Auto-tuning an Electric Guitar via Fast Fourier Transforms and Motor Control: Preliminary Report

Team GA-5

Ryan Barker

Duke Durot

Jules Ebaa

Shane Ma

Michael Norcia

Problem Statement

To review the requirements, the purpose of this project is to design an autonomous tuner system capable of tuning a standard electric guitar to a set of known guitar tunings. The system itself must support three separate tunings (standard E tuning and two others) and may be built around an electric or acoustic guitar. The guitar may be modified to fit the design, and there are no restrictions on hardware. The system is required to be designed for the user to strum the guitar strings with instruction from the user interface, which may be designed as deemed appropriate by our group. All wiring must be contained and neat.

The main emphasis of the project is the accountability for user experience, with the philosophy that the easier the system is to use, the better the system is. Developing a wireless application to interface with the design is much preferred over a clunky user interface. The design preferably should not inhibit a guitarist's ability to play, both while it is being used and while it is turned off. The ability to remove the design is encouraged, but the user should not have to spend a significant amount of time or effort removing or attaching the unit to their guitar. As mentioned, modifications to the guitar are allowed, but significant modifications, such as extensive wood routing to the guitar's body, are discouraged.

Design Objectives

The primary goal for our design is to maximize the practicality of the guitar autotuner by creating a removable and self-contained structure that does not interfere with a guitarist's ability to play their instrument. It will be constructed around an electric guitar with 3+3 tuning peg configuration (or Les Paul style configuration) to support as many guitars as possible. The primary component of the design is to be housed on the guitar's headstock and contain all circuitry, processors, and motors used in the design. The design will support polyphonic tuning, or the ability to tune multiple strings at once. It will be capable of being used while the user is standing or sitting with the guitar in playing position. The user will interface with the design via an Android application available in the Google Play Store, which will allow the user to select three separate tunings and guide the user through the tuning process. The supported tunings will be standard, E flat, and Open G tunings. The control application should be intuitive enough to be used by a guitarist who has never seen it before. Once the tuning process has started, the design should be able to tune the guitar faster than a guitarist with average technical ability (faster than 30 seconds).

The below lists our design principles:

- Above all, make decisions that emphasize user experience.
- Minimize the modifications to the guitar whenever possible.
- Choose commercial-off-the-shelf parts from reputable vendors rather than self-made parts whenever possible.
- When selecting components for the design, value their quality over their price.
- Use hardware solutions to obtain readable frequency data and turn the guitar's tuning pegs, and software solutions to compute frequency analysis on each string.
- Process each string of the guitar independent of the others.
- Test each component of the design modularly. Do not integrate a piece of hardware or software into the design without first testing its functionality independent of the overall system.
- Finish developing and debugging the design on one tuning peg of the guitar before attempting to build the design on all six tuning pegs.

Preliminary Design

The overall preliminary design will be broken into six separate subsystems, as shown in Table B.1 in Appendix B. The operational amplifier and audio filtering subsystem will obtain signal from the guitar, amplify the signal to readable levels, and filter noise from other strings out of the signal. The audio sampling subsystem will be the simplest subsystem, be used to read data from subsystem one at a sufficient sampling rate, and act as a buffer to pass that data to subsystem three. The frequency analysis subsystem will convert the audio signal passed into the Fourier domain, analyze the data to determine how far the guitar is from being in tune, and send commands to subsystem four to tune the guitar. The motor control subsystem will connect to six motors directly connected to the guitar's tuning pegs and be responsible for interpreting commands from subsystem three to tune the guitar. The Bluetooth interface subsystem will maintain all communication between the preliminary design and the Android control application. Finally, the control application subsystem will be housed remotely on an Android device, start the tuning process remotely with a Bluetooth command, receive frequency data from the design, and report to the user when the guitar is in tune. Subsystems one through five will be powered by one set of two to four AA batteries. Figure B.1 in Appendix B shows the subsystem view of the preliminary design graphically.

For the input to the system, a hexaphonic pickup will be used. The primary advantage of using a hexaphonic pickup over a standard, single-coil pickup is that there will be an electrical signal output for each of the six strings of the guitar. A drawback is that without special installation of the pickup to account for the multiple signals, the bridge pickup will be rendered useless when amplifying the guitar for

normal play. This problem could be overcome, as stated, with special installation procedures and additional hardware, but has ultimately been deemed a non-issue for this specific application.

From the hexaphonic pickup, the output of each string will be fed into an operational amplifier and audio filtering subsystem. A diagram for this circuit is found in Figure B.2 in Appendix B. It will utilize single rail op amps and work as follows: Firstly, the circuit will filter out any DC bias the guitar may be applying to its AC output. The capacitors in the design will set up a low pass filter and block any frequency bleed over from the other strings of the guitar. Two resistors will divide the voltage from the positive rail in order to provide a DC offset of 2.5 volts bias to the AC signal to center it. Then, a circuit will be in place to amplify the guitar signal but block DC amplification. Since the DC offset is 2.5 volts, amplifying the bias would cause the circuit to rail. For this reason, the capacitor C_2 and C_3 are grounded, effectively blocking DC amplification. The prior mentioned low pass filter to ground is formed by the capacitor and resistor. The op-amp is set up in a non-inverting amplifier configuration, and R_f and R_i are set to a yet-to-be-optimized voltage gain of $A_v = 1 + R_f/R_i$. A compensating resistor will also be added to the non-inverting terminal of the op-amp to take the brunt of any non-ideal characteristics.

The audio sampling and frequency analysis subsystems will work in tandem to read in and analyze the data from the hexaphonic pickup. Ideally, the overall system would use the same microcontroller, presumably the Tiva, to host both subsystems two and three. However, the Tiva's limiting 32 KB of RAM would bottleneck the amount of samples the system could take, which in turn would decrease the accuracy of the overall system. Instead, our group opted to use a Raspberry Pi Model 2 Version B for all calculations, which has a surplus amount of 1 GB of RAM. The Raspberry Pi's strictly GPIO inputs will not read analog signaling. Therefore, to achieve optimal performance, our group has opted to split audio sampling from the hexaphonic pickup and frequency analysis into two separate subsystems, run the audio analysis subsystem on the Tiva, and run the frequency analysis subsystem on the Raspberry Pi.

The audio sampling subsystem will essentially act as an analog to digital converting interface between the raw AC voltages from the output of subsystem one and the GPIO inputs of the Raspberry Pi. Within this conversion system, we will quantize and sample each analog input signal for at least twice its cutoff frequency to satisfy the Nyquist Criterion. The circuitry itself will be implemented as a simple and relatively small Simulink model downloaded to the Tiva. The data will then be passed into the Raspberry Pi for Fast Fourier Transform Conversion and frequency analysis.

The frequency analysis subsystem will be serve to be the computing power for processing the guitar string signals and determining how close or far each string is from in tune. It will run on the Raspberry Pi and be used to provide input into the motor control subsystem to turn the guitar's tuning pegs accordingly. The subsystem itself will work based on Fast Fourier Transforms, which are discrete

time optimizations of regular Fourier Transforms. It will probably be coded in C using an application programming interface for Raspberry Pi and frequency analysis. Finally, because the Raspberry Pi contains a GPU and a CPU, the goal will be to run this subsystem on the Pi's GPU to free the CPU to run subsystem four.

The motor-controlled tuning subsystem consists primarily of two sub-parts: the mount and the motors. The mount will be a wooden frame attached to the head of the guitar to which the motors will be attached. The mount should be attached to the head at four points, in order to maximize stability and minimize the movement of any of the six motors. If any of them move in the slightest, the tuner will be less accurate and provide us with less than desired results. A spine will run down the back of the guitar head, upon which wings will be attached to hold the motors. Ideally, the spine and wings will not take up too much room, so that it doesn't take away from the playing experience.

The motors themselves will be either stepper motors or continuous-rotation servos, and must be able to handle loads of 40 oz in of torque at minimum. Taking such into consideration, a 64 oz in servo or stepper motor would be preferable. The input shaft of the motor will be coupled to the shaft of the tuning peg using a shaft coupler. The shaft of the tuning peg measures 0.15" (or 3/20"), and the chosen stepper motor has a 5 mm input shaft, so the ideal shaft coupler would be either 5mm to 4mm or 5 mm to 5/32". For the selected servo, couplers are available with a servo input on one side and a 5/32" bore on the other.

The motors should take a signal from subsystem three, and turn the tuning peg the desired amount in order to tune the guitar. To increase the accuracy of our robot, we will be removing part of the tuning pegs, so we may get direct contact with the shafts. Since we will be using a cheaper guitar, the current pegs do not directly turn the shafts. There is small amount of movement in them, thus removing them would remove the movement.

When mounting the six motors, the orientation of them will prove to be a key aspect of the project. Mounting the servos would be the easiest task, because they have small dimensions and can be mounted side by side. On the other hand, the steppers won't be able to fit side by side because of their size. In order to get them to fit, we would need to stagger them in a "V" pattern. The only problems that would arise from this is connecting all the steppers and being able to secure them in place, both of which can be done with some handiwork.

The Bluetooth interface subsystem is to be constructed to allow the user to completely and remotely interact with the overall design once it is powered on. The Bluetooth interface will either be built in or attached as an adapter to the Raspberry Pi Model 2B used for frequency analysis. It must be able be able to transmit two-way between the application and Pi to allow the communication described in the introduction of this section. Speed is key in designing this interface: Our group does not want it to affect the overall performance of the design, so the latency of any message being sent over it should be less than

a millisecond. Also, since keeping the system compact is key, this interface should require no more than two cables (a send and receive) to connect to the Raspberry Pi (or, preferably, be built into the Raspberry Pi itself).

The purpose of the Android application being developed with our design is to make the overall design as simple and easy to interface with as possible. The idea is for guitarists to be able to quickly choose what they want to happen through the software, and then set their phone or tablet on a table to refer to while the autotuner works on the guitar. The application is to be widely available through the Google Play store. Once a user has downloaded the software, to use the design, they will simply need to launch it and select one three buttons which correspond to the three different tunings. At this point, they will be directed to a screen showing them a table for each guitar string, the desired pitch of each string, and the frequency of each string. As the user strums the guitar to operate the autotuner, the table will live-update the frequencies. Once the guitar is in tune, an indicator below the table will notify the user that the system is done. The application will also contain pages to introduce users to the developers of the design with our photos. A home button will be placed on all screens to allow easy navigation back to the central menu options. A basic user interface meeting these specifications has already been developed for the application, which is shown in Figure B.3 in Appendix B. Additionally, though they have yet to be developed, all Bluetooth functions of the design are to be automatically done by the software at the appropriate stages. These include pairing the software to the design's Bluetooth adapter, sending Bluetooth signals to start the tuning process, and receiving Bluetooth signals from the design to update the frequency table. This decision reflects the project's emphasis on user experience, as users should not need to worry about the technical aspects of our application to operate it.

Research and Analysis

The way we are approaching signal processing in this project is unique in several aspects. Firstly, the idea of using two microcontrollers is a design (using the Tiva to sample and the Raspberry Pi to run Fast Fourier Transforms) has never been done before. In fact, prior to this semester, Raspberry Pi's have only been used for very minor applications in the senior design lab and have certainly never been used to control an entire design. For the frequency analysis, we must convert sinusoidal signals coming from an analog to digital converter (the Tiva) into the Fourier domain. Analyzing each signal in the frequency domain rather than the time domain will make them much easier to work with, since the signals will appear as a series of impulses in the frequency domain. The resonant frequency of each signal (which will be used to decide if the guitar string is in tune or not), will be the highest amplitude impulse of these impulses. To supplement these ideas, we have done some research on Fast Fourier transform algorithms, and found that they are largely available inside of programming APIs for popular language such as C or

Python. Upon reviewing our ECE 427: Digital Communications notes, we also realized that we need to sample with the Tiva at a frequency at least twice high than the cutoff frequencies of the low pass filters to satisfy the Nyquist Criterion. We have been and will continue to work closely with the lab assistants to learn more about the capability of the Tiva and to go through some predefined tutorials for the device that could serve useful. Additionally, members of our group attended both official help sessions on the Tiva. Our group admits that there is plenty we still do not know about the signal processing algorithms necessary for the design, but learning and experimenting with this technology is the first thing we want to focus on in the upcoming weeks.

After researching the torque required for the servo and stepper motors, we were able to find motors that would meet the design objectives. While weighing the pros and cons of both types of motors, size came up a lot. As we were leaning towards working with steppers, we knew that they may be too large to mount next to each other. In order to get around these physical limitations, we came up with a plan to stagger them so that they wouldn't be in the way. This would require minimal construction on our part.

Beginning the development for the Android control application required extensive research, as none of the engineers had any academic experience programming application or even programming in Java. Initial research quickly turned up Android Studio, Google's integrated development environment for application development, and its official documentation at developer.android.com. This website's coverage of the Java Android SDK is analogous to the UNIX manual pages coverage for the C programming language. Tutorials on the website were completed to learn the new IDE, its features, best practices for development, and design a basic echo application seen in Figure B.4 in Appendix B. Completing the tutorials allowed a full view of how Android applications are represented in Java: Different screens are referred to as activities, which are represented by classes. XML code is used to specify the format of these screens in logical subgroups called layouts and views. Methods are used and associated to objects in this XML code and their corresponding classes to allow user interaction and bring up new screens. It was this knowledge, and an Android operating system emulator built into Android Studio that allowed the engineers to produce what is seen in Figure B.3.

Though complete Bluetooth integration into the control application is still in progress, developer.android.com's coverage of the Bluetooth functions of the Android API have provided the development engineers several ideas for Bluetooth communication between the application and the main design. Refer to the preliminary design section for full coverage of these ideas. The engineers will use the Windows terminal emulator puTTY to test the Bluetooth features of the control application modularly as they are implemented.

Risk Assessment and Contingency Plans

Certain design choices cause uncertainty in performance. The primary causes of uncertainty, along with potential solutions, are outlined below.

- Motor size: For this application, a stepper motor is preferable over a continuous-rotation due to the inherent angular control. However, the stepper motor chosen for the project could prove to be too large to place in-line to turn the tuning pegs. There are ways to work around the size of the motors, such as staggering the motors such that they don't sit exactly next to each other. If all workarounds fail, it may be necessary to fall back on using a servo.
- Data transfer from Tiva to Raspberry Pi: Transferring data from the Tiva processor to another device is something which has not been previously attempted in this project, and thus will require careful attentiveness. The Raspberry Pi has no analog inputs, so the analog inputs on the Tiva will be used to sample the signals and the sampled data will then be fed into the Raspberry Pi, where calculations and processing will occur. Transferring the data, since the TAs have no experience in it, will be a new experiment, and since it is essential to the project that this is done correctly, is a potential weak point and bottleneck. If this is not accomplished, the program will be impossible to run.
- Motor mount: The motor mounting frame will have to be small enough to fit on the head of the guitar and also fit into a guitar case, but have enough points of contact to be stable and be large enough to fit six motors. These constraints will likely change the materials used and shape of the mount, and the mount poses itself to be a potential weak point as far as durability is concerned.
- Non-ideal characteristics of op-amps: for such small signals as will be output from the hexaphonic pickup, it could be necessary to consider the non-ideal characteristics of op-amps. Such a design change would necessitate using more IC packages, as well as purchasing potentiometers. It is unlikely that the non-ideal characteristics should need to be considered, but there are simple design changes that can help to adjust for these, such as compensation resistors and offset-null adjustments.

Testing and Data Collection Plan

As discussed in the Design Objectives section, central among our design principles is, "Test each component of the design modularly. Do not integrate a piece of hardware or software into the design without first testing its functionality independent of the overall system". Our group wants to know each portion of our design works before moving on to the next. This will allow full system integration to go as painless as possible and also provide debug points inside of the design when a certain piece of circuitry or software malfunctions. All of these ideas will key in navigating our six subsystem design.

Our test plans consist of four different categories: System input test plans, system hardware test plans, system software test plans, and overall system test plans. Input test plans include testing the hexaphonic pickup and subsystem one by taking twenty AC voltage measurements at their outputs and verifying the measured mean voltage amplitudes are at readable levels for subsystem two. Hardware test plans involve testing all solder collections while they are being built and testing subsystem response to various input voltages. We will also verify correct amplifier gains and frequency cutoff points for subsystem one, verify the sample rate of subsystem two, verify subsystem three is receiving the correct frequencies for each string, and verify the angular control of the stepper motor control circuit is functional and reliable. Each test to these subsystems will be run at least 25 times, and the standard deviation and mean of all test results will be computed to ensure suitable mean data values and a deviation of less than 1 unit. Software test plans include simulating all software before physical implementation, testing all cases of conditional logic, testing all Bluetooth communication with puTTY terminal emulator, and ensuring all voltage or current sources inside of software do not exceed values safe for hardware (e.g. verifying there is no way for a Simulink model to output 10 amps of current and blow the Tiva). Finally, overall system test plans will be done in conjunction with the Bluetooth Android control application. The most important of these tests will be speed tests and reliability tests. Speed tests will be centered on verifying we beat our target 30 second tuning time. Reliability tests will involve running the autotuner at least 30 times on each tuning to ensuring the frequency of each string (as shown in the application) is the value used by the tuning itself and have low percentage standard deviation.

Through the testing processes, several pieces of data will be collected to verify results. Any and all voltages or frequencies measured will be recorded and either tabulated or graphed in a shared Dropbox folder for easy access. For graphing, we will either use an oscilloscope or appropriate software such as B^2 Spice or Audacity. All software unit test drivers developed will be available through the Dropbox folder, and any test scripts used will be included in the final report. Circuit diagrams for all hardware will be drawn before circuits are constructed, and a photo of each test circuit used through the semester will be taken for the final report. An overall connection diagram for the design will also be generated.

Cost Accounting

After compiling a list of items needed to complete the project, which can be found in the Appendix, we were able to calculate the Expected Development Cost, Expected Out-of-Pocket Development Cost, and Expected Cost of Artifact.

The Expected Development Cost was $381.21, which was the total cost of all of our items. This includes the Tiva supplied by the ECE Department. The majority of our money was spent on the Epiphone Les Paul Express Electric Guitar. Two other areas where we needed to spend a little extra money were the six steppers and the hexaphonic pickup.

Our Expected Out-of-Pocket Development Cost was $356.22, which is less than the previous cost, because we were able to use donated items. The two donated items were the Tiva, donated by the ECE Department, and the various wires needed, which were donated from the previous semester's project. We did not purchase any spares, because we did not see a need for them. Shall anything need to be replaced, we will make a note of how much it is.

The final cost that we found was the Expected Cost of Artifact. This was the same as the Expected Development Cost ($381.21), because as stated before, we didn't buy any spare parts or supplies for this project. The cost analysis table found in the appendix shows all of the supplies that we will be using for the final project.

Project Schedule

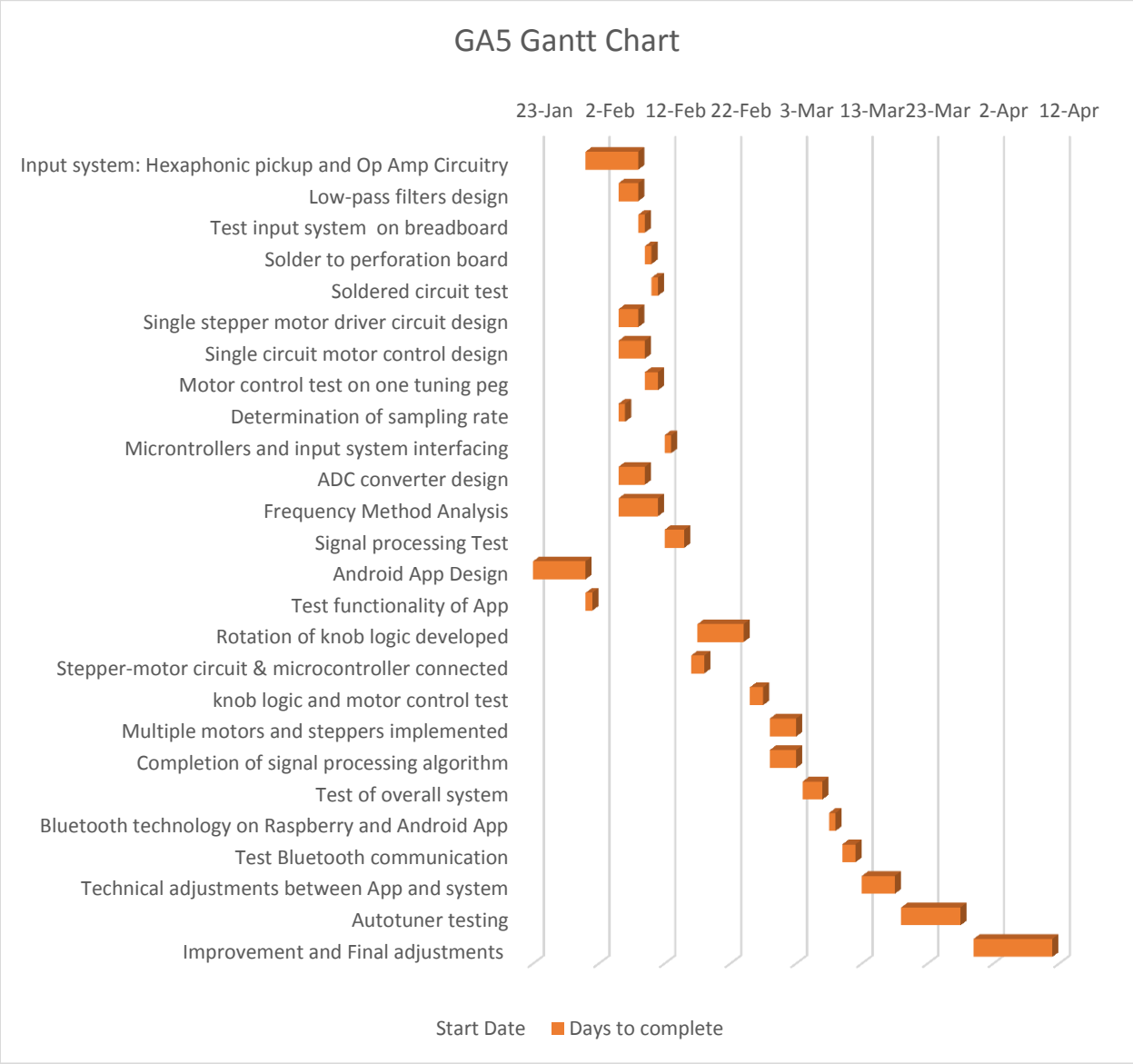

Figure B.5 in Appendix B contains this data in tabular format.

Appendix A – Parts List

Item Description	Part Number	Vendor	Quantity	Total Cost	Payment Type
Epiphone Les Paul Express Electric Guitar	-	Sweetwater.com	1	$119.98	Purchased
PowerGig Hexaphonic Pickup	-	Amazon.com	1	$60.00	Purchased
Stepper Motor	NEMA1740OZ	Amazon.com	6	$83.94	Purchased
Coupler	-	Amazon.com	6	$40.14	Purchased
Raspberry Pi	Version 2 Model B	Adafruit.com	1	$39.95	Purchased
Various Wire and Jumper Wires	-	Adafruit.com	1	$12.00	Donated
Microchip Single Rail Op Amp	MCP602-E/P	Microchip.com	2	$1.26	Purchased
Texas Instruments Tiva Launchpad	-	ECE Department	1	$12.99	Donated
Wood	-	Lowe's	1	$3.00	Purchased
Perma-Proto Breadboard	-	Adafruit.com	1	$7.95	Purchased

Expected Development Cost - Total cost of all items	$381.21
Expected Out-of-Pocket Development Cost - NOT donated items	$356.22
Expected Cost of Artifact - Total Costs of items from final design	$381.21

Appendix B – Figures and Tables

Subsystem Number	Purpose
1	Operational Amplifier and Audio Filtering Subsystem
2	Audio Sampling Subsystem
3	Frequency Analysis Subsystem
4	Motor Control Subsystem
5	Bluetooth Interface Subsystem
6	Android Control Application Subsystem

Table B.1. Preliminary Design Subsystems.

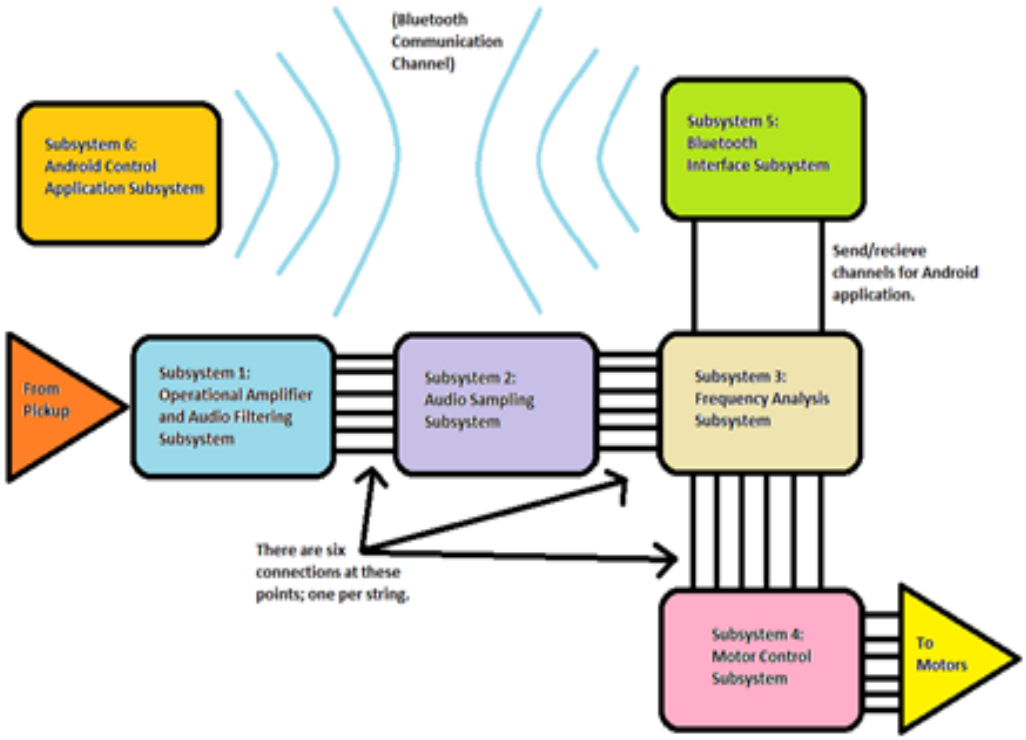

Figure B.1. Design Subsystems.

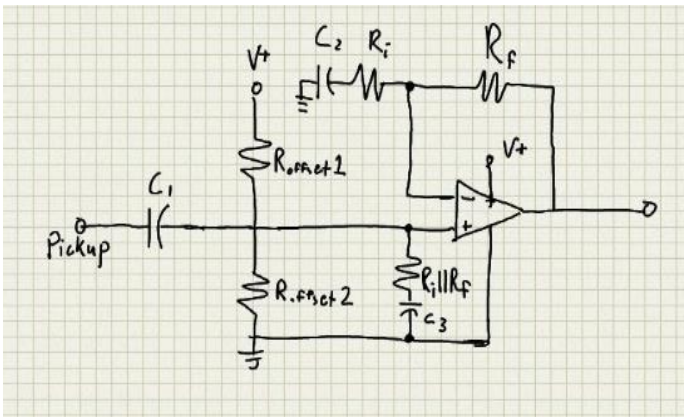

Figure B.2. Amplification Circuit Schematic.

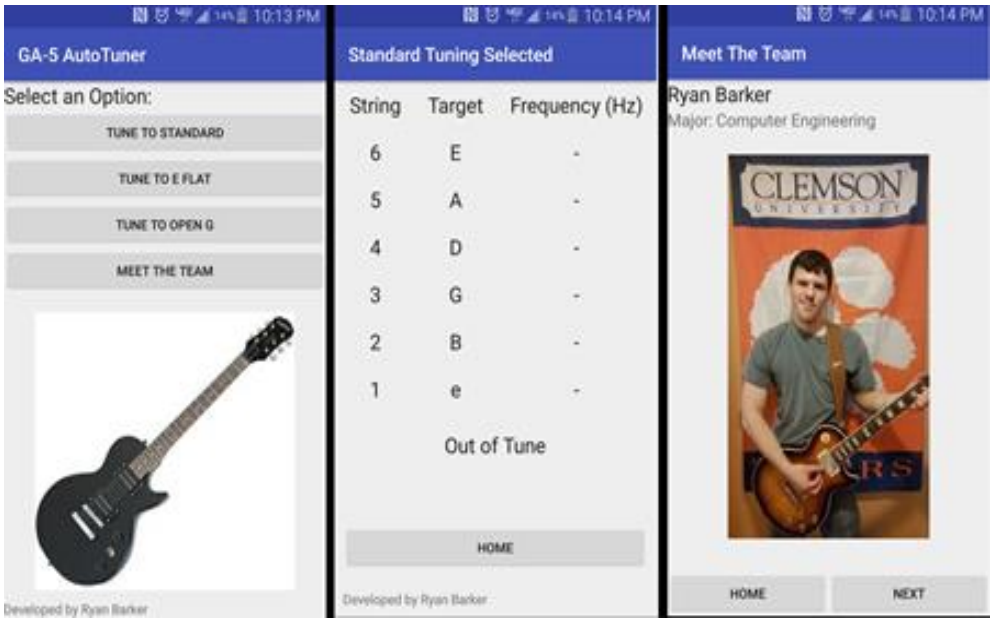

Figure B.3. Control Application User Interface.

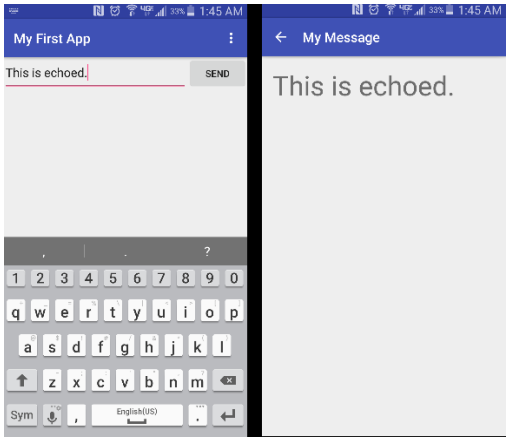

Figure B.4. Simple Tutorial Echo Android Application.

Task	Start Date	Days to complete
Input system: Hexaphonic pickup and Op Amp Circuitry	31-Jan	8
Low-pass filters design	5-Feb	3
test input system on breadboard	8-Feb	1
solder to perforation board	9-Feb	1
soldered circuit test	10-Feb	1
Single stepper motor driver circuit design	5-Feb	3
single circuit motor control design	5-Feb	4
motor control test on one tuning peg	9-Feb	2
Determination of sampling rate	5-Feb	1
Microntrollers and input system interfacing	12-Feb	1
ADC converter design	5-Feb	4
Frequency Method Analysis	5-Feb	6
Signal processing Test	12-Feb	3
Android App Design	23-Jan	8
Test functionality of App	31-Jan	1
Rotation of knob logic developed	17-Feb	7
stepper-motor circuit & microcontroller connected	16-Feb	2
knob logic and motor control test	25-Feb	2
Multiple motors and steppers implemented	28-Feb	4
Completion of signal processing algorithm	28-Feb	4
Test of overall system	4-Mar	3
Bluetooth technology on Raspberry and Android App	8-Mar	1
Test Bluetooth communication	10-Mar	2
Technical adjustments between App and system	13-Mar	5
Autotuner testing	19-Mar	9
Improvement and Final adjustments	30-Mar	12

Figure B.5. Gantt chart Tabular View.