Art for the Disabled

Electric^2 Guitar

Major Qualifying Project Report

in partial completion of the degree requirements for

Bachelor of Science, Electrical and Computer Engineering

Joshua DeNoncour

2019

Approved by:

Stephen J. Bitar, Advisor

Electrical and Computer Engineering, WPI

Abstract

Diseases such as ALS and muscular dystrophy have no cure, and cause a loss of muscle function as the disease progresses. This project was aimed to help those with disabled arms or hands play the bass guitar, creating a prototype bass guitar that received user input from an easy to use controller, designed to mimic bass playing in its design. A microcontroller processes the signals from the user and from the guitar, and controls a motor that adjusts the tension on the string.

Acknowledgements

I would like to thank professor Stephen Bitar for his extensive help in this project, as well as the WPI Electrical and Computer Engineering (ECE) Dept. for providing me with the skills to complete this project, William Appleyard of the WPI ECE store for his assistance in procuring parts, and finally, I would like to thank my friends and family, especially my father, for giving me the passion for music that made this project possible.

Executive summary

Muscular Dystrophy is a debilitating disease that gradually reduces muscle mass in patients, resulting in a loss of motion, strength, and ability. To help combat the mental effects of such a disease, and to give people who suffer a voice through music, a bass guitar was developed and modified to allow a user with Muscular Dystrophy to play.

Force sensitive resistors are used as a form of user input in an array designed to mimic the movements of playing a bass guitar. The prototype designed only utilized a single string, but could be easily modified to accommodate a four string design. User input was read by an Arduino Mega microcontroller, and used to determine appropriate motor reactions.

A guitar signal is fed into a fourth order butterworth filter with a break frequency of 40 Hz. This filter strips the harmonic rich guitar signal down to its fundamental frequency. This fundamental frequency is fed into an optocoupler configured as a zero crossing detection circuit. The optocoupler generates a pulse pattern in sync with the zero crossings of the fundamental frequency, and sends these pulses to an interrupt pin on the Arduino Mega.

The Arduino Mega calculates the frequency of the guitar's fundamental pitch and compares it to the user selected frequency through a series of if statements determining if the frequency is greater, lesser, within the acceptable variation to be in tune, or below the threshold for a played note. Different signals are sent to the motor controller based on these conditions.

The motor controls the tension of the string, and tightens or loosens the string accordingly, altering the frequency of the guitar signal. The actual playing is done by a pair of solenoids that damp the string vibration and strike the string to cause vibration.

Table of Contents

List of Figures

Goal statement

This project aims to create a proof of concept device to allow individuals with Muscular Dystrophy to play the bass guitar. This will be accomplished by creating a unit that alters string tension to change notes. The unit will replace the original bass bridge with no damage to the original guitar. The device will be controlled by a “mouse” that senses finger pressure to determine the note. This unit will also have a piano hammer style actuator that strikes the string to cause vibrations in the string, as well as a muting device to prevent slides between notes if a player cannot mute manually. Both the mute and hammer will attach utilizing the neck screws that hold the neck of the guitar to the body. String tension will be adjusted by some rotary device, possibly another tuning key.

1. Background

1.1 Muscular Dystrophy

Muscular Dystrophy is categorized by a gradual loss of muscle mass and muscle weakness caused by mutations of the genes needed to form healthy muscle. The majority of cases of Muscular Dystrophy occur in males, often appearing during childhood [1]. Muscular Dystrophy causes problems with muscle strength and function, limiting the mobility of an individual, and restricting their ability to perform what was once a common and easy task. This disease has no cure. People with Muscular Dystrophy will eventually experience trouble walking, to the point a wheelchair is necessary, and the tightening of muscles and tendons (called contractures), which further limit mobility. As time progresses they will also begin to experience difficulty with all muscular functions, both voluntary and involuntary, causing scoliosis, breathing problems, and heart problems[1]. Muscular Dystrophy is most common in males, as the disease is linked to a recessive gene on the X chromosome, meaning females, with two X chromosomes would need to have the mutation on both X chromosomes for the traits of Muscular Dystrophy to become apparent, whereas males would only need the mutation to show up on their single X chromosome [2]. Muscular Dystrophy will affect an estimated 1 in every 7,250 males aged 5 – 24 [3].

1.1.1 Duchenne Muscular Dystrophy

Duchenne Muscular Dystrophy is one of many types of Muscular Dystrophy, making up close to 50% of all Muscular Dystrophy cases, and is the most common form of Muscular Dystrophy often occuring before the age of five [2],[4]. No family history of Muscular Dystrophy or Duchenne Muscular Dystrophy is required, as it is a genetic mutation and can occur randomly. Duchenne Muscular Dystrophy is also hereditary, and is often passed down by a mother, as a father cannot pass an X chromosome to a son. Female carriers, or females with one mutated X chromosome can receive a carrier gene from either parent [5].

Duchenne Muscular Dystrophy usually occurs in boys between the ages of three and five. The first symptoms include enlarged calf muscles, difficulty learning to walk, and general clumsiness. As time progresses walking becomes more difficult, and children will begin to walk with a different gait, walking more on their toes or the balls of their feet to counteract leg muscle or tendon weakness. By the time a child reaches the ages of seven to twelve many will no longer be able to walk, and will require a wheelchair. Around this age they will also have developed weakness in the arm muscles, and can find lifting the arms difficult. As a child enters the preteen and teen years more serious, life threatening muscle weakness can occur. Heart, lung, and core muscles can become seriously weakened, sometimes fatally. Most people living with Duchenne Muscular Dystrophy will not live beyond their early twenties.

1.1.2 Psychological effects of muscular Dystrophy

A study found that boys with DMD had a higher likelihood of having depressive disorders as they age compared to boys of the same age and stature without MD. Many were found to have difficulties being social, or interacting with their peers, with approximately 58.8% of the study group admitting that they had troubled peer relations, and 64.7% being observed by a child psychiatrist to be anxious during the study. When compared to the control group, these numbers begin to show the psychological effects that MD can have on a child. In the control group, only 5% described having problems in their peer relationships, and only 25% were observed to be anxious. Observations of the children's moods determined that 41.17% of the children with DMD had a depressed mood, where none of the control group were observed to exhibit this behavior. A similar study on young adults and adolescents with cystic fibrosis conducted by R. Boyle in 1976 found a similar trend among children with disabling and life threatening disorders [6].

1.2 Disabilities and the arts

Art Therapy is a way to help enrich the lives of people by improving cognitive and sensory-motor functions. Self-esteem and self-awareness, as well as emotional wellbeing are also fostered or developed through art therapy. Music, a form of art, is one way people can undergo art therapy.

Many people turn to art as a form of expression, regardless of age, race, or physical or artistic ability. Art gives people a freedom to express themselves and explore their emotions. People with physical disabilities use art as a way of regaining some form of freedom, which they have lost to their disability. Disabilities can also cause social ostracization, especially among children, who perceive a disabled person as different [7]. One musician with Duchenne Muscular Dystrophy states

"*Right when I started Middle School at age 11 I started to use a wheelchair full time. Unfortunately, most other kids were less than accepting and I endured several years of verbal and cyber bullying and social isolation. I even spent some time being homeschooled because of what I was dealing with. Music quite possibly was my only saving grace.*" [8]

Music is such a varied art form, with so many different variations in the way it can be created, that it allows everyone to participate. Disabilities are not a handicap where music is involved, nor even something that should be considered. There are hundreds of musicians around the globe that play music and have some form of disability, ranging from blindness to quadriplegics. Among these musicians, most instruments are represented, from drums, to orchestral instruments, guitar is also among these instruments, but there are almost no guitar players suffering from Muscular Dystrophy or Duchenne Muscular Dystrophy listed among the musicians from the united states [can do musos]. One player, Jay Harris, experienced muscular atrophy in his hands, leading to an inability to play guitar like he used to. Jay had been playing music his whole life, and around the age of 17 muscular atrophy set in, and it devastated him to lose the ability to play music. Jay adapted his playing style and continued to play, until his left hand became paralyzed, and he could no longer play guitar like he used to. Jay has since changed his style again to accommodate his disability. [9]

1.2.1 Existing accommodations

Currently there are not a lot of options for a physically disabled person if they want to play a guitar style instrument. Most people with a physical disability just modify their playing style, or play the instrument in an irregular fashion that gives them the ability to play. Guitarists without hands such as Mark Goffeney have learned to use their feet to play, guitarists like Bill Clements with one hand have learned to play without a picking hand, using a combinations of hammer-ons and pull offs. Then there are drummers like Jason Barnes, who lost his right hand in an electrical accident, who have devices that allow them to play their instrument almost normally. Another prominent example of this is Def Leppard drummer Richard Allen, who plays most of his drum set using his feet. But these are all established musicians, who had been playing before they lost the physical ability. What about those who want to learn to play an instrument, but have a physical disability that prevents them from playing the instrument normally?

The Muscular Dystrophy Association (MDA) is involved with multiple assistive devices for musicians including the Laser Band and Jamboxx. Both of these were designed to create digital music, using a new 'instrument' to interact with a software module to produce the sounds. There is little to no information on assistive devices that modify or attach to a guitar style instrument. Yet, over time, a musician can develop an attachment to the instruments they play, as years of effort and reward become associated with a specific instrument it gains a sentimental value. Sometimes instruments are passed down through a family and gain sentiment in this manner, but a person suffering from Muscular Dystrophy would not be able to play a specific, possibly important instrument due to their disability.

1.3 Music

1.3.1 Musical instruments

There are hundreds if not thousands of different varieties of musical instruments, but the majority can be broken down into a several major categories, those being brass, woodwind, percussion, guitar, bowed stringed, and keyboard styles of instrument.[10] This project focuses on the guitar family of instruments, namely the electric guitar family. Guitar style instruments are played by strumming or plucking one or more strings made of varying materials. A guitar is normally played with the left hand fretting notes on the neck of the guitar, and the right hand either plucking or strumming the strings.
An electric guitar is composed of several different parts. The body houses the electronics, which include the volume and tone controls, and the bridge, which secures the strings to the body [Figure 1]. The neck connects to the body and houses the frets [Figure 3]. At the top of the neck, opposite the body, is the headstock, where the strings terminate at the tuning pegs [Figure 2].

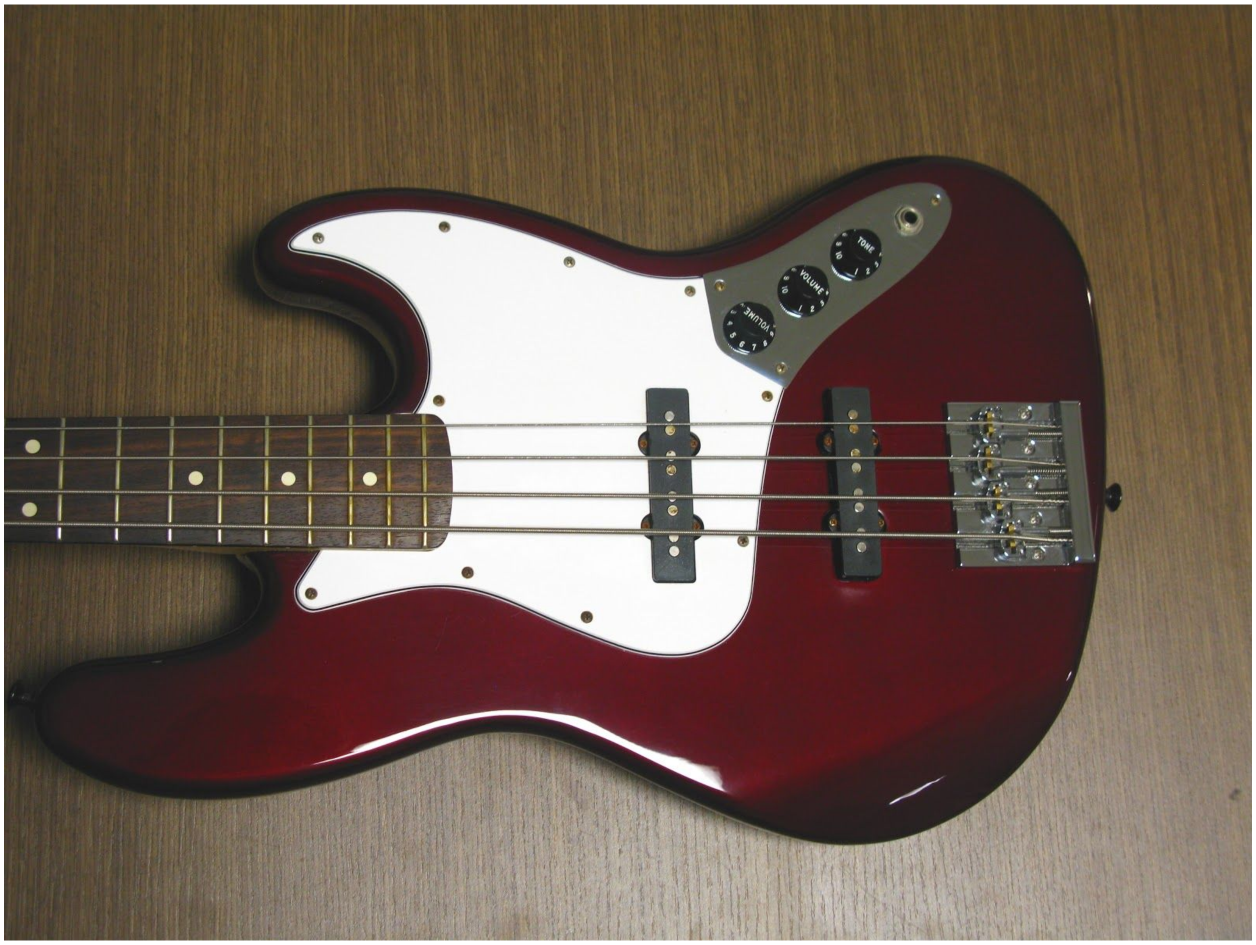


Figure 1:The body of a bass guitar.

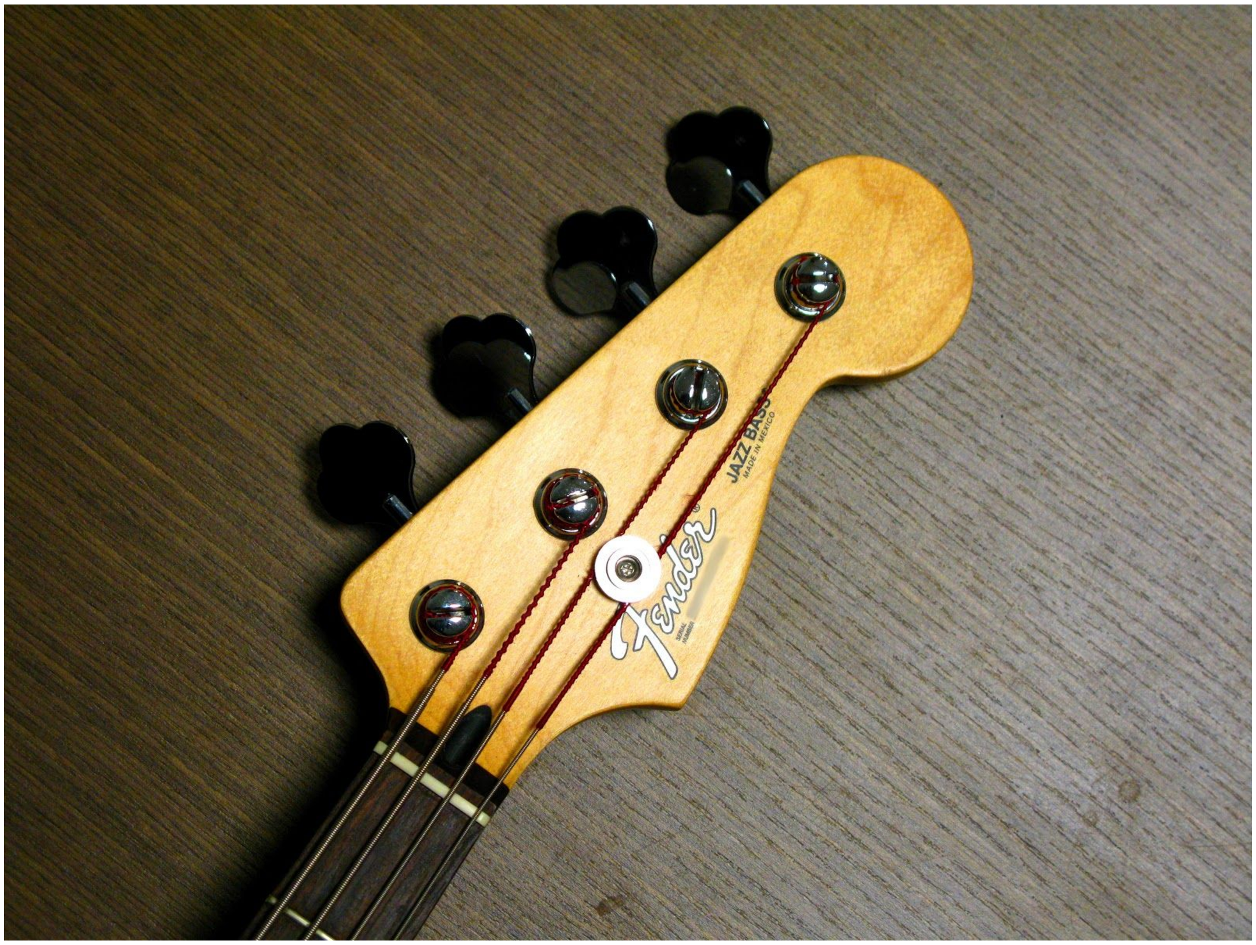


Figure 2: The headstock of a four string bass guitar.

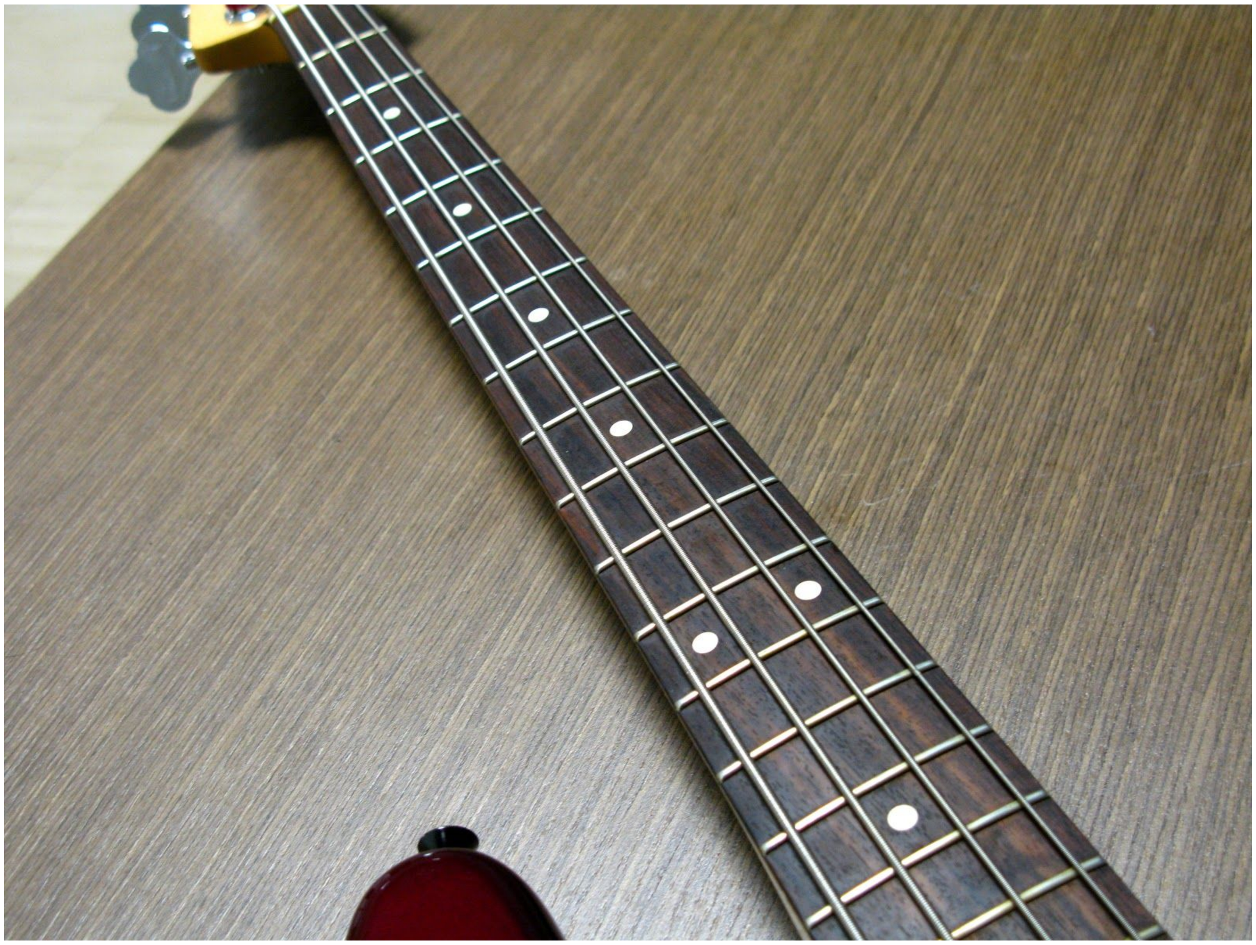


Figure 3: The neck of a guitar, showing the fretboard and frets.

An electric guitar works by picking up vibrations from plucked or strummed strings using an inductive coil pickup. An electric guitar's pickup works by picking up disruptions in a magnetic field generated by the pickup. The movement of a metal object, in this case, a steel string, disrupts the magnetic field, inducing a fluctuating voltage in the pickup coil.[11]

1.4 Bass

Bass guitars are a variation of the guitar with only four strings, tuned an octave lower than a regular guitar. The bass guitar originated from the orchestral upright bass, sometimes called a double bass. The electric bass became popular among musicians for its more compact size and lighter weight than its counterpart, the double bass. The first electric bass guitars were made by Paul Tutmarc in the

1930's, but the electric bass did not see commercial success until the 1950's when Leo Fender created the Fender Precision Bass. [12]

1.4.1 Types of Bass

Bass guitars come in a majority of styles, and are manufactured by hundreds of companies. [13] The majority of basses have between 4 and 6 strings, although basses exist with up to 24 strings. Both acoustic and electric basses exist, although a much greater variety in styles exists in the electric basses than the acoustic basses. Most variations in bass guitars comes down to a preference of the player. For this project, as there was no single specific person this product was targeted towards, the product is designed to fit the greatest number of basses and fit the needs of the majority of players. This means that four string basses are the only ones considered in the design of this project. Four string basses make up 63% of the market for bass guitars.[14]

1.4.2 Parts of the Bass

1.4.2.1 Neck

The neck of a guitar contains the fretboard and headstock. This makes up the majority of the length of a guitar, and is slightly under three feet in length, varying on standard scale guitars by a couple inches. The bass used for testing has a length of 33 inches from the tip of the headstock to the base of the neck. Inside the neck is the truss rod, an adjustable metal rod used to help straighten the neck of a guitar and counteract the tension of the strings.

On the front of the neck is located the fretboard, a hard slab of wood, usually made from rosewood or maple, with pieces of fret wire embedded at regular intervals. The fret board is how a conventional bass alters notes while playing. The fretboard on a bass is two feet from end to end. At the top end of the fretboard is the nut, a piece of plastic, bone, or metal that keeps the string in place laterally, as well as creating a break angle that stops the string from vibrating past a certain point.

Guitar headstocks vary in shape and size based on guitar model and manufacturer. The headstock provides a mounting point for the tuning pegs, which anchor the strings at the top end of the guitar and adjust the tension on each string.

1.4.2.2 Body

The body of the guitar is a solid piece of wood, typically of a dense wood such as alder, basswood or mahogany where all other components of the guitar connect. The neck of the guitar is bolted to the body, the bridge is mounted opposite the neck, and cavities are carved out in the body to house the electronics of the guitar. The electronics of a guitar include the pickups, as well as the volume and tone controls. The bridge is a piece of metal that anchors the strings at the base of the guitar, as well as creating a break angle, stopping string vibration at a certain point.

1.4.3 Range and Tuning

Bass guitars usually range in pitch between 40Hz and 330Hz (Musical notation E1-E4). A four string bass in standard tuning will have the first string tuned to the note E1 (41.2Hz), the second string tuned to A1 (55Hz), the third string tuned to D2 (73.42Hz) and the fourth string tuned to G2 (98Hz). Extended range basses can reach well beyond the listed range, with a standard tuned six string bass reaching as low as 30.87Hz (B0) and as high as 466.16Hz (B flat 4. Some basses are tuned in drop tuning, where one or more strings are tuned to a lower than normal note. Drop D tuning results in first string of a 4 string bass being tuned to 36.71Hz (D1) other strings are the same as standard tuning.

1.4.4 How does a bass work to create a note?

The strings of the bass guitar are made out of a core of wire wrapped with another smaller wire to add mass and prevent a sharp recoil if a string were to snap. Three factors affect the resonant frequency of the string, vibrating length (scale length), string weight, and string tension, change any of these and the frequency of the string will change. Normal bass or guitar playing uses a change in string length to alter the frequency of a note. The exact frequency of a string can be determined through Equation A:

$$(Unit\ Weight(lb/in) * (2 * Scale\ Length(in) * Frequency(Hz))^2)/386.4 = Tension(lb)$$

Equation A: Calculation to determine tension on a guitar string [15]

For this project there are two options to change the played note, altering string tension, or changing the vibrating length of the string. Changing the weight of the string is not a feasible option as adding or removing material from the string is not sensible. Changing string length in the same manner as a regular guitar is a possible option and has been explored in a previous Major Qualifying Project (MQP) where a ukulele was made to accomplish the same goal as this. The assistive playing device for the ukulele had an actuator on each fret of each string. For a bass this would require a large device, and would be rather unwieldy and complex. The second option evaluated to change notes is an alteration of string tension. This method would only require four actuators, one for each string, and would be useable on any length or style of bass, regardless of scale length, neck profile, or manufacturer.

2. Solution: “The Brick”

Two parts are needed for a solution to this problem, an apparatus that takes user input and relays it to the bass, and a unit attached to the bass that interprets the signals gathered from the user and converts it to a change in the bass’s condition, playing the desired note. Multiple avenues were analyzed, and evaluated for potential usefulness to this project. The Brick is a design that adjusts the string tension to change a note, rather than altering string length is done when a player frets a note in the normal style of playing guitar.

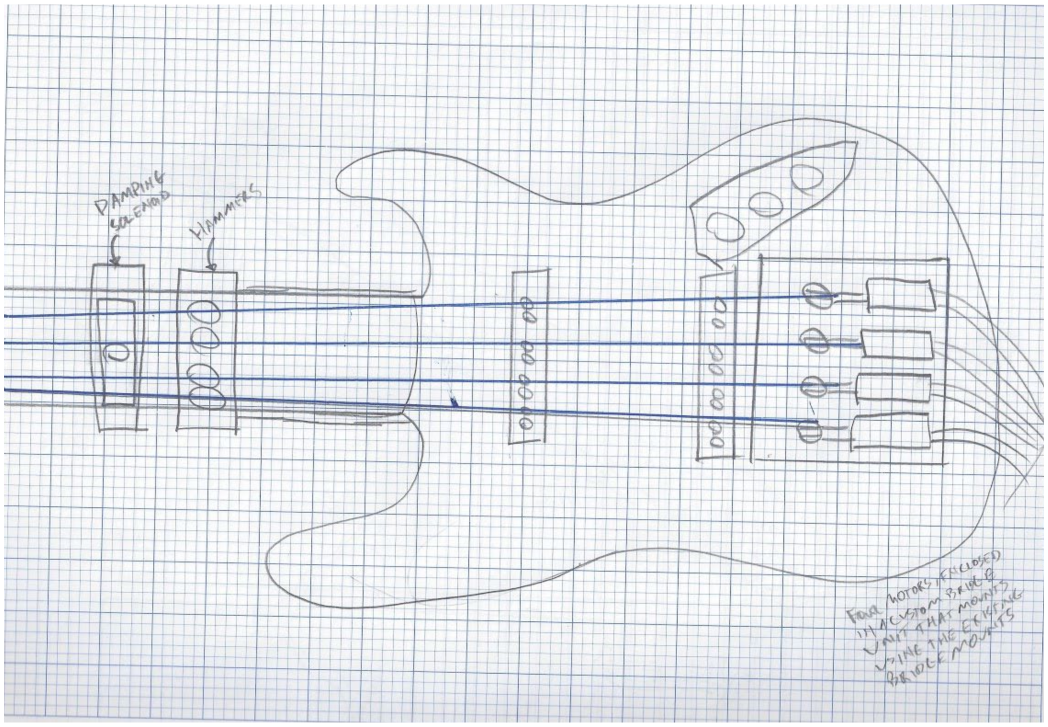

DAMPING SOLENOID
HAMMERS
FOUR MOTORS, ENCLOSED IN A CUSTOM BRIDGE UNIT THAT MOUNTS USING THE EXISTING BRIDGE MOUNTS

Figure 4: An artist rendition of a Fender Jazz bass style guitar with “The Brick” mounted instead of a standard bridge. Damping and Hammer solenoids would mount to the neck plate on the backside of the guitar, utilizing screw holes that already exist.

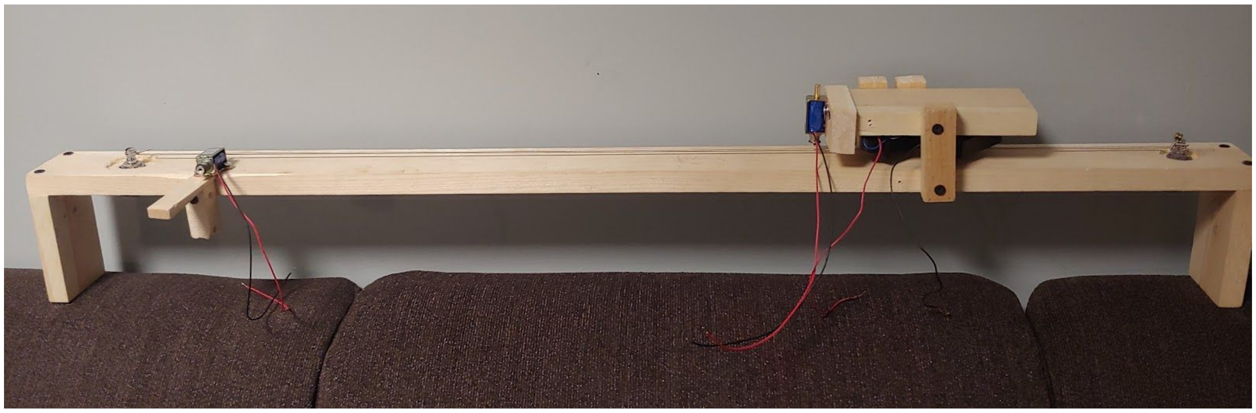


Figure 5: The single string prototype device to be used, showing the hammer (5a), damper(5b), and pickups(5b).

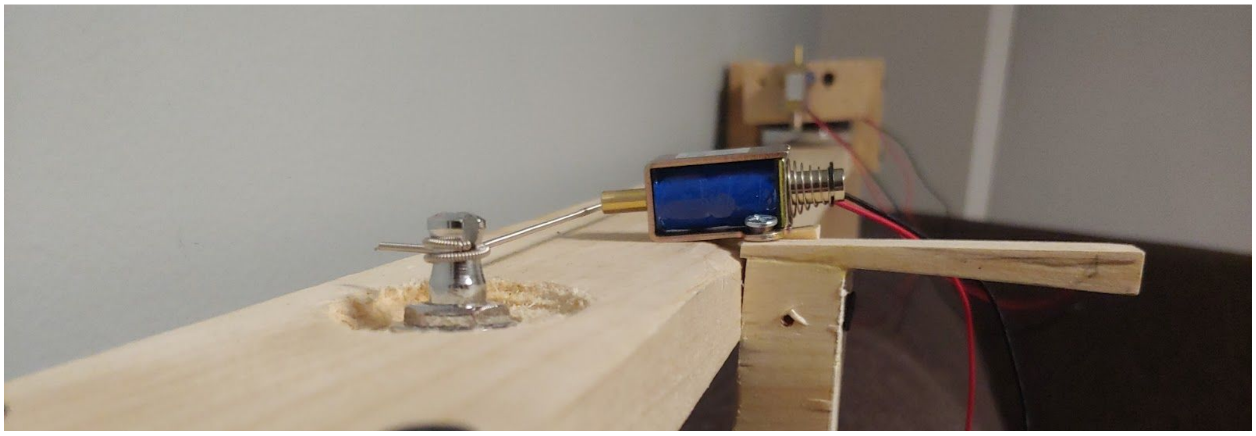


Figure 5a: The hammer solenoid.

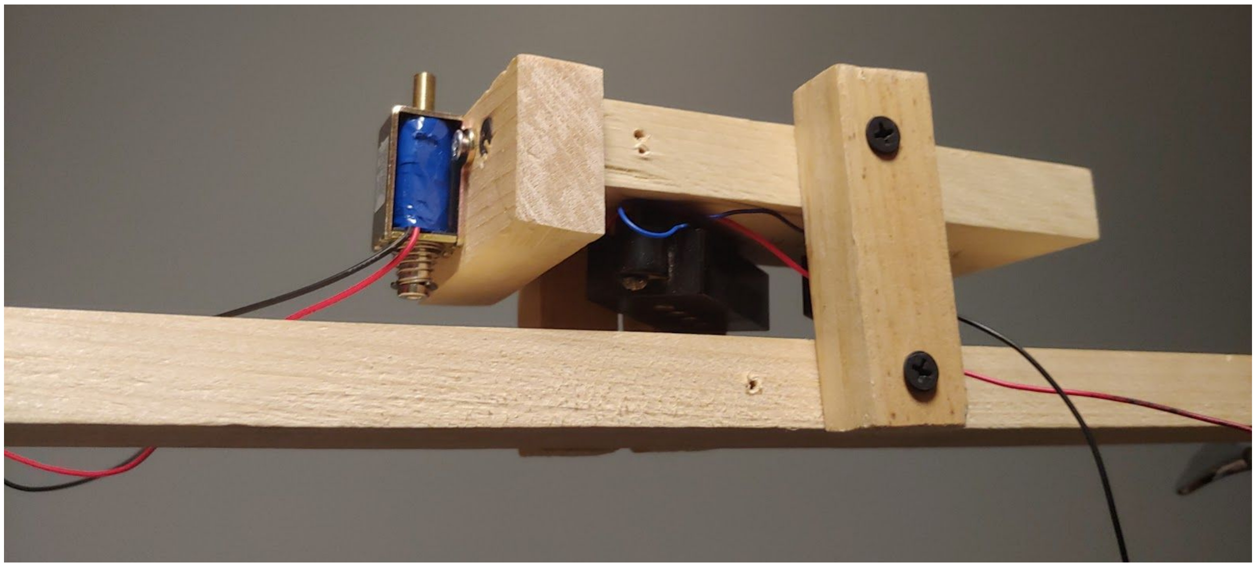


Figure 5b: The pickups and damper solenoid.

2.1 Inspirations

2.1.1 Ukulele MQP

A previous MQP created an assistive playing device for the ukulele. The project design included a computer mouse-like device to receive user input, and a rack of solenoids to press down the strings of the ukulele. The mouse-like design used force sensitive resistors as buttons, requiring little force to register a pressed button. This was found to be a highly effective idea, although a test user suggested the ukulele remember user input until it was changed, using a latching method. This project looks to keep the "mouse" idea for a controller. It seems to be the best method of receiving input from a subject.

The fretting mechanism was found to be too unruly and complicated to adapt to a bass.
One goal of this project was to design the apparatus without causing permanent damage to the bass. The solenoid method used in the Ukelele MQP was found to be too involved to mount easily on a bass with minimal changes. Some form of mounting bracket(s) would need to be screwed on near the headstock, requiring mounting holes that may compromise the strength of the neck, in addition to doing permanent damage to the bass if the apparatus was ever to be removed, holes would be left behind. [2]

2.1.2 Piano mute, hammer

Pianos are a complex musical instrument, with each key containing over a dozen individual parts. Two parts in particular are of interest to this project, the hammer, and the damper. A piano hammer is used to strike the piano strings, causing them to resonate and produce a note. The damper is used to mute the piano strings when a note is terminated. The damper either stops a note or allows it to ring out. A similar method will be implemented in "The Brick" to assist a player that does not have the hand strength to pluck and mute the strings of a bass.

2.1.3 The Washtub Bass (a varied string tension instrument)

The washtub bass is made from a hollow resonating chamber such as a metal washtub, large box, or tea chest, a broomstick, and a string. One end of the string is tied to one end of the broomstick, and the other fastened to the washtub center. The end of the broomstick that isn't tied to the string is placed on the rim of the washtub, pulling the string tight. Notes on a washtub bass are changed by altering the tension on the string.

2.1.4 Gibson Min-Etuners™ (an auto string tuner)

Gibson debuted an auto-tuning product in December of 2007 called the Gibson Robot Guitar. This product tunes all six strings on a guitar at the same time, with high accuracy. The robot guitar used a servo motor on each tuning peg to adjust the tension on the string. The documentation on the original Robot tuner is vague on how the Robot Tuner system monitors the pitch of the string. Below is an excerpt from the archived page on the Robot Guitar.

"Gibson's new Data Transmitting Tailpiece is a hub of activity. First, each string is separated by ceramic insulators that isolate each individual string signal and avoids confusion as to which string is being processed and tuned. There are also special isolating inserts that keep the ball ends commonly found on electric guitar strings from making contact and disrupting signal flow. Underneath the tailpiece is a tiny circuit board that processes each individual signal to the ribbon cable, which is then transmitted to the on-board CPUs, which, in turn, tune the strings."

The system works by monitoring a signal generated by each string as they vibrating, and a set of sensors mounted in the bridge of the guitar interpret these signals and relay information to a CPU unit on the headstock of the guitar, which controls the motors on the tuning pegs. The Robot guitar was a limited edition model, and was not widely available, and the tuning system was not available for separate purchase. If a musician wanted the auto tuning system, they had to purchase the robot guitar. Demand for an aftermarket tuning system lead to the production of the Gibson Min-Etune, a headstock mounted auto tuning unit. [16], [17], [18], [19]

3. Objectives and Constraints

3.1 Objectives

1. Create a device allows use of previously owned bass.
2. Create a device that allows a user with Muscular dystrophy to play the bass in some capacity.
3. Create a device that reacts to real time input from a bass guitar.
4. Create a device that can determine frequency of an unknown signal.

3.2 Constraints

1. Device must not damage the bass.
2. Device must be able to tune specific notes.
3. Device must be safe.
4. Device must be non-exhausting.
5. Device must tune quickly.
6. Device must be small enough to mount on a bass guitar.

4. Testing

The test bass was equipped with a already well used (est. 5 months of playtime) set of medium gauge standard strings (45-105), to simulate a less than ideal string condition to test if the strings would be able to handle the applied tensions while being played. The bass was then tuned to standard tuning. Each string was then individually tuned up to the highest required tension that they would be subjected to under normal playing conditions while being used in conjunction with the tensioner unit. All strings held up under excess tension while being played, even in their worn state. All components of the bass also showed no signs of distress or altered operation under this higher stress, proving that this method of changing string tension is applicable to a regular, unmodified bass. Each string was marked at the nut while in standard tuning and then marked again at the nut while under the highest required tension (Figure 6). This data will later be used in a predictive model for the feedback system to accelerate tuning speed and accuracy.

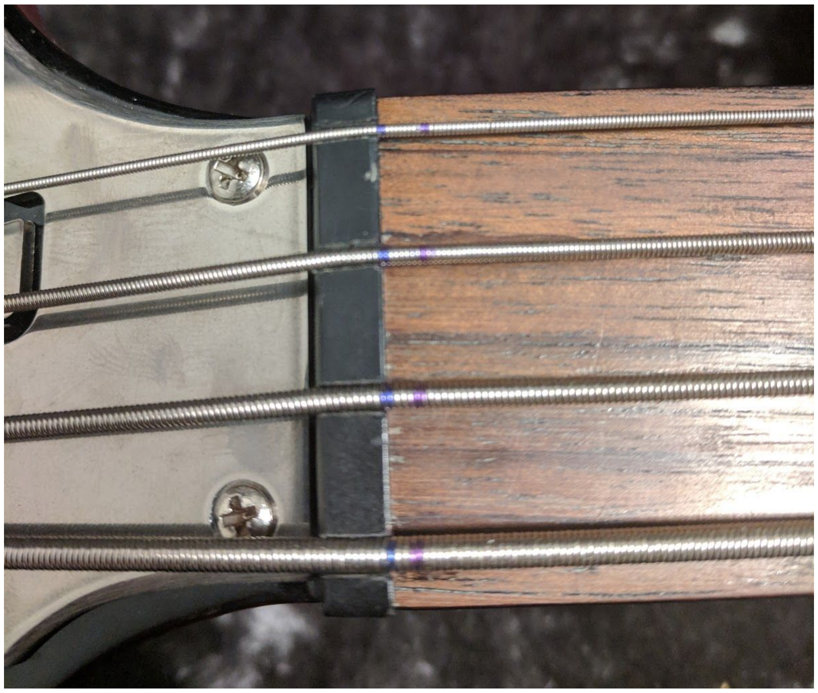


Figure 6: The test bass, with string markings for highest and lowes tensions. Bass is shown with strings at lowest tension.

Bass output varies greatly, namely it decays over time as the momentum of the vibrating string decays. Figure 7 shows the output from a bass immediately after the string is plucked, Figure 8 shows the same string 5 seconds later. Harmonics also change with respect to time, with some harmonics lasting longer than others. Initial voltages can reach the realm of almost 1V peak to peak(pk-pk) on a loud note. When a passive filter was attached directly to the bass output, these voltages dropped to the range of 20mV pk-pk resulting in the need for a unity gain buffer to prevent loading down of the bass signal.

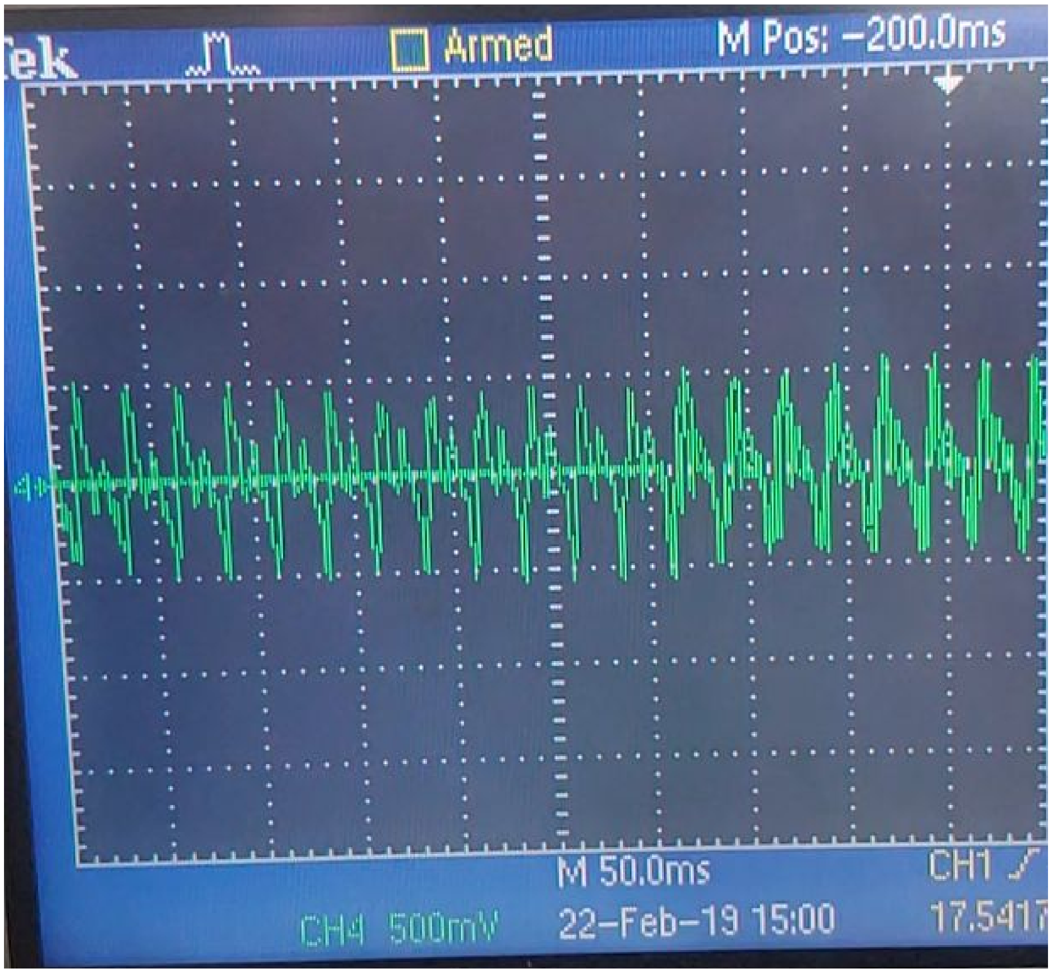


Figure 7: A bass string output immediately after being plucked.

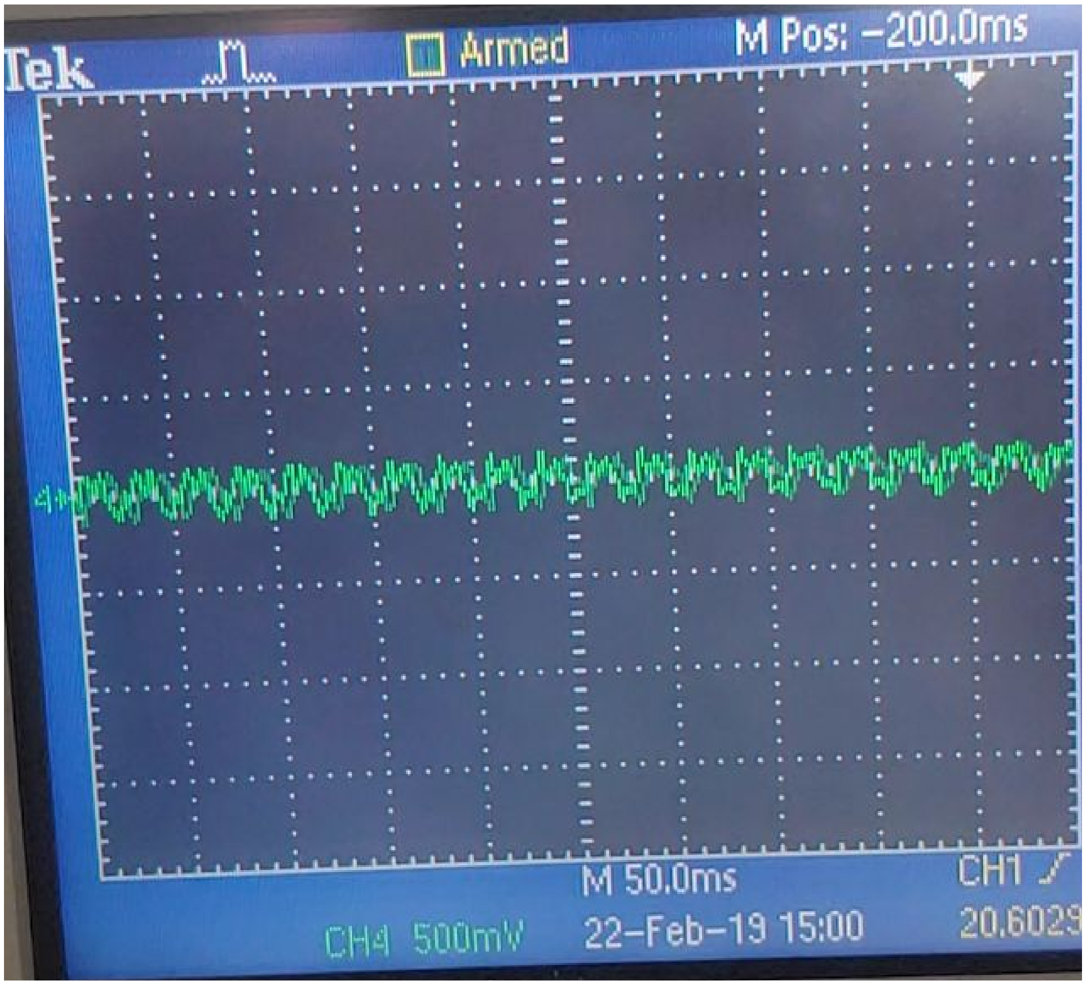


Figure 8: A bass string output five seconds after being plucked. Note, the amplitude has more than halved.

The bass output needs to be filtered to remove the intrinsic harmonics from the string, and isolate the fundamental frequency. Each string has harmonics vibrating at multiples of the fundamental frequency. These muddy up the fundamental frequency during processing and would ideally be reduced to zero through filtering, although this is not entirely possible, a well designed filter can reduce the harmonic signals to a level where their effect on the fundamental frequency is negligible.

4.1 Filter Design

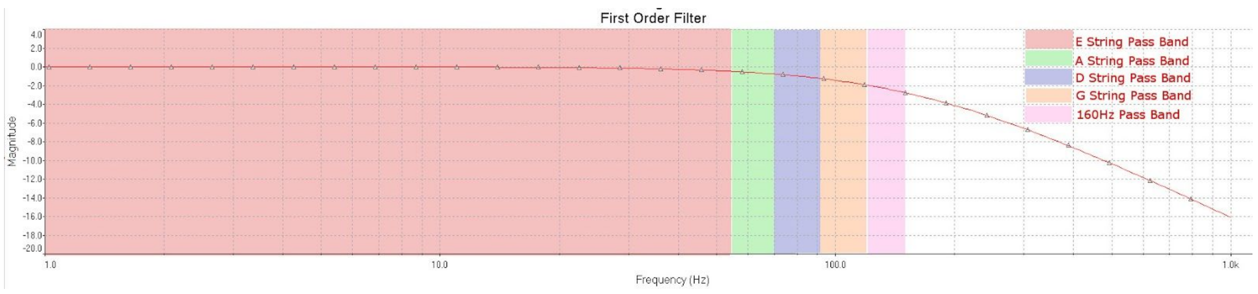


Figure 9: A Bode Plot of a first order filter with a break frequency of 160 Hz. Shown also are pass bands of individual strings on the bass in standard tuning.

Initially a first order capacitive low pass filter was designed using R=100 Ohm and C=10 micro Farad, resulting in a break frequency of 159.155 Hz (Figure 10). A lower break frequency would begin to attenuate fundamental frequencies of the higher strings. The logic behind this filter design and break frequency favored a simplistic approach, utilizing one filter for all frequencies. In testing a first order filter proved to be ineffective at removing the harmonics, especially at lower frequencies, which had harmonics below the breakpoint. It was concluded that each individual string would require its own filter as shown in Figure 9.

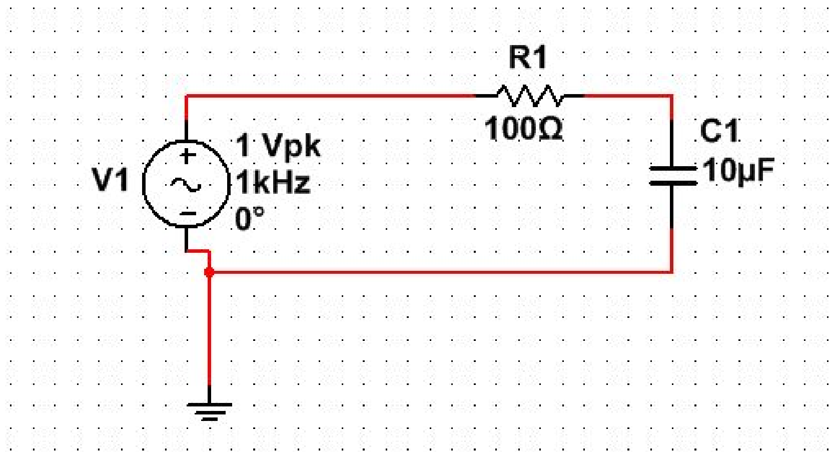


Figure 10: A first order low pass capacitive filter.

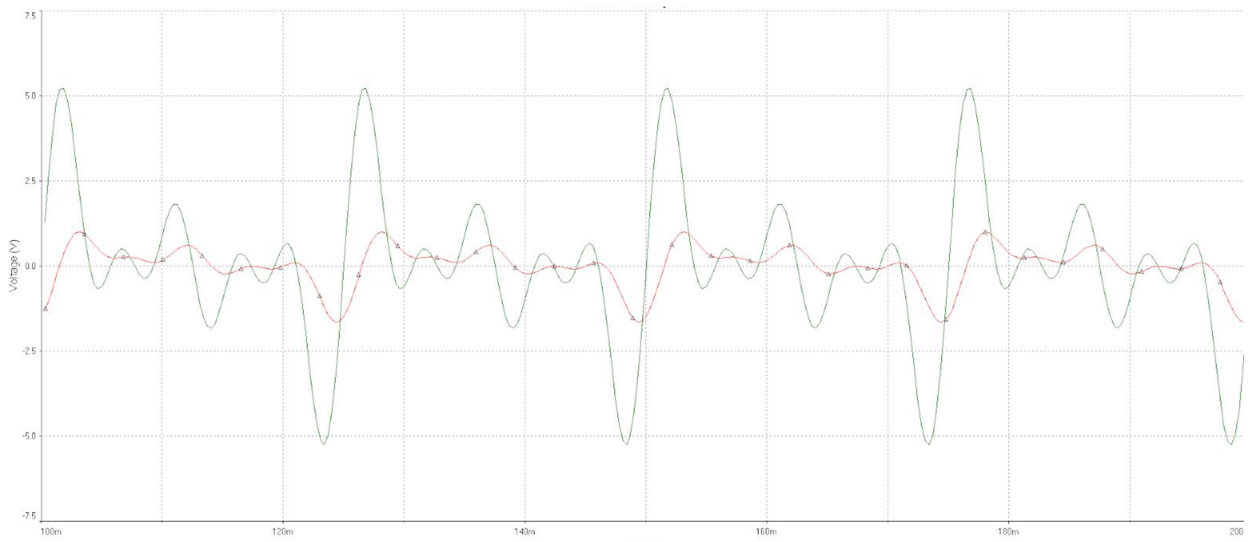


Figure 11: The Multisim Transient analysis of the first order filter shown in Figure 10.

The lowest frequency was chosen for experimentations, with a fundamental frequency of 41.2 Hz. A second order RC low pass filter was constructed with a break frequency of 40 Hz (Figure 12). Tests again proved that the harmonic attenuation was not enough to produce a useable signal, and a much steeper roll-off was required to produce the desired results. Keeping the break frequency at 40 Hz, a 80 dB per decade roll-off was needed to provide a sharp enough filter to have a marked effect at 80 Hz, the lowest frequency harmonic involved. 80dB per decade roll-off means a fourth order filter, or a filter that has four stages.

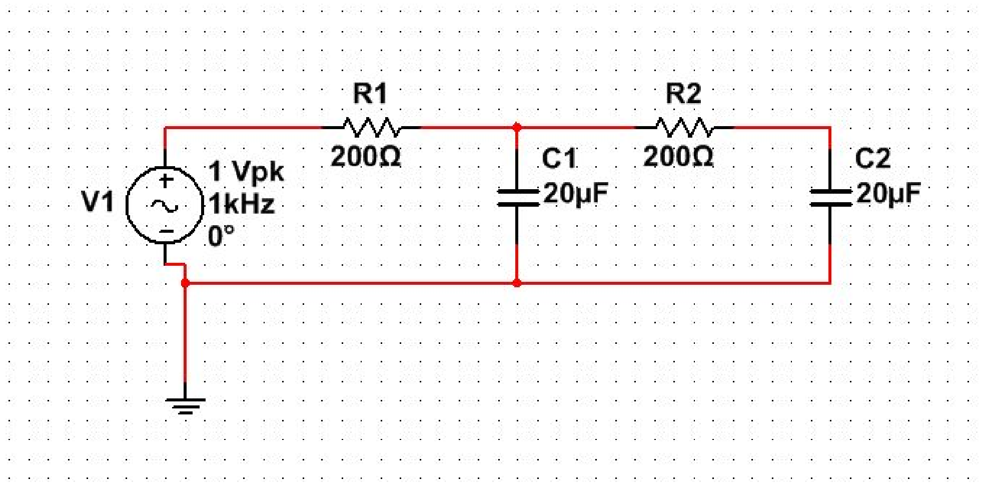


Figure 12: The circuit for the second order filter created. Break frequency: 40 Hz

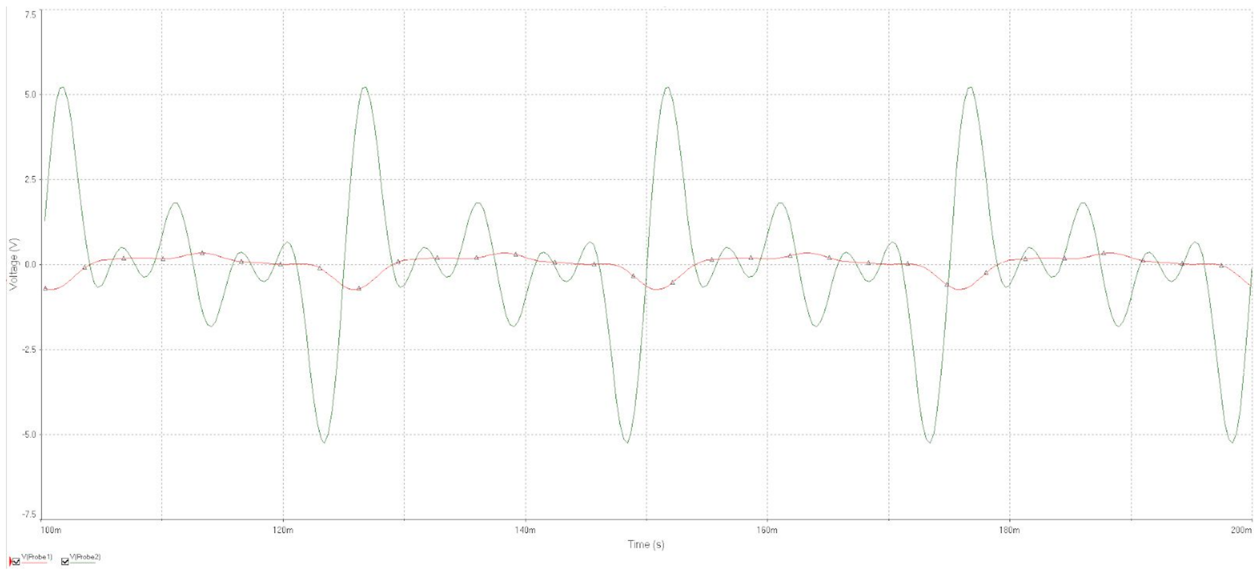


Figure 13: The Multisim transient analysis of the second order filter from Figure 12.

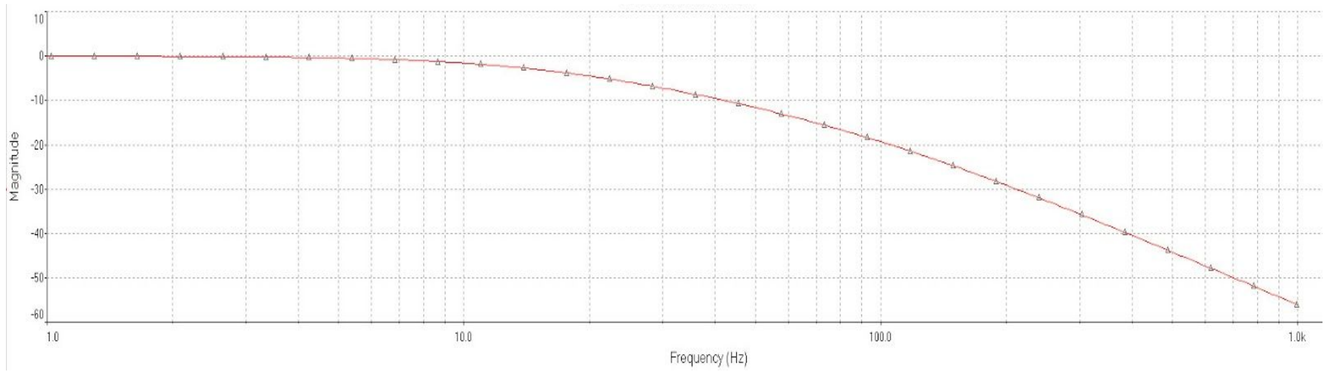


Figure 14: The Bode Plot for the circuit in Figure 12

4.1.1 The Fourth Order Active Butterworth Filter

A butterworth filter is a filter designed to be as flat as possible in the passband and have a smooth roll-off in the stop band. To achieve this the poles of a filter are manipulated. A butterworth filter has poles at even spacing from the imaginary (j) axis, with a fourth order butterworth having four poles spaced at 45° starting at the j axis. To calculate component values two equations are used to give filter components the proper ratio to place the filter poles in the correct locations.

$$tan\theta = \sqrt{\frac{C1}{C2} - 1}$$

Equation B: Determining the capacitor ratio in a butterworth filter where Theta is the angle from the real axis to the pole (Figure 15).

$$2\pi F_b = \frac{1}{\sqrt{R_1 R_2} * \sqrt{C_1 C_2}}$$

Equation C: Once a Capacitance ratio is established a value can be chosen for C1 and C2, and the value of R1 and R2 can be calculated for a specified break frequency (Fb). Note: If R1=R2, as is common practice, the $\sqrt{R1 * R2}$ *term simplifies to R.*

A fourth order butterworth is comprised of two second order butterworths each with its own set of poles. Equations B and C are used to place the poles of a two pole butterworth system. A Fourth order butterworth would use Equation B twice to find the two capacitance ratios, once with a Theta of 22.5° and again with a Theta of 67.5° (Figure 15). Using Equation B for these values of Theta result in a capacitance ratio of C1:C2=1.7157 (θ =22.5°) and C3:C4=6.828 (θ =67.5°) Plugging these ratios into Equation C results in R1 and R2 values of 3,678,470 Ohms when C2=1nF (θ =22.5°) and R3 and R4 values of 2,030460 Ohms when C4=1nF (θ =67.5°) . These values can be scaled by changing the size of the capacitor. A capacitor size increase by an order of magnitude results in an order of magnitude reduction for the resistance values.

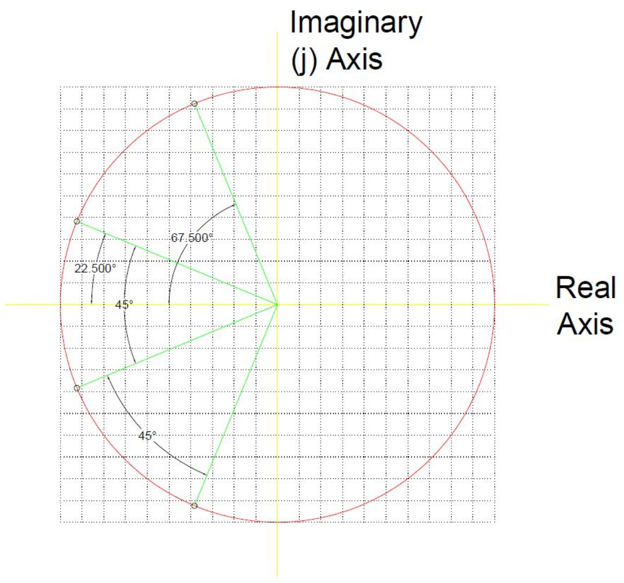


Figure 15: The locations of the poles for a fourth order Butterworth filter.

For a break frequency of 40 Hz very large capacitors would be required to bring the resistors into a reasonable range for use in a filter. This was not the ideal solution, as the capacitors would need to be bipolar, requiring a custom order of very expensive components to create the filter.

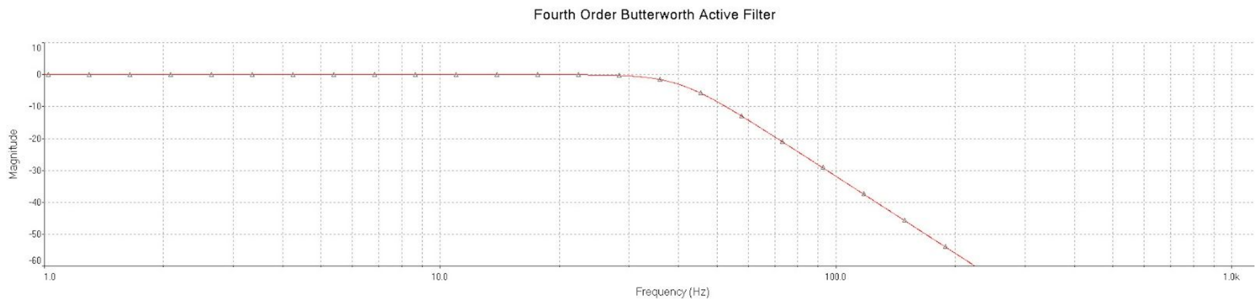


Figure 16: The Bode Plot of the fourth order Butterworth filter.

4.1.2 Fourth Order Butterworth Filter With Gain

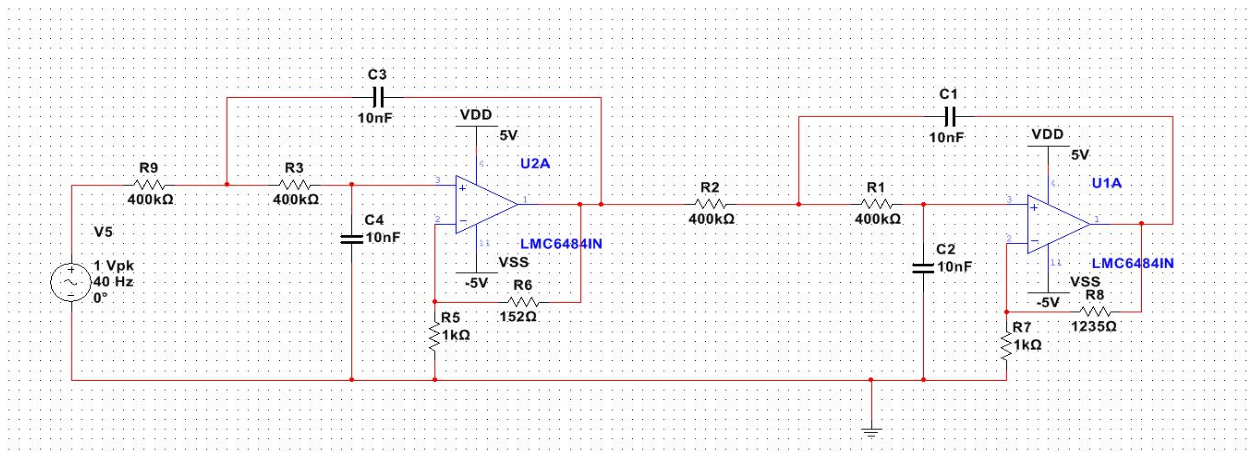


Figure 17: A fourth order Butterworth filter with gain.

Another way to manipulate the poles of the butterworth filter is to use gain in the filter while keeping the R and C values constant. This allows for more reasonable values of R and C to be selected, as well as removing the ratio restrictions required by a unity gain butterworth filter. With gain a fourth order butterworth can be reduced from Equation B and C to Equation D below, for R and C values, and Equation E for pole manipulation.

$$F_b = \frac{1}{2\pi RC}$$

Equation D: Determining resistance and capacitance values for a low pass filter.'

$$K = 3 - \frac{2}{\sqrt{1 + \tan^2 \theta}}$$

Equation E: Determining the gain of the filter (K) for proper pole placement.

Solving Equation E for θ=22.5 and θ=67.5 results in K=1.15224 and K=2.23463. These gains can be achieved by solving Equation F for the ratio of R1:R2.

$$K = 1 + \left(\frac{R_1}{R_2}\right)$$

Equation F: Calculating the gain of a non-inverting Op Amp

Solving Equation F for each K value obtained results in R2/R1=.15224 for K=1.15224, and R3/R4=1.23463 for K=2.23463. Choosing R1=R3=1kOhm results in R2=152.25 Ohm and R4=1235 Ohm.

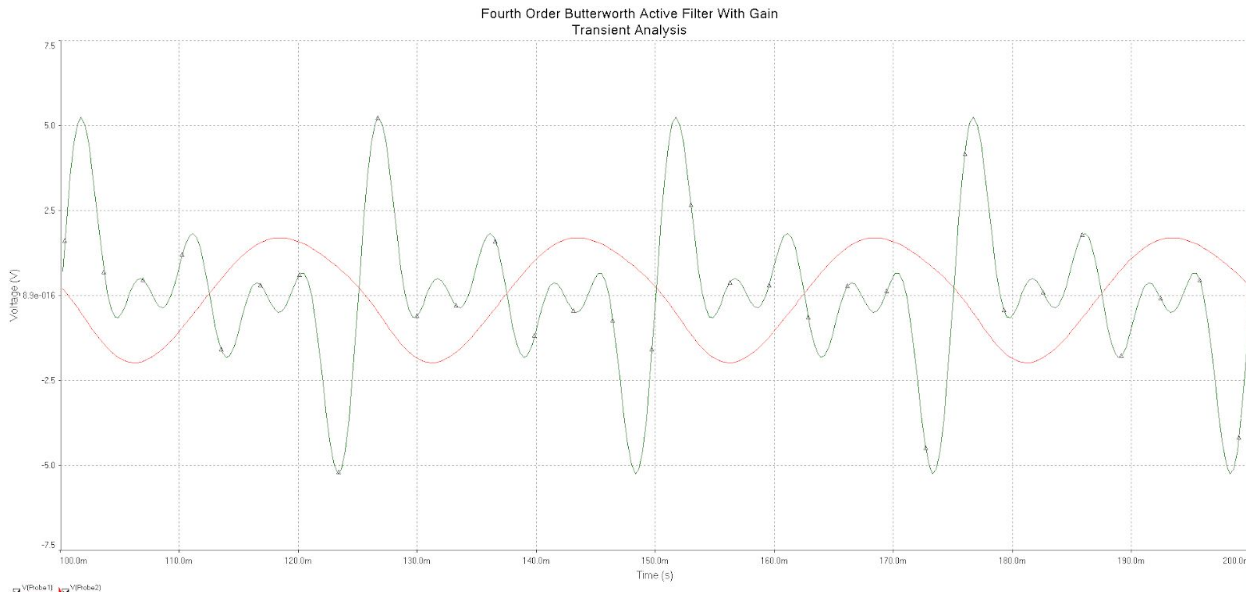


Figure 17b: Transient analysis in Multisim of the filter in Figure 17. Note:Red represents filter output, Green represents fundamental frequency of 40 Hz plus harmonics 1 through 4 using amplitudes that roughly replicate those of a guitar string.

5. Part Decisions

5.1 FSRs

Initially, the OPEN-SMART Round Film Force Sensitive Resistor was chosen due to its similarity to the force sensitive resistors used in the [year] ukulele MQP, as the research they did into the controller was compelling and comprehensive, and in practice proved to be a very successful solution. The Open-Smart FSR has a resistance range of 1k Ohm at 50 Newtons, up to a resistance of greater than 1MOhm under no load conditions. The FSR requires a supply voltage of no greater than 5.5 Volts, and has a sensing area of 9mm. Costwise, this FSR was also the ideal choice, at $3.21 per piece, half the cost of other FSRs on the market. Response time was quick, at less than 10 ms. Overall this FSR was the optimum choice for replicating the control module from the ukulele MQP. Unfortunately, this specific FSR was unavailable due to long lead times in delivery, so a different FSR had to be selected. The Sparkfun Force Sensitive resistor was selected because it closely matched the specifications of the original FSR. The sensing area was a bit smaller, at 4mm, but aside from that, the Sparkfun FSR met all requirements. The FSR taken from the uke MQP measured between 100k Ohm and 300k Ohm while just letting a finger rest, and would drop to around 50k Ohm when being pressed with gentle pressure. Heavy pressure would bring the resistance down to almost 20k Ohms. The FSR ordered for this project measured between 200k Ohm and 600K Ohm while letting a finger rest, and would drop to around 100K Ohm with light pressure, down to around 20k Ohms with heavy pressure

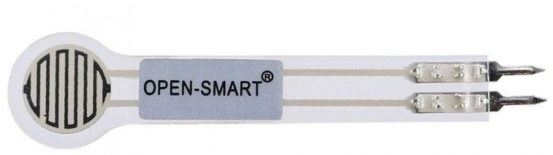


Figure 18: The Force Sensitive Resistor used. https://www.dx.com/p/open-smart-round-film-force-sensitive-resistor-50n-5kg-fsr-sensor-477803?tc=USD&ta=US&gclid=CjwKCAjw39reBRBJEiwAO1m0ORryxvQ_sfpvfE5YW8RLOjwPErUYBcBT7kz-y6485VBLQxJdX8J8OBoCM0IQAvD_BwE#.W9crFWhKhPY

5.2 Zero crossing detection circuit

Several options were considered to determine the frequency of the guitar signal. The first option was peak detection, but this was quickly eliminated due to the harmonic and decaying nature of the bass signal. This would require the circuit to be able to determine the difference between a peak as part of the main frequency, noise, or a harmonic. As time progressed this would become increasingly difficult due to the varying peak voltage over time. The second option considered was detecting the frequency based on zero crossings. This would be easier than peak detection because it intrinsically eliminated any reliance on the voltage of the signal. A filter would be used to reduce the harmonic components of the signal, providing a clean sine waveform at the fundamental frequency.

A common method of zero detection is an optocoupler. An optocoupler is a device that uses optical components to couple two physically unconnected circuits. The particular optocoupler chosen was the H11AA1 Phototransistor Output Optocoupler, which uses a pair of LEDs and a phototransistor to create a pulse pattern in sync with the zero crossings of an input signal. This works by having the LEDs in parallel, with the bias directions going opposite ways. This way, if the input voltage is positive, the forward facing LED is lit, and the phototransistor output remains low. If the input voltage is negative, the reverse oriented LED is lit, and the phototransistor remains low. The phototransistor only goes high when both LEDs are off, which occurs when the input voltage falls below the bias voltage for both LED. The phototransistor output looks like Figure 19 for a regular sine waveform. This pulse pattern helps remove the effects of the overlying harmonics that would be able to make it through the filtering system. The optocoupler output would be fed into the ADC of the microcontroller, where the time between high outputs would be measured using the built in timers, and then inverted and doubled to find the frequency of the original sine wave input to the optocoupler.

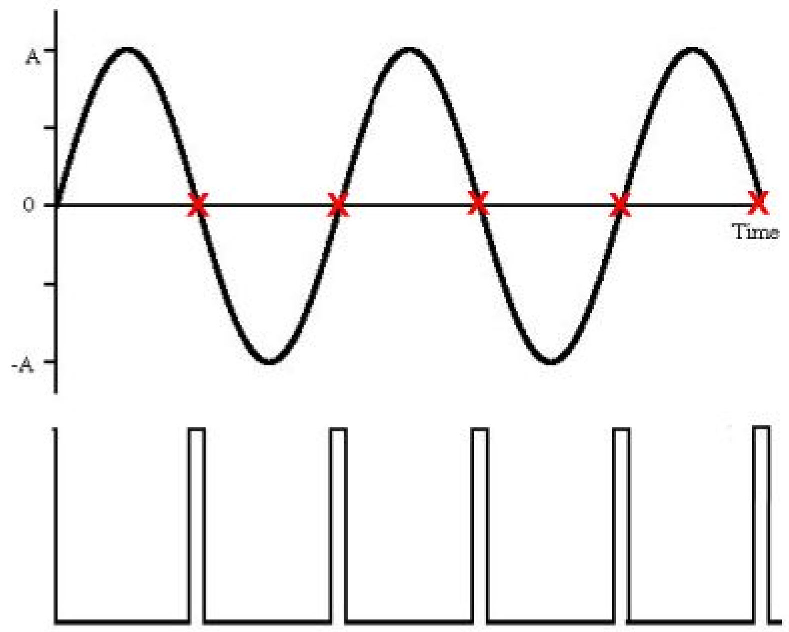


Figure 19: The output of an optocoupler(bottom) with a sine wave input(top).
http://www.bristolwatch.com/ele2/zero_crossing.htm

5.3 Motor

The motor chosen was the DF Robot GB37Y3530-12V-251R 12V Geared Motor, a motor intended for moderate sized robotics use, capable of [torque stats] and up to 251 RPM at the output shaft. The motor also included a built in Hall style encoder, and the ability to be driven forward or reverse, just by switching the polarity of the drive voltage. The project required a bidirectional DC motor to control the tension/tuning mechanism, which needed to be able to rotate one direction to increase tension, and the opposite direction to relieve tension.

The GB37Y3530 was the largest motor considered, and despite its size, provides all the needed functions that the smaller, weaker motors could not, and at a reasonable price tag of only $29. The student working on this project already had one of these motors in his possession, and bench testing proved that the GB37Y3530 would fit the bill.

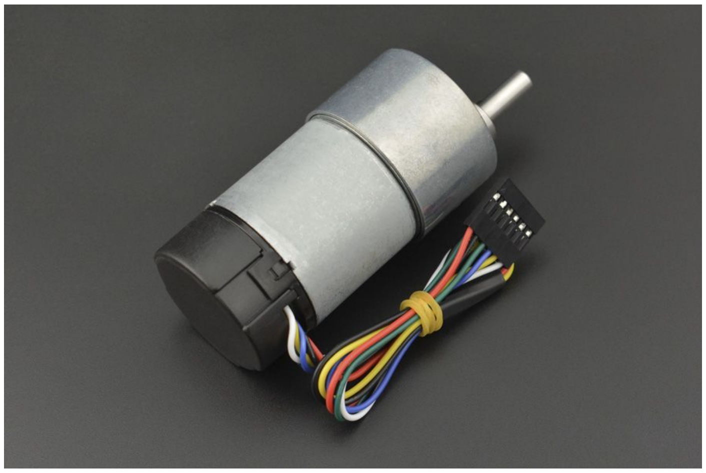


Figure 20: The DF Robot GB37Y3530-12V-251R 12V Geared Motor

https://www.dfrobot.com/product-634.html#.UWE4PNb8n5-

5.4 Solenoids

The solenoids were selected for their push-pull action and relatively small size. Arm extension reach and applied force were not crucial, as both these attributes can be compensated by attaching a small armatur or lever to the solenoid shaft. A small 12V solenoid was selected from Adafruit.

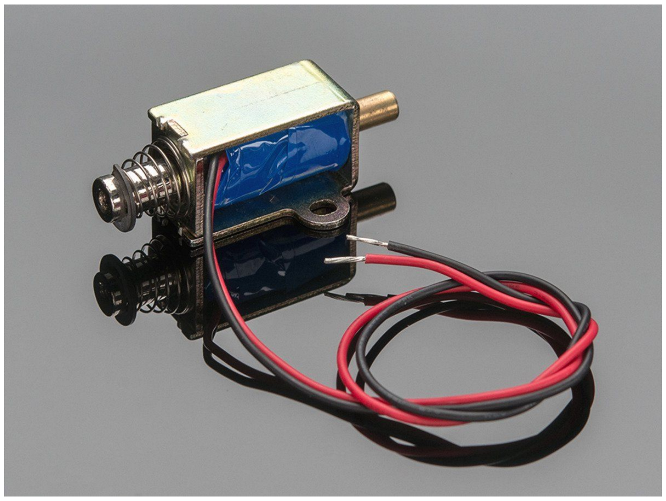


Figure 21: The Adafruit solenoid selected.

https://www.adafruit.com/product/412?gclid=Cj0KCQiA6JjgBRDbARIsANfu58HIesd3vAYudBA5NwwbrWoej147Qi7ktL5fmmsqCA62efOFltTmCvsaAmAQEALw_wcB

5.5 Microcontroller

Arduino microcontrollers were the main brand considered for this project, based on their ease of use, low price, and wide variety of microcontrollers and development boards to accompany the chips. Several other brands were considered, including the BeagleBone Black development board, but these were all eliminated early on. Four Arduino boards were considered for use in the finished product, the Uno R-3, the Due, the Leonardo, and the Mega.

5.5.1 The Uno R-3

The Uno is a small board, and a popular choice for small projects. A strong contender, offering 14 5V digital I/O pins and 6 analog inputs, the Uno was an appealing choice. Costing an average of $24, the Uno was a solid choice.

5.5.2 The Due

The Arduino Due is a lower voltage board, operating at 3.3V on the digital I/O pins. 54 Digital I/O pins and 12 analog inputs made the Due more than capable to handle all the inputs and outputs the prototype or full product would need. Unfortunately, the 3.3V restriction removed the Due from consideration, as the filter unit could put out a voltage up to 5 volts, potentially damaging the board.

5.5.3 The Leonardo

The Leonardo, another 5V board, offered a total of 20 digital I/O pins and 12 analog pins, more than the Uno, and the ability to handle the higher voltages the Due could not. This board only cost $20, making it ideal for the prototype as designed at the time, but not allowing for any room to expand if new designs evolved..

5.5.4 The Mega

The Arduino Mega is a larger version of the Uno. boasting 54 digital I/O pins and 16 analog inputs, this board provided more than enough inputs and outputs. A 5V operating voltage meant that this board could handle the necessary voltage from the filter output. At $40 this board was the most expensive option considered, but te large number of inputs would allow the same board to be used to operate the final product as well as the prototype.

Figure 22: The Arduino Mega.

5.6 Power system

A 24V AC to DC laptop charger was used to supply power to the circuits. The 24V DC was stepped down using a series of voltage regulators to provide the appropriate voltage to each branch of the circuit. A 12V regulator supplied power to the motor, solenoids and +/-5V regulators, and the +/-5V regulators supplied power to the filter ICs and user input FSR array.

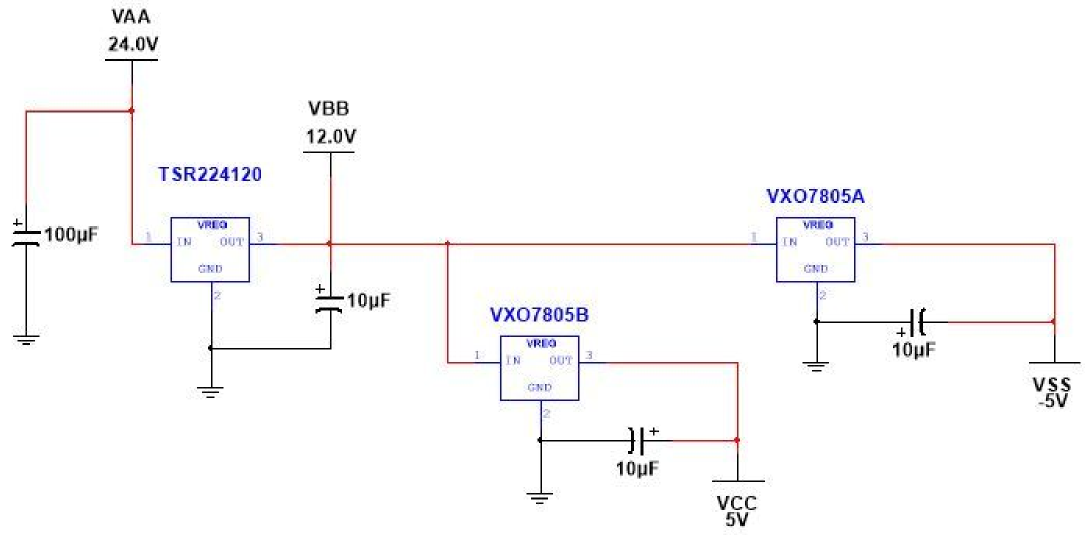


Figure 23: The power rail design for the circuit.

6. Coding

6.1 User Input

User input from the left hand is received through a series of voltage dividers hooked to individual arduino input pins, and collectively connected to a single interrupt pin (Figure 24). This common connection, with the individual input pins isolated by diodes, allows the left hand to react as soon as a user provides a change in input. Using the arduino attachInterrupt() function to assign a rising edge trigger to the interrupt, a latch method was constructed for the left (fretting) hand. This method allows users to release pressure on the left hand while playing, only requiring the user to interact with the controller when a different note is needed. Upon triggering this interrupt the system reads the value from the assigned individual pins for each input of the left hand. Multiple inputs are ignored and the lowest of the selected frequencies is chosen (Figure 25), as a bass cannot play two notes on the same string at the same time. This value is then converted into the desired frequency and compared to the frequency of the guitar input.

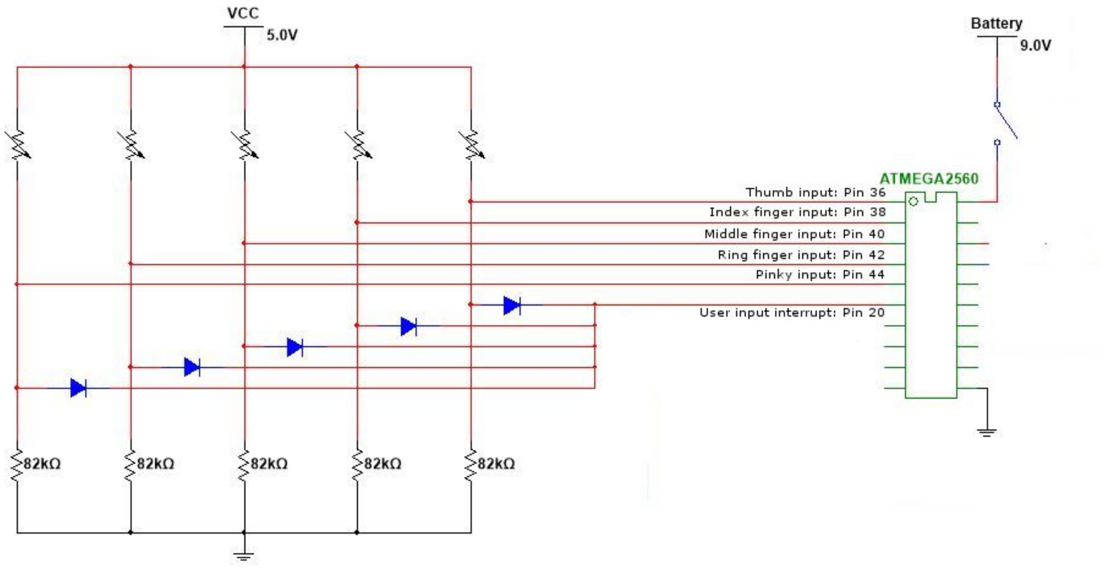

VCC
5.0V
Battery
9.0V
ATMEGA2560
Thumb input: Pin 36
Index finger input: Pin 38
Middle finger input: Pin 40
Ring finger input: Pin 42
Pinky input: Pin 44
User input interrupt: Pin 20
82kΩ
82kΩ
82kΩ
82kΩ
82kΩ

Figure 24: Left hand input.

User input from the right hand is directly connected to another interrupt pin, and an interrupt is generated both rising and falling edges of the signal. The right hand input is not a latching input, the user needs to maintain pressure on the FSR to sustain the note. This decision was made based on the playing style of the bass. Often the notes are short and quick, especially in genres such as rock or pop. At higher tempos, or faster paced basslines, a latching input would require far more effort to operate. The single input from the right hand provides the signal actuating both the hammer and the damper. After actuation the hammer is retracted automatically, while the damper will stay lifted until the user releases pressure from the FSR.

Input	Possible combinations (Binary concatenation)	Programmed reaction
Thumb	1xxxx	Selects E (41.2 Hz)
Index Finger	01xxx	Selects F (43.65 Hz)
Middle Finger	001xx	Selects F# (46.25 Hz)
Ring Finger	0001x	Selects G (49 Hz)
Pinky	00001	Selects G# (51.91 Hz)

Figure 25: The programmed responses to user input (left hand).

6.2 Frequency detection

Frequency is detected by counting the zero crossings of the guitar signal and comparing the amount of time that has passed. It is well know that:

$$Frequency = \frac{1}{Period}$$

So a frequency of 40 Hz, or forty cycles in a second, takes 1/40 of a second to complete one cycle. Each period contains two zero crossings in a sine wave, resulting in 80 zero crossings per second for a 40 Hz signal. The arduino microcontroller counts the number of zero crossings, and after a set number of crossings, checks how much time has passed between the first and the last crossing. These two values are all that is required to find the frequency. Before we discuss the math in detail, first we must discuss the set up of the timer and the optocoupler in this application.

6.2.1 Optocoupler

The optocoupler generates a pulse pattern at twice the frequency of the original signal, and is read by an interrupt pin on the arduino board, ensuring that a pulse is not missed due to polling. The use of interrupts also allows more accurate frequency readings, as the Interrupt Service Routine (ISR) is called the instant a pulse is received. As the math depends on the timer, the less extraneous time that passes between receiving the input and calculation, the more accurate the detected frequency will be.

6.2.2 Timer

The arduino timer used in the frequency calculation operation is a sixteen bit timer, set to a period of 1 second. The one second time, allows the frequency to be calculated in Hz. A second sixteen bit timer is used to time the hammer, measuring an interval of .1 seconds before retracting the hammer after a strike. This delay gives the solenoid enough time to fully extend and strike the string, and retract quickly, before the hammer can cause interference in the vibration of the string.

6.3 Frequency Calculation

Two options were considered for the frequency calculation equation, the first being how many zero crossings pass in a given timeframe, utilizing a set length of time to find the frequency, and counting the number of pulses detected during that time frame, giving the equation

$$Frequency = \left(\frac{timer\ max}{timer\ count}\right) * \left(\frac{pulses}{2}\right)$$

Equation G: Frequency Calculation Equation

In code this was written as :

```
void count() {
  if (TCNT1 == counter) { //every [counter] seconds, calculate frequency
     frequency = ((optocount * 62500) / (2*TCNT1)); //calculates frequency
based on cycles over set time
    TCNT1 = 0; //set timer count to zero
    optocount = 0;
  }
}
```

The readings from this method were somewhat irregular, and did not provide an accurate enough reading of the frequency. The cause of this was the small space of time between the conditional logic being met and the actual event that checked if the logic had been met.

A second method was devised, using the pulses from the optocoupler as the reference, and the time passed as the variable. The same math was used as Equation G. Instead of waiting for a polled check of the conditional logic, the polling was moved into the ISR for the optocoupler pulses, meaning every time a pulse was recorded the software checks if it is ready to calculate frequency, and if it is, does so immediately. This method reacts faster than the previous design, requiring less time as the frequency increases. The new code reads as below:

```
void count() {
  optocount ++;
  if (optocount == counter) { //every ten pulses, checks frequency
    cycles = (optocount / 2); //calculates number of cycles (currently default
5)
    frequency = ((cycles * 62500) / TCNT1); //calculates frequency based on
amount of time passed during cycles
    timercount = TCNT1; //store time passed since start of detection
    TCNT1 = 0; //set timer count to zero
    optocount = 0;
  }
}
```

6.4 Tuning logic

The logic used to calculate if the bass is in tune consists of a series of simple if statements, found below, each setting a series of variable to convey the tuning state. First the desired frequency and the detected frequency are compared to find the greater value, this series of statements sets the ‘greater’ or ‘lesser’ flags, which control the motor direction. Then the difference between the desired and detected frequencies is compared to a threshold variable to determine if the note is tuned, and if it is tuned motor rotation is stopped, and the tuned flag is set, meaning that as long as the user does not select a new note, the bass will not react to detected variation in the note. Variations as mentioned before can be caused by plucking, striking or bending the string, all of which are inherent bass playing and musical techniques that the bass should not interfere with once tuned.

```
void tune() { //full tuning function
```

```
  if (frequency >= freqcomp) {
    greater = 1; //sets greater logic flag
    lesser = 0;
  }
  if (frequency < freqcomp) {
    lesser = 1; //sets lesser logic flag
    greater = 0;
  }
  if (tuned || (frequency < 30)) { //stops freq below 30hz from being accepted
as a tuning
    check = 1; //test and debug variable
    digitalWrite(tuneup, LOW);  //stop motor
    digitalWrite(tunedown, LOW);
  }
  if (((frequency - freqcomp) <= tuningthresh) && greater) {
    tuned = 1; //nothing to do here
    check = 6; //test and debug variable
  }
  else if (((freqcomp - frequency) <= tuningthresh) && lesser) {
    tuned = 1; //nothing to do here
    check = 7; //test and debug variable
  }
  else if (frequency < freqcomp) {
    tuned = 0;
    check = 2; //test and debug variable
    uptune(); //call the uptuning function
  }
  else if (frequency > freqcomp) {
    tuned = 0;
    check = 3; //test and debug variable
    downtune(); //call the down tuning function
  }
}
```

6.5 Tuning threshold

The tuning threshold is the acceptable variation between the detected frequency and the desired frequency. If this threshold were zero the bass would oscillate between slightly too high of tuning and slightly too low, leading to unplayable behavior where the pitch is always varying. While this is necessary too large a difference will lead to the bass being out of tune. To find the max threshold the difference in Hz between the lowest two notes was found, and divided in half, giving the approximate division between the note being “tuned but not exactly perfect” and “out of tune”. The difference between E and F is 2.45 Hz (43.65-41.2), and half of that is 1.225 Hz, meaning that any note that is more than 1.225 Hz off from its target frequency is out of tune enough to be easily detected by the human ear. As a compromise between tuning stability and tuning accuracy, 1 Hz was chosen to be the threshold level. As frequency increases so will the accuracy of tuning.

7. Implementation

7.1 Assembly

For full integration testing a single string bass prototype (Figures 5, 5a, 5b) was created out of pine boards. This prototype included precision bass style, split coil humbucker, and standard tuning pegs. The bridge and nut were excluded from the build as the brick made these components obsolete. A second tuning peg was added in place of the bridge, and was modified to be driven by a motor attached to the prototype. The hammer and damper solenoids were mounted near the headstock and pickups respectively.

7.2 Testing

Each block of the system was tested individually before being connected to the other blocks to help isolate problems, as well as discover a baseline for each blocks regular operation. The filter unit was tested using a lab bench frequency generator first, to ensure the correct passband and attenuation of different frequency signals. Then an actual bass was hooked up and tested, to see the output from the filter. The signal quickly dropped below the threshold needed to operate the optocoupler, so an amplifier with a gain of 11 was added before the filter stage. This extended the frequency detection time available from less than a second to up to three seconds of active detection time, more than enough for the frequency calculations to take place once the string settled from any initial abnormalities caused by striking or plucking the string. The optocoupler output pattern for a known, constant frequency was measured on an oscilloscope to determine the accuracy of the filter and optocoupler unit. A frequency of 40 Hz generated by a Tektronix frequency generator was routinely read as 40.34 Hz an error of .34Hz (0.85%) assuming the function generator is exactly accurate. According to manufacturer datasheets, these generators may have a variation of 1% or less. A second test was performed, this time using a Fender jazz bass to generate the input waveform. When tuned to E1 the frequency was detected to be 41.3 Hz well within the acceptable range of frequencies. Each discrete note frequency was tested in this manner to ensure that accuracy did not vary with frequency, and no variations were found.

The power system was simply tested to ensure it was providing the proper voltage levels to the proper components, 12 volts to the motor controller, 5 volt regulators, and both solenoids, and 5 volts to the positive rail of the chip power supply, and negative 5 volts to the negative rail.

Motor control was tested using an oscilloscope attached to the two motor outputs on the L293 Chip. A comparison frequency was chosen, and an actual frequency was provided. By varying the given frequency so it varied beyond the tuning threshold in both directions provided the necessary input to stimulate all motor reactions. When the frequency was greater than the desired frequency one motor output went high, and if the frequency was lower than desired, the other mor output would go high. Problems were encountered when the motor was attached, as spinning in a clockwise direction drew too much current and caused a droop in the voltage supply rails, leading to missed optocoupler readings and an incorrect frequency reading.

User input was tested via the built-in serial communication on the Arduino Mega. The values of the variables used to store the left hand user input were read out via serial every time through the loop, and whenever a FSR was pressed, the change could be seen in the change of output. The interrupt worked perfectly, and whenever an FSR was pressed the values would update instantly. Two inputs at the same time could not be replicated because of the speed that it updated at.

The solenoids provided the most challenge when trying to implement and test their operation. A second timer had to be instantiated to time the retraction of the hammer. When and where to call the hammer to retract also provided some difficulties at first, as it was tried to retract it automatically, but the delay was not long enough and the solenoid did not receive enough voltage to activate. Therefore a timer had to be instantiated to control the hammer's retraction. A clock divider of 100 was used on a timer that could count a full second between interrupts, and every ten cycles it calls the hammer to retract, providing a tenth of a second activation time for the hammer, when activated.

-

8. Obstacles

The motor causes the entire power system to 'droop' when turning, resulting in missed opto pulses, causing a low frequency reading, resulting in temporary rotation in the opposite direction when the detected frequency is above the desired frequency. Tuning to a lower frequency did not cause a power droop, and the frequency readings would remain normal.

The motor chosen for this project turned out to have a high current draw during its initial spin up from standstill, or when reversing rotation direction. This leads to the motor driver chip slowly burning out and becoming unable to provide the required current to the motor. A solution to this would be to build a higher current full H Bridge driver out of MOSFETs and modify the code to run a full H bridge rather than a commercially designed driver chip. This being the ideal solution, the motor selection could be improved and a motor that draws less current could be used, as this motor can draw almost all the current the power supply can provide when reversing direction or simply starting up. If a better motor could not be found, a higher current power supply would need to be designed and is necessary to solve the power rail droop.

To solve the excessive current draw when the motor would start spinning, four 1 Ohm, 10 Watt power resistors were put in series with the motor, limiting the current to two amps, and successfully reserving enough power to operate the remainder of the system. This peak draw of 2 Amps was still too much for a single H bridge in the L293 motor controller, so an entire chip, consisting of 2 full H bridge drivers, was wired in parallel to drive the motor, and lower the current draw on each individual H bridge.

The budget also proved an obstacle for this project, as WPI funds MQP projects by the member, regardless of project. A team of three would have had much more headroom in the project budget. Music equipment is relatively pricey, and can quickly eat away at a budget.

9. Final Design

Figure 26 shows the final circuit schematic for the prototype design. The design consists of five distinct segments connected to the microcontroller. The five segments that make up the design are the motor control unit, the power system, solenoid control circuitry, user input circuitry, and signal processing unit.

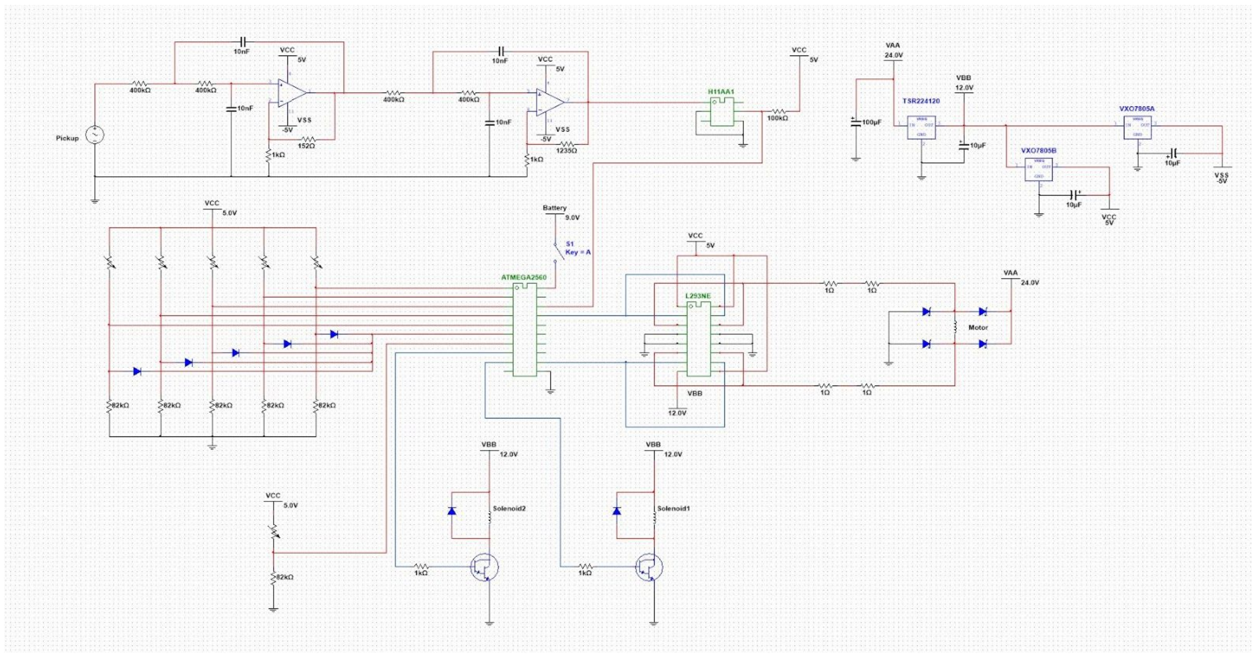


Figure 26: The full circuit schematic.

Red wires are inputs to the microcontroller, blue wires represent outputs to control circuitry, and black wires represent connections to ground.

9.1 Motor Control Circuit

The motor control circuit used an L293 quad half H bridge motor driver wired in parallel to prevent burning out the chip during use.The chip is powered by two different sources, VCC at 5V for chip power, and chip enable, and VBB at 12V to provide power to the motor. The motor itself had to be wired with both current limiting power resistors, to keep current draw within the limits of the supply, and protection diodes to prevent inductive spikes from damaging the circuit. The control circuit works by connecting a set of BJTs to either power or ground based on the input from the two control pins attached to the Arduino. Low, high, or floating signals at both input pins will result in no reaction from the controller. A high signal on only one of the control pins will configure the chip connecting a one of the two outputs on a single side of the chip to VBB, and connecting the other to ground. Wired in parallel, a high signal causes each half of the chip to react in the same manner, doubling the behavior of a single H bridge, and decreasing current load on the chip.

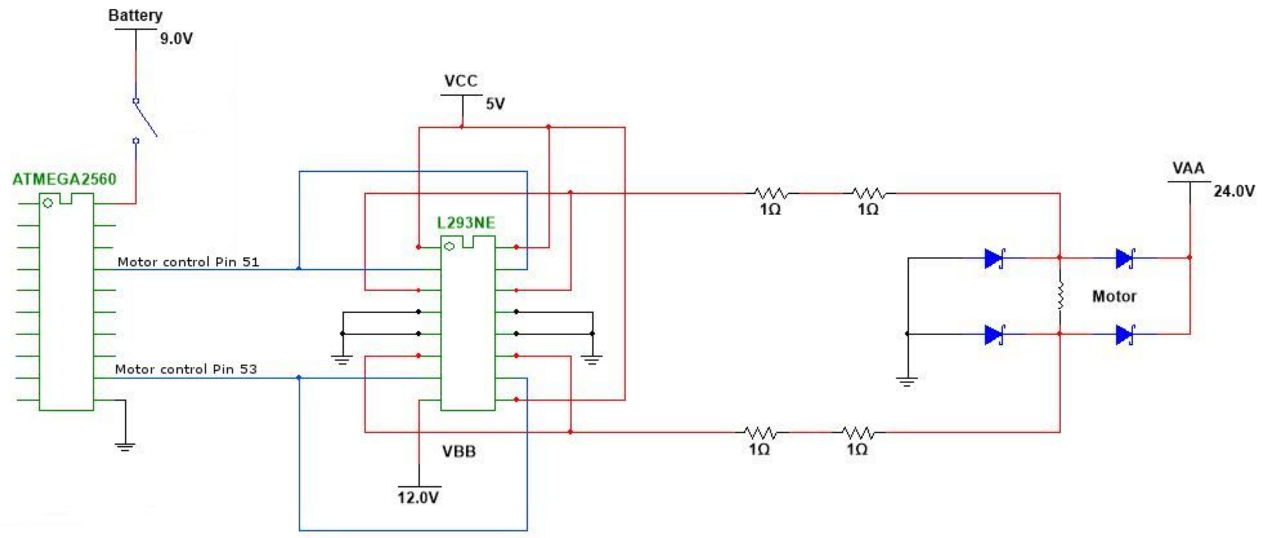


Figure 27: The motor control circuit.

9.2 Power System

The power system is comprised of a modified laptop charger and three voltage regulators. The laptop charger provides up to 2 Amps of 24V DC to a 12 volt regulator. The 12 volt regulator provides power to two +/-5 volt regulators that make up the power rails for the IC chips and the FSR array. The 12 volt regulator also provides power to the motor and solenoids. Filter capacitors were added to help reduce noise and maintain a level voltage supply.

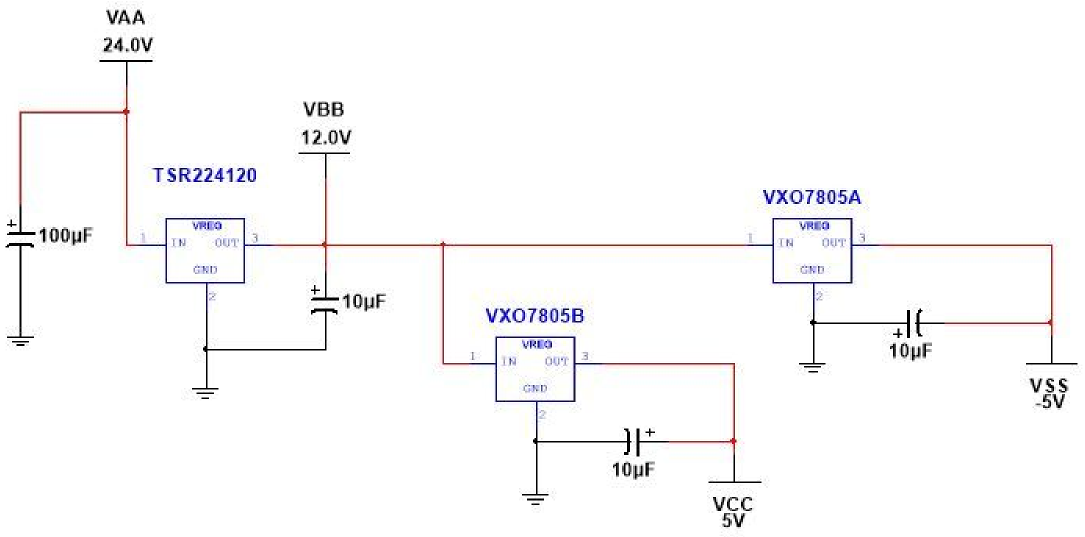


Figure 28: The power supply circuit.

9.3 Solenoid Control

The solenoids are controlled with a simple transistor setup. A signal from the Arduino turns on the transistor, allowing current to flow through the solenoid, creating a magnetic field, activating the solenoid. A flyback diode is wired in parallel to each solenoid to disperse inductive spikes.

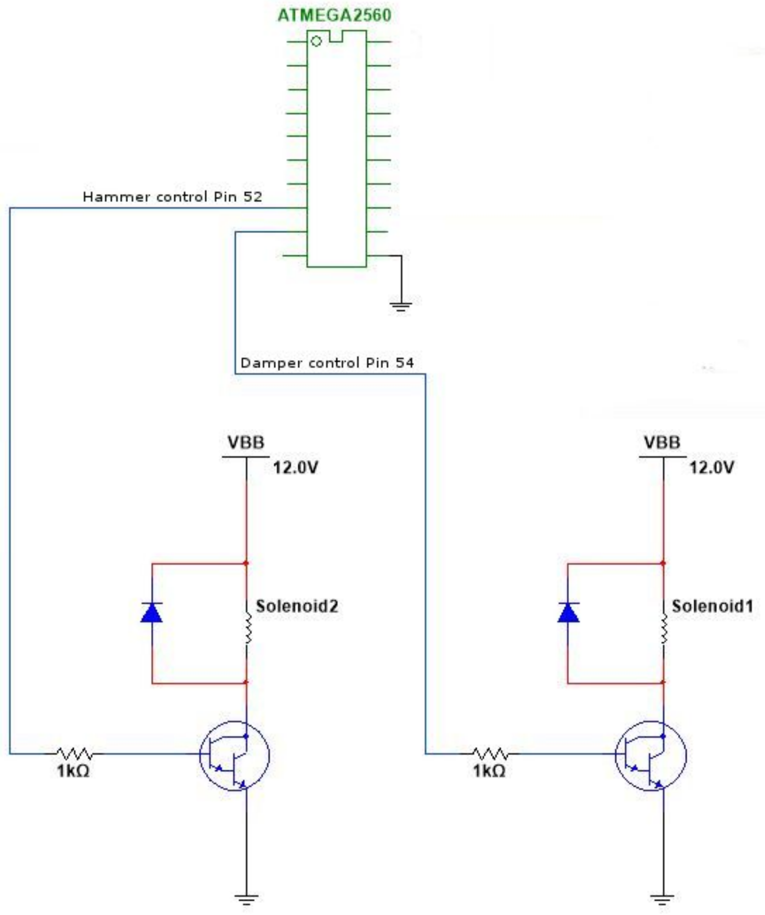


Figure 29: Solenoid control circuit.

9.4 User Input

User input utilizes six force sensitive resistors as buttons. The left hand array consists of five voltage dividers, with the signal being sent in parallel to both an universal interrupt pin, and a unique input pin on the Arduino. Diodes isolate the unique inputs from the universal signal, where all five signals are sent to the Arduino, one indistinguishable from the other. The right hand input (not pictured) consists of a single voltage divider, of the same design as the left hand array voltage dividers, but with just one output to the Arduino.

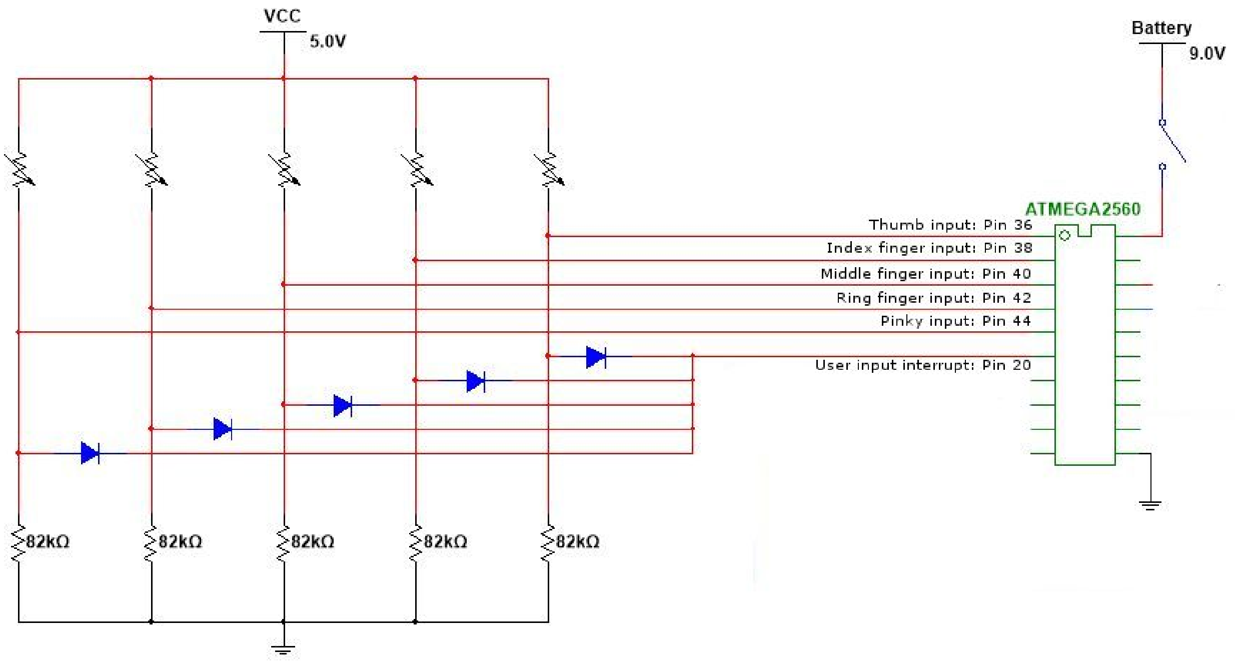

VCC
5.0V
Battery
9.0V
ATMEGA2560
Thumb input: Pin 36
Index finger input: Pin 38
Middle finger input: Pin 40
Ring finger input: Pin 42
Pinky input: Pin 44
User input interrupt: Pin 20
82kΩ
82kΩ
82kΩ
82kΩ
82kΩ

Figure 30: Left hand input circuit.

9.5 Signal Processing

The signal processing unit is made up of the filter and the optocoupler. The filter is designed as a fourth order butterworth filter with gain, having a break frequency of 40 Hz. This filter strips a guitar note down to its fundamental frequency and feeds it into an optocoupler. The optocoupler is a phototransistor controlled by a pair of opposing biased LEDs. These LEDs are powered by the output of the filter, resulting in the transistor being on the majority of the time, giving a low output reading to the Arduino. Only when both LEDs are off does the transistor turn off, and read high. This signal provides a pulse to the Arduino at each zero crossing.

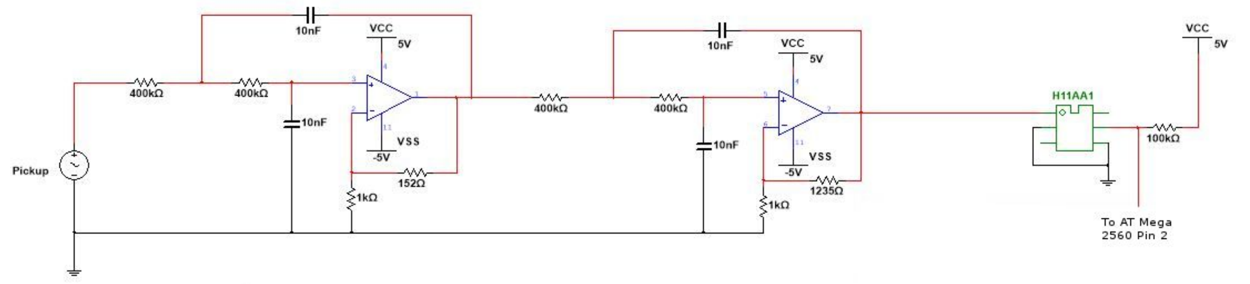


Figure 31: Fourth order filter and zero detector.

9.6 Prototype

The physical prototype was constructed using a 1” X 3” pine board (actual dimension 2.5”X.75”). Approximately 5’ in length of the board was used to assemble the prototype. A set of precision bass pickups were mounted 8” OC from one of the tuning pegs. On each end tuning pegs were mounted, with one being attached to the motor used to control string tension. The other tuning peg was left as is to allow string tension adjustments outside of the motor’s operation, if necessary. Refer to Figures 5, 5a, and 5b for visual reference. The motor shaft was coupled to the tuning peg using ABS plastic, and gorilla glue. This plastic proved to be too soft, and quickly lost its hold, and glue was not strong enough to hold it. While the glue held, the motor was operated, and used to tension the string from below 40 Hz up to 52 Hz, the full range of the string being tested. Beyond the shaft coupler failure, the entire prototype held up.

10. Discussion

10.1 Changes made

The unit gain buffers initially included on the FSR array for user input were found to be unnecessary. Without the buffers, user input was not bogged down when connected to the microcontroller. This elimination reduced the power draw on the 5V supplies by a large margin, removing two thirds of the IC chips on the 5 V rails. Removal of the unity gain buffers also reduced the circuit footprint, furthering the goal of a device that can fit on a bass guitar. Through testing, it was found that an amplification stage was needed before the filter, to ensure that the bass signal stayed at a high enough peak to peak value to be read by the optocoupler for a period of time long enough to interpret a nice clean frequency. The initial design called for the built in Hall encoder to be used to track motor movement and direction. This was never implemented due to motor issues, as it could not be tested or properly programmed to react with the rest of the system.

A current limiter was installed on the motor, reducing the speed of the motor, but also relieving an excessive current draw when the motor would start spinning. This change could be reversed in a future design if a more powerful AC to DC supply was used or built, capable of handling 8 Amps or more.

Two L293 H bridges were wired in parallel to control the motor, rather than the originally planned single H bridge design. This change was implemented due to the high peak current loads on motor startup. While the L293 can handle a peak of 2 Amps, the motor would draw at this level too much, and would wear out the L293, causing chip failure in a matter of hours of use. Two L293 H bridges can provide up to 4 Amp peak current, well above the 2 Amp limited draw of the motor.

10.2 Success and Failures

10.2.1 Success

The frequency detection and filtering was very accurate, and surprisingly fast, detecting frequency in 1/8th of a second. I was very satisfied with the way it operated and how it could detect frequency, even from an actual guitar being played.

The user interface was designed to mimic that of a bass, with input from one finger causing the same result in the prototype, as would normal playing of a bass. For example, fretting the first fret would utilize the index finger, and if the user gives input with their index finger, the prototype will tune as if you were fretting at the first fret. This input system also latched as much as possible, allowing a single button press to choose a note, and only requiring a drawn out press to sustain a note.

The fourth order filter worked exactly as it was designed, providing a nice clean sine form wave to the zero crossing detection unit.

Modifying Arduino's timers is not an easy task, as it is not well documented, nor the intended use of the arduino platform. The arduino platform is aimed more at high level functionality of calling functions, rather than the low level hardware manipulation end of embedded systems. I was successful in manipulating them, and creating timers that could be used for my specific task.

The motor was successfully implemented with a current limiter, removing the power rail droop, and still operating at a good speed.

10.2.2 Failures

The power system was not the best designed branch of this project, and definitely contributed to some of the problems experienced. In addition to the poorly designed supplies, the motor chosen, while wholly suitable for the task, lacked in depth documentation, leading to power supply issues, and drawing more current than recommended though the motor driver, causing some chips to burn out. The third failure of the project was the hammer. The solenoid chosen just did not have a large enough throw distance to cause a large vibration in the string, and while it could cause an audible vibration, there is a lot more volume that could be utilized if a better hammer were introduced.

The prototype could not be operated with the motor altering tension on the string for very long, as the plastic coupler was too soft to utilize the initial pressure fit couple, and quickly stripped out, requiring a bonding compound to hold the coupler to the tuning peg. Unfortunately a strong enough bonding compound could not be found.

10.3 Possible revisions

To a future team that may have an interest in working on this project, modifying it, or furthering the research and design already done, I suggest that a more robust power system be designed, namely something capable of handling a current well in excess of estimated current draw. The current 2A power supply is not enough to run a powerful motor for this application without interfering with the other circuitry. A different motor controller would be another revision to include, as the one selected is only borderline capable of running the selected motor without failing.

Less critical revisions should include a hammer solenoid with greater mass and throw length, to improve the effect of the hammer on the string and maximize the initial input signal to the filter. The current hammer is too weak for its intended purpose. A Schmitt trigger could be used in place of an optocoupler, if a rising (or falling) edge interrupt were attached to the signal. In this configuration a Schmitt trigger would work just as well as an optocoupler, and may simplify the design, and maximize the time that a signal can be read for.

The coupler, designed and 3D printed to allow the motor to control the tuning peg, was made out of ABS plastic, proved to be too soft for the torques applied, and quickly stripped out the tuning peg end. A harder material, such as aluminum, would be advisable for a better functioning shaft coupler.

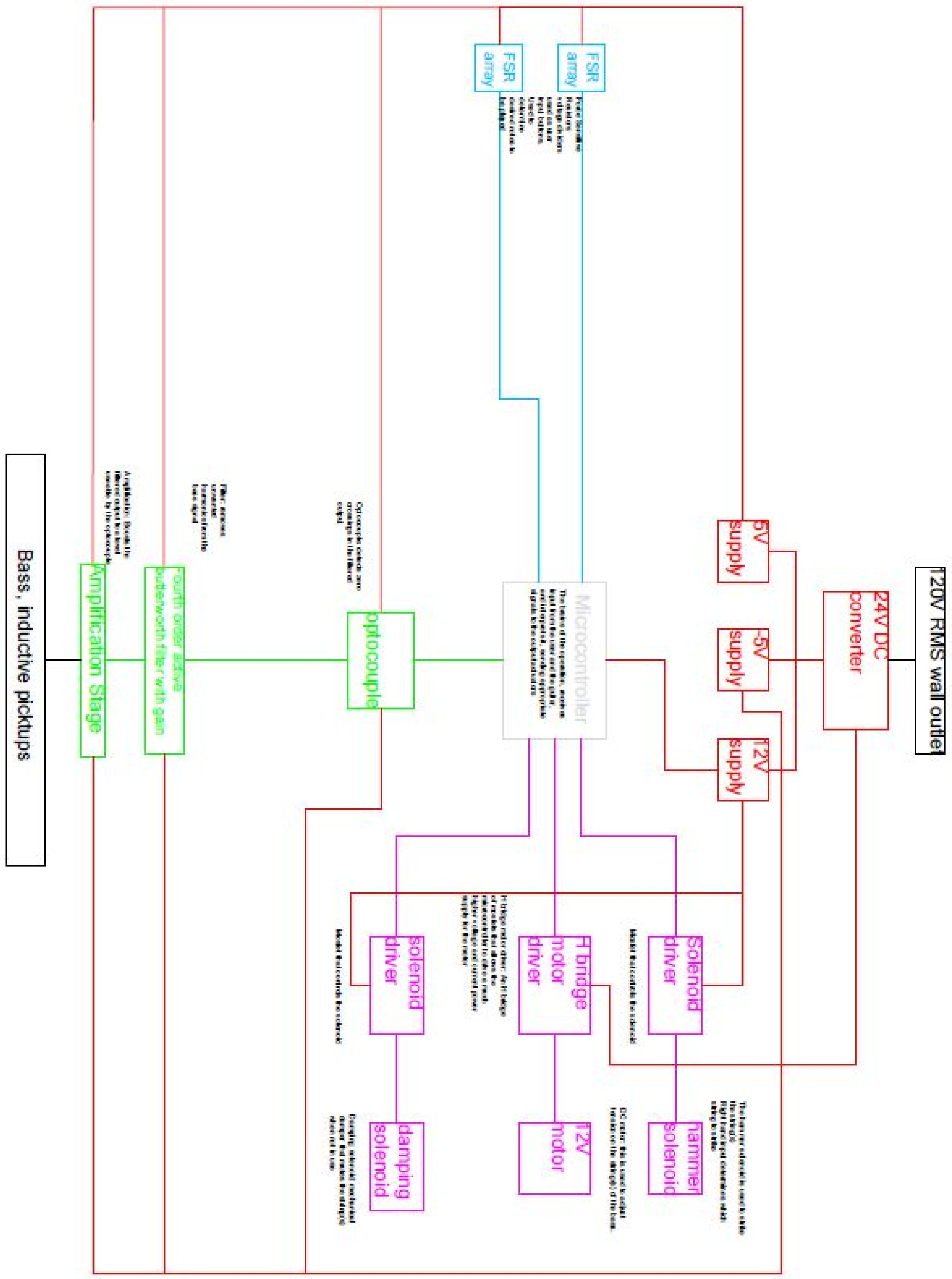
11. Appendix

11.1 Appendix A: Bill Of Materials

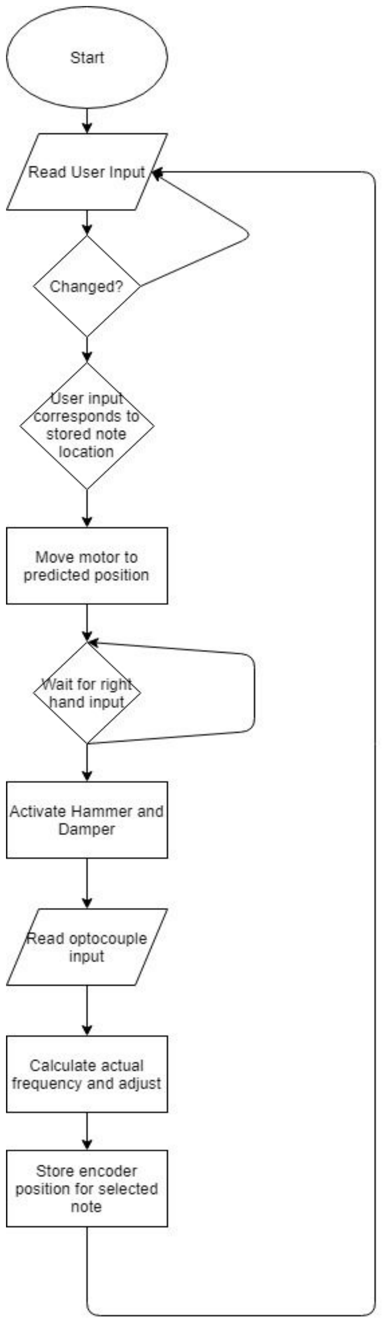
Part Number	Part Name	Quantity	Price (each)	Use
001	Force Sensitive Resistor	6	$2.75	User Input
002	82K Ohm Resistor	6	$0.10	User Input
003	1N4004 Diode	7	$0.22	User Input, Solenoid Control
004	TIP 102 BJT	2	$0.77	Solenoid Control
005	1k Ohm Resistor	4	$0.10	Solenoid Control, Filter
006	200K Ohm Resistor	8	$0.10	Filter
007	1240 Ohm Resistor	1	$0.10	Filter
008	150 Ohm Resistor	1	$0.10	Filter
009	200 Ohm Resistor	1	$0.10	Filter
010	2k Ohm Resistor	1	$0.10	Filter
011	10KpF Capacitor	4	$2.39	Filter
012	LMC6484IN	1	$2.82	Filter
013	H11AA1	1	$1.31	Optocoupler
014	100k Ohm Resistor	1	$0.10	Optocoupler
015	L293NE	1	$3.68	Motor Driver
016	SB2H100	4	$0.44	Motor Protection Diodes

017	GB37Y3530	1	$29.00	Motor
018	VXO7805-500	2	$2.47	+/-5V Regulator
019	TSR 2-24120	1	$12.50	12V Regulator
020	24V 2A Laptop Charger	1	$8.99	AC to 24V DC Converter
021	10 Microfarad Capacitor	2	$0.45	Power System
022	100 Microfarad Capacitor	1	$0.47	Power System
023	AT Mega 2560	1	$38.50	Microcontroller
024	Tuning Pegs	2	$11.99	Prototype
025	Inductive Pickups	1	$20.00	Prototype
026	12V Solenoid	2	$7.50	Hammer, Damper Solenoids
027	1 Ohm 10W Power Resistor	4		Motor Current Limiter
028	3-D Printed Motor Shaft Coupler	1	$0.30	Shaft Coupler
029	Foam	1	$0.00	Damping Material
030	Pine Board	1	$8.00	Prototype
031	Guitar Jack	1	$2.39	Input
032	Bass String	1	$5.00	Prototype
033	Breadboard	2	$2.72	Breadboard

11.2 Appendix B: Block Diagram



11.3 Appendix C: Software Flow Chart


Start
Read User Input
Changed?
User input corresponds to stored note location
Move motor to predicted position
Wait for right hand input
Activate Hammer and Damper
Read optocouple input
Calculate actual frequency and adjust
Store encoder position for selected note

11.4 Appendix D: Filter values

R=400k Ohm, C=10nF, Fb=40 Hz

Pitch	Attenuation
41 Hz (Fundamental frequency)	~0
82 Hz (Second Harmonic)	~40dB

R=300k Ohm, C=10nF, Fb=53 Hz

Pitch	Attenuation
55 Hz (Fundamental frequency)	~0
110 Hz (Second Harmonic)	~40dB

R=200kOhm, C=10nF, fb=80 Hz

Pitch	Attenuation
74 Hz (Fundamental frequency)	~0
148 Hz (Second Harmonic)	~40dB

R=163k Ohm, C=10nF, fb=98 Hz

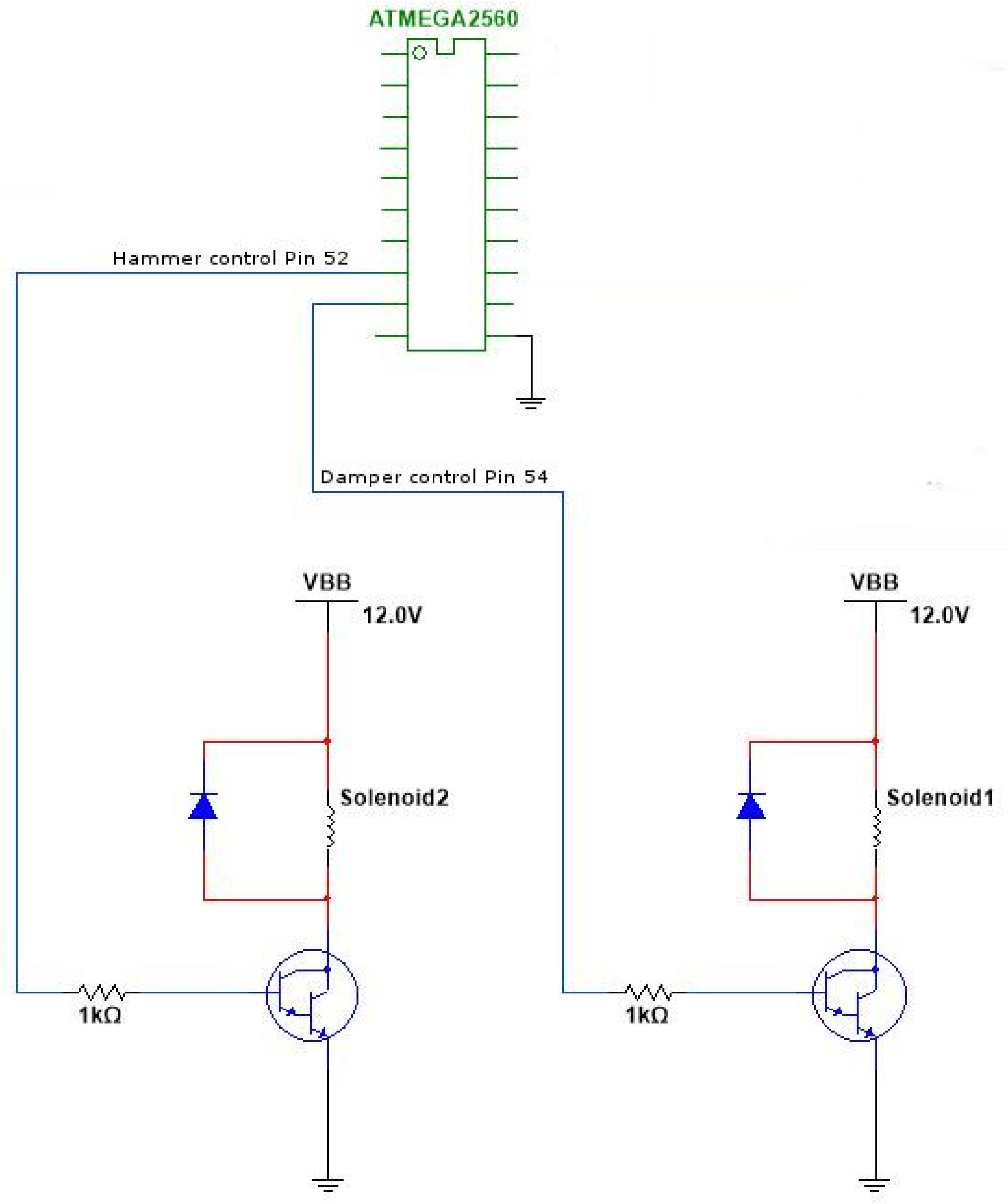
Pitch	Attenuation
98 Hz (Fundamental frequency)	~0
196 Hz (Second Harmonic)	~40dB

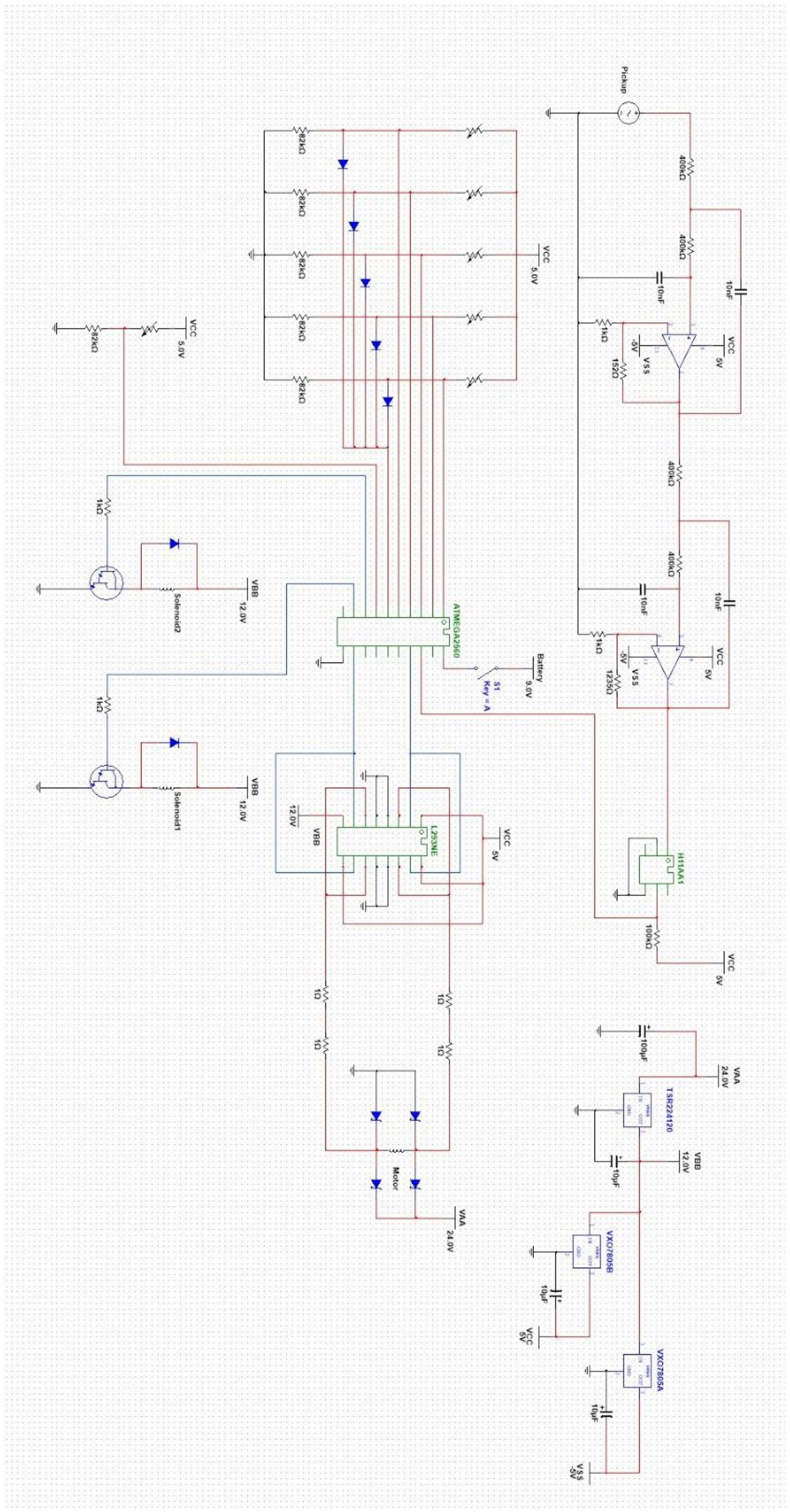
11.5 Appendix E: Full Code

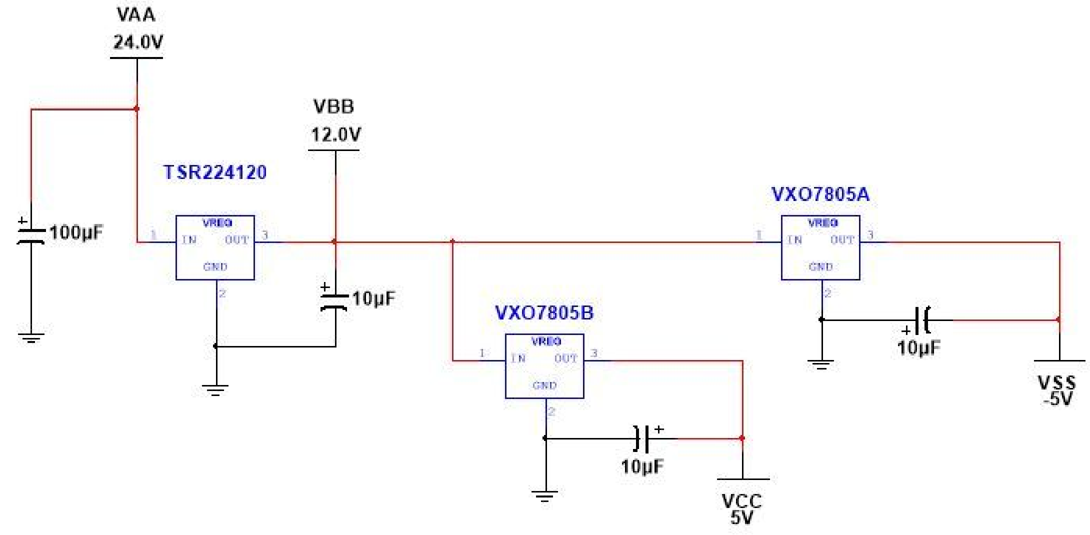
```
/*
   Read user input : working
   React to user input : working
   Calculate frequency: working
   Move motor forward: logic working, successfully runs motor
   Move motor backward: logic working, power rail trouble operating
   motor, causing controller burnout
   Logically determine motor direction needed: working
   Activate hammer on command: working
   Activate damper on command: working
   Adjust frequency: logic working
*/

//VARIABLES
int timerdiv = 1; //clock divider for timer1
int timer5div = 100; //clock divider for timer5
volatile byte flip = LOW; //damper flag
int hampin =  52; //hammer pin
int damppin = 54; //damper pin
int UIpin1 = 38; //finger 1 pin (index)
int UIpin2 = 40; //finger 2 pin (middle)
int UIpin3 = 42; //finger 3 pin (ring)
int UIpin4 = 44; //finger 4 pin (pinky)
int UIpin5 = 36; //thumb pin
int UIpin6 = 21; //plucking input pin
int tunepin = 20; //left hand input flag (20)
int tuneup = 51; //motor positive rotation pin
int tunedown = 53; //motor negative rotation pin
int val1 = 0; //finger 1 input
int val2 = 0; //finger 2 input
int val3 = 0; //finger 3 input
int val4 = 0; //finger 4 input
int val5 = 0; //thumb input
int optopin = 2; //optocoupler input to mcu
int optocount = 0; //optocoupler pulse count
int counter = 10; //MUST BE EVEN NUMBER
//number of zero crossings to determine frequency
//more cycles will give more accuracy but slower recognition
float cycles = 0; //used in math for frequency
volatile float frequency = 0; //frequency found
volatile float freqcomp = 0; //frequency comparison variable
float tuningthresh = 1; //acceptable variation of tuned note to
perfect pitch
int tuned = 0; //tuned flag
int greater = 0;//freq comparison var
int lesser = 0; //freq comparison var
int d = 0; //hammer check var
int valflag = 0; //flag indicating user input, for use in predictive
model
int oldval = 0; //previous valflag value, for use in predictive model
int timercount = 0;

//TEST VARIABLES
//int times = 0; //timer count var
int check = 0; //test var
volatile byte active = LOW; //test var

//FUNCTIONS

void count() {
  optocount ++;
  if (optocount == counter) { //every five cycles, checks frequency
```

```
    cycles = (optocount / 2); //calculates number of cycles (currently
default 5)
    frequency = ((cycles * 62500) / TCNT1); //calculates frequency
based on amount of time passed during cycles
    timercount = TCNT1;
    TCNT1 = 0; //set timer count to zero
    optocount = 0;

  }
}


void play() { //hammer and damper activate
  flip = !flip; //raise/lower damper, also damper flag
  if (flip == 1) {
    check = 1;
  }
  digitalWrite(damppin, flip);//actvate damper
  digitalWrite(hampin, flip); //activate hammer
}

void readinput() {
  //read user input and interpret as a  note
  oldval = valflag;
  val1 = digitalRead(UIpin1);
  val2 = digitalRead(UIpin2);
  val3 = digitalRead(UIpin3);
  val4 = digitalRead(UIpin4);
  val5 = digitalRead(UIpin5);
  tuned = 0;
  //priority given to lower notes by arbitrary decision
  if (val5) { //{E = 41.2Hz}
    valflag = 0;
    freqcomp = 41.20;
  }
  if (val1) { //{F = 43.65Hz}
    valflag = 1;
    freqcomp = 43.65;
  }
  if (val2) { //{F# = 46.25Hz}
    valflag = 2;
    freqcomp = 46.25;
  }
  if (val3) { //{G = 49Hz}
    valflag = 3;
    freqcomp = 49.00;
  }
  if (val4) { //{G# = 51.91Hz}
    valflag = 4;
    freqcomp = 51.91;
  }
  /* //Predictive model
      if (valflag > oldval) {
     //tighten string;
    }
    else if (valflag < oldval) {
     //loosen string;
    }
    else if (valflag > oldval) {
     //do nothing
    }*/
}

void uptune() { //runs motor to tighten string
  digitalWrite(tunedown, LOW);
  check = 4;
  digitalWrite(tuneup, HIGH);
}

void downtune() { //runs motor to loosen string (turns left when
freqcomp>actual)
  digitalWrite(tuneup, LOW);
  check = 5;
  digitalWrite(tunedown, HIGH);
}

void tune() { //full tuning function
  if (frequency >= freqcomp) {
    greater = 1;
    lesser = 0;
  }
  if (frequency < freqcomp) {
    lesser = 1;
    greater = 0;
  }
  if (tuned || (frequency < 30)) { //stops freq below 30hz from being
accepted as a tuning
    check = 1;
    digitalWrite(tuneup, LOW);
    digitalWrite(tunedown, LOW);
  }
  if (((frequency - freqcomp) <= tuningthresh) && greater) {
    tuned = 1; //nothing to do here
    check = 6;
  }
  else if (((freqcomp - frequency) <= tuningthresh) && lesser) {
    tuned = 1; //nothing to do here
    check = 7;
  }
  else if (frequency < freqcomp) {
    tuned = 0;
    check = 2;
    uptune();
  }
  else if (frequency > freqcomp) {
    tuned = 0;
    check = 3;
    downtune();
  }
}

void resetopto() { //watchdog to reset opto
  optocount = 0;
  frequency = 0;
}



void setup() {
  //TIMER 1 SETUP
  noInterrupts(); // disable all interrupts
  TCCR1A = 0; //clean timer 1 register A
  TCCR1B = 0; //register b
  TCNT1 = 0; //set timer count to zero
  OCR1A = 62500 / timerdiv; // 1 second interval divided by the
preselected divider
  TCCR1B |= (1 << WGM12); // CTC mode //TCCRx timer control register
  TCCR1B |= (1 << CS12); // 256 prescaler //TCCRx prescaler setting
  TIMSK1 |= (1 << OCIE1A); // enable timer compare interrupt //TIMSKx
interrupt mask register
  //TIMSKx interrupt enable for timer x

  //TIMER 2 SETUP
  TCCR5A = 0; //clean timer 1 register A
  TCCR5B = 0; //register b
  TCNT5 = 0; //set timer count to zero
  OCR5A = 62500 / timer5div; // 1 second interval divided by the
preselected divider
  TCCR5B |= (1 << WGM12); // CTC mode //TCCRx timer control register
  TCCR5B |= (1 << CS12); // 256 prescaler //TCCRx prescaler setting
  TIMSK5 |= (1 << OCIE1A); // enable timer compare interrupt //TIMSKx
interrupt mask register
  //TIMSKx interrupt enable for timer x
  interrupts(); // enable all interrupts


  //PIN SETTINGS
  pinMode(hampin, OUTPUT);
  pinMode(damppin, OUTPUT);
  pinMode (tuneup, OUTPUT);
  pinMode (tunedown, OUTPUT);
  pinMode(optopin, INPUT);
  pinMode(UIpin1, INPUT);
  pinMode(UIpin2, INPUT);
  pinMode(UIpin3, INPUT);
  pinMode(UIpin4, INPUT);
  pinMode(UIpin5, INPUT);
  pinMode(UIpin6, INPUT);

  //ATTACH INTERRUPTS
  attachInterrupt(digitalPinToInterrupt(optopin), count, RISING);
  attachInterrupt(digitalPinToInterrupt(tunepin), readinput, RISING);
  attachInterrupt(digitalPinToInterrupt(UIpin6), play, CHANGE);

  //SERIAL START
  Serial.begin(9600);
  Serial.println("Begin");
  Serial.println("detected frequency, target frequency, check, tuning
threshold, tuned, greater, lesser");
}

//Timer ISR
ISR(TIMER1_COMPA_vect) // timer compare interrupt service routine
{
  //watchdog style timer to stop rogue adjustments if bass is not
being played
  resetopto();
}

ISR(TIMER5_COMPA_vect) //Hammer reset
{
  if (d == 10) {
    d = 0;
    digitalWrite(hampin, LOW);
  }
  else {
    d++;
  }

}


void loop() {
  tune();
  //Serial.println("detected frequency");
  Serial.print(frequency);
  Serial.print(",");

  //Serial.println("target frequency");
  Serial.print(freqcomp);
  Serial.print(",");
  //Serial.println("check");
  Serial.print(check);
  Serial.print(",");
  //Serial.println("tuning threshold");
  Serial.print(tuningthresh);
  Serial.print(",");
  //Serial.println("tuned");
  Serial.print(tuned);
  Serial.print(",");
  //Serial.println("greater");
  Serial.print(greater);
  Serial.print(",");
  //Serial.println("lesser");
  Serial.println(lesser);
  //Serial.println("optopin");
  //Serial.println(optopin);


}
```

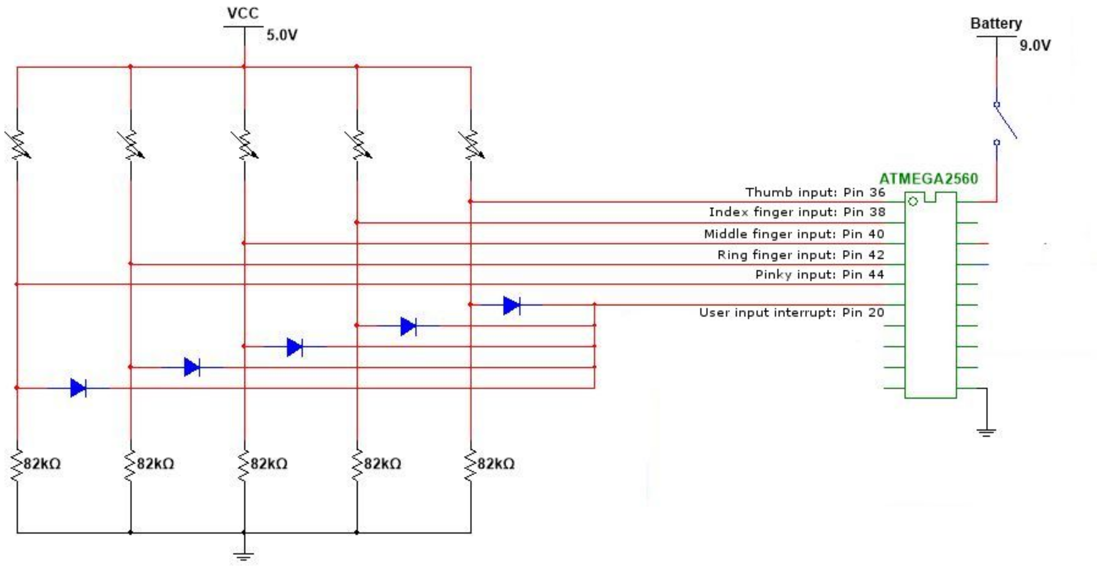
11.6 Appendix F: Circuit Diagrams



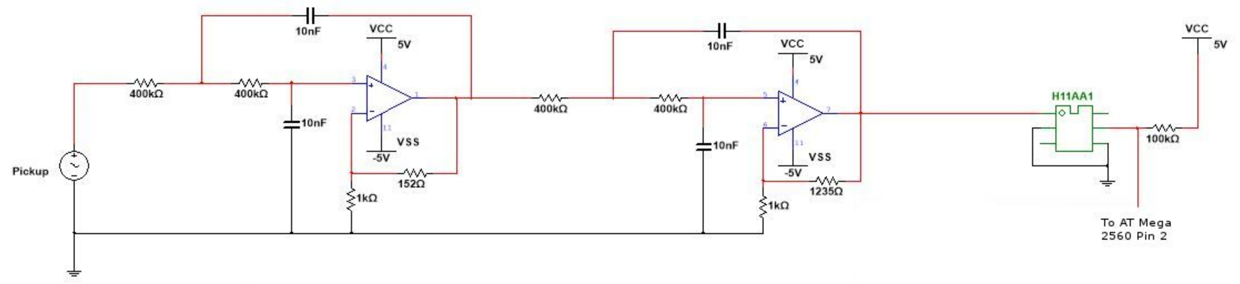

Pickup
400kΩ
400kΩ
10nF
10nF
1kΩ
VCC 5V
VSS -5V
152Ω
400kΩ
400kΩ
10nF
10nF
1kΩ
VCC 5V
VSS -5V
1235Ω
H11AA1
100kΩ
VCC 5V
VAA 24.0V
TSR224120
100µF
10µF
VBB 12.0V
VXO7805B
10µF
VCC 5V
VXO7805A
10µF
VSS -5V
VCC 5.0V
82kΩ
82kΩ
82kΩ
82kΩ
82kΩ
VCC 5.0V
82kΩ
1kΩ
1kΩ
Solenoid2
VBB 12.0V
Solenoid1
VBB 12.0V
ATMEGA2560
Battery 9.0V
S1
Key = A
L293NE
VCC 5V
VBB
12.0V
1Ω
1Ω
1Ω
1Ω
Motor
VAA 24.0V

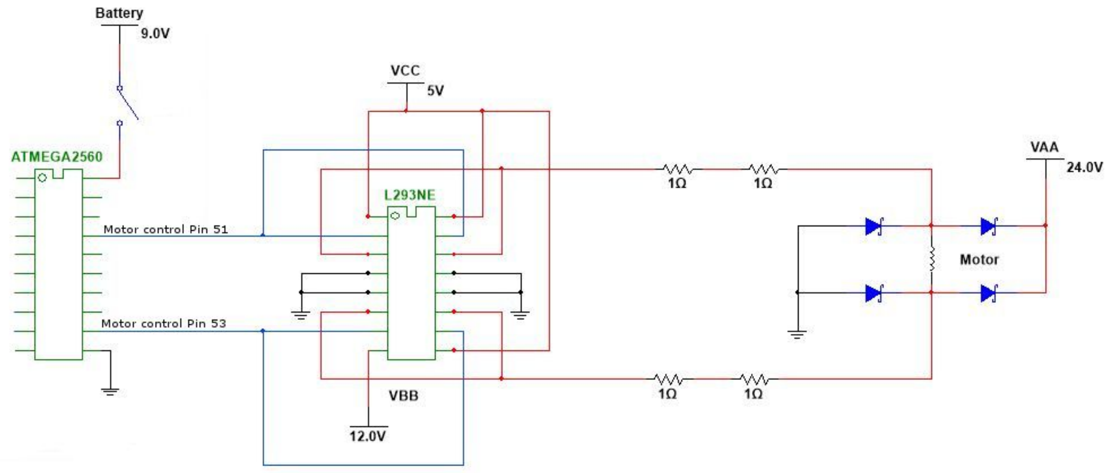


VAA
24.0V
VBB
12.0V
TSR224120
100µF
10µF
VXO7805A
VXO7805B
10µF
10µF
VSS
-5V
VCC
5V



VCC
5.0V
Battery
9.0V
ATMEGA2560
Thumb input: Pin 36
Index finger input: Pin 38
Middle finger input: Pin 40
Ring finger input: Pin 42
Pinky input: Pin 44
User input interrupt: Pin 20
82kΩ
82kΩ
82kΩ
82kΩ
82kΩ


10nF
VCC
5V
400kΩ
400kΩ
10nF
VSS
-5V
152Ω
1kΩ
Pickup
400kΩ
400kΩ
10nF
VCC
5V
10nF
VSS
-5V
1235Ω
1kΩ
H11AA1
VCC
5V
100kΩ
To AT Mega 2560 Pin 2


Battery
9.0V
ATMEGA2560
Motor control Pin 51
Motor control Pin 53
VCC
5V
L293NE
VBB
12.0V
1Ω
1Ω
1Ω
1Ω
VAA
24.0V
Motor

11.7 Appendix G: Terminology

Term	Definition
Standard Tuning	The accepted ‘normal’ tuning for a bass guitar.
Fret	Small pieces of wire embedded in the fretboard at regular intervals. Used to alter the pitch of a note.
Neck	Long thin piece of wood that connects to the body and contains the fretboard.
Body	Large slab of wood that houses the electronics, as well as the bridge and neck.
Bridge	One of two termination points for the strings.
Tuning Peg	Located on the headstock, second termination point for the strings, used to adjust tension in the string.
Headstock	Top end of the neck, mounting point for the tuning pegs.
Pickup	Part of the electronics of the guitar, this picks up the vibration of the string and converts it to an electric signal.
Tone Control	A capacitive low pass filter
E String (First string)	The thickest string on a standard 4 string bass, resonates at a frequency of 41.20 Hz (E_1)
A String (Second string)	The second thickest string on a standard 4 string bass, resonates at a frequency of 55 Hz (A_1)

D String (Third string)	The third thickest string on a standard 4 string bass, resonates at a frequency of 73.42 Hz (D_2)
G String (Fourth string)	The thinnest string on a standard 4 string bass, resonates at a frequency of 98 Hz (G_2)
Nut	A small piece of bone or plastic located at the top of the fretboard. Keeps the strings in place laterally.
Fretboard	The large slab of wood underneath the strings, containing the frets.
Fretting	The action of pressing down on a string to change the note.
Octave	The musical note of the same name, twice the frequency of the previous note.
Hammer-on	A playing technique where the musician presses down on the string quickly with the fretting hand, causing the string to vibrate upon impact with the fretboard. Allows notes to be played without picking or strumming.
Pull-off	A playing technique where the musician pulls the string with their fretting hand before releasing the note, adding tension which upon release, causes the string to vibrate. Allows notes to be played without picking or strumming.
Neck Profile	The cross sectional shape of the neck.
Break angle	The angle created by either the nut or the bridge. The bend in the string is what stops the rest of the string length from vibrating. The break angle, if

	not sharp enough, will cause strings to buzz.
Picking	A method of playing where a musician ‘picks’ individual notes using a guitar pick
Strumming	A method of playing where a musician plays multiple strings at once.
Optocoupler	A device used to couple two physically unconnected circuits through optical components
O.C.	On center, a measurement used to indicate distances between the center of two round or irregularly shaped objects.

11.8 Appendix H: Sources

[1]“Muscular dystrophy,” *Mayo Clinic*, 06-Feb-2018. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/muscular-dystrophy/symptoms-causes/syc-20375388. [Accessed: 09-Sep-2018].

[2] E. Lacroix, A. Sylvia, R. Yang, S. Arce, and K. Nazareth, “Assistive Aid for Playing the Ukulele by Persons with Duchenne Muscular Dystrophy,” 28-Apr-2016.

[3]“Muscular Dystrophy,” *Centers for Disease Control and Prevention*, 10-Apr-2018. [Online]. Available: https://www.cdc.gov/ncbddd/musculardystrophy/data.html. [Accessed: 26-Sep-2018].

[4]“Duchenne Muscular Dystrophy (DMD),” *Muscular Dystrophy Association*, 22-Jun-2018. [Online]. Available: https://www.mda.org/disease/duchenne-muscular-dystrophy. [Accessed: 26-Sep-2018].

[5] “Duchenne Muscular Dystrophy (DMD) - Causes/Inheritance,” *Muscular Dystrophy Association*, 31-Jan-2018. [Online]. Available: https://www.mda.org/disease/duchenne-muscular-dystrophy/causes-inheritance.

[6]C. Fitzpatrick, C. Barry, and C. Garvey, “Psychiatric Disorder Among Boys With Duchenne Muscular Dystrophy,” *Developmental Medicine & Child Neurology*, vol. 28, no. 5, pp. 589–595, 1986.

[7]H. Okobokekeimei, “People with Disabilities as Social Outcasts: Shifting the Perspective from Victim to Advocate,” *The Huffington Post*, 12-Oct-2013. [Online]. Available: https://www.huffingtonpost.com/helen-okobokekeimei/people-with-disabilities-_1_b_3744152.html. [Accessed: 27-Sep-2018].

[8]*American Pianist Brandon Myers | Inspirational Musician | Musicians with Disabilities | Can Do Musos*. [Online]. Available: http://www.candomusos.com/profile-brandon-myers.php. [Accessed: 27-Sep-2018].

[9]"American Guitarist Jay Harris | Inspirational Musician | Musicians with Disabilities | Can Do Musos," *American Pianist Brandon Myers | Inspirational Musician | Musicians with Disabilities | Can Do Musos*. [Online]. Available: http://www.candomusos.com/profile-jay-harris.php. [Accessed: 27-Sep-2018].

[10]"Welcome to Naxos Records," *Musical Instruments*. [Online]. Available: https://www.naxos.com/education/music_instruments.asp. [Accessed: 20-Sep-2018].

[11]"Getting Started," *Seymour Duncan*. [Online]. Available: https://www.seymourduncan.com/support-pickups-101/getting-started. [Accessed: 12-Oct-2018].

[12]P. Blecha, "Audiovox #736," *Vintage Guitar® magazine*, 11-Dec-2001. [Online]. Available: http://www.vintageguitar.com/1782/audiovox-736/. [Accessed: 04-Oct-2018].

[13]"List of Bass Guitar Manufacturers," *101 Basses*, Jan-2016. [Online]. Available: https://101basses.com/list-of-bass-guitar-manufacturers/. [Accessed: 04-Oct-2018].

[14]"Basses," *Bass Guitars | Guitar Center*. [Online]. Available: https://www.guitarcenter.com/Bass.gc?typeAheadRedirect=true. [Accessed: 15-Oct-2018].

[15]"EXL165 Nickel Wound Bass, Custom Light, 45-105, Long Scale," *D'Addario Strings :: Balanced Tension*. [Online]. Available: http://www.daddario.com/DADProductDetail.Page?ActiveID=3769&productid=138&productname=EXL165_Nickel_Wound_Bass__Custom_Light__45_105__Long_Scale. [Accessed: 07-Oct-2018].

[16]“Min-ETune™,” *Gibson Guitar*, 2013. [Online]. Available: http://www.gibson.com/Products/Min-ETune.aspx. [Accessed: 05-Oct-2018].

[17]“Gibson Robot Guitar: First Run Limited Edition,” *Gibson Guitar: Electric Guitar, Acoustic Guitar, Slash Guitars, How To Play Guitar Lessons, Baldwin Pianos, Amps, Strings, Stories and Music News*. [Online]. Available: http://archive.gibson.com/RobotGuitar/guitar.html. [Accessed: 09-Oct-2018].

[18]“Gibson Proudly Presents The Robot Guitar- A First Run Limited Edition,” *Gibson Guitar: Electric Guitar, Acoustic Guitar, Slash Guitars, How To Play Guitar Lessons, Baldwin Pianos, Amps, Strings, Stories and Music News*. [Online]. Available: http://archive.gibson.com/RobotGuitar/story.html. [Accessed: 09-Oct-2018].

[19]M. Parker, “Gibson's Min-Etune – the basics,” *MusicRadar*, 30-Nov-2012. [Online]. Available: https://www.musicradar.com/news/guitars/gibsons-min-etune-the-basics-567831. [Accessed: 09-Oct-2018].