8266 with both a Windows based machine and Mac-OS based machine and determined that due to the Arduino IDE framework/platform, this resulted in the MCU being platform independent and able to be programmed by either operating system, if the correct drivers and libraries were installed correctly as it was discussed in the Prototype Construction ESP8266 section. Once we had determined that this preliminary testing was a success, we began to add more and more components, and software, to the breadboard prototype until we ended up finalizing a working breadboarded prototype.

18.2.2 Piezo Sensor and Filter Testing

The Piezo sensors that were chosen were tested primarily by Team Members Paul Grayford, and Luis Vargas. Since it wasn't immediately clear which piezo sensors would be needed, the team procured a Snark ST-8 Super Tight Clip-on Tuner to disassemble, to attempt to reverse engineer the method in which this device operates. This Snark ST-8 operates using a silicone-like covered clip that would be placed on the headstock of a stringed instrument, like a guitar, and then would pick up on the vibration waves carried from the instrument to then display whether the string strummed is in correctly in tune or not. The waves would travel from the headstock to the silicone clip, then through a ball-and-socket joint made of similar silicone/plastic material, and then through the ST-8 housing where it would be picked up by the sensor that would transmit the information on to determine the tuning. Knowing this, we disassembled the ST-8, and determined that the sensor used by the ST-8 is a ceramic disc piezo sensor. We utilized this information as encouragement that the disc sensor that we already had would likely be the correct choice but kept the option for utilizing the polymer film sensor as well just in case.

In both the piezo disc and the film sensor, we had to solder on some longer wires to be able to use as leads so that they may not only fit the breadboard, but also reach it to begin with as the leads that were soldered on during manufacturing were insufficient in length and thickness to be able to be used appropriately during testing. The testing of each sensor was initially carried out in a manner to determine if they were operational, then to determine how the analog input of the ESP8266 would read the information generated by the sensor, then how the sensitive the sensor itself would be based on testing methods.

The operational tests of the sensors were simple, we used a circuit with a resistor of 1MΩ between the positive lead of the sensor and ground to attenuate the generated signal, and then connected this into the analog input pin, A0, of the MCU board, with the negative terminal of the sensor into ground with a ground pin from the MCU board into the breadboard to complete the circuit. Once this was done, we initialized the Arduino IDE, and utilized the sketch example "Knock", which is included with the Arduino IDE, but also describes how to set up the analog sensor so that the program can recognize the input from it. Following this, the Knock program was compiled and uploaded onto the MCU so that we could begin testing of the sensors.

Once the IDE informed us that the upload completed successfully, we opened the Serial Plotter that is a part of the Arduino IDE, and we noticed a single line in a graph that would greet us whenever we ran this program. In doing so, we then began to test the sensor by ensuring that it was lying flat upon a wooden desk/table, and we would knock on the table at varying forces and repeated this process for both the Piezo disc and film sensors. From this we determined that the disc sensor would be optimal in detecting the vibrations from the stringed instrument, since the film piezo requires that the film itself vibrate/resonate for it to detect the necessary information, so it would not be used.

For the next test of the piezo sensor chosen, we separated the main housing of the ST-8 from the clip and utilized the ball-and-socket joint with the clip to simulate a string vibration to see if the sensor/MCU combo would be able to pick up on the signal at all. To do this, we took the clip and covered an iPhone XS earpiece speaker and used the YouTube app to play a series of tones at different frequencies, 440Hz, 1000Hz, and 323Hz. Once those frequencies were playing, the ball end of the stem attached to the clip that is then attached to the iPhone XS speaker, is placed on top of the sensor as the Knock program is still running to see if we would be able to detect the tone from the frequency as it travels through a different medium.

This test proved to be successful, and we were able to see in the Serial Plotter that the different frequencies on the tones generated different analog inputs on through the sensor, which meant we were in the right direction. However, it was at this point that we also noticed that the sensor was always introducing some sort of noise into the signal, and we devised a simple filter to try and attenuate that noise from the sensor into the Analog input of the MCU.

Using the filter mentioned in the previous section and repeating these two tests, we determined that the noise introduced was reduced, and the analog signal read by the analog pin was not adversely affected. To solidify our initial findings, we utilized an oscilloscope to determine whether the piezo sensor was transmitting the correct frequency into the analog input, and we found that immediately after the sensor we were matching the frequency from the tone. This same test, we repeated using two channels to test both the signal coming off of the piezo sensor directly and then after the filter, as a comparison, and the resulting frequency reading at both ends was matching to the tone being generated. Figure 21, Figure 22, and Figure 23 below, serve to demonstrate the prototyping breadboard set up, before including the motor for movement, and the oscilloscope response when reading from the piezo lead wires, and then after the filter is added.

This testing done on the piezo sensor the base point for the testing for the rest of the devices used. Since we are able to accurately read a vibration at a specified frequency, and correctly transmit that information to the MCU via the analog pin, we decided to move on to not only testing the other devices, but further refining this basic operation of the piezo sensor in our prototype.

While completing further technical research, the general consensus was determined to be that we should add a second piezo sensor to our prototype/device. Initially, while using one sensor we determined that the sensitivity while knocking on a wooden table, and through the iPhone XS ‘test’ employed previously yielded satisfactory results. However, once we procured the Jowoom Guitar tuner, and subsequently disassembled the product, one of the immediate things that stood out to us is that the Jowoom employs the use of two piezoelectric disc sensors simultaneously, however they are connected in parallel as seen in Figure 36 below.

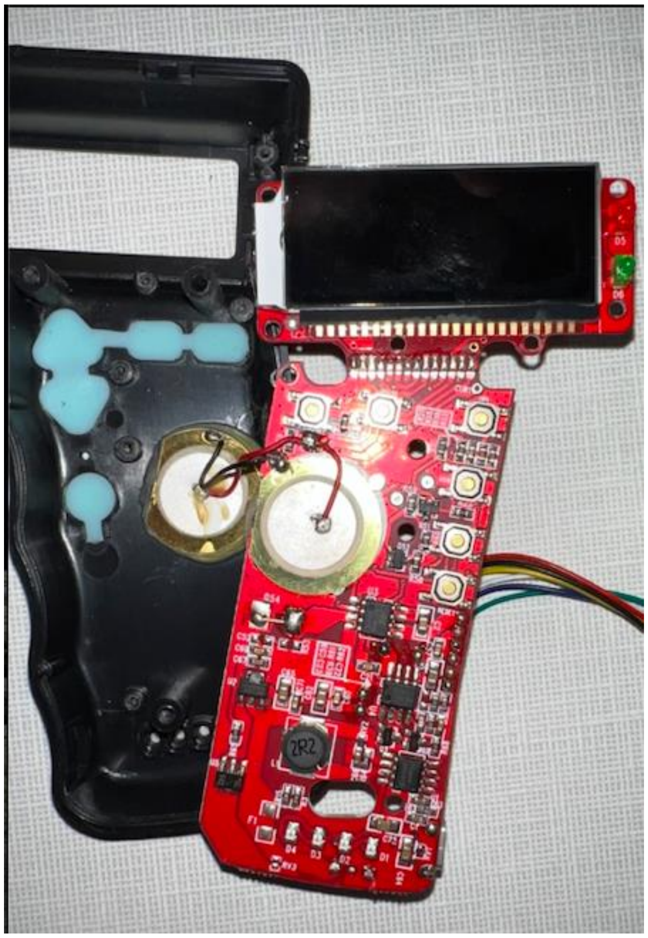


Figure 36, Jowoom Automatic Tuner (deconstructed) showing two piezoelectric sensors in parallel

Without being able to compare it to other in-market competitors, like the Roadie3 Tuner, to see if these other competitors employed this parallel-piezo design the team decided to implement this piezo sensor set up to see how the sensitivity and accuracy would be affected by connecting the two in parallel.

Through further technical research, our group determined that the implementation of the piezo sensors in parallel, without being able to 100% confirm with the manufacturer of the Jowoom tuner, was a design decision to produce more consistent results. As can be seen in Figure 36 above, there is one piezo sensor directly attached to the housing of the Jowoom device, and another to the PCB, when noticing this specifically it led us to look into "seeing" the piezo sensor as a device akin to an electric guitar's pickups, where wiring pickups in parallel is a fairly standard procedure since you can modulate the incoming signal from the strings using a switch and the pickups on the guitar to obtain a different sound. The concept of adding a secondary piezo sensor is similar to this, when considering the piezo sensors as pickups rather than the individual electric sensors that they are. So, this reasoning leads us to believe that the integration of a secondary piezo sensor would assist in further isolating the incoming analog signal while also boosting the necessary sensitivity so that we are able to take the analog input from the piezo sensors and obtain a more accurate representation of the incoming frequency when repeating the iPhone XS "test" that previously yielded satisfactory results.

With further testing methodologies implemented, we found that the input of the oscilloscope was introducing unnecessary noise into our piezo sensors, giving us erroneous readings. It was through this discovery that we decided to then simplify our design and remove the filters altogether since there was no real discernible difference in accuracy when testing the piezoelectric sensors with or without them.

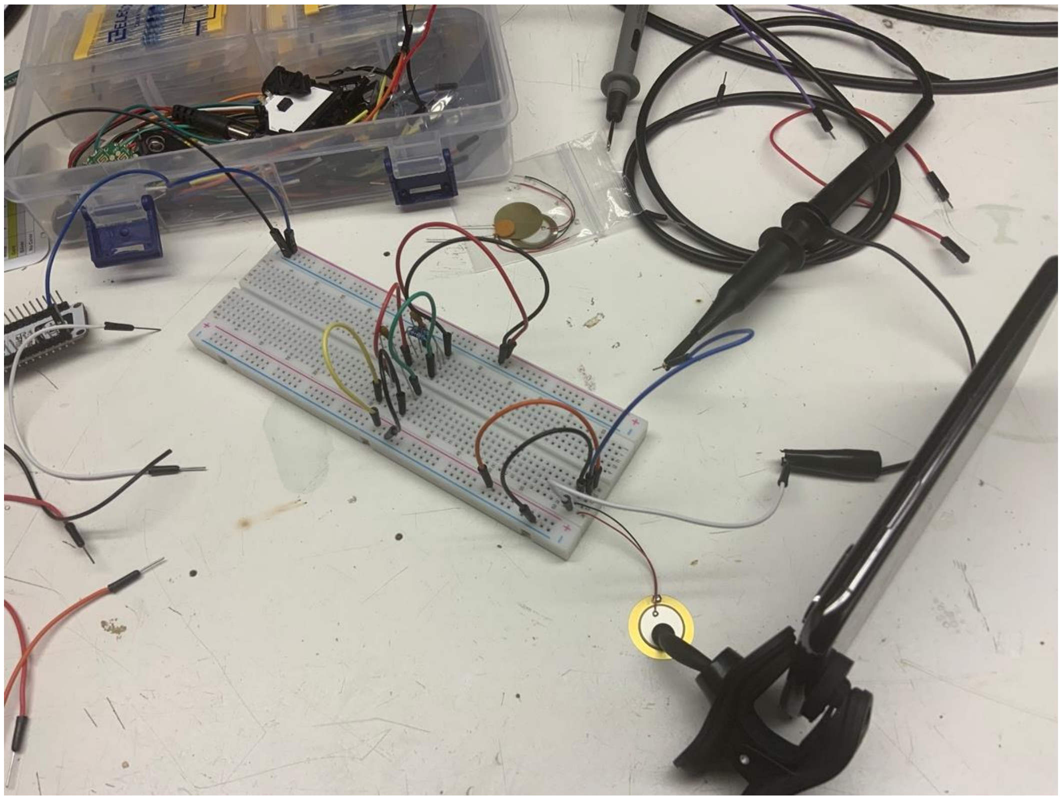


Figure 37, Breadboard Prototype Demonstrating iPhone Speaker Vibration/Tone Test

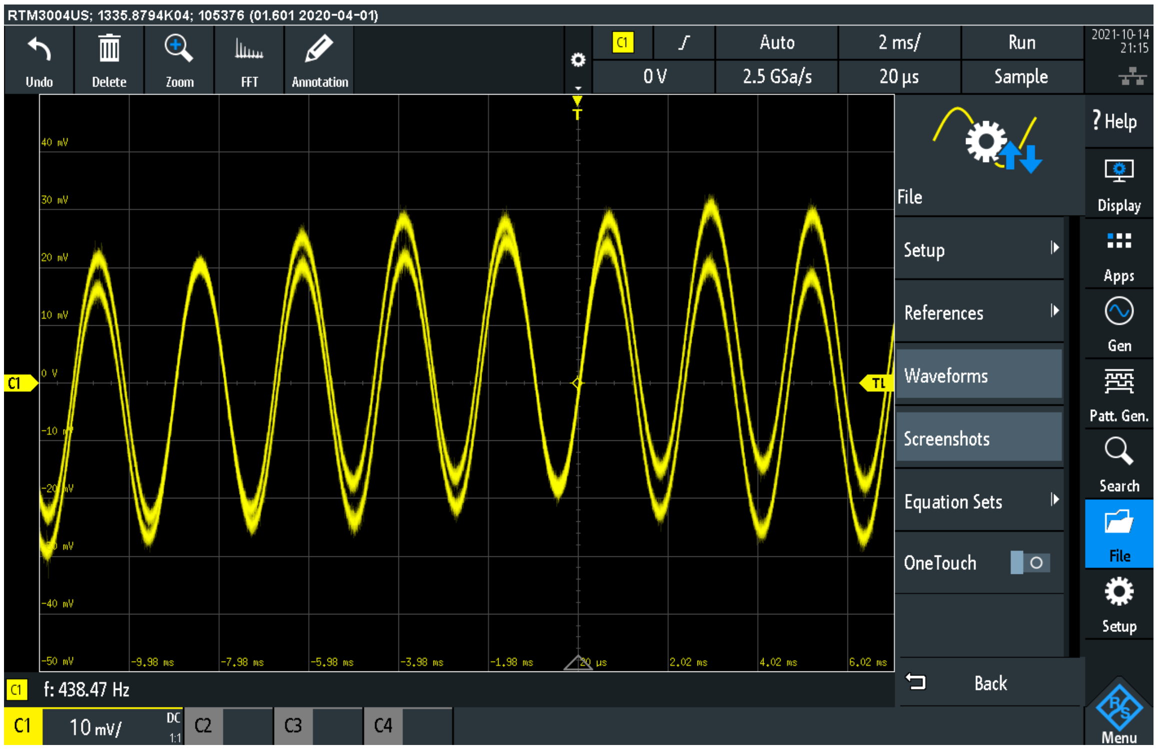


Figure 38, Oscilloscope Frequency from Piezo Sensor, iPhone Test

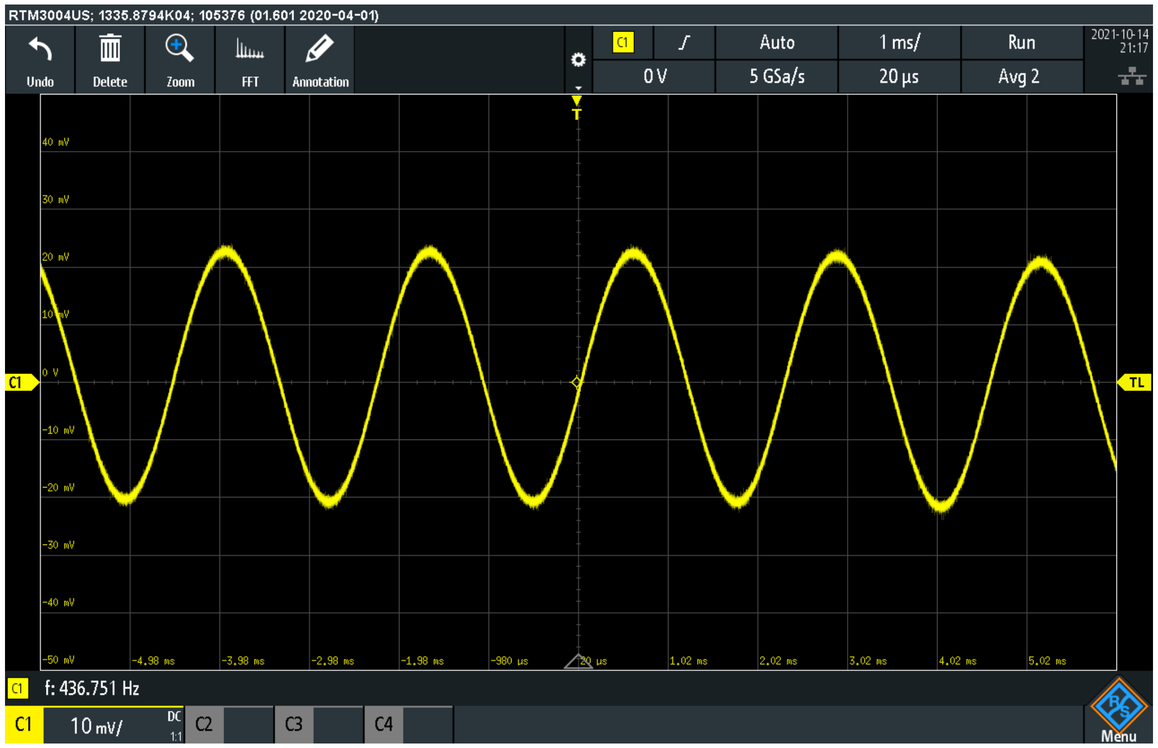


Figure 39, Oscilloscope Frequency from Piezo Sensor, iPhone Test, with Filter

18.2.3 Motor Testing

The testing of the motor is very similar to that of the Piezo Sensor. It follows a step-by-step sequencing of attempting to determine the operational capacity, then introduction of variability to determine if the device is adequate for our final operational purposes. To test the motor/driver, aka motor, similarly to the piezo, we utilized the Arduino IDE provided example of “stepper_oneRevolution” program and modified it slightly to match the operational capacity of the motor so that the steps for one revolution and speed in the program matched the specifications of our motor, and we also changed the pins used in the program to those we would use on the MCU.

After doing so, we wired up the motor driver board to the appropriate pins so that in the driver PCB pins IN1, IN2, IN3, IN4 correspond with physical pins D1, D2, D5, D6 on the MCU prototyping board. Once done, we added the necessary power using the DC-IN board mentioned earlier and attempted to compile the program so that we could get the motor turning. After some debugging of the program, due to non-matching GPIO pins in the code, we were able to get it turning. The testing of the operational capacity of the motor was a success.

It was at this point that we would attempt to run the device again, however we would attach the 3D Printed Peg-Winder onto the stem to see if we could turn a tuning peg on a guitar. After completing this test, we determined that this specific motor, the 2BYJ-48, does not produce enough torque to turn the tuning peg thus we would have to do more technical research to choose a motor that would work. However, with these findings, we decided that it would be beneficial to test the integration of the motor into the prototype so that we can test to see if the ESP8266 can transmit the information so that we can turn the motor based on whether the sensor detects a knock, or a tone.

So, we first set up the testing environment first so that the filtered sensor inputs into the analog input of the MCU, and with the sensor lying flat on the table to sense a knock on the wooden table. In doing so, we also wired up the motor so that it could also be used when the analog input is recognized by the MCU the motor would rotate. We modified the existing code and integrated the working parts of the “Knock” Arduino example so that we could turn our motor head whenever there was a knock sensed. After compiling and uploading the code, we opened the Serial Plotter and began knocking the table to find if the integration of the two devices worked, and it did. The motor moved whenever a knock was sensed from the table.

Building upon this success, we took the sensor set-up and modified it so that it would now resemble the testing setup for the detecting the tone generated from the iPhone speaker. Using this set-up, we began playing a 440Hz tone and placed the ball-and-socket ball end on the sensor and saw that the sensor was picking up the tone successfully, and the motor was moving consistently since the vibration was non-stop. This successful test demonstrated to us that the integration of the three devices, so far, was successful. From here, however, since this motor was not adequate to turn the device, it was determined that further testing on the motor would be suspended since we would need to replace it with a different one.

Once we changed the motor, we re-conducted the above testing, with the appropriate pin and software changes, and we were able to achieve the same response based on input from the piezoelectric sensor. Further into the testing, of the motor, we integrated the three main components, ESP8266, piezo sensor, and motor, even further through implementing a simple fast Fourier transform function that would allow us to determine an input frequency reading and seeing if we would be able to relay a PWM signal to the motor so that it would move if a specific frequency was detected. This all proved to be successful so far, and so we decided that we would then spin the motor for a small amount of time to see if it was strong enough to turn the tuning pegs of the guitar. Once both of these tests were successful, we moved into the next testing.

18.2.4 LCD Display Testing

The testing of our LCD Display is very similar to that of the Piezo sensor. We essentially attempted to follow a step-by-step sequencing to determine the operational capacity of the procured piece. This was done in order to ensure that the LCD that we received was not a dead-on-arrival piece. To test the device, successfully, we had to solder on the header pins onto the device in order for us to be able to interface with it through the ESP8266 and the Arduino software. Once the header pins were soldered on successfully, it was noticed that the LCD Display was able to accept a voltage input of 3V to 5V DC and was capable of outputting a regulated voltage of 3.3V, thus reducing the need for an added voltage regulator in the overall project by one.

To really begin testing the LCD, we used the guide that Adafruit has for wiring it to an Arduino development board and made the necessary pin adjustments in order for our ESP8266 development board to connect and interface with the display via SPI. Initially, the wiring diagrams were seemingly correct and were properly implemented when conducting the previously mentioned adjustments, however there was no backlight after connecting the correct DC voltage to the appropriate pins. Thinking that this may be due to the controller for the LCD not being directed to utilize the supplied voltages to turn on, we proceeded to attempt the connection of the two devices through the Arduino software using the libraries for this LCD display supplied by Adafruit for use with the Arduino IDE. These libraries, “Adafruit_GFX”, “Adafruite BusIO”, “Adafruit Zero DMA”, “Adafruit ST7735 and ST7789”, “Adafruit SPIFlash”, and “SdFat – Adafruit Fork”, are the ones that would initially be used to try to interface the ESP8266 MCU with the LCD display through the ST7789 TFT driver that this LCD display uses.

Further following the supplied Adafruit guide for testing this LCD, we attempted to utilize an Arduino IDE example provided by the “Adafruit ST7735 and ST7789” library called “graphicstest”. This example driver code needed to be updated/modified according to the specific display and display driver that was being used, since this is a general-use code provided by the manufacturer. Once this code was updated, we ran into significant issues trying where the compiling of the program would fail, and we were unable to determine the operational capacity of the LCD display. In order to determine the source of the issue so that we could come to a solution, we began testing each and every one of the libraries being used in the “graphicstest” example code and determined that the very library that provided the example, was causing compilation errors on a MacBook Air.

Further testing was done on the Arduino environment on another computer, but the same errors were run into when trying to flash the example code to the processor. These issues led to the switching of testing the LCD screen on the uPyCraft environment with a self-created test code in MicroPython. The SPI connection was researched and implemented based off similar displays and processors. However, further issues were run into when testing the display on the new environment.

The SPI connection was successfully created, but unfortunately the test code was not drawing onto the display correctly. After conducting further research to iron out all the issues, since the tuner is heavily reliant on visual feedback to the users, we determined that the cause of our issues was software based, and not hardware based, so it would be a relatively easy fix. Once the connections were successfully made, we were able to begin drawing onto the screen and we created our own modular GUI functions that would draw the necessary info onto our display.

18.2.5 FFT Input Testing

Alongside testing the input piezo sensor, we shall also test the implementation of the fast Fourier transform on the input data sent from the sensor. We shall use this information to then display the computed frequency onto the display once they are all connected. The procedure would be the same as the testing of the piezo sensor. However, we shall have a program sampling the analog voltage input and performing the transform to test the accuracy of the sensor and the accuracy of the written code. This would let us use an average of the input voltage to be able to determine what the frequency being read is. The implementations that we used after testing and modifying some of the parameters led us to being within 0.05Hz accuracy within certain testing cases.

18.2.6 USB-C Charging Circuit and Battery Testing

The testing of our USB-C Charging Circuit and battery was more for a verification of operational capacity to determine if either of the two devices were dead-on-arrival, then finally overall use time. Further integration into the system will determine the runtime on a single charge, the time needed to charge the battery with both a 100mA charging current as well as with a 500mA charging current from 0% to 100%, amongst other tests. We have determined that the battery and charging circuit both work since we are able to connect the battery and charge it and determine that it has been charged with an external voltmeter used once the parts were acquired.

Additionally, since the charging circuit we are prototyping with has two data line pads, D+ and D-, it is our group's intent to utilize these pads during testing to try and determine the current status of the battery, i.e., whether it is charged or not and the percentage of battery remaining on the device. Ultimately, due to not being able to find appropriate datasheets that pertained to these data lines, we were unable to fully integrate the battery percentage/status into our finalized version.

Finally, once we had fully integrated and defined all the components into our breadboard prototype, we fully charged the battery, and we found that even when heavily using the motor, starting, and restarting the device, it was not necessary to recharge the battery for 4+ hours. This came as a pleasant surprise because it proved to us that we didn't need to keep charging the device in between uses. This was determined by the fact that the motor was the component that had the largest current draw and thus being able to run it consistently would be no challenge for our battery.

18.3 Next Prototype Testing Steps and Ending Notes

Once the previous preliminary testing was completed, we began testing the breadboarded prototype thoroughly. We began by determining the parameters for which we would measure accuracy of our tuner, and then how we would refine the motor movements based on the inputs received from the vibration sensors/input jack.

The usage of the Fast Fourier Transform is what allowed us to get an accurate representation of the input frequency by converting the analog signal input into the digital signal needed to be able to use our software-based calculations to analyze it, and therefore, adjust the necessary motor movement so that we could "officially" tune a guitar. Since we have determined that the motor had sufficient torque to turn the tuning pegs on the guitars, we had available for testing, we decided that the overall integrating and testing of our device would begin, and we began collecting the necessary data and information to ascertain the overall effectiveness of our device.

Through what essentially was trial and error we were able to fully accomplish what our project requirement specifications, and more importantly meet the tuning accuracy and speed of tuning required.

19.0 Software Block Diagram

The main purpose of the software component for the project is to take in inputs from both the user and the sensor. This will provide a created algorithm with the desired string tuning frequency and what frequency the string is currently tuned to. This will allow the algorithm to detect the difference between the two frequencies and then turn the motor accordingly to wind the tuning peg to the user desired frequency. The software will also communicate with the display to show any measured inputs and an interface for easier access to options for the user to select their input.

Our software will have preset frequency values for the user to choose from when tuning strings along with past used frequencies, stored in a library. The program will display to the user the frequencies in order of most used for convenience. This will also save some time since our users will most likely be re tuning instruments to their previous frequency with our project. After the frequency is selected, the software will prompt the user to pluck the string that they are tuning, and our sensor will pick up its frequency with a fast Fourier transform to save computation time.

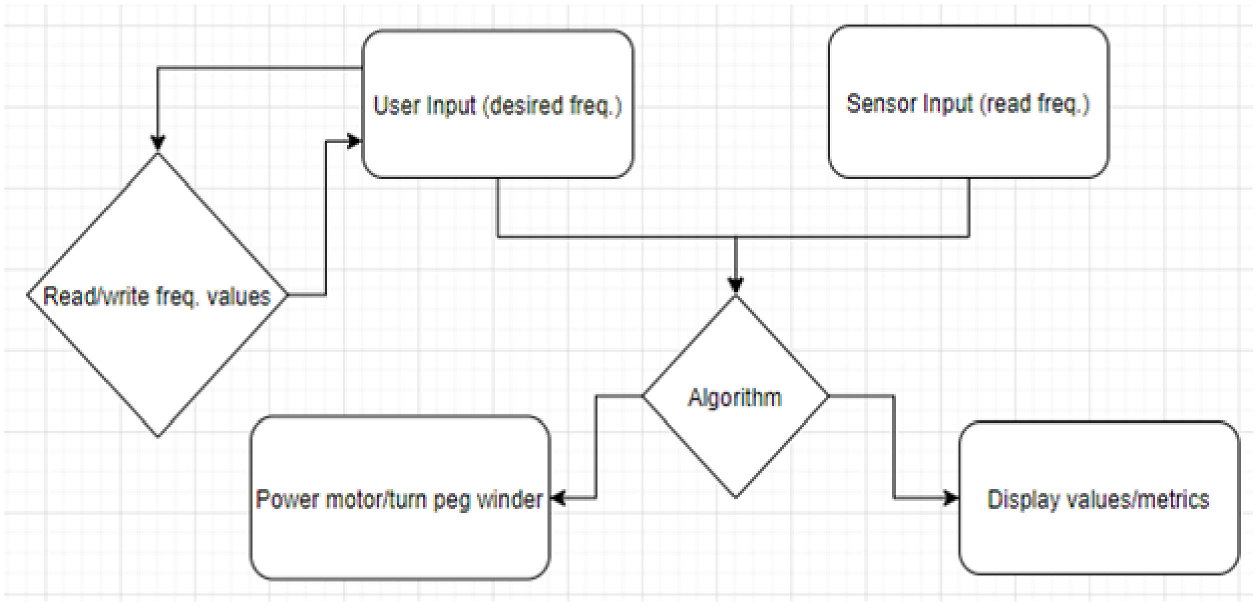

User Input (desired freq.)
Sensor Input (read freq.)
Read/write freq. values
Algorithm
Power motor/turn peg winder
Display values/metrics

Figure 40, Software Block Diagram

20.0 Algorithm

Using data science, we shall find a correlation between a string's gauge and the torque needed to tune that string, C_{gauge}. This correlation will then be translated to voltage supplied to the motor to turn the string to the correct frequency dependent of the gauge. We then can use this data and test our motor to find the relationship between voltage supplied to the motor and the change in frequency of the string that was winded, C_{factor}. This will allow our algorithm to scale with many string gauges.

The process for acquiring the constant, C_{gauge}, is to study the relationship between torque needed to turn the string based on the certain string's gauge. With this torque relationship between gauges, we can then translate that into a factor of voltage needed to send to the stepper motor for the certain gauge. Further researching and testing will have to be done on the stepper motor to find out the steps needed for a certain amount of torque applied to the string peg.

The process for acquiring the constant, C_{factor}, will be plotting steps sent to the stepper motor against the change in frequency on the string being winded. Using this data and plotting the results will help find a correlation between the two values and that correlation will present a constant that is present within the points and will allow more accurate calculations in the algorithm.

The algorithm will have two inputs, f_{sensor}, f_{user_input}, and use them alongside the defined constants/correlations, C_{gauge}, C_{factor}, to output the correct voltage to the motor to turn the peg winder so the string reads the desired frequency, V_{motor_output}. The function created is the difference between the desired frequency and the frequency of the string that was read by the sensor. This difference is the main factor of how far we need to turn our peg winder to match the desired frequency. This difference is then multiplied by the constant and correlation defined to create the correct voltage to send to the motor/motor driver based on the string's gauge and how much torque is needed to correct the string's frequency.

$$V_{motor\_output} = \left(f_{user\_input} - f_{sensor}\right) \times C_{gauge} \times C_{factor}$$

Equation 1. Algorithm for Motor Output

Ultimately, due to time constraints, and inability to obtain sufficient data points to be able to create an accurate data model representation for use during our motor movement, we ultimately decided that a time-based motor control would also work. This cut down the necessary time in software testing and implementation as we could have a pre-set time input that is determined by the calculation of the distance from what our desired tuning is in relation to what the device is currently reading as an input from the guitar.

21.0 Software Features/Functions

21.1 Fast Fourier Transform

When reading the filtered analog voltage from our vibrational sensor, in order to use it we must convert it to the frequency domain to send to the algorithm for accurate tuning. The input from the vibrational sensor is in the time domain and a Fourier transform would convert that reading from the time domain to the frequency domain. However, the discrete Fourier transform has a time complexity of $O(n^2)$ where *n* is the size of the input data being read. The fast Fourier transform will be used in our project as it has a time complexity of $O(nlogn)$ which is significantly faster and will save computation time when tuning each string. This transform will help satisfy our multiple time-restraint technical requirements with fast computation at each string. During the tuning process, the sensor will sample the readings from the plucked string over a certain amount of time. After the reading is complete, we then perform the fast Fourier transform on the wave and obtain the frequency to send over to the main algorithm to finish tuning the string.

21.2 Tuning a New String

The Pick Pocket Tuner will have an option for tuning a brand-new string. This means that when a string is being replaced, the user will select an option for tuning a new string. When the user selects this option, it will prompt the guitar player to select the string they are replacing on the interface, and then select a “go” option for when the string is ready to be wound. The motor will automatically start to turn, bringing the string up to tension. The device will then ask the user to start plucking the string of the guitar. As the string gets tighter, the device will be able to read the frequency with the sensor or the output jack and tune the string in one easy step. The goal is to be able to complete this process in under a minute. The challenge will come with the quality of the electrical components, as well as with the quality of the software behind this option. The way that this code is written is a crucial part in how fast and accurately the tuner can tine the string. This feature will be required, because the competition does not have a setting like this, thus causing our product to have a competitive edge.

When this option is selected, the software will turn the peg winder a certain amount without prompting to get close to the lowest frequency among the six strings. This amount will be calculated with further testing and researching. We shall find the average frequency of a newly installed string from a sample of users, and we shall use that average frequency to calculate the difference between this and the resulting low frequency. After the string is brought to the low frequency, the software will work as normal and prompt the user for the desired frequency of the string and tune it.

Making this process as automatic as possible is key when trying to stay under roughly the same time frame when tuning a guitar with six new strings as opposed to a guitar with six strings that are only slightly loosened over time. This will provide ease-of-use for all occasions and reasons to tune multiple strings on a guitar and make our product more valuable.

21.3 Display Interface

The interface for the device is where some artistic liberty can be taken. The final interface was constructed towards the end phase of the project. The group wants to make sure that user input with the interface works before the display is modern looking and streamlined. However, this will be a fun part of the project, as it allows the device to have a personality, and has the potential to be an effective selling point to a customer. Most interfaces today follow a simplistic approach, using neutral colors, and following a defined color scheme. Even though this does not seem to have too much of an importance on the project, it can really elevate the user’s experience and make it more enjoyable to use. Similar to the construction of the interface, our finalized style was determined late into the project, and inspiration was gathered from devices such as the Roadie3, smartphones, and other small handheld appliances with screens on them.

Creating this seamless and modern looking display was a challenge as we only have three buttons for the user to interact with our device. These buttons will be two directional buttons with a selection button as the third. The interface will play to these directional buttons to create simplistic menu changes and user selections for choosing modes of string tuning and frequency selection. Knowing the button layout before creating the menu hierarchy will play a big role in creating the aesthetic ease-of-use display for our users and clients.

Creating a simple interface also helped us achieve our technical requirement for ease-of-use display and our timing requirement for tuning all the strings on a guitar. If the user can access the desired frequencies and menus to tune the strings to those frequencies, the faster our device will tune the user’s guitar. Our device will also allow the user to save past used frequencies for easier access, at a later iteration so this is more of a stretch goal. This will provide even faster access to string tuning as our device will be used mostly for retuning guitars back to the past frequency they were set at as the peg will naturally loosen over time. If we allow for more saved frequencies, then the user will have an easier time with tuning their guitar to multiple sets that they enjoy playing. Instead, what we implemented to have a quicker interface time we decided to create a pre-stored library of common tunings that are used.

The Adafruit graphics library is created for Arduino, and this is essential to our project as the display driver library is well programmed, and our group will have access to many high-level graphic drawing functions to create our simple and ease-of-use interface. Ultimately the GUI design was determined to be the final step in completion, and we took a simple approach that allowed us to display the necessary, and relevant, information on the screen during use. This modular approach allowed us to make changes on the fly and give us freedom for using and reusing display elements to achieve a user-friendly design.

21.4 Tuning Libraries (Stretch Goal)

One of the stretch goals that was implemented into our project is to create different tuning libraries, to be able to accommodate a wide range of guitar players of varying skill levels. The purpose of this is to be able to store a preferred tuning of an electric guitar on the Pick Pocket Tuner’s internal memory so that the user will to be able to change from standard tuning (EADGBe) to an alternate tuning, for example DADGAD, with ease. Additionally, this feature would allow the user to store a preferred tuning for use with different guitars, or even different use cases like performances, recording, or just practicing. In order to achieve this, we stored on the device memory a pre-set library of 12 tunings that would be able to be accessed by the end user whenever they needed or felt like changing the tuning on their guitar.

21.5 Tuning Other Instruments (Stretch Goal)

An additional stretch goal that we would like to implement in the future, is to have the capability to tune different stringed instruments, including but not limited to banjos, Ukuleles, Cello, Violin, etc. The software needed for these other instruments would change slightly since different frequencies would be necessary to be read due to the pitch changes based on the size/length of the instrument, and the torque output to the tuning pegs would also need to change based on the type of strings used. What this means, for example, is that tuning a string to a G note on a guitar requires “x” ft. Lbs. of torque, meanwhile tuning a ukulele to that same G would require “y” ft. Lbs. of torque.

Ultimately, during testing we found that our device’s software was able to analyze the input of a ukulele so adding this instrument, for example, would entail updating the libraries to accommodate for a 4 stringed instrument and the corresponding notes of the device in essence not changing the core components of our software, but instead adding/changing profiles to correspond with the number of strings and what the frequencies expected are.

21.6 Programming Language(s)

The programming language(s) to be used and implemented are still in a research/to-be-determined stage due to still picking out the necessary parts, and testing for the parts as they are acquired. As it currently stands, the Piezoelectric Sensor (a.k.a. PES) and most other parts would be able to be utilized and interfaced via the Arduino environment using the Arduino IDE.

However, as further development and investigations occurred, the members of Group 42 were able to make informed decisions on the software development process as needed. These decisions, impacting what languages or methods to use for software implementation, are sometimes simple determinations made for the group by the algorithm that we wish to employ itself, as is the case with the fast Fourier transform.

Additionally, since we know we are utilizing the Arduino IDE for prototyping, we know that both the Arduino-based framework language will be used, as well as C/C++, especially since the Arduino-based framework language is directly based off of C/C++. As stated previously, we are not limiting ourselves to any one set of languages, and during the early stages of development primarily, during the integration of all the different components with the ESP8266 and other functions, this led to changes in our choices. We were initially set in utilizing Python for most of our project, however we found that there were issues using Python and the display we selected, amongst others, and so we opted to utilize C/C++ as our final programming language for our project.

21.7 Programming/Software Libraries

During the prototyping stages of our project design is where we ultimately decided on what to specifically use, however it is known that there are various libraries in existence already that would not only aid in facilitating the mathematical processes for analyzing the frequencies but would aid in the integration and interfacing of the hardware that we are to use.

Additionally, we are utilizing some Arduino libraries in conjunction with the listed C/C++ libraries to accomplish some of the fine control that is necessary for the motor, and in order to interface with the display correctly.

These libraries in the Arduino/C/C++ suite are Stepper.h, math.h, Adafruit_GFX.h, Adafruit_ST7789.h, Adafruit_ST7736.h, and SPI.h, and other open-source libraries that are available for use. The one other library/software packages that we thought about using were the MicroPython or CPython, where these two libraries are optimized for operating on a microcontroller as they are lightweight and quick in program execution.

However, as mentioned in the Programming Language(s) section, we found that using Python was not possible when considering the necessity for our display to work as intended. Ultimately since we decided on using the Arduino/C/C++ libraries available, we made the decision of implementing a modular approach to our GUI and created custom functions that would handle the formulas and calculations necessary to meet the tuning parameters described.

Figure 2. Software flow of system

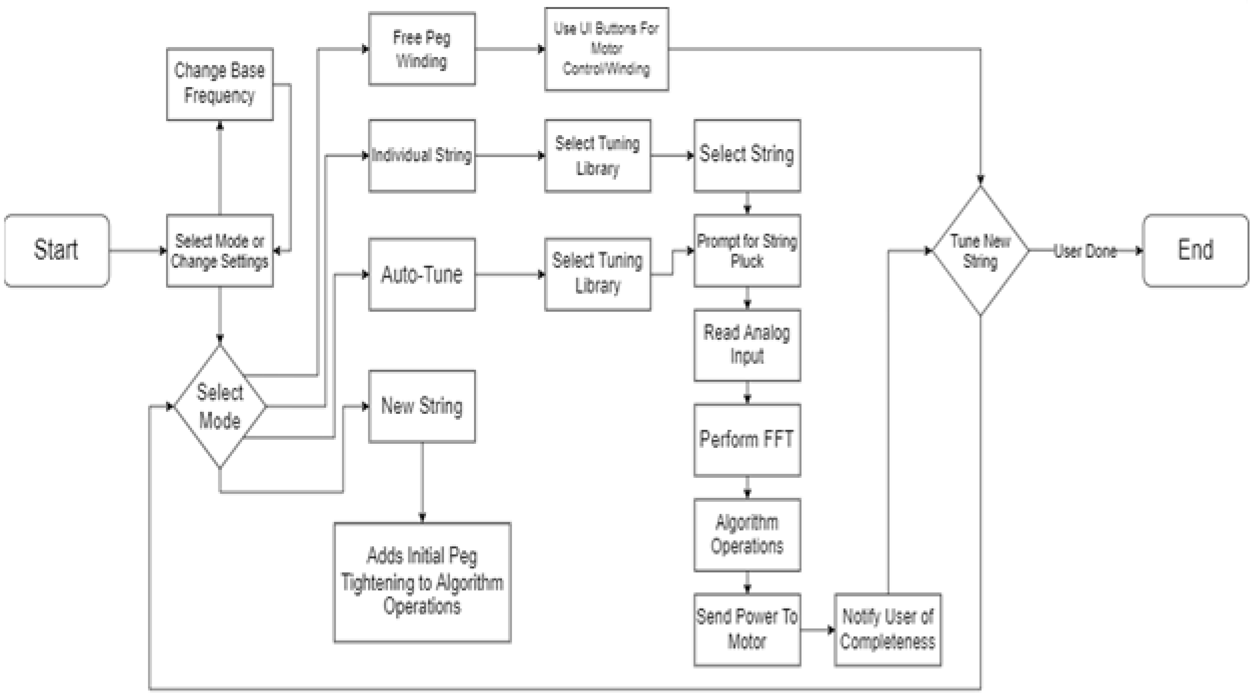


Figure 41, UI/UX Software Hierarchy/Design

22.0 UI/UX Software Hierarchy and Design

22.1 Software Hierarchy Description

In Figure 41 above, we have decided to visually mark how the overall UI/UX Software Hierarchy is loosely intended to work. The intended method of interfacing between the Pick Pocket Tuner and the end-user is dependent on the utilization of buttons that are on the device itself and the software necessary to receive the inputs from said buttons. Through all this, the device displays all the information back through a GUI that is designed by the members of Group 42 in an attempt to relay the most information in an accessible and readable manner. The LCD display software hierarchy/design will be discussed in a different section; however, it is relevant to mention at this juncture since almost all of the feedback of the tuner and interfacing depends on the performance and implementation of the LCD display. Additionally, it is worth mentioning that the assumption for user inputs to the device are performed strictly through the buttons on the device as there is no other input required of the user. At the moment, this implementation allows for further refining and addition of other elements to facilitate in providing a positive, intuitive user interface. After testing and integration, we have achieved providing a usable and easy-to-navigate user interface for our device.

In discussing the specific flow this diagram, we have determined this pathing as it is relatively self-explanatory. Once the device boots up, at the Start block, the software package will automatically load the method that would enable the automatic tuning to occur in order to tune the guitar to E Standard. Then, the UI will prompt the user for input to tune the correct string and thus proceed through the tuning process. Next, the device contains a switch that allows the user to choose whether they want to use the 1/4" input or the vibration sensor. Through a combination of buttons, they also have to option to navigate between Free Mode, New String, or Regular (a.k.a. Automatic) tuning. Once the mode of operation is chosen, the display will proceed through the necessary software steps to accomplish the tuning process, and loop back to select the string, in other words move the Pick Pocket Tuner, to the next string to tune, or if the user is done with all the strings, to the end of the process.

22.2 Software Tuning

The process of tuning a string on the software side requires communication between the piezo sensor (or the ¼" input), the display, and the ESP8266. When the user is prompted to pluck the string, the software will read in the input analog voltage from the piezo sensor over a determined duration. The duration of the analog reading is currently 0.5 seconds, s this was the most accurate option. This also is fast enough to meet both of our technical requirements of accuracy within 4 cents and a short time duration to tune all strings on the guitar in 3 minutes.

For electric guitars, the ¼" input jack will be used to bypass the piezo sensor due to the guitar's internal circuit. Reading the string frequency from the guitar's circuit will result in a more accurate reading than if the string pluck was read by the piezo sensor. However, the ¼" input jack will send its analog voltage into the same ESP8266 pin resulting in the same steps with tuning an electric guitar. This allows our product to have multiple ways of tuning and allow multiple instruments to be tuned without multiple steps of code for the differing steps.

To test the frequency and accuracy of the fast Fourier transform, we shall connect the piezo sensors to the NodeMCU and created an Arduino program using the arduinoFFT function that is already built into the IDE to read in the input. We shall perform the transform on the input sensor values and graph the resulting frequency value for accuracy.

This input frequency value is then compared to the frequency chosen by the user, which is the desired string frequency. This comparison will result in a difference of frequency values that will be used to calculate the amount of voltage needed to send to the motor which will then turn the peg winder. If the voltage sent to the motor was correct to tune the string to the desired frequency, the buzzer will have voltage sent to it to provide the user feedback. The amount of voltage needed to send to the motor will be tested as it is related to the torque of the winded peg and to the change in frequency onto the tensioned string.

22.3 Software Display

During the tuning process, the display shows the menu options for the user to traverse through with directional buttons. The menu options will be displayed in a hierarchy that can be selected with the selection button. Backwards menu traversing will be made possible with the user using the left most button to travel back to the previous menu screen.

Reusing buttons for other actions while traversing through the menu allows our product to have a simpler design that is not too complicated for the user to understand and use. These will both help result in a faster tuning time and an easier product for all users to use.

After the user has selected their string and desired frequency, the display will then prompt the user to pluck the string. Further iterations of this prompt are desired to be more aesthetically pleasing, such as an image or an animated image. However, after the string is plucked, the display will then show the read frequency, the string being tuned, and the desired frequency. This text will be shown alongside a graphic that shows the range of the tuned string within its sharp/flat frequency.

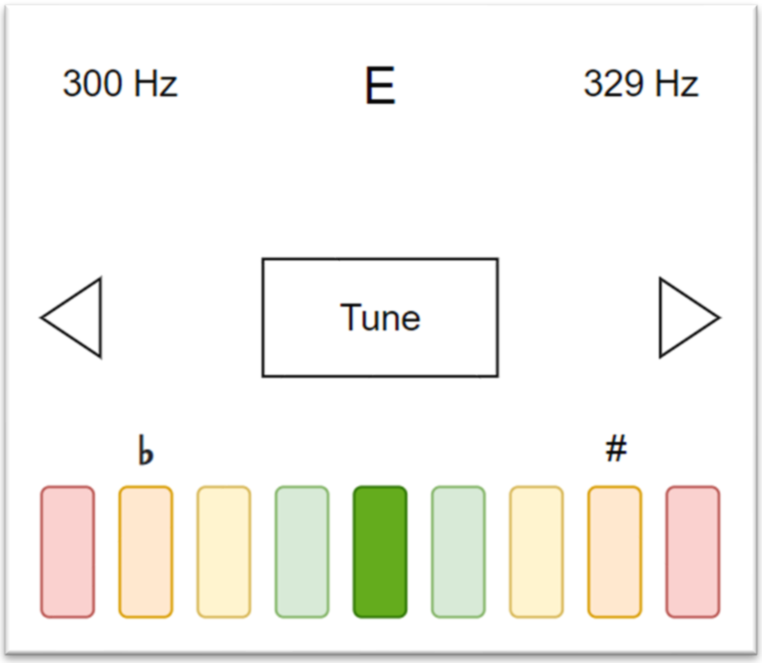


Figure 42, TFT display GUI Layout During Tuning

Having the diagram display this graphic shows the user the whole tuning process with a simplistic display with the green bar being the tuned frequency. With the middle rectangle being the correct and in-pitch frequency. The diagram also displays the flat and sharp frequency placements for comparison. These being on the display also allows for users to have more knowledge in their string frequencies if they are tuning to certain flat and sharp notes compared to the natural frequency.

Using this diagram as opposed to simply displaying the frequency values in text form will result in a more aesthetically pleasing display to the user. This diagram will also work as more interactive feedback, as frequency values are not as well-known while flats and sharps are more relatable information when tuning certain instruments.

This will also show the accuracy of our product’s tuning more graphically with the midpoint lines shown within the arc of the flat, natural, and sharp frequencies. Since a big technical and customer requirement is accuracy, this diagram hits all the points of a simplistic, yet effective, design.

For more aesthetic appeal and to have more unique designs, a stretch goal for the display would have graphics during user downtime. This would include the device powering on/off and during the string frequency reading processes. Having these graphics would help bring more appeal to our product as opposed to our competitors and would allow for graphical uniqueness on the market for users to pick from.

Throughout the process of tuning strings and navigating the menu, another stretch goal is for the display will be displaying battery indications. This would preferably be in the top right corner as users are more familiar with that location for any battery indications for handheld devices. This indication will also be useful for users to know if they will need to charge the device before planning any events that our product will be needed for.

This can be implemented with our current recharging circuit as it has two data pads (D+ and D-). We intend to use these pins to communicate the battery percentage/status to the display and further provide the user with knowledge of the device and how much longer it is needing charging.

22.4 Handling Flat and Sharp Notes

The purpose of this section is to define and identify a potential issue with how different notes are interpreted by the system in order to display the correct feedback to the user based on the specific pitch of the string that is actively being tuned and the desired pitch that the user wishes. For simplicity, Figure 27 below is what will be used to reference the information to be described here. The content of this section will be explained as if it were a keyboard, however, since this information is based on music theory the same information applies universally across instruments.

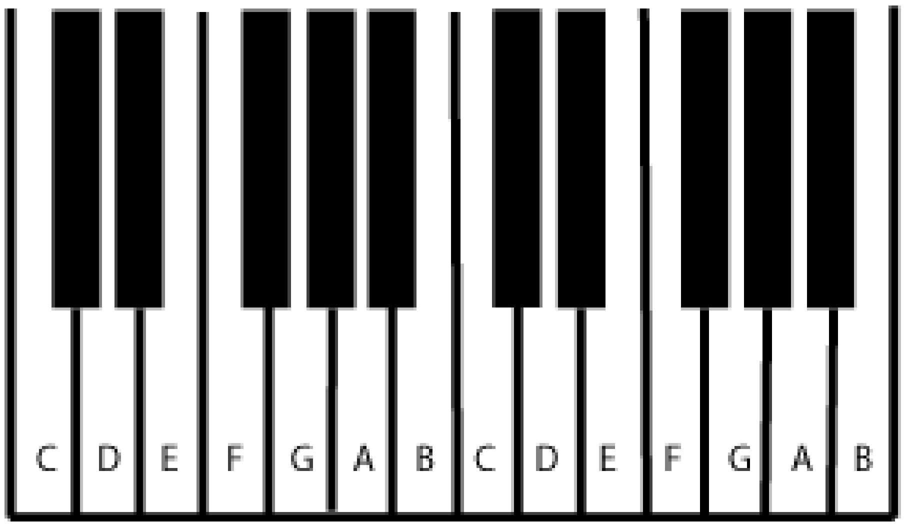


Figure 43, Natural Notes Written on the Keys of a Keyboard

As can be seen above in Figure 27, we can see that the white keys all repeat periodically, from the letters A through G. These seven letters name all the natural notes in music. The black keys in this instance are the remaining 5 notes that are in an octave, these black keys are referred to as sharp or flat corresponding to either the preceding or following letter. When describing a sharp note, like A#, it means that the note A# is one half step higher than the natural note A, and when the note is flat, like Ab, it means that the note Ab is one half step lower than the natural note A. One distinction to be made, which can be seen in the figure above, is that not all natural notes have intermediate steps to go through, like between natural notes E and F, and between B and C, so in this instance when referring to E sharp, E#, it would mean the same as if one were referring to the natural note F since there is only one-half step between the notes E and F, likewise with notes B and C.

Flat Note - Frequency	**Natural Note - Frequency**	**Sharp Note - Frequency**
Cb – 246.94 Hz	C – 261.63 Hz	C# - 277.18 Hz
Db – 277.18 Hz	D – 293.66 Hz	D# - 311.13 Hz
Eb – 311.13 Hz	E – 329.63 Hz	E# - 349.23 Hz
Fb – 329.63 Hz	F – 349.23 Hz	F# - 369.99 Hz
Gb – 369.99 Hz	G – 392.00 Hz	G# - 415.30 Hz
Ab – 415.30 Hz	A – 440.00 Hz	A# - 466.16 Hz
Bb – 466.16 Hz	B – 493.88 Hz	B# - 523.25 Hz

Table 11, Flat/Natural/Sharp Frequency Example starting at Middle C

With that brief music theory out of the way, the idea behind how this is being utilized is when specifically determining what tuning a string is in, it is necessary to be able to distinguish whether the tuning is natural, sharp, or flat, and thus be able to relay that information to the user so that it can be used. Since each note has a specific frequency, we can determine an offset for the device to display back to the user whether the note will be flat, natural, or sharp. Typically, a difference in 10-15Hz, either higher or lower, would cause the note to become flat or sharp, as shown in found. Above, when considering starting at what is referred to as ‘Middle C’.

From this table it is clearly shown that the range for where a note is determined to be flat or sharp varies. So, between each half step, it would be necessary to find a range so that when it crosses that 10-15 Hz threshold the information relayed to the user would display the note as being sharp, natural, or flat. Similarly, to what was previously stated, this threshold has been adjusted and tested to fully determine what the most appropriate ‘stepping’ for the frequencies to go through is and to determine the pitch accurately.

22.5 Motor communication with FFT

The determining factor for how we shall tune our guitar will be the implementation of the Fast Fourier transform with the DC motor. The Transform will be constantly picking up frequency signals from the vibration of the strings, and the motor must adjust according to if the string is flat or sharp. To make this work together, our team came up with a code that relates cents to the amount of time the motor would spin for.

The code would read how many cents from the desired frequency the string was and react with turning the motor in a direction that would get the string closer. The software has a tolerance for different cent ranges such as turn X long if string is X flat or sharp. Through testing we found that the motor turned the guitar peg faster when unwinding he string and slightly slower when winding the string. Therefore, we made an adjustment in the code to turn the motor less if the guitar was sharp so that the cent change would be so drastic. This feature allowed us to get the guitar to tune faster and more accurately at the same time.

22.6 New String Mode

This mode of operation, at first, would be limited to just guitar and bass, however, could be expanded to include additional stringed instruments determined by testing and further refining of the software. Since we have identified the rudimentary frequency threshold to determine whether the string is flat, sharp, or natural one of the first methods for utilization and improving the UI/UX with the Pick Pocket Tuner is our implementation of a New String mode.

The intended method of accomplishing this gives us the capabilities of stringing a guitar faster than a human, and faster than the competitors already in the market. The manner in which the software method would accomplish this would take into account octaves on a guitar/bass, and music theory.

As seen in Figure 27, each note repeats, which means that when the note interval repeats are that it’s changing octave. This means that each frequency also repeats by a specific factor, the distinguishing feature between a higher/lower pitch is whether or not the frequency of that same note is doubled or halved.

For example, if a C note at 65.4Hz, one octave higher means the frequency doubles to 130.8Hz, and one octave lower means that the frequency halves to 32.7Hz. Table X below shows this for other notes to further exemplify this. When each octave moves from one to the next, the pitch is what changes, and since the pitch determines the sound, we hear as either being high or low, it corresponds to the frequency.

Octave Lower - Frequency	Note - Frequency	Octave Higher - Frequency
A – 220.00 Hz	A – 440.00 Hz	A – 880.00 Hz
B – 246.94 Hz	B – 493.88 Hz	B – 987.77 Hz
C – 261.63 Hz	C – 523.25 Hz	C – 1046.50 Hz
D – 293.66 Hz	D – 587.33 Hz	D – 1174.66 Hz
E – 329.63 Hz	E – 659.25 Hz	E – 1318.51 Hz
F – 349.23 Hz	F – 698.46 Hz	F – 1396.91 Hz
G – 392.00 Hz	G – 783.99 Hz	G – 1567.98 Hz

Table 12, Frequency Examples of One Higher and Lower Octave For Each Frequency

We can use this information to set an arbitrary 'new string mode' note to achieve with the correct octave, to not over tighten the string causing the string to snap, and conversely to not leave the string so loose so that the vibrations won't be picked up by the tuner due to low frequencies.

When in the "new string mode" we are to guide the tuner to wind a string to one or two steps below the E standard tuning, during the initial winding. What this looks like is that if the string is meant to be tuned to G, this mode of operation would wind the string to E and then ask the user to move to the next string, the octave and thus frequency which is tuned to would be determined by which string is being tuned.

When doing this, it allows the guitar body and other strings to ease into the tension that is required to tune it to E Standard and help in mitigating damage. While undergoing this process, the UI/UX would prompt the user to either change strings or to continue plucking the current string until the preset tuning is achieved. The following table visually demonstrates the tunings which would be produced given the one or two steps below E Standard.

E Standard Tuning	1 Step Lower (D Standard)	2 Steps Lower (C Standard)
Low E	Low D	Low C
A	G	F
D	C	A#
G	F	D#
B	A	G

High E	High D	C

Table 13, Demonstrating One and Two Step Offset Tuning from E Standard Tuning

All of this winding is a response to the strumming that generates the soundwaves that are then read and analyzed by the rest of the system and the motor responds in a way to turn the tuning peg. All of this, would be used to facilitate the tuning mode the relayed information to the user would be displayed through the LCD screen so that the user will be able to know when to strum or switch to the next string. Additional feedback could be implemented by adding a buzzer and/or an LED that could also be used to alert the user that the tuning of that string is complete.

22.7 Autotune Mode

The purpose of this section is to demonstrate the way we intend to implement the autotune mode for a positive, intuitive UI/UX experience. At the time of writing this document, we intend to make the Autotune mode the initial mode of operation that the Pick Pocket Tuner would load into after booting up. The idea behind this mode is that the user can just turn on the device and tune to whatever the closest string is at, be able to use the on-device buttons to tune to a preset tuning chosen from the library of tunings available, or to choose from a user-defined preset tuning that is recalled from the device memory.

When using the autotune mode without the presets, the user would utilize the tuner, assuming that the string isn’t too far from whatever the intended string is supposed to be tuned to, by just placing the tuning peg in the peg winder of the Pick Pocket Tuner, and be prompted by the LCD screen to strum so that the device can begin tuning the guitar, as is shown when following the flow chart in Figure X above. Once complete, as with all other modes, the user can either move on to the next string needing to be tuned, or end the tuning process altogether.

The second sub-mode of operation, would entail having accomplished our stretch goal of including/implementing a tuning library that could store both preset tunings, and/or the storage of user-defined tunings for their guitar, or other stringed instruments. What this sub-mode of operation would entail is adding an additional step. This step would be where the user, through interacting with the buttons on the housing of our Pick Pocket Tuner, would lead them to select a tuning to tune to or create a new preset that would be stored based on the tuning that they tell the device what tuning they want as they move through all the strings.

If choosing from the preset tunings, all the device would do is load the tuning and prompt the user to start from the lowest string, and pluck the string to begin reading/analyzing the frequency, inform the user through the LCD screen and/or a buzzer/LED that the tuning of that string is complete, and then prompt to move on to the next string. If there is no next string then the user can simply be done with tuning.

If the user would like to create a new tuning preset for them to use, they would instead select the necessary prompt so that they can begin setting the string as necessary. They can then start to proceed through selecting the note they want the guitar tuned to, tune the guitar, then once tuned save that string in that preset, and move on to the next, repeating the process until all strings are saved/completely tuned.

To recall this user-defined tuning, all it would be is choosing it from the library of tunings as if it were a preset tuning from the library of available tunings. As with the other tuning methods, the feedback for creating a new user-defined tuning would be through the LCD display where the necessary prompts would display to the user instructing them what to do, but it would also give the user the feedback from a buzzer and/or an LED relaying that the step they are currently on is complete, until no other strings remain, or the user is done.

22.8 Free Move Mode

If the user selects the free move mode, the directional UI buttons are used to turn the peg winder both clockwise and counterclockwise. For example, the “up” directional button would turn the peg counterclockwise, tightening the string, and the opposite for the “down” directional button, by loosening the string. This mode is also very helpful when the guitar player wants to unwind the strings or wind them when they are changing them. It loosens and tightens the strings very quickly, which makes changing the strings more desirable and take less time.

23.0 Software Summary

Software was implemented to our project with the ESP8266 microprocessor and with C++ as our programming language. Using the Arduino IDE programming environment, C++ was utilized to easily write the software onto the processor. This software setup was chosen as opposed to the Micropython environment and their programming language Python. Arduino’s environment was ultimately used because of the very easy to implement Fast Fourier Transform. Since this transform was an important aspect of the project, C++ was implemented and tested on.

The software for the Pick Pocket tuner consists of hierarchal menus to guide the user to tuning the correct string at their desired frequency. This menu is displayed to a TFT LCD screen via a SPI connection. The input vibrational sensor sends its frequency reading to the processor via an analog voltage. The software will read this value for a specified polling time and the resulting voltage wave will be transformed into the frequency domain. After the algorithm decides the amount of voltage to be sent to the motor/peg winder, the software will update the user onto the LCD screen visually and audibly with a voltage sent to a buzzer located on the housing. The software will then repeat with tuning the remaining strings until the user is done.

24.0 Project Budget Estimates

For the project budget estimations, we researched most of the components we intend to use for this project. The final table lists roughly how much each component will cost, and the quantity used for the final product. We expect to keep the budget of our final prototype below $100, but the overall budget of the project will be greater because of early prototypes and testing materials. The table below shows the optimistic cost for the final prototype. This budget will likely be subject to change, especially since we do not have the full design down yet, however we are hopeful about keeping the cost to a minimum.

24.1 Final Prototype Cost (Estimated Range)

Components	Quantity	Costs
Battery	1	$12
Charging Circuit	1	$10
Motor	1	$20
Display screen	1	$20
Regulator	2	$5 (total)
Vibration sensor	2	$2(total)
On/off switch	1	$1
buttons/inputs	3	$0.50 (each)
Peg winder head	1	$1
Housing	1	$1
Micro controller	1	$2
PCB	1	$2
Shipping Costs	1	$20
Total	15	$96.50

Table 14, Estimated Final Cost for Device

Some of the budget estimations may vary due to some supply chain issues that have arisen. Most parts just not being in supply or have too long of expected restock dates that are too far out where some range over a year and other alternative methods for items are required. For one instance, the voltage regulators are out of stock on most suppliers and even third-party suppliers and the back order are multiple weeks out if not even more than a year out of stock. Most of the other parts are attained from Amazon as the main supplier for most parts except specific ones such as the motor and semiconductor devices.

24.2 Project Development Costs

Along with other project requirements there are the development costs that came in and are larger in costs than the actual prototype. These costs mostly came from buying testing equipment to make sure the specific components work properly to fit the design specifications. These costs varied largely as the project proceeds in the right direction. The standard development estimate range is around $500. Another part of the development costs is time to learn how the parts work and the time it takes to research the right parts that are need for the job.

24.3 Development Cost (Estimated Range)

Components	Quantity	Costs
Snark ST 8 Clip-on Super-Tight Chromatic Tuner	1	$14
ADXL345 Digital Accelerometer	1	$17.50
Nema 16 Stepper Motor	1	$12
Nema 17 Stepper Motor	1	$15
3D print material 1kg	1	$20
Node MCU ESP8266	1 (3 pack)	$15
SN754410NE	5	$3
L298N Motor Driver	1 (4 pack)	$10
Piezo Vibration Sensor	1	$4.95
Piezo Element	4	$1.50
28byj 5V DC Stepper Motor with Drivers	1 (5 pack	$12
12V DC Brushless Stepper Motor	1	$20
Stepper Driver	1	$35
Jowoom	1	$34
Guitar Strings	1 (3 pack)	$17
Electric Guitar Strings	1 (3 pack)	$15
Lithium-ion Polymer battery	1	$12
Boost up Converter 2V-24V	1 (10 pack)	$11
Adafruit 4410 Micro LiPo charger USB type C	1	$9
Rocker on /off switch	1 (8pack)	$9
Tactile Momentary Push Buttons	1 (25 pack)	$8
¼" Output Jack	1	$5
Total Estimated Development Cost Range:	29	$316.45

Table 15, Estimated Cost for Development of Project

Most of the research costs are going to go towards the vibration sensor which is a key component that is needed for the device to work the best. There are multiple vibration sensors that need to be tested and only on will be chosen which meets the best requirements for the overall system. Another big research cost is going to be getting ahold of the parts themselves due to production delays or costs from manufacturers themselves. Another development cost will be finding the torque needed to turn the pegs of multiple string instruments.

Some instruments mostly being the orchestra styled instruments being the violin or cello, the tuning pegs work slightly different where the pegs must be pushed inwards and turned at the same time to properly be adjusted and tuned. Whereas most of the basic stringed instruments can simply be turned at the peg heads. Now not all the stringed instruments will be tested or can even be tested for since some will not be compliant, for example the harp is a stringed instrument and is somewhat well known, but it does not have peg heads to tune them, and it would take a long time to tune all the strings so it's not a logical choice. Other instruments that will not be tested are instrument that are more oriental or have oblong peg heads that are meant to be tuned by hand or with a very special tool. As many things that have a basic or friendly peg head design and is a well-known instrument will be trialed and tested. The music department at UCF has agreed to lend us any aid in testing stringed instruments through UCF's music program's students and supply any" Subjects," we may need, which will save costs of having to go out and rent instruments.

There have also been a few issues with the development costs that are big things that should try to be avoided but arise in any case. Most of the issues are buying things then realizing they are wrong or not going to end up working for the project. Most of these issues came from trying to pick already end version types of products than testing purpose types of products. Most of them came down to the motor and battery type for what was going to be used.

The next part was finding the components that were mostly going to be compatible to work along-side with the software. There were some issues at first when deciding on the motor which was planned for testing a stepper motor that would meet the torque requirement at first. The first motor that we ended up getting was a Nema 17 stepper motor that did over the amount of torque that was needed to potentially help reach the stretched goals right away. There was one main issue with it being a little too big for the design and heavy also. Then there was the decision to step down from the Nema 17 to a Nema 16 which was about half the size and half the torque but still had enough torque for what was desired.

There was one good thing that came out next and that was buying a Jowoom and seeing the internals of what they used and how they implemented everything together. Getting the Jowoom also saved a substantial amount of spending on the development costs from saving the group of buying a Roadie 3 which would have costed $129 plus shipping off Roadies website, but instead we were able to find a Jowoom on eBay for only $32. That alone saved the spending costs of $97 for the development costs section. Buying and taking apart the Jowoom gave a lot of insight to finalize the decision of the motor for the final design. Besides accidentally buying a few extraneous parts the budget has mainly been on a good track of getting most of the other proper parts.

25.0 Project Milestones

Project Task	Assignee	Date Started	Date Due	Status
Decide project idea	All	08/27/2021	09/17/2021	Completed
Identify Parts	All	08/27/2021	10/01/2021	Completed

Final Report Subtasks				
D&C v1.0	All	09/07/2021	09/17/2021	Completed
D&C v2.0	All	09/17/2021	10/01/2017	Completed
60-page Draft	All	10/01/2021	11/05/2021	Completed
100-page Draft	All	11/05/2021	11/19/2021	Completed
Final Document	All	11/19/2021	12/07/2021	TBA
Final Design Parts				
Motor/Motor Driver	Lucas	09/10/2021	10/01/2021	In Progress
Vibration Sensor	Luis	09/10/2021	10/01/2021	Completed
Display	Jamie	09/10/2021	10/01/2021	Completed
System Controller	Luis	09/10/2021	10/01/2021	Completed
Voltage Regulator	Paul	09/10/2021	10/01/2021	Completed
Input Buttons	Jamie	09/10/2021	10/01/2021	Completed
Switch	Lucas	09/10/2021	10/01/2021	Completed
Battery	Paul	09/10/2021	10/01/2021	Completed
Housing	Lucas	09/10/2021	10/01/2021	Completed
PCB	Paul	09/10/2021	10/01/2021	Completed
End of Project Design Goals				
Acquisition of Parts				Completed
Final PCB Design				Completed
Working Prototype				Completed

Table 16, Project Milestones Chart

26.0 Hardware & Software Requirement Specifications Tables

This section is to be utilized as a simple-to-navigate portion of the present documentation, in conjunction with the following section, which relates to the final testing plan that will determine a successful and/or failed implementation of the corresponding Requirement Specifications.

26.1 Hardware Requirement Specifications

No: 26.1.1
Statement: The Pick Pocket Tuner shall tune the entire guitar within a maximum amount of time.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: 26.1.3
Conflicts: None

Supporting Materials: In-Market Competition
Evaluation Method: The User will be able to tune a guitar in 5 minutes, ±1 minute
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.1.2
Statement: The Pick Pocket Tuner shall tune a new, individual, string within a maximum amount of time.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: 26.1.3
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The User will be able to restring a string and tune it in 3 minutes, ±1 minute.
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.1.3
Statement: The Pick Pocket Tuner shall tune the strings to the correct frequency for that string.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: None
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The User will be able to tune the string of the instrument (guitar) accurately, within ±5 cents of accuracy
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.1.4
Statement: The Pick Pocket Tuner shall be able to tune more than 100 strings.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: 26.1.3
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The User will be able to tune more than 100 strings before needing to recharge the device
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.1.5
Statement: The Pick Pocket Tuner shall weigh less than 3 pounds.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: None
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The user will not need much strength to use the device.
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.1.6
Statement: The Pick Pocket Tuner shall have the ability to tune to alternative tunings.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: 26.1.3
Conflicts: None
Supporting Materials: In-Market Competition

Evaluation Method: The User will be able to tune to tunings such as Drop-D, Drop-C, etc.
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.1.7
Statement: The Pick Pocket Tuner shall have an easy-to-use physical User Interface.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: None
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The User will be able to easily reach and use all the physical components of the device
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.1.8
Statement: The Pick Pocket Tuner shall be ergonomic, and small enough to fit in one hand.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: 26.1.7., 26.1.5
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The User will be able to hold and use the device comfortably with one hand
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.1.9
Statement: The Pick Pocket Tuner shall have an easy-to-read display.

Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: None
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The User will be able to read the screen in most light conditions
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.1.10
Statement: The Pick Pocket Tuner shall be able to use a ¼ -inch input jack with a ¼ -inch instrument cable.
Source: Team Member Discussions
Dependency: 26.1.3
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The User will be able to use the jack/cable combo to tune the guitar as a back-up method of tuning.
Revision History: Luis Vargas, 11/29/2021, Created Table

26.2 Software Requirement Specifications

No: 26.2.1
Statement: The Pick Pocket Tuner shall have an easy-to-understand software menu structure
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: 26.1.3
Conflicts: None
Supporting Materials: In-Market Competition

Evaluation Method: The User will be able to easily choose the desired mode of operation
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.2.2
Statement: The Pick Pocket Tuner shall be able to choose between four available modes of operation.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: 26.2.1
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The User will be able to choose how they want to utilize the device to tune.
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.2.3
Statement: The Pick Pocket Tuner shall display the current tuning to the display.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: 26.1.3
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The LCD Display will read the current tuning and the user will be able to determine if the tuning is correct or not
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.2.4

Statement: The Pick Pocket Tuner shall be able to distinguish between flat and sharp notes
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: 26.1.3
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The device will be able to display sharp and flat notes to the display for the user to see
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.2.5
Statement: The Pick Pocket Tuner shall be able to differentiate between octaves of the same note.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: 26.1.3
Conflicts: None
Supporting Materials: In-Market Competition
Evaluation Method: The device will be able to tune a string at an octave-length-difference based on frequency
Revision History: Luis Vargas, 11/29/2021, Created Table

No: 26.2.6
Statement: The Pick Pocket Tuner shall report accurate notes and frequency values.
Source: Team Member Discussions, In-Market Competitors, Technical Research
Dependency: 26.1.3, 26.2.3, 26.2.4, 26.2.5
Conflicts: None
Supporting Materials: In-Market Competition

Evaluation Method: The device will move the motor according to the guitar's tuning frequency read
Revision History: Luis Vargas, 11/29/2021, Created Table

27.0 Final Testing Plan

27.1 Objective for Final Testing

The Objective for the hardware and software testing is a simple, and straightforward one, which is to ensure that the final device outlined in this documentation meets the requirements, specifications, and successfully executes/implements the necessary combinations of hardware and software to accomplish the outlined goal: tune a guitar/bass automatically. The assumption for this Test Plan, for all intents and purposes, is that in the final version, the device will accomplish the goal within a reasonable "error-free" experience.

27.2 Description of Final Testing Environment

For the Pick Pocket Tuner, the test environment will realistically be physically anywhere that the group will be able to use a guitar/bass. Ideally, since the device is meant to be hand-held and portable, we can implement our testing procedures in various spaces, and settings, including but not limited to a garage, a recording studio, an auditorium, to an empty lecture hall on the UCF campus. The benefit of this is that the testing parameters aren't all necessarily limited by environmental factors, unless specifically and discretely noted. On both the hardware and software sides, the testing parameters outlined throughout this documentation details how the intended use of each component is, and in this section specifically it is assumed that the previously mentioned parameters will be included in further testing.

For Hardware the Testing Environment is more closely related to the physical environment, in which case, for optimal results, the group will be testing the device in the same location, however using varying guitars with different tuning pegs. Ideally, we shall be reaching out to the UCF College of Music to inquire about potentially using one of their practice rooms for testing since they are built for the separation of sound between those being emitted from the inside of the practice room versus those originating from the surrounding hallways. The guitars that are to be initially used for testing will be provided by Lucas Grayford, Paul Grayford, and Luis Vargas.

Lucas' guitar was manufactured by Behringer, model METALIEN. The METALIEN is an electric guitar with an unknown manufacturer of tuners, and as an electric guitar it has a ¼" jack for a ¼" instrument cable to be used with it. Additionally, this guitar also has a floating bridge, which is also non-branded. Since the bridge is floating, we are going to be "blocking" the floating bridge essentially turning it into a standardized "traditional" static bridge. Paul's guitar was manufactured by the Indiana Guitar Company, model Scout Black Acoustic. The Scout is an entirely acoustic guitar, which means that there are no electronic components with which to interact with, and has also non-branded tuners. The bridge on this guitar, since it is acoustic, is a standardized "traditional" non-floating bridge and as such, there are no necessary alterations needed to be made. Luis' guitar was manufactured by the ESP guitar company, model is the LTD-DJ600 and for tuners it uses Sperzel Locking Tuners. The DJ600 is an electric guitar, which means it has a ¼" jack for a ¼" instrument cable, and it also has a Floyd Rose Tremolo Floating Bridge, this typically indicates that the tuning process is significantly more complex as a traditional tuner, however, for Luis' guitar in particular, this Bridge has been blocked off, and it does not "float" but instead operates in the same manner as a traditional standardized guitar bridge.

The brand and specific model of guitar strings to be used for testing is at this time still to be identified, investigated, and reviewed by the group members, due to availability, cost, and/or delivery constraints. For final verification of tuning accuracy, the tuning of the strings individually will be checked against the AXE I/O guitar interface. The AXE I/O interface houses a chromatic tuner, which through the electric guitar's ¼" output and an instrument ¼" cable, will let us confirm if the pitch that our final device adjusts the string to is correct or not. Additional tools for confirming hardware testing successes and/or failures include but are not limited to items such as oscilloscopes, voltmeters, stop-watch timers, and/or a power supply to be used, as needed for different testing steps. These devices will be implemented in an as needed basis, and shall be used as a 'safety net' for additional confirmation beyond that of what the Pick Pocket Tuner and the AXE I/O reports as the tuning of the guitar string.

For the software aspect of our Testing Environment, our intended testing environment is a straightforward one. The devices used to interface directly with the ESP8266 will be the extent with which the device will need to be 'ran' on. The compilation of the final software to be tested will be done on these machines, which will then upload onto the ESP8266 for the device to run on-device directly, with the available on-device memory. Information regarding the hardware of the ESP8266 can be referenced above in the section detailing the specifications of the ESP8266. The machines who will be used with interfacing with the ESP8266, shall use an operating system agnostic environment for compiling and uploading the code. This environment, which will also be used for testing is the most recent, up-to-date version of the Arduino IDE software. For both Windows (7 and newer), as well as for MacOS (10.10 or newer) it is Arduino IDE v1.8.16. Beyond these mentioned devices/interfaces, the software testing will be more-or-less done simultaneously with the hardware testing so initially the physical environment, as well as guitars, and hardware used will be the same as the hardware testing environment.

While the testing environments isn't meant to be a 'catch-all' for all situations, the reason for detailing this information is to layout a baseline from which to begin the determinations for the device to be marked as a successful product. Additionally, detailing these tools and environments will aid in reducing the number of extraneous variables that could introduce unneeded errors that can affect the results of our testing. As previously stated, at the time of writing this document, this is what we have currently deemed necessary to complete the final testing for the Pick Pocket Tuner. However, it is entirely within a sizeable possibility that between now and final delivery of the product, the information in this document could be outdated or have been altered to suit the needs of the Senior Design team.

27.3 Stopping Criteria

The stopping criteria for each test case will be dependent on a Pass/Fail system. This system will be used for test cases throughout the entire project, to determine how well the individual components will be integrated into the overall system, however they will be incredibly important when the final testing development phase has been reached. If the test case at hand results in a “Pass” rating, we shall then move on to the next test case, until a test case “Fail” rating is reached. If a “Fail” rating is reached, the failing test case will be documented, and the hardware/software feature and/or module dependency for that test case is determined. From there, the dependency is then analyzed as part of the system and independently to determine the cause of failure. Once said cause of failure is determined through said analysis, an attempt to address it shall begin immediately. If it can be addressed immediately, we shall return the affected to the developer/integrator to determine a solution. If it is determined that the fix cannot be done now, whether due to missing/yet-to-be-implemented features/modules, or as an example due to waiting for hardware to be acquired, the reasoning for this is also documented so that the team can continue with other hardware/software feature/module implementations.

In addition to this, if no errors are found, and integration/development is still underway, the team will re-assess the system and any currently unlisted/undetermined test cases will be constructed to attempt to filter out any false positive “error free” test runs. If all features/module implementations have been completed and the test case run returns with all test cases receiving a “Pass” rating, additional test cases will be constructed at that time to determine the reliability and robustness of the system. Once this is secondary, more extensive testing is completed, if the reports return that all test cases, this includes the original set and the new set, receive a “Pass” rating we shall then assess and determine if the product is ready to be considered as a deliverable item. However, if there is a “Fail” rating in these test cases, depending on the test case and cause for failure, we shall either work to find a fix or find a work-around as a resolution for the issue, if it is an issue that is determined to be a necessary issue to be fixed. Once no known errors are found after fixing all the “Fail” ratings, the device will be submitted for approval for consideration as a deliverable item.

27.4 Description of Individual Test Cases

As it was previously mentioned, this list is neither exhaustive, nor is it a comprehensive list of test cases. At the time of writing this document, we are still determining additional Test Cases to satisfy the Specification Requirements of the Pick Pocket Tuner, so for the time being, the test cases we are initially considering are listed in the tables below.

Test Case Number	1
Test Case Objective	Battery & System Test
Test Case Description	The user will flip the switch on the device providing power to the entire device and it will boot to our “beginning” screen
Test Case Conditions	See Testing Environment.
Expected Results	The device turns on successfully.

Test Case Number	2
Test Case Objective	User Interface Test
Test Case Description	The user will use the display and interface elements to navigate through the menus.
Test Case Conditions	See Test Environment.
Expected Results	The user will be able to move in and out of all the menu hierarchies.

Test Case Number	3
Test Case Objective	User Interface Test
Test Case Description	Using the UX/UI and display the user will be able to change the tuning method to utilize.
Test Case Conditions	See Test Environment.
Expected Results	The user will be able to change the mode of operation from one of those available to be selected.

Test Case Number	4
Test Case Objective	Tuning Test
Test Case Description	The user will be able to use the device and tune a string as the string is being plucked.
Test Case Conditions	See Test Environment.
Expected Results	The guitar will be in tune when the strings are strummed at once, or individually.

Test Case Number	5
Test Case Objective	Motor/Tuning Test

Test Case Description	The user will use the device to signal the device that there is a new string added.
Test Case Conditions	See Test Environment.
Expected Results	The new string will be re-strung to a yet-to-be-decided frequency. Left to be ready for “final tuning”

Test Case Number	6
Test Case Objective	Tuning Test
Test Case Description	The device will assist the user in tuning the guitar faster than the human would be able to do so manually.
Test Case Conditions	See Test Environment.
Expected Results	The time it takes to tune a single string will be under 30 seconds.

Test Case Number	7
Test Case Objective	Restring/Tuning Test
Test Case Description	While in the appropriate tuning mode, the user will be able to restring and tune a new string on the guitar faster than a human would be able to do so manually.
Test Case Conditions	See Test Environment.
Expected Results	The time for this process will be under 3 minutes.

Test Case Number	8
Test Case Objective	Motor/User Interface Test
Test Case Description	The user will be able to use the user interface to manually drive the motor
Test Case Conditions	See Test Environment.
Expected Results	The motor turns in one of two directions for as long as the corresponding button is held down.

Test Case Number	9
Test Case Objective	Tuning Test
Test Case Description	The device’s components correctly transfer the vibrational frequencies

	through the housing/components to the piezoelectric disc for analysis.
Test Case Conditions	See Test Environment.
Expected Results	The device can interpret a string vibrating at a given frequency, after the string is plucked.

Test Case Number	10
Test Case Objective	Tuning Test
Test Case Description	The user is able to use the interface to tune strings to different, i.e. alternate, tunings
Test Case Conditions	See Test Environment.
Expected Results	The tunings of the strings reflect the desired tuning from the user.

Test Case Number	11
Test Case Objective	Battery Test
Test Case Description	The battery life will be tested to determine how long the operational capacity will be.
Test Case Conditions	See Test Environment.
Expected Results	The device will be able to run all the components simultaneously and tune strings on a guitar.

Test Case Number	12
Test Case Objective	Weight Test
Test Case Description	The device will be weighed once all components are acquired.
Test Case Conditions	See Test Environment.
Expected Results	The device will be weighed with the housing to determine the overall weight of the device.

Test Case Number	13
Test Case Objective	User Interface Test
Test Case Description	The UI of the device will be determined how user-friendly and intuitive the controls are based on how long it takes to

	change the mode of operation until the device is ready to tune a string.
Test Case Conditions	See Test Environment.
Expected Results	The user will be able to change between menus/modes of operation and select the desired information in a easy-to-use manner.

Test Case Number	14
Test Case Objective	Tuning Test
Test Case Description	The device will be able to analyze the lower frequency from a bass guitar string and tune it.
Test Case Conditions	See Test Environment.
Expected Results	The bass guitar string will be in tune

Test Case Number	15
Test Case Objective	Ergonomics Test
Test Case Description	The device will take shape based on designs described as ergonomic and comfortable.
Test Case Conditions	See Test Environment.
Expected Results	Comfortable to hold and use

Test Case Number	16
Test Case Objective	User Interface Test
Test Case Description	The buttons on the device will be able to navigate in the desired direction and select the desired item.
Test Case Conditions	See Test Environment.
Expected Results	The user will be able to use buttons and the display to navigate and use the device

Test Case Number	17
Test Case Objective	User Interface Test
Test Case Description	The necessary, relevant, information will be displayed on the device's display.
Test Case Conditions	See Test Environment.

Expected Results	The user will be able to use visual feedback to make a decision if the string is in the tuning they want yet or not.

Test Case Number	18
Test Case Objective	Tuning Test
Test Case Description	The tuning process of the string doesn’t take a long time for each string. The device is able to handle the calculations needed, and provide the necessary signals to turn the motor appropriately
Test Case Conditions	See Test Environment.
Expected Results	The tuning is a lag free experience, and the tuning isn’t plagued with processing lag.

Test Case Number	19
Test Case Objective	Tuning/ESP8266 Test
Test Case Description	The device is able to store a predetermined tuning for ease-of-use
Test Case Conditions	See Test Environment.
Expected Results	The default tuning to tune to when the device boots up is E-Standard Tuning.

Test Case Number	20
Test Case Objective	Display/Tuning Test
Test Case Description	The device will display the correct tuning, and frequency, when note is flat.
Test Case Conditions	See Test Environment.
Expected Results	The display will inform the user that the string being plucked is X note but flat

Test Case Number	21
Test Case Objective	Display/Tuning Test
Test Case Description	The device will display the correct tuning, and frequency, when the note is sharp.
Test Case Conditions	See Test Environment.

Expected Results	The display will inform the user that the string being plucked is X note but sharp.

27.5 Trace of Individual Test Cases to Requirements

The purpose of this table below, was to map the previously outlined test cases to the requirement specifications. This served as a starting point to be able to determine if the Pick Pocket Tuner was ready to receive the mark of “Complete”.

Req. ID	Req. Description	Test Case Reference	Status
26.1.1	The Pick Pocket Tuner shall tune an entire guitar within a maximum amount of time.	1, 3, 4, 6, 7, 9, 10, 11, 14, 18, 19, 20	Complete
26.1.2	The Pick Pocket Tuner shall tune a new, individual string, within a maximum amount of time.	1, 3, 4, 5, 6, 7, 9, 10, 11, 14, 18, 19, 20	Complete
26.1.3	The Pick Pocket Tuner shall tune the strings to the correct frequency for that string.	1, 3, 4, 5, 6, 7, 9, 10, 11, 14, 17, 18, 19, 20	Complete
26.1.4	The Pick Pocket Tuner shall be able to tune more than 100 strings.	1, 3, 4, 5, 6, 7, 9, 10, 11, 14, 18, 19, 20	Complete
26.1.5	The Pick Pocket Tuner shall weigh less than 3 pounds.	1, 3, 4, 11, 13, 14, 15, 16, 17,	Complete
26.1.6	The Pick Pocket Tuner shall have the ability to tune to alternate tunings.	1, 2, 3, 4, 6, 7, 9, 10, 11, 14, 17, 18, 19, 20	Complete
26.1.7	The Pick Pocket Tuner shall have an easy-to-use Physical User Interface.	1, 2, 3, 4, 6, 7, 8, 9, 10, 13, 15, 16, 17,	Complete
26.1.8	The Pick Pocket Tuner shall be ergonomic, and small enough to fit in one hand.	1, 2, 3, 4, 8, 9, 11, 13, 14, 15, 16, 17,	Complete
26.1.9	The Pick Pocket Tuner shall have an easy-to-read display.	1, 2, 3, 4, 5, 6, 7, 9, 10, 14, 17, 19, 20	Complete

26.1.10	The Pick Pocket Tuner shall be able to use a ¼” input jack with a ¼” instrument cable.	1, 4, 6, 7, 10, 11, 18,	Complete
26.2.1	The Pick Pocket Tuner shall have an easy-to-understand software menu structure.	1, 2, 3, 5, 6, 7, 8, 10, 14, 16, 17,	Complete
26.2.2	The Pick Pocket Tuner shall be able to choose between four available modes of operation.	1, 2, 3, 5, 6, 7, 8, 10, 16, 17,	Complete
26.2.3	The Pick Pocket Tuner shall display the current tuning to the display.	1, 3, 4, 5, 6, 7, 9, 10, 11, 14, 17, 19, 20	Complete
26.2.4	The Pick Pocket Tuner shall be able to distinguish between flat and sharp notes.	1, 3, 4, 5, 6, 7, 9, 10, 14, 17, 18, 19, 20	Complete
26.2.5	The Pick Pocket Tuner shall be able to differentiate between octaves of the same note.	1, 3, 4, 5, 6, 7, 9, 10, 14, 17, 18, 19, 20	Complete
26.2.6	The Pick Pocket Tuner shall report accurate notes and frequency values.	1, 3, 4, 5, 6, 7, 9, 10, 14, 17, 18, 19, 20	Complete

28.0 Conclusion

The experience of designing this project has been very educational and encouraging. It allowed our team the opportunity to learn much more than we did before, not only in the designing aspect, but also in the day-to-day real world challenges engineers face on a regular basis. Whether that be solving a design flaw, supply chain issues, team meeting efficiency, collaboration with teammates and many other aspects we do not learn about from a textbook, this experience provided us with valuable knowledge on how to tackle common issues in the engineering realm. While our academic curriculum has given us the necessary tools to successfully complete this project, the reality is that this project has exposed each-and-everyone of us in the group to the different steps necessary in the product development process, from inception to final product delivery. We are excited about the outcome of our project, and are looking forward to what is next for the engineers in this group.

Bibliography

Ada, Lady. "Li-Ion & LiPoly Batteries." *Adafruit Learning System*, 6 Dec. 2021, https://learn.adafruit.com/li-ion-and-lipoly-batteries/proper-charging.

"A440 (Pitch Standard)." *Wikipedia*, Wikimedia Foundation, 27 Nov. 2021, https://en.wikipedia.org/wiki/A440_(pitch_standard).

"Brushless DC Motor vs. AC Motor vs. Brushed Motor." *Oriental Motor U.S.A. Corp.*, https://www.orientalmotor.com/brushless-dc-motors-gear-motors/technology/AC-brushless-brushed-motors.html?gclid=CjwKCAiAhreNBhAYEiwAFGGKPDWLT4VICIx25Se5uYDdfa0hdppbdn4xKDQiR9ayByDR-rdY74ZJbBoCpK4QAvD_BwE.

"Designing an Electric Guitar with Shapes." *TeachRock*, 22 Feb. 2021, https://teachrock.org/lesson/designing-an-electric-guitar-with-shapes/.
"ESP LTD DJ-600 Dan Jacobs | ZZounds." *Www.zzounds.com*, https://www.zzounds.com/item--ESPDJ600.

"FIT0441_BRUSHLESS_DC_MOTOR_WITH_ENCODER_12V_159RPM." *DFRobot*, https://wiki.dfrobot.com/FIT0441_Brushless_DC_Motor_with_Encoder_12V_159RPM.

"Fourier Transform in Python – Vibration Analysis." *AlphaBold* , 2 Mar. 2021, http://www.alphabold.com/fourier-transform-in-python-vibration-analysis/.
"Guitar Tuning Machines from Sperzel." *Www.sperzel.com*, www.sperzel.com/guitar-tuners.php.

"How to Properly Connect Lithium Batteries in Series and in Parallel." *Wsd*, 1 Mar. 2019, https://www.lithium-battery-factory.com/ithium-batteries-parallel/.

"How to Tune the Guitar to Standard Tuning." *Www.guitar-Chord.org*, www.guitar-chord.org/articles/standard-tuning.html.
"How to Use the Stratocaster Pickup Selector Switch." *Www.fender.com*, www.fender.com/articles/tech-talk/sounds-aplenty-the-stratocaster-pickup-selector-switch.

"Inductor Placement: Basic Knowledge." *ROHM TECH WEB: Technical Information Site of Power Supply Design*, https://techweb.rohm.com/knowledge/dcdc/dcdc_pwm/dcdc_pwm04/9321.

"JOWOOM T2 Smart Automatic Guitar Tuner Peg String Winder for Guitar Chromatic Us." *EBay*, https://www.ebay.com/itm/133889033602?chn=ps&_trkparms=ispr%3D1&am

data=enc%3A15-djjHG3TxCbDkMqqaaAAw95&norover=1&mkevt=1&mkrid=711-117182-37290-0&mkcid=2&itemid=133889033602&targetid=1262906534602&device=c&mktype=&googleloc=9012409&poi=&campaignid=15275224983&mkgroupid=131097072938&rlsatarget=pla-1262906534602&abcId=9300697&merchantid=101492064&gclid=CjwKCAiAhreNBhAYEiwAFGGKPFQFowQx8HS73pbnVZhCFv1ftcTEQuzyFqCv9ZSv4oiV203XVDM_iRoCwjsQAvD_BwE.

“Measuring Vibration: The Complete Guide | Brüel & Kjær.” *Www.bksv.com*, www.bksv.com/en/knowledge/blog/vibration/measuring-vibration.

“Piezo Vibration Sensor Hookup Guide - Learn.sparkfun.com.” *Learn.sparkfun.com*, https://learn.sparkfun.com/tutorials/piezo-vibration-sensor-hookup-guide/all.

“Piezoelectric Pressure Sensors.” *Avnet*, https://www.avnet.com/wps/portal/abacus/solutions/technologies/sensors/pressure-sensors/core-technologies/piezoelectric/.

“Piezoelectric Sensors - How Do Piezoelectric Sensors Work.” *Sensor Works*, 3 Dec. 2019, www.sensor-works.com/how-do-piezoelectric-sensors-work/.

“Placement of Inductors: Basic Knowledge.” *ROHM TECH WEB: Technical Information Site of Power Supply Design*, Tech Web, 7 Dec. 2017, https://techweb.rohm.com/knowledge/dcdc/dcdc_pwm/dcdc_pwm03/3254.

“Pure Tone Mono Multi-Contact 1/4″ Output Jack.” *Pure Tone Technologies*, https://puretonetechnologies.com/products/pure-tone-multi-contact-1-4-output-jack.

“Scout BK.” *Indiana Guitar Company*, www.indianaguitarcompany.com/scout-black.html.

“Software.” *Www.arduino.cc*, www.arduino.cc/en/software.

“Stepper vs Servo.” *Tutorial: Stepper vs Servo*, https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/stepper-vs-servo/.

“The String Family: Instruments, History & Facts - Video & Lesson Transcript | Study.com.” *Study.com*, 2019, https://study.com/academy/lesson/the-string-family-instruments-history-facts.html.

“What Is a Piezoelectric Sensor? - Utmel.” *Www.utmel.com*, www.utmel.com/blog/categories/sensors/what-is-a-piezoelectric-sensor.

AXE I/O, http://www.ikmultimedia.com/products/axeio/index.php?p=specs.

Belokobylskiy, Ivan. “St7789py_mpy.” *GitHub*, https://github.com/devbis/st7789py_mpy/blob/master/st7789py.py.

Frequencies of Musical Notes, https://pages.mtu.edu/~suits/notefreqs.

Industries, Adafruit. “Buzzer 5V - Breadboard Friendly.” *Adafruit Industries Blog RSS*, https://www.adafruit.com/product/1536?gclid=CjwKCAiAhreNBhAYEiwAFGGKPMRSK4GWxT5ykCDRAdUEalYp7kv5BtQm__5SA5EDeYfPiEk45-zi7BoCQkAQAvD_BwE.

Industries, Band. “Roadie Automatic Guitar Tuner.” *Roadie Music's Automatic Tuners*, https://www.roadiemusic.com/roadie3?gclid=CjwKCAiAhreNBhAYEiwAFGGKPMOMM9OG3GPRvfUm8dTntVZuZueJNEqLsKVcnPJUSMaBilh0qSjVTRoCX-YQAvD_BwE.

Kuklovskai͡a Elizaveta. “DP.” *Amazon*, "Books by Mail" Pub. Co., https://www.amazon.com/dp/B00J2QET64?psc=1&ref=ppx_yo2_dt_b_product_details.

Kuklovskai͡a Elizaveta. “DP.” *Amazon*, "Books by Mail" Pub. Co., https://www.amazon.com/dp/B015RQ97W8?psc=1&ref=ppx_yo2_dt_b_product_details.

Kuklovskai͡a Elizaveta. “DP.” *Amazon*, "Books by Mail" Pub. Co., https://www.amazon.com/dp/B01E38OS7K?psc=1&ref=ppx_yo2_dt_b_product_details.

Kuklovskai͡a Elizaveta. “DP.” *Amazon*, "Books by Mail" Pub. Co., https://www.amazon.com/dp/B07BK1QL5T?psc=1&ref=ppx_yo2_dt_b_product_details.

Kuklovskai͡a Elizaveta. “DP.” *Amazon*, "Books by Mail" Pub. Co., https://www.amazon.com/dp/B07RNBJK5F?psc=1&ref=ppx_yo2_dt_b_product_details.

Kuklovskai͡a Elizaveta. “DP.” *Amazon*, "Books by Mail" Pub. Co., https://www.amazon.com/dp/B07XC5KB8D?ref=ppx_yo2_dt_b_product_details&th=1

Kuklovskai͡a Elizaveta. “DP.” *Amazon*, "Books by Mail" Pub. Co., https://www.amazon.com/dp/B08168GWVJ?psc=1&ref=ppx_yo2_dt_b_product_details.

LeVan, John, et al. “The ABCs of Output Jacks.” Premier Guitar, 9 Sept. 2021, https://www.premierguitar.com/diy/guitar-shop-101/guitar-jack-wiring.

Libretexts. “Pitch- Sharp, Flat, and Natural Notes.” *Humanities LibreTexts*, Libretexts, 11 Apr. 2021, https://human.libretexts.org/Bookshelves/Music/Book:_Understanding_Basic_Music_Theory_(Schmidt-Jones)/01:_Notation_-_Pitch/1.03:_Pitch-_Sharp_Flat_and_Natural_Notes.

Ltd, Magnolia International. “Behringer | Product | GPK836BK.” *Www.behringer.com*, www.behringer.com/product.html?modelCode=P0836.

Maklin, Cory. “Fast Fourier Transform.” *Medium*, Towards Data Science, 29 Dec. 2019, https://towardsdatascience.com/fast-fourier-transform-937926e591cb.

Rembor, Kattni. “Adafruit 1.3’ and 1.54’ 240x240 Wide Angle TFT LCD Displays.” *Adafruit Learning System*, https://learn.adafruit.com/adafruit-1-3-and-1-54-240-x-240-wide-angle-tft-lcd-displays?view=all.

Rowley, Jared. “What Is the Difference Between a Mono And Stereo Jack?” *Guitarandampparts.com*, Sheehan Enterprizes, 24 June 2019, https://guitarandampparts.com/2019/06/24/what-is-the-difference-between-a-mono-and-stereo-jack/.

Simple Projects, et al. “Interfacing Arduino with ST7789 TFT Display - Graphics Test Example.” *Simple Projects*, 3 Sept. 2019, https://simple-circuit.com/arduino-st7789-ips-tft-display-example/.

Study.com, 2021, https://study.com/cimages/multimages/16/violinconstruction.jpg.

Success, Dan-Fret. “What Are the Guitar String Frequencies?” *Fret Success - Guitar Tuition*, 26 Mar. 2019, http://fretsuccess.com/what-are-the-guitar-string-frequencies/.

Swagatam. “Designing a Customized Battery Charger Circuit.” *Homemade Circuit Projects*, 31 Oct. 2020, https://www.homemade-circuits.com/designing-customized-battery-charger/.

Taifur, and Instructables. “Seven pro Tips for ESP8266.” *Instructables*, Instructables, 24 Apr. 2018, https://www.instructables.com/ESP8266-Pro-Tips/.

Team, The Arduino. “Stepper Speed Control.” *Arduino*, https://www.arduino.cc/en/Tutorial/LibraryExamples/StepperSpeedControl.

Thingiverse.com. “Superior Universal Guitar String Winder - Quick Tuner by Mordraden.” *Thingiverse*, https://www.thingiverse.com/thing:46231.