WPI NECAMSID

Assistive Bass Guitar for Those Affected by Muscular Dystrophy

A Major Qualifying Project Report
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science by:

Chloe Adler-Mandile
Michael Altavilla
Candan Iuliano
Steven Lussier

Faculty Advisor:
Professor Stephen Bitar

Sponsored By:
The New England Center for Analog and Mixed Signal Design (NECAMSID)

Date: March 27, 2020

Abstract

Muscular Dystrophy is a group of disorders in which there is a progressive loss of muscle mass and strength. The most common form, Duchenne Muscular Dystrophy (DMD) tends to affect young males and has no cure. DMD causes a decrease in motor control over time, so those affected with talents in the arts may lose their ability to express themselves in this way. This project creates an assistive aid for those affected with DMD which allows them to play the bass guitar. Using a combination of motors, solenoid plungers, and a microcontroller, the user is able to interface with a tablet and create music.

Acknowledgments

We would like to give a special thanks to all who helped us during this MQP especially our team advisor, Professor Stephen Bitar, whose weekly meetings and insight we could not have completed this project without. For his wealth of knowledge regarding robotic instruments, we would like to thank Professor Scott Barton, who provided us with invaluable advice regarding tuning design. Additionally, for exceptional guidance in our printed circuit board design, we would like to thank PhD Candidate and Research Assistant Ian Costanzo. For his generosity of materials, knowledge, and time, we would also like to thank Electronics Technician III, William Appleyard.

Thank you to everyone who helped to make this project a reality, we couldn't have done it without you.

Table of Contents

Table of Authorship

Section	***Primary Author***	***Primary Editors***
Abstract	Candan Iuliano	Steven Lussier
Acknowledgments	Chloe Adler-Mandile	Michael Altavilla
Executive Summary	Chloe Adler-Mandile	Steven Lussier
1. Introduction	Steven Lussier	Candan Iuliano
2. Literature Summary		
2.1 NECAMSID Information	Steven Lussier	Candan Iuliano
2.2 The Arts and Disabilities	Steven Lussier	Chloe Adler-Mandile
2.3 Muscular Dystrophy	Steven Lussier	Chloe Adler-Mandile
2.4 Musical Background	Chloe Adler-Mandile	Candan Iuliano
2.5 Prior Art	Candan Iuliano	Michael Altavilla
2.6 Design Constraints	Candan Iuliano	Steven Lussier
3. Procedure		
3.1 Mission Statement	Steven Lussier	Chloe Adler-Mandile
3.2 Design Constraints	Steven Lussier	Candan Iuliano
3.3 General Architecture	Chloe Adler-Mandile	Michael Altavilla
3.3.1 Fretting Design	Candan Iuliano	Steven Lussier
3.3.2 Plucking Design	Candan Iuliano	Michael Altavilla
3.3.3 Tuning	Michael Altavilla	Steven Lussier
3.3.4 Inputs/Android App	Chloe Adler-Mandile	Steven Lussier
4. Methodology		
4.1 Design Objectives	Chloe Adler-Mandile	Steven Lussier
4.2 Design Constraints	Steven Lussier	Candan Iuliano
4.3 General Design Aspects	Candan Iuliano	Michael Altavilla
4.3.1 Hardware Inputs	Michael Altavilla	Candan Iuliano

4.3.2 Hardware Outputs	Candan Iuliano	Michael Altavilla
4.3.3 User Interface	Steven Lussier	Chloe Adler-Mandile
4.2.4 Signal Analysis	Chloe Adler-Mandile	Michael Altavilla
4.3.5 Tuning Hardware	Michael Altavilla	Steven Lussier
4.3.6 Additional Design Features	Michael Altavilla	Steven Lussier
4.4 Summary	Steven Lussier	Candan Iuliano
5. Realization and Results		
5.1 Finalized Design	Michael Altavilla	Steven Lussier
5.2 Testing Period		
5.2.1 Tuning Implementation	Candan Iuliano	Chloe Adler-Mandile
5.2.2 Mechanical Work	Michael Altavilla	Steven Lussier
5.2.3 Circuitry	Chloe Adler-Mandile	Steven Lussier
5.2.4 Mobile Application	Candan Iuliano	Chloe Adler-Mandile
5.2.5 Sound Testing	Steven Lussier	Candan Iuliano
5.2.6 Combined Module Testing	Chloe Adler-Mandile	Candan Iuliano
5.2.7 Overall Implementation	Steven Lussier	Michael Altavilla
6. Discussion		
6.1 Achievements	Steven Lussier	Michael Altavilla
6.2 Shortcomings	Candan Iuliano	Steven Lussier
6.3 Impact for Assistive Technology	Steven Lussier	Candan Iuliano
6.4 Manufacturing and Costs Evaluation	Michael Altavilla	Steven Lussier
6.5 Limitations	Chloe Adler-Mandile	Candan Iuliano
6.6 Future Work	Steven Lussier	Chloe Adler-Mandile
7. Conclusion	Chloe Adler-Mandile	Michael Altavilla
Appendices		
References		

Executive Summary

Art in all its forms, is a way for someone to express themselves and realize their identity. People with disabilities are prone to struggles with self esteem and identity. This project, sponsored by the New England Center for Analog and Mixed Signal Design (NECAMSID), aims to create an Assistive Bass Guitar in order to provide a creative outlet for people with Muscular Dystrophy.

Muscular Dystrophy is a group of diseases which causes the affected to progressively lose muscle mass and muscle strength. This can affect range of motion, and limit the person's ability to perform certain things independently, including playing an instrument such as the bass guitar. A standard four string bass with 24 frets covers 38 separate notes, or 3 octaves with 2 extra semitones. The product described in this report is limited in the number of notes it can produce, but with less limitations on time and budget, this product could easily be improved to more closely resemble a standard bass guitar. The description below highlights and outlines the ***Mission Statement*** for the creation of the assistive device:

- *This project is an assistive aid that will improve the ability of someone affected with Duchenne Muscular Dystrophy (DMD) to manipulate a bass guitar. This will include plucking, tuning and fretting so that the user would be able to play a variety of songs that range in structure and notes. The device will allow those with an affinity to play music with the inability to play to have an outlet for their musical expression.*

Design Objectives

Objective 1: Wireless Application

Objective 2: Automatic Tuning

Objective 3: Clean Output Sound

Objective 4: Ease of Use with Low Mobility

General Block Diagram

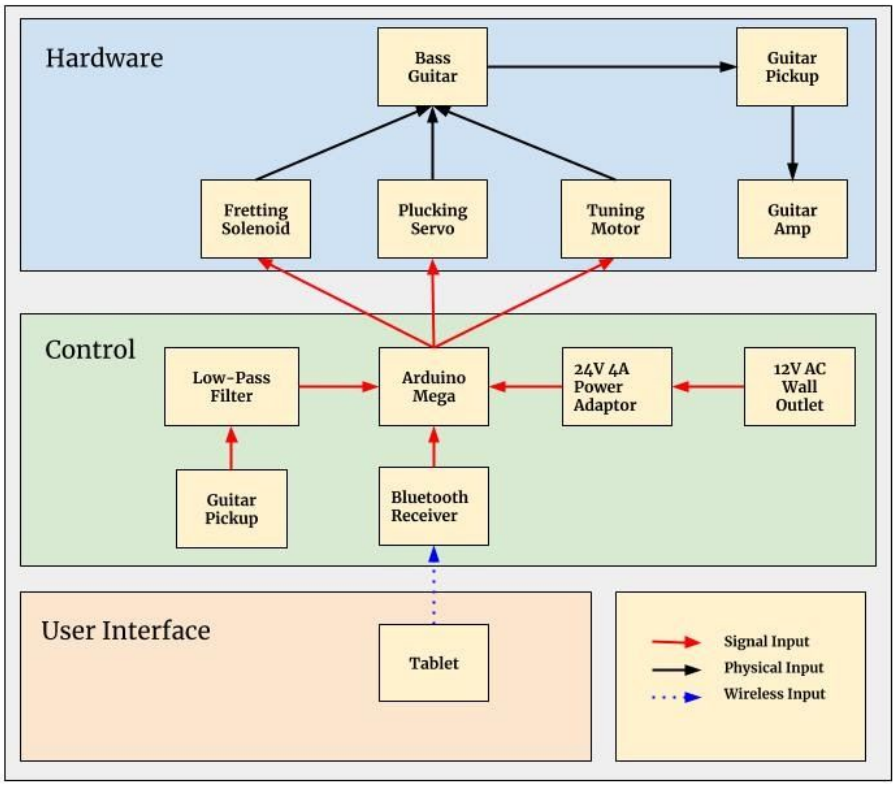


Figure I: General Block Diagram

Microcontroller

The deciding design feature in the selection of a microcontroller was the amount of digital I/O pins. This led to the decision to use the Arduino Mega 2560, with the 12 volt operating voltage and the ample amount of extra digital I/O pins for expansion.

Filter

The filter block of our design contains an active low-pass filter with a gain of 20 and a cut-off frequency of 170 Hz as it is intended to take a 150 mV peak input signal from the bass pick-ups and output a filtered signal ranging from 0 to 5V. The internal source resistance of the pickup is modeled by a 10 kΩ resistor.

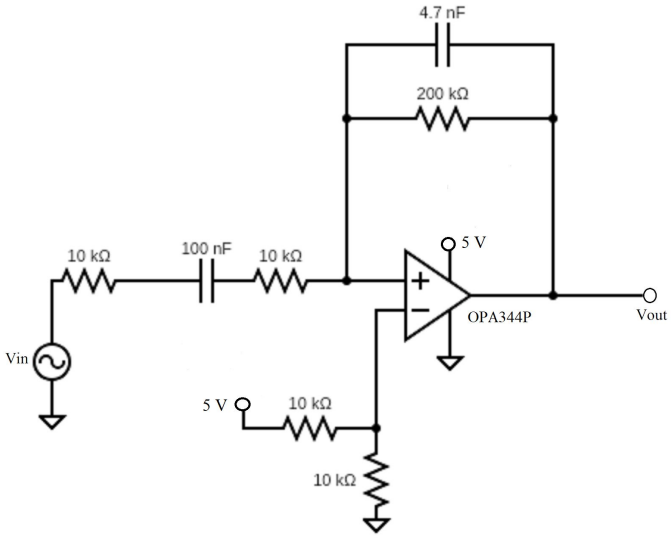


Figure II: Signal Amplifier and Filter Circuit Schematic

Power Switching

The majority of our circuitry is powered by five volts, so we employ a buck-converter module to step down a twelve volt supply and create a five volt rail. The module we use is part number 106990003 from Seeed Technology Co., Ltd.

Motor

In order to control the direction of the DC motor in our tuning block, we utilize the HiLetgo BTS7960 motor driver module which employs two BTN7970 PN half bridges created by Infineon.

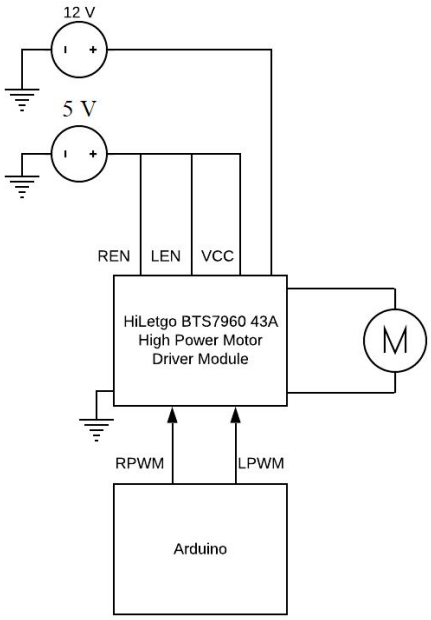


Figure III: Motor Control Diagram

Bluetooth

The HC-06 bluetooth module is powered by six volts. The HC-06 uses 3.3 V logic, but the Arduino uses 5 V logic. Because of this, a voltage divider is used between the transmit of the Arduino and the receive of the bluetooth to step the 5 V logic down to 3.3 V.

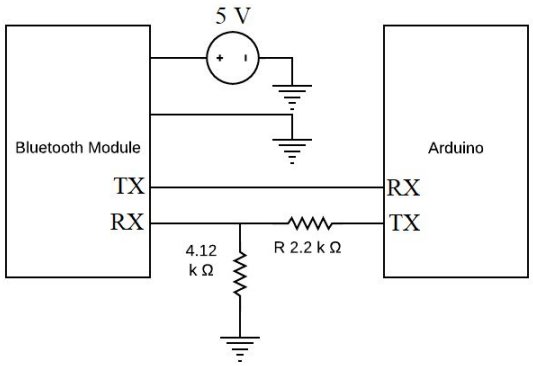


Figure IV: Bluetooth Module Schematic

Servo

The plucking mechanism of our system consists of a servo motor which is powered by a five volt rail and is connected to the Arduino through a digital pin for controlling purposes.

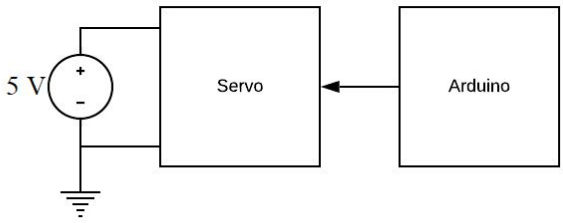


Figure V: Servo Schematic

Solenoids

The fretting of the string is executed by five push-pull solenoids, one for each fret of our design. Each of these solenoids is controlled through the gate of a MOSFET connected to a digital pin on the Arduino with a diode connecting the source to the twelve volt supply. This ensures voltage only flows into the positive bank to account for kickback from the solenoid when it turns off.

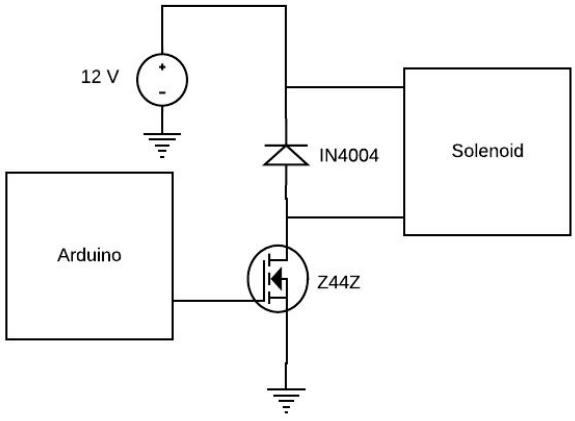


Figure VI: Single Solenoid Schematic

The final design meets the objectives of this project, but leaves plenty of room for future work. Future work for this project would include the implementation of real-time tuning, the development of a professional mobile application, and the testing of the product on people affected by range-of-motion limitations.

With future work completed, the device proposed in this paper has the potential to assist not only people affected by Muscular Dystrophy, but an entire community of people experiencing limited range of motion, to express themselves through music.

1. Introduction

Our project is an assistive aid that will improve the ability of someone affected with Duchenne Muscular Dystrophy (DMD) to manipulate a bass guitar. DMD is a degenerative muscular disease that affects young males ages 3-5. As they age they lose more and more muscular strength and control, eventually resulting in their premature death. Those who possess talent in art forms such as musical instruments can no longer experience their passion due to their limitations. This project will allow those afflicted to be able to play a bass guitar with minimal movement.

2. Literature Review

2.1 NECAMSID Information

NECAMSID, short for The New England Center for Analog and Mixed Signal Design at WPI, is a research laboratory at WPI for industry-sponsored research and project work. The lab sponsors many student projects and research, including both senior projects and graduate work. In the past, multiple company sponsors have been a part of NECAMSID, including Allegro Microsystems, BAE Systems, Texas Instruments and Analog Devices. The type of work within the lab revolves around real world applications and limitations of circuits and systems, with students testing limitations in these systems. With this type of work and sponsorship involving the common interest of analog design and mixed signal design, sponsors and students can share information and work together effectively [26].

2.2 The Arts and Disabilities

Art is used as a way to express oneself through the means of creative freedom. With art, people are allowed to express themselves and connect with their emotions to create different forms of art, whether it be in the form of drawings musical based. Due to this cathartic experience with creating art, art therapy has been used to provide emotional stability and wellbeing to those who have trouble expressing themselves. Due to this, art has been used as a form of therapy to those with disabilities. Those with disabilities struggle with a sense of normalcy and identity, and can help progress the disability identity over time [21]. From art and art-based learning, both self-esteem and confidence levels rose in those who had disabilities as this expression through certain art forms allowed those with disabilities to express themselves where they previously could not. Engaging youths with the arts allowed them to "adopt a more positive, inclusive and potentially multi-identity perspective" [23]. Those with disabilities who are prone to struggles with identity and self esteem issues can utilize art as a means to build esteem and confidence by freely expressing their emotions through art therapy.

2.2.1 Physiological Effects related to Muscular Dystrophy

Art can be used to help alleviate behavioral issues amongst those with DMD. Studies have shown that there is an increased occurrence of concerning neurobehavioral issues among young boys and males with Muscular Dystrophy. These neurobehavioral issues include anxious behavior in social situations, along with depressed thoughts. In a study involving 23 boys who have Muscular Dystrophy between the ages of 8 and 15, an assessment of occurrence of these concerning behavioral issues was conducted against a control group [22]. The results show a significantly more occurrence of both depressive disorders and dysthymic disorders among the males with Muscular Dystrophy(6). From this study, 64.7 % of those in the study reported being anxious at the time. 58.8 % also stated they have anxiety in social situations, especially those in peer to peer relationships. The study also elaborated on the issues of depression within the study group as 41 % of the males were found to have a depressed mood [22].

2.3 Muscular Dystrophy

Muscular Dystrophy is a group of diseases which causes the affected party to progressively lose muscle mass and muscle strength. A genetic mutation causes interference in the creation of protein synthesis within the cells. These inhibited proteins are needed to build muscles and maintain healthy muscle development. This abnormal production of proteins is the root cause for the progressive loss of muscle mass and strength over time. Other symptoms that form in early childhood are troubling walking, pain within the muscles, and difficulty standing [11]. Because of these symptoms, people resort to the use of a wheelchair for mobility. As of right now, there is no cure for this disease, with physical therapy and other medical treatments used to slow the progression of the disease. Muscular Dystrophy is a genetic disease, meaning that the mutated gene can be passed down from family members. Muscular Dystrophy typically affects young children, mainly young boys, with the possibility of 1 in every 5,000 males being affected [11]. There are many types of Muscular Dystrophy, all involving the similar symptoms of the progressive loss of motor control and muscle loss. The most common form, however, is Duchenne type Muscular Dystrophy (DMD). Other types of Muscular Dystrophy occur later in life, and are usually more gradual in effect.

2.3.1 Duchenne Type Muscular Dystrophy

The most common form of Muscular Dystrophy is Duchenne type, which affects 1 in every 3500 males. The dystrophin gene, which is the largest human genome, is mutated to cause the blockage of protein synthesis. The dystrophin gene occurs on the X chromosome, making Duchenne Muscular Dystrophy an X-linked recessive disorder [17]. This is why males are more affected by the disorder, as they only need one mutation on the X chromosome due to their X-Y genome make-up, while females would require two mutations on both their X chromosomes.

Symptoms begin to show in young boys between the ages of three and five, with a waddling gait, clumsiness and enlarged calf muscles being indicators to the disorder. Around the age of 12, the children become wheelchair bound due to the lack of muscle mass in the legs to

allow the child to walk. The lack of motor control also affects involuntary muscle movement as well, where involuntary muscle movements like that of the heart and lungs become severely weakened. Due to this, life-threatening complications like heart issues and breathing difficulties occur. The life expectancy of a person with Duchenne Muscular dystrophy is 27 years old, however with newer medical advancements like that of stem cell therapy and gene targeting, the outlook of the disorder is changing for the better [17].

2.3.2 Range of Motion

The progressive loss of motor control causes a person's range of motion to be severely reduced. Most with Duchenne Muscular Dystrophy are limited to a wheelchair by the age of 12, due to weakness in the legs that prevents them from walking correctly. However, the most well preserved muscles in a person with DMD is within the wrist and hand muscles [24]. In a test conducted by Hiller and Wade [20], 28 boys with DMD at an average age of 14 years old were asked to help perform a study regarding arm, hand and wrist movements. A test known as the Jebsen Hand Function test was performed, which assesses a broad range of hand functions using activities that would occur in daily living. The study reported that many of the subjects performed well in activities such as stacking checkers, writing and page turning. However, many subjects had difficulty lifting both light and heavy objects against gravity [24]. The research shows that lifting objects is a difficulty amongst people with DMD amongst all ages, however hand, wrist and finger movement can still be relatively preserved over time. This allows those with the disorder to utilize everyday objects, and other interfaces.

2.4 Musical Background

In order to create an assistive bass guitar, the fundamentals of music need to be understood and considered as well as the physical design of a standard bass guitar and the common techniques a musician of this instrument would master. The physical forces of the interactions between the musician and their bass guitar must also be analyzed in order to be

reproduced in this project. Once the music theory, artist's techniques, physical theory and design are understood, an assistive bass guitar can be created which will most closely achieve the same techniques as a standard bass guitar.

2.4.1 Fundamentals of Music

Several important fundamentals include tone, pitch, scale and mode, rhythm and melody, timbre, dynamics and articulation.

2.4.1.1 Tones and Semitones

A musical tone is a steady periodic sound and is characterized by its duration, pitch, and timbre. A pure tone is a sinusoidal waveform while a complex tone is a combination of two or more pure tones. Semitones represent the distance between two tones in a scale and are the smallest step that can be taken between pitches in music.

A tone's pitch is its fundamental frequency. Frequency refers to the number of vibrations that a string makes in a specific period of time. Due to the sinusoidal behavior of tones, their fundamental frequencies can be analyzed using the Fourier theorem. Often the lowest of the frequencies, the fundamental frequency, and pitch, is also the inverse of the period of the waveform [12].

2.4.1.2 Pitch

A string vibrates with a particular fundamental frequency (pitch). It is possible, however, to produce pitches with different frequencies from the same string. The four properties of the string that affect its frequency are length, diameter, tension, and density.

Pitch is a perception of a tone's lowness or highness. The difference in pitch between two notes is called an interval. The most basic interval is unison, which occurs between two notes of the same pitch. The octave interval is two pitches that are either double or half the frequency of one another. Specific frequencies are assigned letter names. For example, A4 (the A above

middle C on the piano) is commonly assigned to the frequency of 440 Hz. Therefore, a note of half or double the pitch frequency (ie 220 Hz or 880 Hz) would be in octave intervals of A4 and would be considered A3 and A5. Pitches in octave intervals of each other are grouped into a single class determined by their assigned note letter. So, the notes with fundamental frequency (pitch) of 220 Hz, 440 Hz, and 880 Hz are all grouped into the “A” pitch class [15].

2.4.1.3 Scale

Notes can be arranged in a variety of scales. A scale is a set of musical notes ordered by fundamental frequency or pitch. Western music theory generally divides the octave into a series of twelve tones, called a chromatic scale, within which the interval between adjacent tones is called a half step or semitone.

Most scales are octave-repeating, so their pattern of notes is the same in every octave. An octave-repeating scale can be represented as a circular arrangement of pitch classes. For instance, the increasing C major scale is C–D–E–F–G–A–B–[C], with the bracket indicating that the last note is an octave higher than the first note. The distance between two successive notes in a scale is called a scale step.

2.4.1.4 Timbre

Timbre is the principal phenomenon that allows us to distinguish one instrument from another when both play at the same pitch and volume. Although timbre can be accurately described and analyzed by Fourier analysis and other methods because it results from the combination of all sound frequencies, attack and release envelopes, and other qualities that a tone comprises, it has no standard nomenclature.

Timbre is principally determined by the relative balance of overtones produced by a given instrument due to its construction, and the envelope of the sound (including changes in the overtone structure over time). Timbre varies widely between different instruments, voices, and to a lesser degree, between instruments of the same type due to variations in their construction, and significantly, the performer's technique. The timbre of most instruments can be changed by

employing different techniques while playing. For example, the timbre of a trumpet changes when a mute is inserted into the bell.

2.4.1.5 Articulation

Articulation refers to the duration of the notes a musician plays. The most commonly performed articulations of all instruments from long to short include legato, tenuto, marcato, staccato, and martelé. Many of these articulation techniques can be combined to create certain crossovers of style.

In bass guitar, the more common forms of articulation are staccato and legato. Staccato notes are described as separated or detached. They can be created by plucking a note and then muting the sound after the desired duration, causing quick and discrete notes in the melody. Legato notes are described as smooth and connected. They can be created by plucking a note and then immediately plucking the next, creating a fluid pattern of notes with no discernible disconnect between them.

2.4.2 The Standard Bass Guitar

The many aspects of the design of the modern bass guitar that make it a popular instrument in the music industry today should be understood when creating an assistive bass guitar for this project. The techniques used by musicians when playing this instrument must also be understood in order to enable the user to master the same techniques.

2.4.2.1 Physical Design

A standard four string bass requires a neck with frets, 4 strings, a pickup, and a tuning mechanism.

2.4.2.1.1 Neck with Frets

A standard four string bass with 24 frets covers 38 separate notes, or 3 octaves with 2 extra semitones. Figure 1 shows a diagram of the notes available on a 24 fret, 4 string bass guitar.

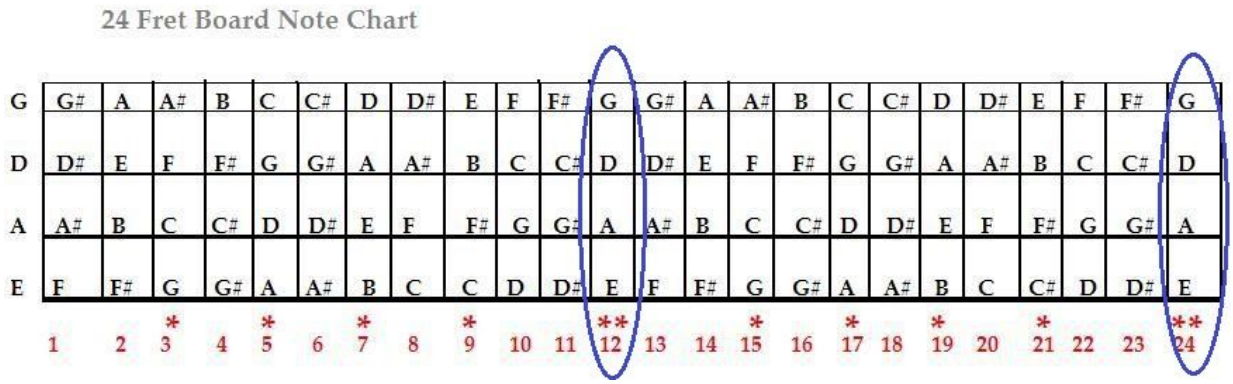


Figure 1: Notes of a 24 Fret Bass

Retrieved from

http://smartbassguitar.com/bass-essentials-series-pt-3-bass-guitar-notes-fretboard-radius-and-neck-profile/#.XZo-iEZKg2w

As stated in section 2.4.1.2, four properties of the string that affect its frequency are length, diameter, tension, and density. When a musician presses their finger on a string, they shorten its length, increasing the pitch. [27]

2.4.2.1.2 Pickups

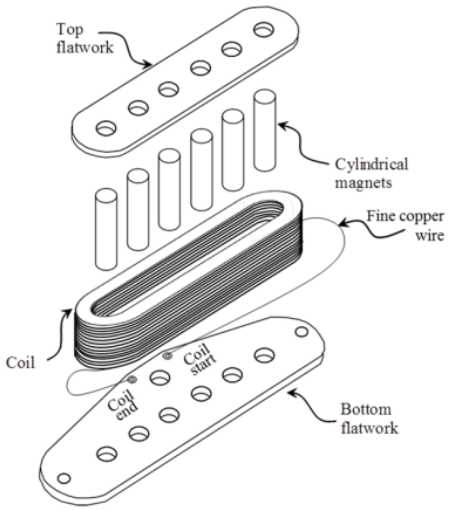


Figure 2: Guitar Pickup

Retrieved from https://www.guitarworld.com/gear/how-does-a-guitar-pickup-really-work

A pickup sits underneath the strings on the body of the bass guitar and senses the vibrations in the string and converts it to an electrical signal that can be amplified and played through an electric speaker. There are many different physical designs of a pickup including magnetic, piezoelectric, and optical.

A magnetic bass guitar pickup generally uses a sensor which measures changes in magnetic reluctance caused by the vibrations in the strings. It is essentially a coil wound around a permanently magnetized probe which sits below a string made out of a magnetic metal. When the string vibrates through the probe's magnetic field, the flux density is modulated. This induces AC voltages in the coil. It is this signal that gets amplified to create the sound of an electric guitar.

2.4.2.1.3 Tuning

A bass guitar should be tuned every time it is played. The standard tuning for a 4 string bass is E, A, D, G from thickest to thinnest string. The bass strings are tuned in fourths. All-fourths tuning is based on the perfect fourth (five semitones) meaning the consecutive open

notes of all-fourths tuning are spaced apart by five semitones on the chromatic circle, which lists the twelve notes of the octave. This is the reason why the A note can be heard on the 5th fret of the E string and the D on the 10th and so on. See Figure 2 to visualize the standard all-fourths tuning of a bass guitar.

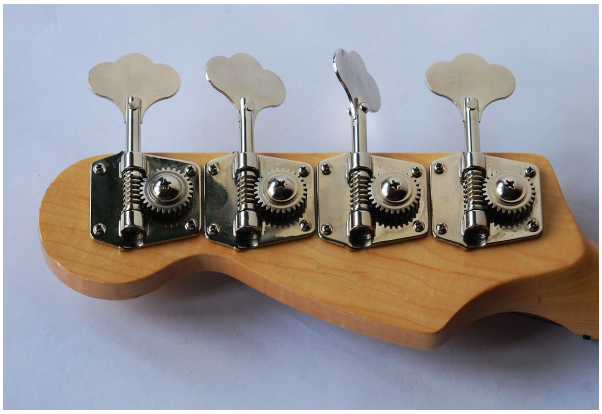


Figure 3: Tuning Pegs

By Pastorius - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=3233695

A bass guitar string is tuned using a tuning peg attached to a worm drive, shown in Figure 3. A worm drive is a gear arrangement in which a screw (called the worm) interacts a gear (called the worm gear). Like other gear arrangements, a worm drive can reduce rotational speed or transmit higher torque.

As stated in section 2.4.1.2, four properties of the string that affect its frequency are length, diameter, tension, and density. This tuning mechanism affects the tension in the string in order to control the frequency. A string stretched between two points will have tension. Increasing tension in the string gives it a higher frequency while loosening it lowers the frequency. [27]

Overall the tuning of a string follows the following form:

$$f_1 = \frac{\sqrt{\frac{T}{m/L}}}{2L} \qquad (1)$$

Where T represents the string tension, m represents the string mass, and L represents the string length.

2.4.2.2 Musical Techniques

There are many popular fretting techniques for musicians to master with a standard four string bass, including hammer-on and muting.

A hammer-on is a common articulation on bass which produces a note by pressing down (hammering) a string which has already been plucked. For example, if a musician plays a note on the third fret, they could then simply press the fifth fret of the same string without plucking again and create a second hammered note.

Muting can be useful in two ways, to cut off the ringing at the end of a note's desired length and to create muted notes. A musician would mute the end of a note by playing that note and then applying pressure to the string to damp the vibrations and end the note. This can provide staccato articulation to the melody. A musician would play a muted note by applying some pressure on the string while playing, creating a muted, or "dead", note.

2.5 Prior Art

After conducting research in the assistive instrument field, we found that there were many other aids to help those disabled play music including past MQPs.

2.5.1 Art for the Disabled MQP

The idea for our project came from a past MQP named Art for the Disabled, which was done by Joshua R. Denoncour. Our project is essentially a continuation of his where we will make improvements to his original design. The project consisted of a single string that would be plucked and muted through the use of solenoids. The project also tuned the string with the use of a metal gear motor. He had used force sensitive resistors as input for plucking and muting using a mouse like design for the player to play with.

This MQP is very helpful for us moving forward with our project. Josh layed out many things for improvement upon his design. From his recommendation we will use a higher amperage power supply to power the components in our system. We also know the various torque and pressure requirements needed for tuning the string. His filtering will also be used as a model for how we will filter our incoming signals for better tuning. This MQP is a very valuable resource for us in terms of our progress moving forward [1].

2.5.2 Assistive Aid for playing the Ukulele

Another previous MQP conducted was the assistive aid for playing the ukulele. Although this is a different instrument than a bass guitar, many of the concepts used here will be useful to our design. This MQP implements a fretting design over the neck of the ukulele to push down on the string for fretting. This design consisted of solenoids suspended by a machined structure to put direct pressure on the strings. Another aspect which Joshua also used in his project was the use of the force sensitive resistors as the systems inputs. They designed a mouse-like controller so the player would only need to use his fingers to play.This seemed like the best method to limit the amount of movement needed.

One of the main differences with this MQP to ours is the string material. The ukulele uses nylon strings whereas a bass guitar has metal string.This means our pressure requirements will be much greater to properly fret. This project also had to create a structure to hold the solenoids above the neck for fretting. Learning from this we decided to rotate our instrument sideways to avoid the laborious process of making an apparatus for the solenoids. Overall this MQP was very successful with the only issue being in buzzing caused by the solenoids when firing. [2]

2.5.3 Other Assistive Musical Accommodations

On the market currently there are very few aids to assist those affected by movement limitations to play bass guitar. One of the most common options would be to use a digital music player. One option is a touch pad music controller. This is essentially a layout of buttons that can be programmed to output a particular sound. This allows the addition of several musical sounds and notes including a bass. These run anywhere from $50 to $500 depending on the quality.

Though this may be a solution, it is only just digital music and doesn't give the same sound and freedom of actually playing a bass.

Figure 4: Touchpad Music Controller

Another example that was created in association with the Muscular Dystrophy Association (MDA), is called the Laser Band, shown in figure 5. This project uses a device which contains four low-intensity lasers to play music [quest.mda.org, 2012]. If a laser is blocked by the player, it will play notes corresponding to what was programmed to that laser. This device is mostly used to collaborate with professionally created music, but it can also be used to create any music one has in mind. It was designed with disabled people in mind, offering multiple means to operate the device if user mobility is limited. However, this still would require the movement of the players arms and hands, whereas we want our design to cover the most extreme cases of limited movement. [3]

Figure 5: Laser Band

2.5.4 Improvements

The first thing our project aims to improve is the amount of technique we can play the bass guitar. Joshua's design was very limited in that it only had muting and plucking. Through the use of fretting we will be able to perform lagato, stacado and hammer ons. The implementation of fretts will also allow a larger variety of notes. Another improvement will be the plucking mechanism. To improve upon Joshua's design we will use a servo rather than a solenoid to pluck string. We also plan to cut down on the amount of hardware needed to use the device. Our design will use an android tablet device that can operate the guitar over bluetooth. There will be an app designed so that it can accommodate the players hand position so that they can more easily play the device.

2.6 Design Constraints

In order to build the prototype, we needed to design multiple circuits that work together through a central microprocessor. The first stage will be the tuning of the circuit. The main frequency of the string will be detected and will adjust the string tension accordingly through the use of a metal gear motor. The inputs of the circuit will be which frets the player intends to play. This could be achieved by wireless transmission through bluetooth from an app to the guitar or through the use of force sensitive resistors as inputs. Plucking and fretting will be conducted through a combination of solenoids and servos that will be tied through the microprocessor to the inputs.

2.6.1 Tuning

To tune the bass string we will need to use a filter to isolate the main frequency of the bass string.

When the string is plucked there will be harmonics that surround the main frequency of the string's vibration. The frequency range of a bass string will range from 40hz - 400hz, meaning that we will need to use a low pass filter.

After reading through Joshua R. Denoncour's MQP report for his MQP "Art for the Disabled", he did testing with many low pass filter designs. His final design used a fourth order butterworth filter. He chose the fourth order butterworth due to its sharper break frequency roll off. A butterworth filter also has a flat frequency response meaning no ripples in the pass band. There is also a zero role off response in the stop band. The frequency response of the filter can be seen in figure 6 This filter is a very commonly used digital filter in audio circuits and was chosen for the wide range of resources to help with our implementation. [8]

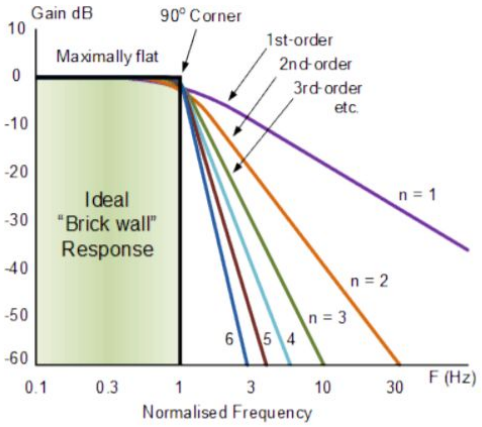


Figure 6: Frequency response of a Butterworth Filter

However, after researching we determined that an active lowpass filter would serve our project just as well. The lowpass filter will utilize an operational amplifier to increase the signal to read into the analog to digital converter of the arduino. A typical guitar pluck produces around a 150mv signal which we will increase using a filter with a gain of 20. Using the signal fed into the arduino we can use auto-correlation and peak detection algorithms to isolate the main frequency. If we are using an A-String a good range of frequency would be about 40-70hz which helped us choose our cutoff frequency.

To tune the string we will use a 12v metal geared motor seen in figure 7. The motor has a 43.8:1 metal gearbox ratio and an encoder that provides a resolution of 16 counts per revolution of the motor shaft. It can provide 18kg.cm of torque and up to 251 rpm at the output shaft. The econder in the motor allows the ability to move either forward or backwards which

will be needed when tuning the string. The encoder will be useful with our filtering system as it can be rotated specific amounts to better dial in to a specific note or tone. This motor costs around $29 which is a reasonable price and provides all the functionality needed to tune the guitar as desired. [9]

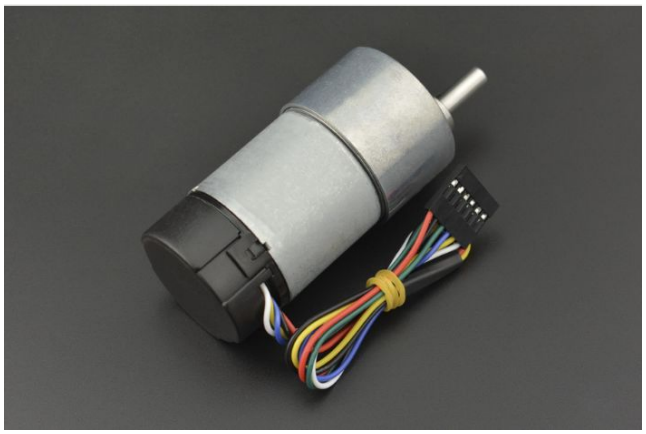


Figure 7: Metal DC Geared Motor w/Encoder - 12V 251RPM 18Kg.cm

2.6.2 Inputs

We currently have two devised methods of inputs for this circuit, the first of which is an analog input. Previous MQPs used force sensitive resistors (FSR) as inputs to the plucking and tuning mechanisms for the bass. This acts as a variable resistor with it's resistance fluctuating depending on the amount of pressure on it. Within the resistor there is a conductive layer that generates more current with a larger force on it. The player would press down on the corresponding sensor and it would press on the desired fret using a signal from the sensor to the solenoid. Each fret would have a corresponding input sensor and use a wired connection to transmit the signal. Each input would be connected to a separate digital I/O pin. Using equation 1 we can determine the voltage values of the FSR that correspond to the tuning tension needed [10].

$$Vo = Vcc\,(R / (R + FSR)) \qquad (2)$$

Figure 8: Force Sensitive Resistor

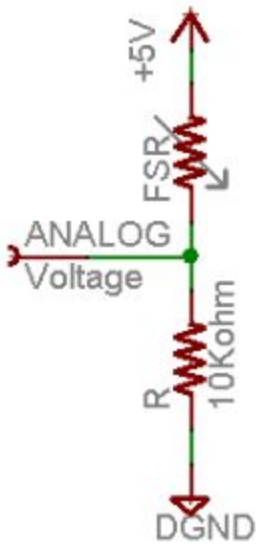


Figure 9: Force Sensitive Resistor Circuit Setup

The other method we are focusing on for input is the use of an android app through a tablet. This would utilize an HC-06 bluetooth chip that is compatible with our arduino microprocessor. The app would be designed in a premade program called MIT App inventor. This program has many interface options and functionality for what we need to accomplish. The app would have a layout that had each fret input layed out and a button for plucking. An issue that may arise with this method comes in the form of latency between input to output. Our group

does not have extensive experience with app programming and therefore are not certain this will be implemented. As of now we will work towards the android app input but will use the analog input as a backup method and a testing method.

2.6.4 Fretting

A typical bass guitar has 24 frets on its neck. For each fret there will be a solenoid pressing down on the string to change its frequency. Solenoids are electromagnets that contain a coil of copper wire with an armature in the middle. When the coil is energized, the slug is pulled into the center of the coil. This makes the solenoid able to pull or push. The armature for these solenoids is 30mm. The solenoids have a newton starting force at 12V. Each solenoid will be a seperate input that corresponds to a finger press from the player. Though guitar playing has more technique to how much pressure is put on fretting each string, we are treating this as a binary application. This means that the solenoids will be either pushing on not pushing on the string. After conducting tests on force needed to push down on the fret we found we would need around 8 newtons. The servo we will use will have 10 newtons of force. The solenoid will need a transistor and a diode for it to work with an arduino. The transistor will be used as a current amplifier to charge up the magnetic coil within the solenoid. The diode will be used to prevent the coil from discharging back into the arduino. The circuit setup for the solenoid connected to an arduino can be seen in figure 14. [5]

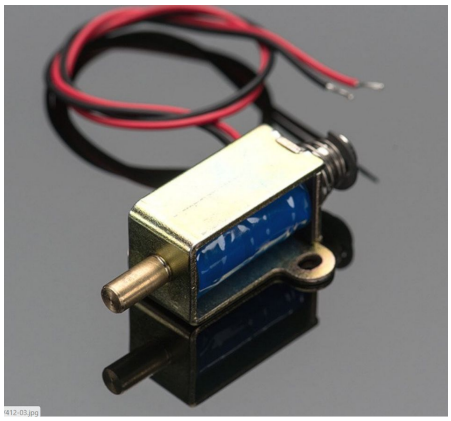


Figure 10: Push Pull Solenoid

2.6.5 Plucking

To pluck the guitar, there will be using a TowerPro SG-5010 servo.A Servo is a small device that is driven by a DC motor and gear train to drive an output shaft. This servo has a 180 degree rotation (90 degrees in either direction) which will more than cover the distance needed to pluck the string. Through testing we determined that we shouldn't need more than 90 degrees of rotation. The servo works with 1-2ms pulses that are well within the rotation range we need for plucking. [4] The servo will be mounted to the far end of the string so as to not interfere with the solenoids during plucking. There will be a guitar pick attached to the output shaft which will be the contact point for the string. The servo will also be connected to our arduino but will not require any external components for operation as shown in figure 11.

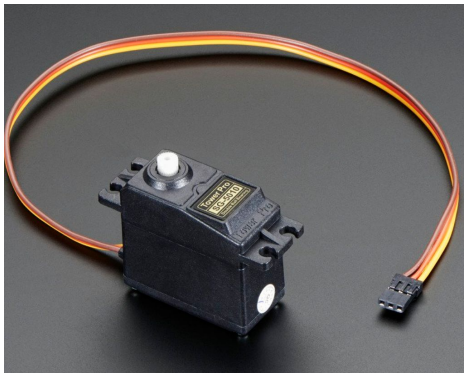


Figure 11: TowerPro SG-5010 servo

2.6.6 Arduino Mega

The arduino mega was chosen as our microprocessor due to it's large number of inputs. It has 54 digital I/O pins which is much more than we need for our 10 planned inputs. However, the extra input pins allows for more expansion later on if we decide to add the ability for new playing techniques or adding more strings to the guitar. The mega offers 5V output digital I/O pins that will work well with many small voltage solenoids. It also runs on relatively low power, needing a 7-20v operating voltage. The mega is also cheap costing around $30. The mega should be able to account for every need moving forward with this project. [7]

Figure 12: Arduino Mega

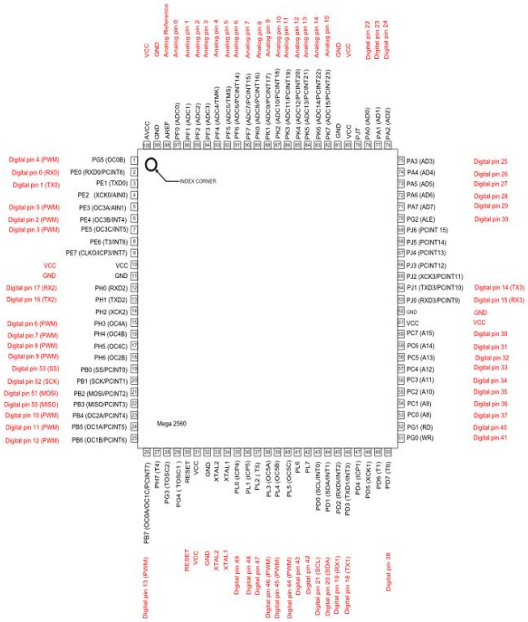


Figure 13: Arduino Mega Pin Layout

2.6.7 Component Power

The DC geared motor that will be used in the tuning design runs on 6-12V. We plan on running it at its maximum voltage for a higher running speed. At rest this will draw around 350mA. Under load it can draw up to a maximum of 5.1 amps. Adjusting the load of a bass string will draw much lower current which will range from 1.1 amps to 2.1 amps. This will only be used at the beginning when tuning before playing. [9]

The push pull solenoids that we will use run on 12V with a current draw about 10mA at idle 250mA during movement depending on how it is being operated. We plan on using 10 -20 meaning that we will potentially have a minimum of 0.1A draw to 2.5A maximum draw. The maximum value will most likely never be used due to the way a bass is played. Typically only one to two will be pressed at once meaning that the maximum draw will not be achieved. [5]

To pluck the string a servo will be used. This has a 10mA current draw when idle and a 100-250mA draw when in use. This will draw more current than a single solenoid as it will be in use more plucking on every note.[4]

All of these components will be connected to the arduino mega which runs on a recommended 7-20 volt operating voltage.[7] The motor will be run on 12 volts. The power will be supplied by a 12V 5A adaptor. The power in will be split to the motor, the arduino and the solenoids. This will be a tricky configuration seeing as we will need to divide this up amongst 20 components.

3. Procedure

3.1 Mission Statement

This project is an assistive aid that will improve the ability of someone affected with Duchenne Muscular Dystrophy (DMD) to manipulate a bass guitar. This will include plucking, tuning and fretting so that the user would be able to play a variety of songs that range in structure and notes. The device will allow those with an affinity to play music with the inability to play to have an outlet for their musical expression.

3.1.1 Goals

1. **Playable bass guitar with two octave range**

   Due to the single string implementation, creating a diverse note and frequency range will be a challenge. Acquiring a large enough octave range will allow the user to play a more diverse assortment of songs.

2. **Usable with low mobility**

   The main functionality of this instrument is to allow users with MD to play a bass guitar. The implementation of a user friendly assistive aid mobile app is a necessity.

3. **Instantaneous input to sound**

   Latency will be a persistent issue with this automated device. Integrating software properly to connect the interface and the electro-mechanical device with minimize latency will help reduce lagging sound.

4. **Clean output sound with minimal noise**

   A clear output sound without static and interference is essential in creating a playable bass guitar. The use of low noise devices and dampening functionalities will reduce unwanted vibrations and interference.

3.2 Design Constraints

When designing and prototyping with the assistive bass guitar, many design constraints had to be followed in order to meet the proper project requirements to satisfy the needs of the target audience. The bass guitar needs to allow users with DMD to easily play the guitar through the medium of a user friendly tablet application. Functionality issues also complicate the design of this device, as the single string design causes certain frequency and note limitations. The complex usage of multiple mechanical and electrical devices in tandem leave room for latency errors that also contain the design of the device.

3.2.1 Muscular Dystrophy

One of the main functions of this device is to aid those with the disorder of DMD. This requires the device to have certain assistive aids that allows the user to properly utilise the instrument according to their specific needs and movement ranges. Due to this, one of the top priorities in constructing the assistive bass is to have an interface that is easily accessible to a person with a limited range of motion.

The user interface for the bass guitar will be a specialized app on a tablet connected via bluetooth. In the app, there will be a limited and simple display that has only the necessary buttons to allow the playing of the bass guitar. Due to the constraints of the range of motion of the user, the buttons will be mapped on the app based on natural hand placement of those with MD. Each button will be relatively big in size as to make button pressing easier. Each button will be mapped to a certain fret solenoid, and when pressed, will activate that solenoid and the plucking device to allow a note to be played. There will also be option buttons for the other playing aspects of the bass guitar, like that of hammer-ons.

3.2.2 Functionality

Due to the proposed design of our project, there will be a few limitations in the functionality when compared with the functionality of a standard bass guitar. These limitations will include limitations in frequency, possible techniques, and playing speed.

The frequency range of our assistive bass guitar will be limited by the number of strings as well as the number of solenoids we use. In order to create a simple and low cost design, we propose the use of a single “A” string with 12 solenoids. This design will cover one full octave from A1 (55 Hz) to A2 (110 Hz). Ideally, the assistive bass would cover two full octaves, but this would require 24 solenoids and may not be cost effective for this prototype.

The techniques which will be possible with the assistive bass guitar include hammer-ons and muting. The assistive bass guitar will not have functionality for the player to perform sliding. This is due to the solenoid design for fretting.

The final design limitation of the assistive bass guitar is the playing speed. This will be limited by both the speed of the DC motor which controls the plucking, the speed of the solenoids which control fretting, and the latencies between the application interface, the microcontroller, and the motors themselves.

3.2.3 Resources Limitations

Another constraint that the team abides by is the allocation of certain resources. The main constraints faced involve time and budget. Both of these constraints shaped certain design choices and material selections.

The time period allowed for the creation of this device is relatively long, as three terms are allocated to designing and building. This means a time period from August to March is allocated. However, the acquisition of certain materials causes delays in the project development. For example, materials need to be ordered ahead of schedule to make sure their arrival is on time, as well as to test their functionalities. Due to procurement of materials, the design stage needs to be expedited in order to allow proper time to be given to the ordering of materials.

Budget is also a constraint faced on the project. Each team member has a budget of $200. With a team of four members, the budget is $800 for the team. While a large amount of money, finding certain materials that will function properly, like a powerful solenoid, is rather difficult. Keeping a budget along with finding the best materials for the automated bass will be a challenge.

3.3 General Architecture

Overall there are 4 main areas that make up the design on the bass guitar: fretting, plucking, tuning and user interface. All of these aspects need to be controlled through a central microprocessor, in which we chose the arduino mega to fill that role. Each aspect needs to work simultaneously to all the use to play the bass. The inputs will be placed using an android app which will each control a component within another portion of the bass. The whole guitar will be contained within one platform and be powered from an external source. Figure 14 shows a general block diagram of the proposed design.

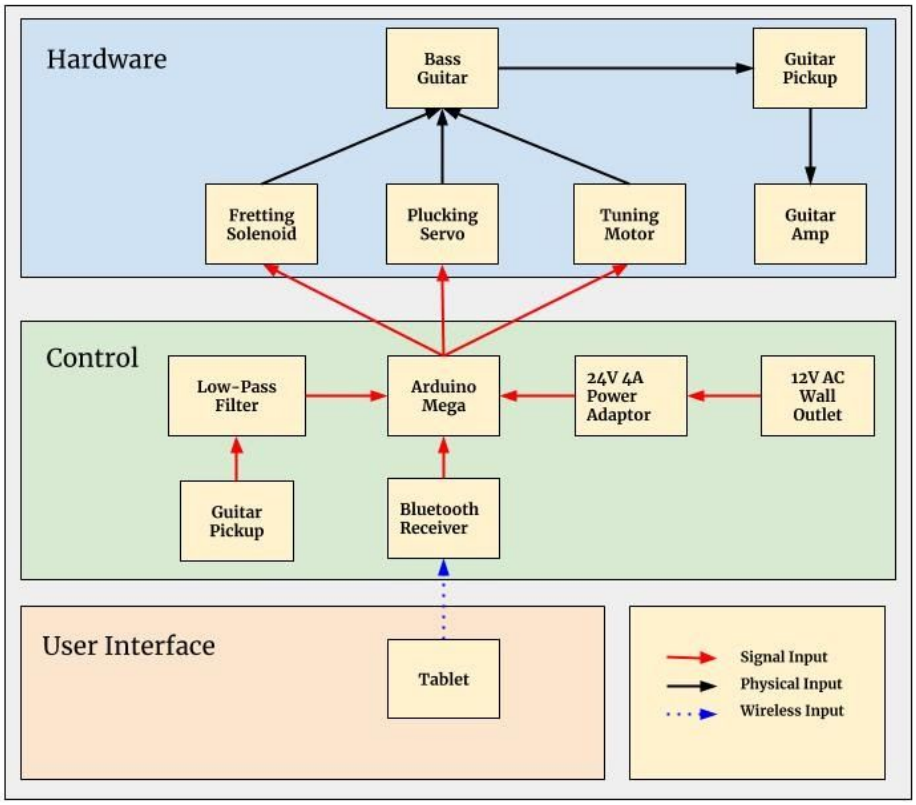


Figure 14: General Block Diagram

3.3.1 Fretting Design

The fretting aspect of the design originally consisted of two options, using servos or solenoids to push down on the string. After meeting with Scott Barton, he recommended the use of solenoids over servos. The reasoning behind this decision was that servos are much louder than solenoids when operating. We would want our design to be as quiet as possible so that we can hear the music clearly, which is why we are moving forward with solenoids.

After conducting research of solenoid use with an arduino controller, there were many projects that already existed that were along the lines of what we needed to build. The code used to program the solenoids is a modified led flashing code. Essentially, we power the solenoid and use an input signal from the arduino to activate it. Since the arduino can only output 5v, we needed to use an external power source to bring it to its operating voltage of 12v. Another

constraint was the amount of current needed to activate the solenoid. To solve this we use a IRFZ44Z mosfet to amplify the current. Since we don't want to draw current from the arduino, this mosfet works very well with our design since the mosfet can be activated with a voltage.The supply voltage will be positive which will bias the gate terminal positive and attract electrons under the gate region towards it. The charge differential generates a current across the channel to charge the coil within the solenoid. Applying a voltage to the base of the mosfet opens the gate and allows the solenoid to be connected to ground and gets powered by the external source. The final component to this design was a diode. The diode is there to prevent the solenoid from discharging back into the circuit. Since it only lets current pass from one end, it cycles around and discharges from the resistance in the solenoid rather than back into the mosfet. The circuit design for this can be seen in figure 15. The preliminary design can be seen in figure 16. This design currently uses push buttons for testing. The pushbuttons bring the logic high or low for the input signal to the gate and turn on their respective solenoids.

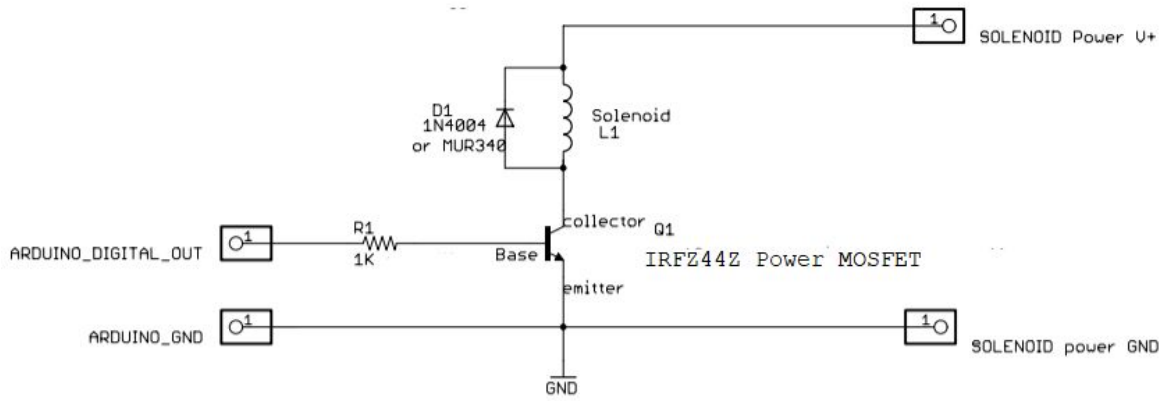


Figure 15: Solenoid to Arduino Connection

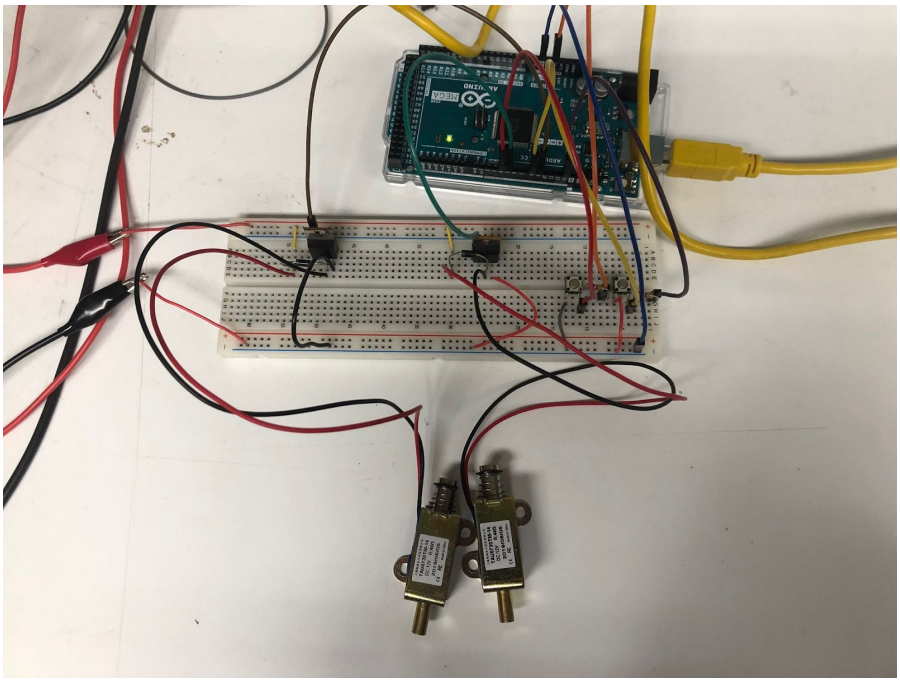


Figure 16: Preliminary Solenoid Fetting Test Circuit

3.3.2 Plucking Design

The plucking portion of the guitar is the simplest portion of the design. The plucking is performed by a servo with a pick attached to the armature. Figure 18 shows the servo attached to the pick. The servo will also be wired into the arduino board. The servo will have 3 connections, power, ground and the signal input to when it needs to rotate. The circuit layout can be seen in figure 19. The servo operates on 5V and operates using 1-2ms pulses to rotate within 180 degrees. The servo can rotate 90 degrees on a 2ms pulse and -90 degrees on a 1ms pulse. For the purpose of plucking the string we only need to rotate within 30 degrees of motion. With a smaller range, the speed in which plucking can occur is increased and doesn't require multiple servos.

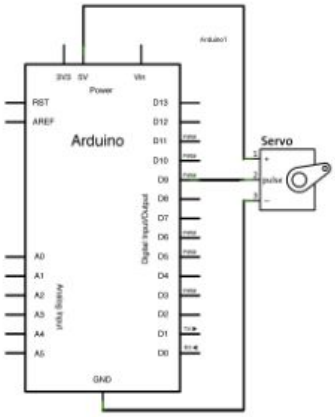


Figure 17: Servo Connected to Arduino

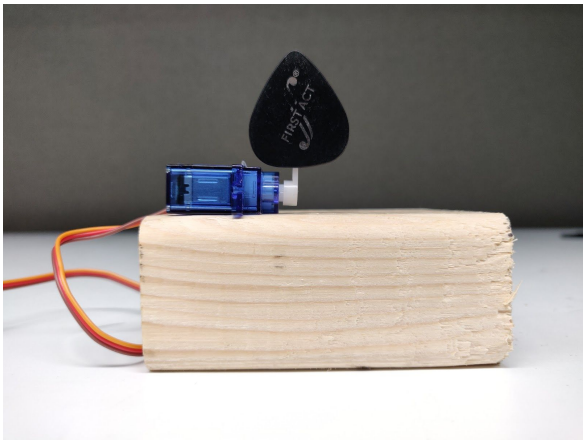


Figure 18: Servo with pick attached

3.3.3 Tuning

The main concept that set our project apart from many other automatic guitars is the fact that we are designing a completely self sufficient model. This means that regardless of temperature changes, stretching of the string, or settling of the physical design, the bass will always keep its one string in tune. The single string design was confirmed during our meeting with Scott Barton, where he agreed that the ability to play two octaves worth of notes would be sufficient for creating a usable instrument. With this in mind, as a team we realized that the

concept of self tuning would arguably be the most challenging concept of the project, and require significant testing in an isolated test setup before being integrated into the overall design. Doing so would allow team members to focus on the mechanical design and coding aspect of self tuning simultaneously.

The first step of creating the test was mocking up connections between the geared DC motor and a dowel for tensioning the string. During initial research, our team determined that the use of worm gears has two major benefits in the design of guitar tuning. On one hand the use of a vertical worm gear to horizontal rotor allows for a design that mitigates shear forces between gears. While this is important, we found in our early testing that the true benefits of using worm gears was the fact that once the tension was set on the string, the DC motor no longer required power and the mechanical advantages of the worm gear would take over and hold the string at the set tension. As a team we developed an initial design which consisted of a modified six string guitar tuning peg coupled to the rotor of the high torque motor. Initially, a strong metal based epoxy was used to conjoin both shafts, but after testing the material began to break down. We took this time to utilize the Washburn laboratories and speak to the professionals in the machining shop who helped source a 6mm collar with double set screws to conjoin both shafts. The test platform seen below shows our general goal for creating a setup focused solely on self tuning. The design itself will take on a few changes as the project develops, but the use of wood, sheet metal, L-brackets, and miscellaneous hardware allowed us to create an effective design quickly so that team members could distribute the workload and begin coding while further improvements were made to the mechanical design.

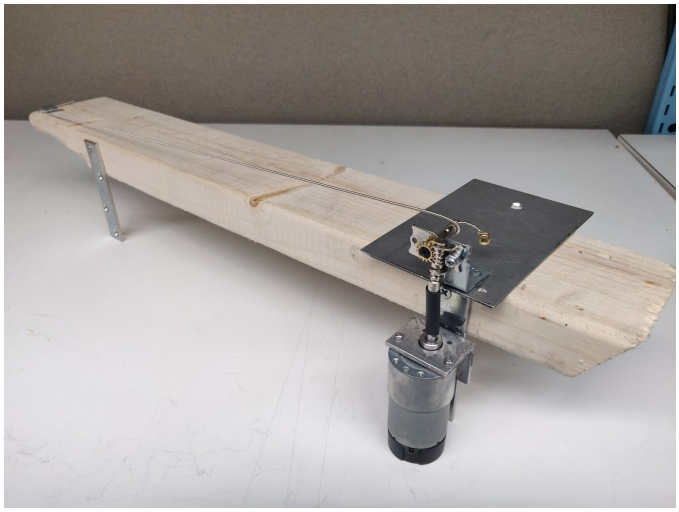


Figure 19: Self Tuning Test Platform

With a basic platform to experiment with self tuning, our team began exploring options for the software side of the system. Looking at the picture above, it is worth noting that the guitar pickup is not installed in the test platform. This development will be addressed in the near future, as it is required to self tune the system. In our case, the pickup is directly connected to a typical auxiliary jack that would go to a normal guitar amplifier. These leads will connect to both the arduino and a seperate amplifier circuit for creating audio. Taking the outputs of the guitar pickup and sending them to the arduino will allow our team to perform signal analysis and manipulation in order to reach an acceptable frequency range.

Using the simulink packages for Arduino, the basic concept of self tuning our team applied required that at first the string be set to an approximate benchmark tension. The code's goal is then to pluck the string to receive an audio signal. This signal is registered by noting whether the amplitude is outside of a set threshold slightly greater than the resting magnitude of the input scope. Once a closed system of input signals and scope readings is established, the code then analyzes the period of the wave, as period and pitch are directly related when regarding sound waves. This portion of the process requires “pitch estimation”, where the motor compensates to ensure the wave period is within an acceptable set threshold of equivalent pitch.

3.3.4 Inputs/Android App

For the application that will be created for the device, there will be multiple inputs that are mapped to certain functionalities of the assistive bass guitar. The application will need to be designed in a way that is accessible to the player with muscular dystrophy. Our proposed design is shown below in Figure 20.

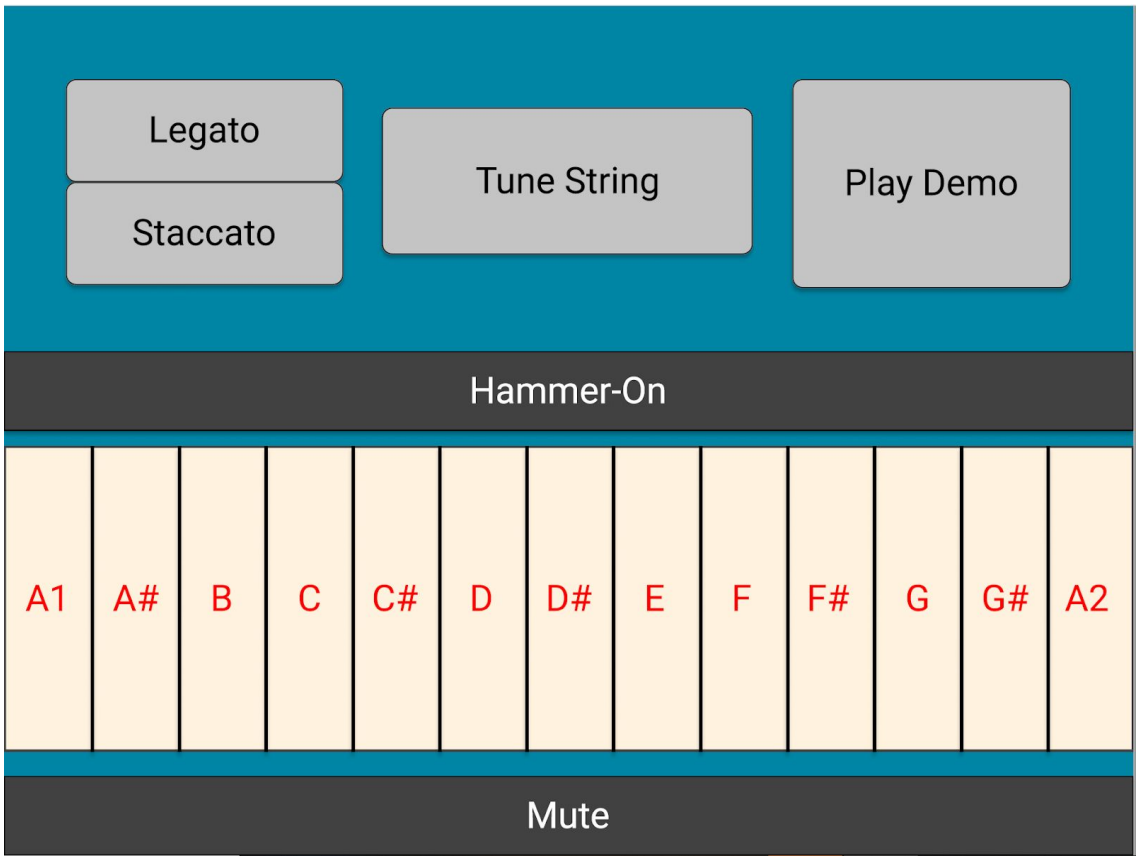


Figure 20: Interface Design

The interface design includes inputs for each note as well and options to mute or hammer-on. The interface design also includes an input to tune the string and to select legato or staccato playing. Lastly, we would like to include an option to play a demo, mainly for

presentation purposes. A basic flowchart which outlines the processes started by each user input is shown below in Figure 21.

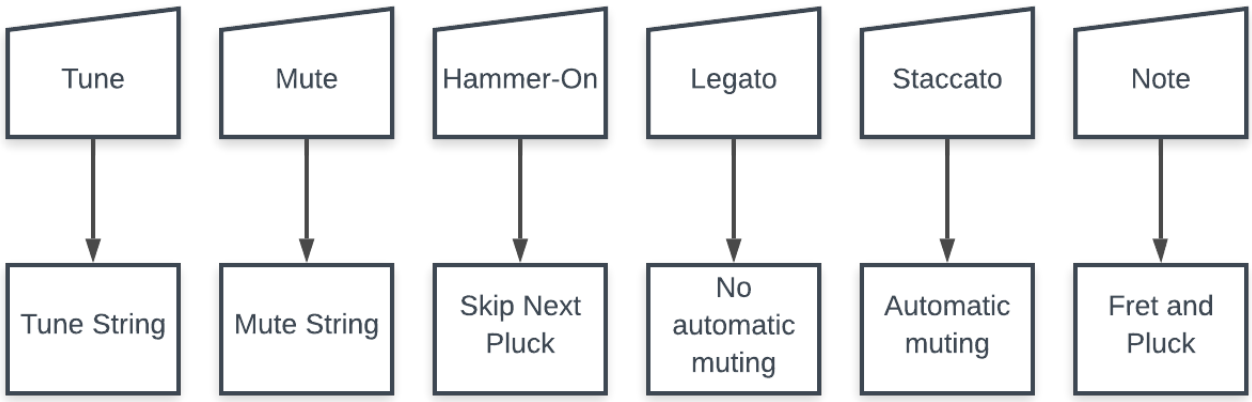


Figure 21: Input/Output Flowchart

4. Methodology

In designing an assistive bass guitar to allow those with limited motion within their hands and fingers, a set of design objectives was formulated to help guide the project in a way to maximize its accessibility for those with Muscular Dystrophy. Each of these objectives were set in order to analyze its functionality, and ultimately tested leading to a finalized prototype piece. While these design objectives were used to fulfill the overall project goal of allowing those with Muscular Dystrophy to effortlessly play a bass guitar, certain constraints known prior to the design period, along with constraints found during testing, limited the project's design freedom. These constraints resulted from mechanical and electrical limitations, along with constraints set by the user audience of the assistive instrument. Both the design objectives and constraints on the project created the parameters needed to build a finalized prototype. These parameters are discussed in detail below, and outline the process and guidelines the team followed to design and build the assistive bass guitar for those with Muscular Dystrophy.

4.1 Design Objectives

With the goal of creating a successful and marketable product for our customer base, the team created a list of objectives for the design. These objectives were determined early in the design process and were re-assessed throughout in order to ensure their success.

Objective 1: Wireless Application

- The instrument should follow a wireless design. The tablet which the user interacts with to control the instrument should communicate with the instrument through bluetooth. This is important to our customers because it allows the same portability and freedom as a traditional bass guitar.

Objective 2: Automatic Tuning

- The instrument should be able to tune its string automatically before the user plays. This is an important aspect because the act of tuning the bass guitar manually would be difficult for our customer with MD.

Objective 3: Clean Output Sound

- The instrument should produce a clean output sound similar to a traditional bass guitar. The notes it plays should be distinguishable and clear with minimal noise interference.

Objective 4: Ease of Use

- The application should be simple and easy to use with a minimal range of motion required to play the range of notes available. This is an important objective as it relates closely to the user's disability.

4.2 Design Constraints

The following constraints were also taken into consideration throughout the design period and the testing period of the project. These limitations helped to keep the project within its original scope.

Constraint 1: Force and Torque Requirements

- Torque requirements apply to the encoded motor that automatically tunes the A string, along with the servo at the base of the guitar designed to pluck the string. The angular force required from the bass guitar string has to be taken into consideration when sourcing motor and servo. Force requirements also are required for linearly pushing down on the string to hit the frets on the bass guitar neck, and require the solenoids to have enough force to hold the string down to make a note play.

Constraint 2: String Selection

- String selection is an important part in this design, as frequency range and string tension affect the overall function of the bass guitar. The string chosen needs to have a large enough range of playable notes to allow common songs to be played, while the tension in the string needs to be manageable to allow the motor and servo to be able to pull and pluck the string.

Constraint 3: Noise Interference

- As a musical instrument, the bass guitar should have a clean output sound with minimal noise coming from other sources. This is why during the design portion of the project, low-noise emitting parts were given a high priority, resulting in the selection of solenoids over servos to push on the frets.

Constraint 4: User Mobility

- The intended audience of this device are those with limited mobility within their hands due to the disorder known as Muscular Dystrophy. Due to this, the User Interface of the device needs to accommodate limited hand motion, making assistive touch functionality a big requirement for the design.

Constraint 5: Power Requirement

- The power supply chosen needs to convert the 120 V AC voltage from an outlet, to a 12 V DC voltage that has a high enough current rating to allow all the electrical devices connected to be powered. This maximum current calculation was done in preliminary stages to procure a power supply early on in the project, and re-calculated later to finalize its design details.

4.3 General Design Aspects

The main design architecture for the assistive project can be classified into different areas. The hardware design aspects involve both input and output hardware to the device. Software involves both user interaction and signal analysis. The additional features section covers extraneous features regarding the mechanical design of the assistive bass guitar.

Hardware inputs involve the work required to operate the device, which includes powering the device, along with the android device that is used to provide the user input to the device. The hardware outputs include devices used to produce the physical process of playing the bass, along with the circuitry required to drive these outputs.

The software of the device includes both the user interface to allow the user to play the device, and the signal analysis to ascertain the fundamental frequency of the string in its current state in order to automatically tune the string. Both areas of software require a microcontroller to read analog and digital inputs in order to control the hardware outputs. The last area of the

general aspects of the device involves the additional features of the bass guitar design. This mainly includes the mechanical design and construction of the device

4.3.1 Hardware Inputs

4.3.1.1 Power Requirement

In order to support the variety of components in the system, a high current power supply was chosen based on the needs of the circuit. In our design, the components that require power are the solenoids, plucking servo, tuning motor, bluetooth chip, filtering op-amp and the arduino mega.

The largest power requirements of the system are the fretting solenoids. Each of the solenoids has an individual current draw of approximately 1.6A and are rated for 12V. To accommodate this massive power requirement, the code for the solenoids is set up to only allow one of them to fire off at once. Therefore, we can never have two solenoids extended at the same making our maximum current drain for the fret 1.6A. Bass playing technique also only uses one fret at a time making this set up even more favorable to the design.

The controller of these solenoids is the arduino mega board which itself draws 70mA on a 5V supply. Each digital I/O pin can supply a maximum of 40mA with a maximum board supply of 200mA. To minimize current drain from the arduino, we will be using external power supplies to power and components controlled from the board.This will eliminate any issue of overdrawing from the board making it only supply a digital signal to each component.

The plucking servo operates on a 4.8-6.8V supply. To have the servo pluck as fast as possible it will operate at the maximum rated voltage. The servo will not be draining power constantly much like the solenoid. When operated, it draws about 300mA of current.

For the bluetooth chip we are using an HC-06 bluetooth module. This model runs well with arduino as it operates from 3.3V-5V and draws a maximum of 30mA when operating.

Finally we must account for the tuning aspect of the guitar. The plan for tuning the guitar is that it will be performed before playing and will not require any power once tuned. This aspect

of the design has two parts to power, the amplifier for the guitar signal and the encoded motor to tighten the string. To power the amplifying chip, we are using a 5V supply. The current draw for this device is so small that it is insignificant. The DC motor uses a 12V supply and drais 150mA under no load. After monitoring the motor under the load of the tuning peg it drains approximately 200-300mA depending on the tension.

$$\textit{Total Approximate Current} \rightarrow 1.6A + 0.07A + 0.3A + 0.03A + 0.250 \approx 2.25A$$

While it would potentially be possible to operate the system using a 3A 12V power supply, there were a few considerations that ultimately allowed up to choose a 12V 5A supply to use with the system. The mosfets used in tuning and operating the solenoids themselves have an inherent current draw. Being able to support these sub-circuits of the system are important, and in order to accommodate them and the other devices at their maximum current draw values, we found that a margin of about 2X was acceptable. The choice was solidified by the fact the 5A power supply was a common output rating and cost no more than less ample models.

We also have to focus on the individual requirements of each component. The solenoids and the motor run on 12V. This won't be an issue to power as our main power rail is 12V however, our other components run on smaller voltages. Therefore, we are going to utilize buck boost converters to step down the voltage to our required needs. The design will use two converters, one to make a 5V rail to power the op-amp, bluetooth chip and the arduino. The other converter will be used to power the servo at 6.8V. All the components will be wired in parallel to each get recommended operating voltages and to split incoming current accordingly.

4.3.1.2 Microcontroller Selection

A major aspect of this project is the digital interface between the user input and the bass guitar, along with the frequency analysis of the bass string. To accomplish both of these concepts, a microcontroller is needed. More specifically, this microcontroller is needed to have both analog and digital I/O pins to take input readings. The analog input reading is the sinusoidal waveform outputted from the bass guitar pick-up itself filtered through the active low-pass filter.

This analog input will be converted from an analog signal into a digital signal through an A/D converter, which will allow software to be written to take the digital signal and find the frequency through peak to peak detection. The digital I/O pins are needed for our hardware. During the selection process, our design at the time required 8 digital I/O pins. These were for the five solenoids, one for the bluetooth module, and two for the h-bridge module controlling the encoded motor. Along with these, the microcontroller ideally would be powered by a 12 volt source, and be able to have a power rail to power the bluetooth module and servo.

The team also took into consideration the amount of experience members had with certain microprocessors and coding environments. Many of the team members had prior experience in the arduino environment, and were accustomed to its functionality and IDE coding environment. By taking into consideration the specifications needed to be satisfied, along with the prior knowledge of the team, the arduino family was chosen as the type of microprocessor the team would utilize. In the figure shown below, the specifications of six different microcontrollers from the arduino family are listed in a chart. Each one of the microcontrollers below was a candidate for the processor we would use.

Name	Processor	Operating/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
Mega 2560	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4
Micro	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	1	2.5	32	Micro	1
MKR1000	SAMD21 Cortex-M0+	3.3 V/ 5V	48MHz	7/1	8/4	-	32	256	Micro	1
Pro	ATmega168 ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	0.512 1	1 2	16 32	-	1
Pro Mini	ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	1	2	32	-	1
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	1	2	32	Regular	1

Figure 22: Design options for Microcontroller

In selecting the microcontroller, the main design feature that decided the selection was the amount of digital I/O pins, as throughout the design and testing process, more I/O pins may

need to be available. Due to this, the Arduino Mega 2560 was chosen as the microcontroller the team would use, with the 12 volt operating voltage and the ample amount of extra digital I/O pins for expansion.

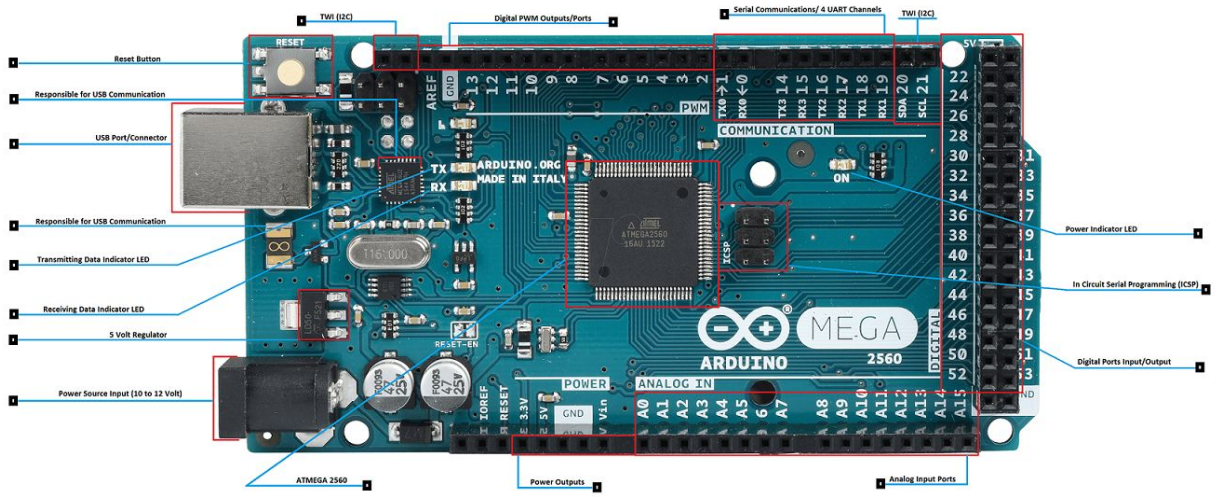


Figure 23: General Pinout for the Arduino Mega 2560

4.3.1.2 User Interface Input

The preliminary design feature to interface the user to the bass guitar was a touch screen design. A digital interface with a touch-enabled design was chosen due to the ease of use the touch screen would provide for the user with Muscular Dystrophy. Also, a touch screen design provided a cleaner, more aesthetically pleasing device, as we can eliminate wired connections and circuitry required to connect the user's physical input to a microcontroller. From our preliminary concept idea, the team decided to use a tablet as the device to allow for the interaction between the user and the device. A tablet was chosen due to its large surface area, which will allow for a larger application with larger input buttons to be created. The larger surface area for the application allows for the user to more easily control the bass guitar, and more readily allows a user with limited motion within their hands to control the bass with the input touch-screen buttons.

The next design decision was to choose the type of tablet, which was between an IOS based tablet or an android based tablet. In choosing the type, the team based its decision on the ease of interfacing the tablet with an external microcontroller with bluetooth. An android tablet has more documentation and accessibility in connecting a device using user created applications than an IOS device. This was the deciding factor in choosing a tablet, as an IOS device would prove difficult in interfacing an application with the device. From this, an android tablet was chosen as the user interface input. Another factor that went into this decision was the devices available to the team for testing. A group member had a pre-existing Samsung Galaxy tablet that could be used as a testing environment for the project, which solidified the team's design choice.

With the tablet, an android application will be pre-installed onto the device that will allow the user to connect through bluetooth to the assistive device, and allow the user provided inputs to the microcontroller, which will control the physical actuation of the hardware outputs of the device. A button press on the tablet corresponds to a digital signal that is sent through bluetooth, and sets a digital high on the digital out pin on the microcontroller, which will in turn control the device. The software and microcontroller operation is described in depth in the proceeding sections of the methodology.

4.3.2 Hardware Outputs

The hardware outputs of the guitar will be what we use to produce the sound of the string. The two outputs that we will have will be fretting and plucking. To accomplish this, we need to make sure that we account for the player’s limited range of motion. We ideally want the player to move as little as possible but still use devices that simulate a human playing. Our solution was to control the fretting using solenoids and the plucking using a servo. These devices are more binary than the actions of a human but with enough precision can come very close to performing the same as a human.

4.3.2.1 Solenoids

To perform the fretting on the bass guitar we came up with the solution to use solenoids to press down on the guitar string at the desired points. A solenoid is essentially an electromagnet that contains a coil of copper wire with an armature in the middle. When the coil is energized, the slug is pulled into the center of the coil and fires off the armature pushing it outward. We needed to simulate a human finger pressing down onto the string and keeping constant direct pressure. Using a solenoid was a clear solution as once it was powered and extended, it could hold the string to change the tension in the string for a new note. For the purposes of this project we are using the solenoids in a binary sense where they are either retracted or putting tension on the string. Human fingers provide much better manipulation of the string but we needed to limit human movement making the solenoids the best replacement. The solenoid being used in our project can be seen in figure 3.

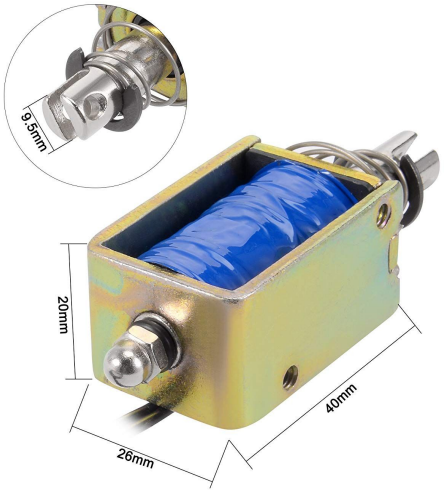


Figure 24: 12V 25N Push Pull Solenoid

The solenoid we chose needed to be strong enough to press down on the string to have it make contact with the frets on the neck of the guitar. We did some testing using pressure gauges and determined that we would need approximately 12-15 newtons of force to press down. We decided to use a 25 newton solenoid to make sure that we had more than enough force to

accomplish this. This solenoid had an operating voltage of 12V and a current draw of approximately 1.6A.

Since a solenoid is an inductor inside, we would need enough current flowing through it fast enough to create an electric field to fire it off. This would require us to use a power MOSFET to amplify the current. To expedite the process of our design choice, we tried to use MOSFETS available to us in the NECAMSID lab to avoid waiting for more parts to ship. We needed a MOSFET that could output at least 1.6A of current using a gate to source voltage of 5V. This gate to source voltage value was determined as this is the digital output value the Arduino Mega's digital I/O pins use. The Arduino would be our controller that opened the gate for the solenoid to activate. After exploring our options we chose the IRFZ44Z as it more than met our specifications. The datasheet showed that it outputs about 2A from drain to source with a 4.5V gate to source voltage which was more than enough for our application. This can be seen in the voltage to the current graph in figure 4.

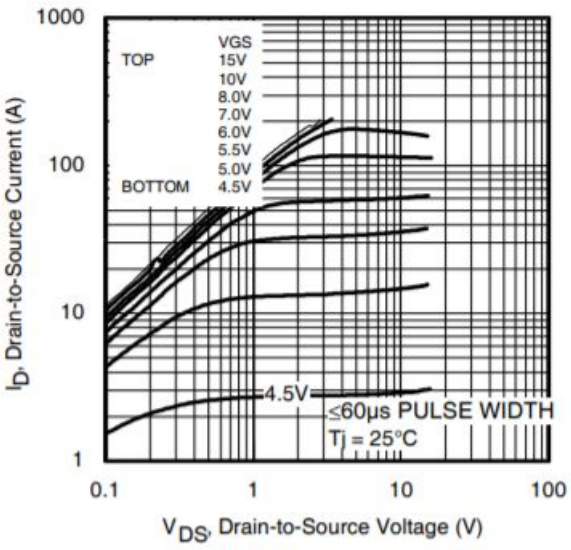


Figure 25: Voltage to Current Graph for IRFZ44Z

The last component needed for the solenoid circuit was a diode. This was used across the solenoid to prevent it from discharging back into our microcontroller. The diode would make the

current discharge across the solenoid in a loop preventing it from going back to the gate. Again using the components available to use we used a basic N4007 diode. The entire circuit layout for a single solenoid can be seen in Section 3.3.1 as Figure 15.

For our guitar design we plan on using 5 solenoids for fretting. We chose this amount for a variety of reasons. We believe that 5 solenoids will give a large range of notes across the A string. We also didn't want to overcrowd the neck of the guitar as the solenoids are large and each require the circuit setup shown in figure 15. Power issues were one of our concerns seeing as each one darain 1.6A. Our solution was to set up the guitar to only be able to use one solenoid at a time. The solenoids use no current when off and when playing a bass guitar you would only use one fret at a time.

4.3.2.2 Plucking Servo

To pluck the string of the guitar we are using a metal gear servo. A servo is a device that utilizes a combination of dc motors, gear train and a potentiometer to drive an output shaft. The servo used in our design can be seen in figure 6. This servo was chosen for a number of reasons. We had originally purchased a smaller servo, but the tension on the bass string was too much for it to pluck. We decided to get a much larger servo that could provide 20kg of torque. The servo has a metal armature that will be attached to the main output shaft. A guitar pick will be affixed to the end of it which will be the contact point to the string. A servo was the best device to use for this application as its high torque and range of motion easily simulates the plucking motion of a guitar pick on a string.

Figure 26: Plucking Servo

The servo has 3 connections, power, ground and the signal input to when it needs to rotate. It operates from 4.8-6.8V and uses pulse width modulation through the signal input wire. The servo uses 1-2ms pulses to rotate within 270 degrees. The servo can rotate 90 degrees on a 2ms pulse and -90 degrees on a 1ms pulse. For the servo's operating voltage it takes 0.16 sec to rotate at 5V and 0.14 sec at 6.8V. The servo will be operated at 6.8V as we want it to be able to pluck as fast as possible with as little larceny. For the purpose of plucking the string we only need to rotate within 30 degrees of motion. With a smaller range, the speed in which plucking can occur is increased and doesn't require multiple servos. The servo will be set up to rotate back and forth to pluck the string to avoid having it rotate all the way around. It will oscillate back and forth within the 30 degree range. The circuit layout can be seen in Figure 17 in Section 3.3.2.

4.3.3 User Interface

The user interface for the Assistive Bass Guitar was created with design goals that would allow ease of use for the user. More specifically, the design needed to be created in order to allow users with Muscular Dystrophy, those with limited motion within their extremities, to readily play the instrument without difficulty. From this, a preliminary concept using a touch screen interfaced using a bluetooth connected chip to the microprocessor was imagined. This would allow the user to connect with the instrument wirelessly from a distance, with minimal motion needed to press the required touch screen buttons to activate the proper notes. With this

in mind, the first step in the process was to acquire and program a bluetooth chip in order to interface a secondary application to the electronic circuit created to play the instrument.

4.3.3.1 Bluetooth Connectivity

The bluetooth module ordered was an HC-06 bluetooth module. This chip was selected due to its ideal low-distance communication of under 50 meters, along with its low-power mode configuration. When acquired, the bluetooth chip was integrated into the solenoid electronic circuit. Testing was done using basic circuit elements, like that of LEDs to test the functionality of the short-range communication. A simple code was programmed to allow the connection of the bluetooth chip to set the digital out pin 9 of the Arduino Nano to turn on the LED by setting the Digital Out pin to a high state. This testing went as planned, and allowed for the next step in the design process.

The HC-06 module transfers data at a 2.4 Ghz band and has a Baud rate set to 9600. To find the sampling frequency, we will use the Baud rate and the number of integers we send over the network which will allow us to get the speed at which the data is sent. The HC-06 transceiver is connected to the VCC 5-volt operating pin of the Arduino, along with the GND pin of the Arduino. Also, the Tx and Rx pins are connected to allow communication between the two devices.

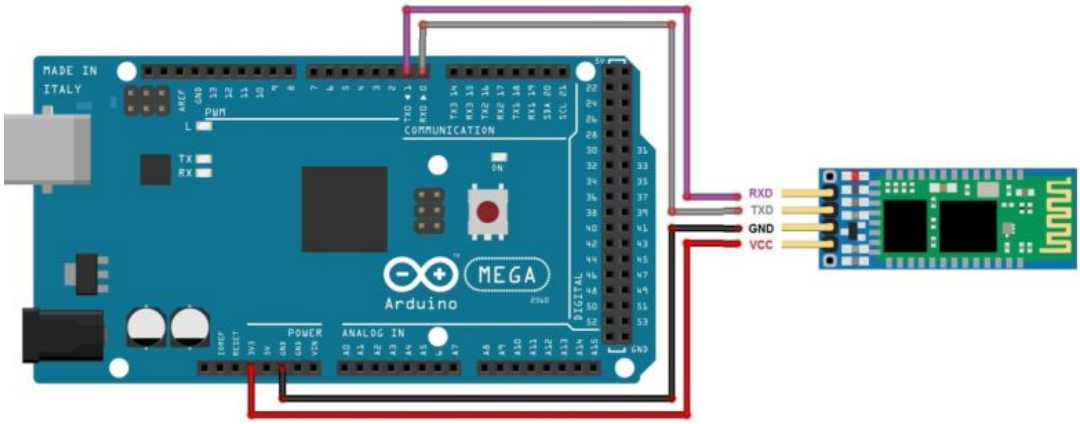


Figure 27: Connection of HC-06 Bluetooth module to an Arduino Mega

The figure above demonstrates the connections that are set up between the HC-06 and the Bluetooth transceiver. The Vcc, or 5v source, is connected between the devices, along with the GND pins on both. Also, the Tx and Rx pins on both devices are connected to each other, which allows the transmitting and receiving of data between the devices.

4.3.3.2 Android Application

The next step involved creating an application to communicate with the bluetooth module. To create the app, research was done to find an environment that would allow our team to create a simple application in order to allow the user to press a button that would correlate to a solenoid hitting a note on the bass guitar. This research led to the adoption of using the environment MIT App Inventor, which is a programming environment that allows the creation of a wide range of applications that can be connected to an arduino using bluetooth. Once this was chosen to program the interface, preliminary code was created to test the functionality of the application.

Since our team has limited experience in designing and creating mobile applications, much research went into studying and learning the MIT App Inventor environment, with multiple tutorials being completed in order to become familiarized with the process. From this, the code to interface the app with the bluetooth chip was learned. From here, the next step in the design process was undertaken. This involved creating buttons to correlate with each solenoid. These buttons were programmed to each respective digital out pin from the Arduino, allowing separate solenoids to be activated based on the button pressed. For testing purposes, one button was programmed and connected through bluetooth commands within MIT App Inventor to test functionality and feasibility.

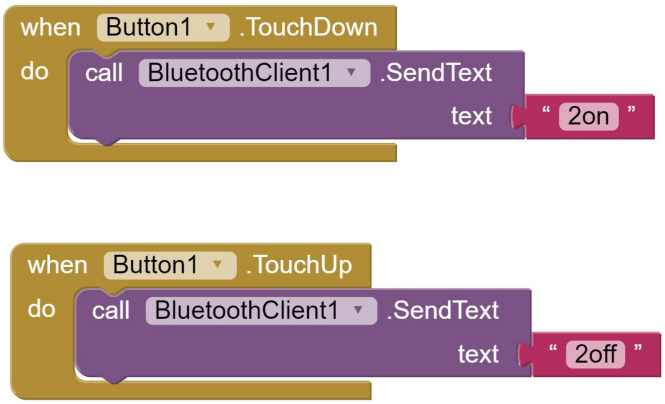


Figure 28: Sample of Android Application code within MIT App Inventor IED

The figure above is a section of the code written using the MIT App Inventor to interface an android application to the Arduino Mega using the bluetooth module. The code is repetitive in nature, as each piece calls for a button press to correspond to sending a "digital high" or "digital low" signal to the microprocessor.

For testing purposes, a button press would allow the solenoid to activate over bluetooth, allowing latency tests between the app and the arduino to analyze the latency between the app and the firing of the solenoids. This latency was problematic at first, as a significant delay was found between the button press and the solenoid activating. To fix this issue, the program created was re-organized to allow minimal use of delays within the Arduino code, along with setting the delays to the lowest possible delay setting without causing activation issues with the solenoid. This fixed the latency issues, and allowed the final stages of the app to be designed.

The final steps involved repeating the code to allow for all the solenoids to be activated with a button press, with five buttons being programmed to correlate with the five different solenoids. This was an easy process after the first button was programmed, as the code was just repeated with each separate button within the MIT App Inventor. Once this was complete, the aesthetics and functionality of the app was worked on to allow the user to easily be able to press the buttons with minimal motion of their hands. This was done by increasing the button sizes and spacing the buttons in a way that matches the shape of a hand. This allows limited motion from the user, and also limits the room for error of a button press due to the increased size of the

buttons. Lastly, a dark theme was applied to the application mainly for aesthetic purposes to allow the user to clearly see the buttons, with the text within the buttons that indicated the note that button controls.

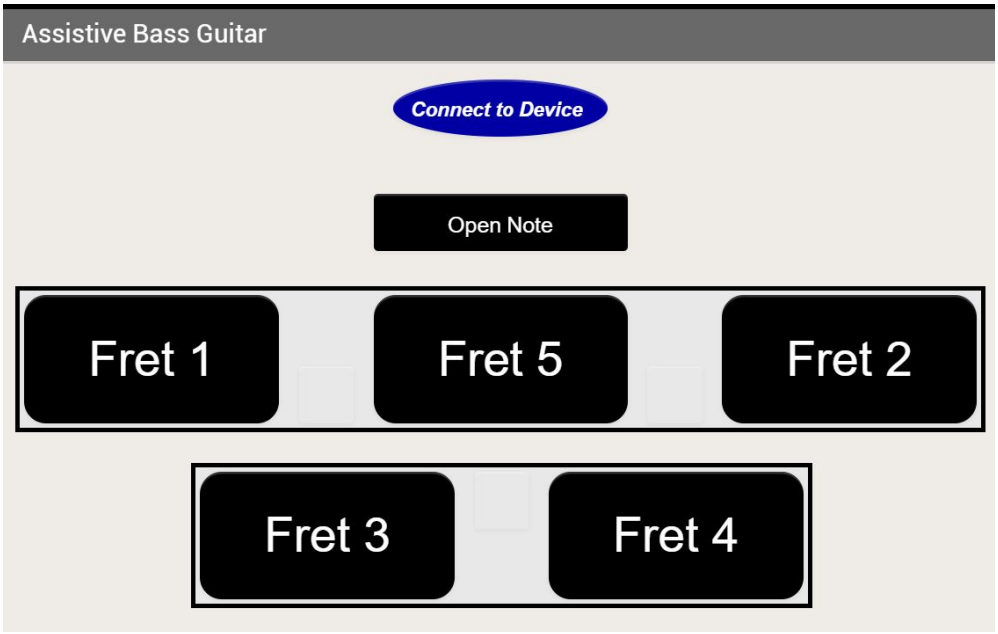


Figure 29: Snapshot of designer block of application

Figure 10 shown above is a preliminary version of the android application created for the assistive bass guitar. A connect to device button is placed in order to connect the device to the arduino using the HC-06 module. The open note button is used for tuning purposes, as the servo will pluck the string in order to find the frequency of the string in its current state.

At the completion of the testing period of the application, an android tablet was acquired to allow the app to be downloaded and installed. A Samsung Galaxy Tab E was used to download and install the created application, and testing was done to make sure the application fit the screen size of the tablet. MIT App Inventor has tools to auto-fit the application to the connected device, along with screen-rotation tools, so not much programming needed to be done in order to fit the bigger tablet display compared to that of the android phone previously used to test the application. With the finished application running on the tablet, the next step in the

design process was to have a person with limited motion in their hands to test the design and provide feedback to test whether the application created allows the user to easily control the instrument through the mobile application.

4.3.4 Signal Analysis

To tune the string automatically, the small AC signal from the pick-up on the bass must be amplified, biased, and filtered so that it can be read on an analog input pin of the Arduino Uno. This signal must then be analyzed within the arduino code to determine the frequency of the string. This data is then used to control the tuning hardware described in figure 11.

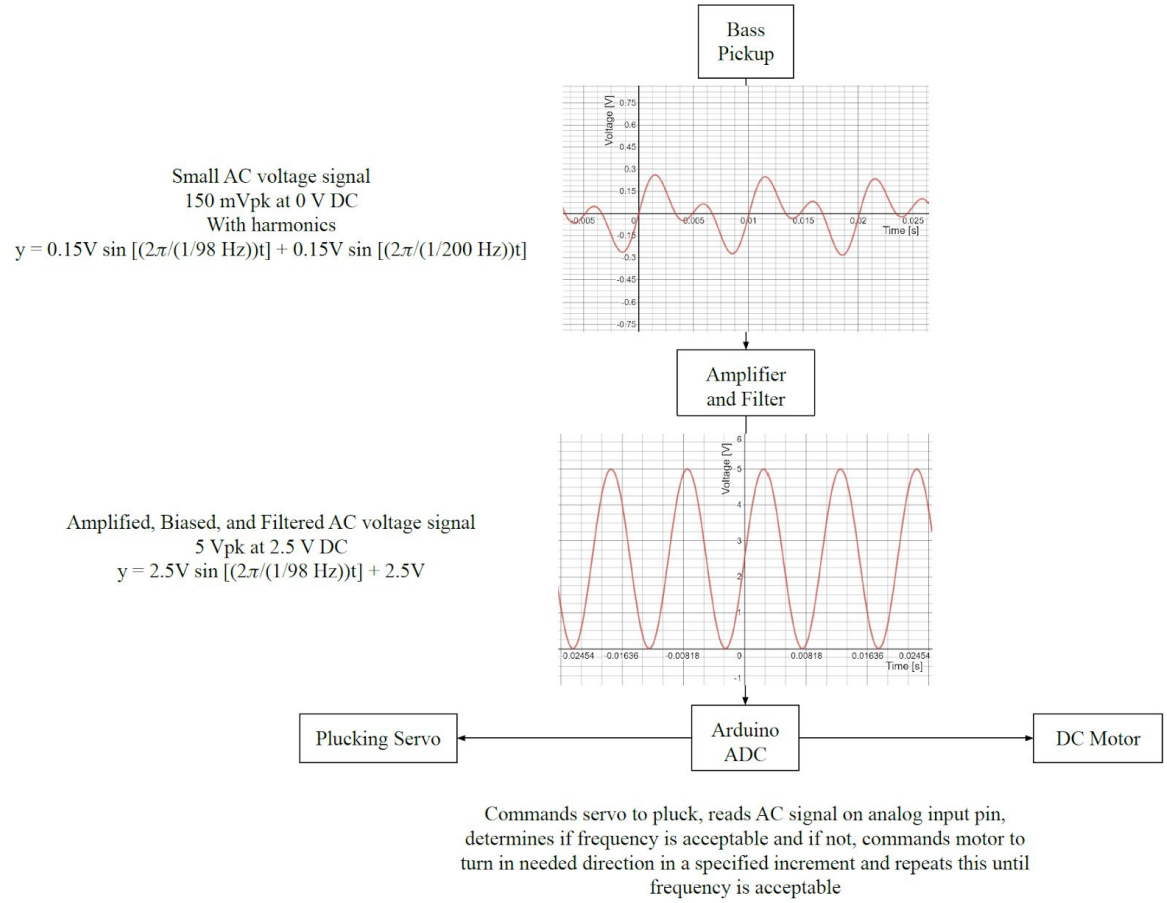


Figure 30: Ideal Tuning Block Diagram

The amplifier block consists of an inverting operational amplifier with negative feedback and a DC biasing network to take the small AC signal at 0 V DC and convert it to a 2.5 V peak signal at 2.5 V DC. It also contains a low-pass filter with a cut off frequency of 170 Hz. This is a reasonable cut off frequency since the string of the bass guitar being used is A, of 55 Hz.

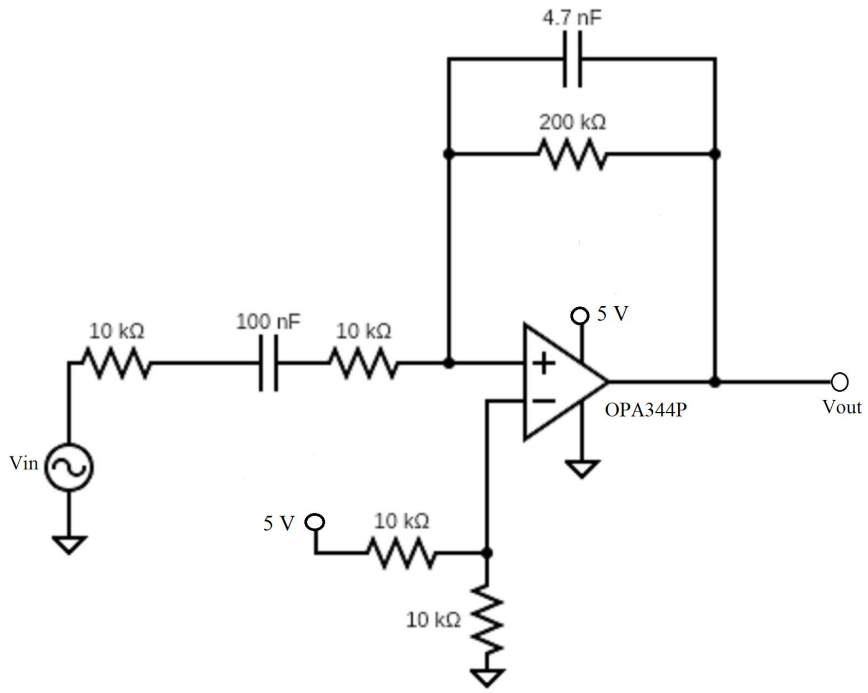


Figure 31: Signal Amplifier and Filter Circuit Schematic

This analog voltage signal, once within the 0 to 5 V range of the arduino, is fed to an analog input pin. Within the arduino, this signal is analyzed for its frequency. This analysis is conducted by sampling the signal at a specified sampling frequency given the equipment used. The sampling conducts peak detection, meaning it detects the peaks of the signal. The algorithm used to determine the frequency is called autocorrelation (see Appendix A for the source code used). This means that the arduino creates a copy of a portion of the input signal to be analyzed, and then measures the correlation between the two signals at the peaks by adding their amplitudes. At correlated peaks, the sum of the amplitudes is very high. The arduino then moves the copied signal by a determined period (determined by the sampling frequency) and at each

sample, sums the signals. It continues to do this until the entire portion is analyzed. The resulting sum waveform contains periodic impulses created by the correlated peaks. The frequency of the signal is then equal to the sampling frequency used divided by the period of these peaks (Reliable Frequency Detection Using DSP Techniques).

Once the signal is analyzed for its frequency, this frequency is compared to the desired frequency of the string, 55 Hz. Depending on the relationship between the measured frequency and the desired frequency, the arduino communicates with the motor to turn the tuning peg in a specified direction for a predetermined increment.

The arduino then commands the plucking servo to pluck again, and this signal analysis process is repeated until the frequency is within the desired range.

4.3.5 Tuning Hardware

Tension Mechanism - The mechanical electrical design of the automatic tuning portion of the project consists of two main components, a geared DC motor and a modified bass tuning peg. The concept of tuning a bass guitar is contingent on one concept, holding tension and therefore staying in tune. A regular bass guitar is capable of holding tune through the use of the worm gear design of the tuning peg. Knowing this, we felt as though it was most logical to modify a typical tuning peg, and adapt the gearset to be applied to the DC motor. The motor used was chosen for its low speed (RPM) and high torque gearbox output. The motor alone could not be directly matched to the tuning peg. Using a basic one-to-one motor shaft coupler, we were able to directly connect the motor's rotor to the shaft of the tuning peg where the butterfly knob would typically be placed. In doing so there is a two part step down of the original DC motor's output. First through the designed gearbox on the end of the motor, and second through the step down gear set of the worm gear. By combining the lockout properties of both pieces, there is no need for constant power to the motor. Once the string is within tuned frequency, power can be removed and the string will hold tension.

Motor Control - As described in the previous section on signal analysis, the motor is controlled by the arduino uno. In order to allow the motor to turn in either direction, an H-bridge is used. The H-bridge module in our design is a HiLetgo BTS7960 43A High Power Motor Driver Module, as shown in Figure 13.

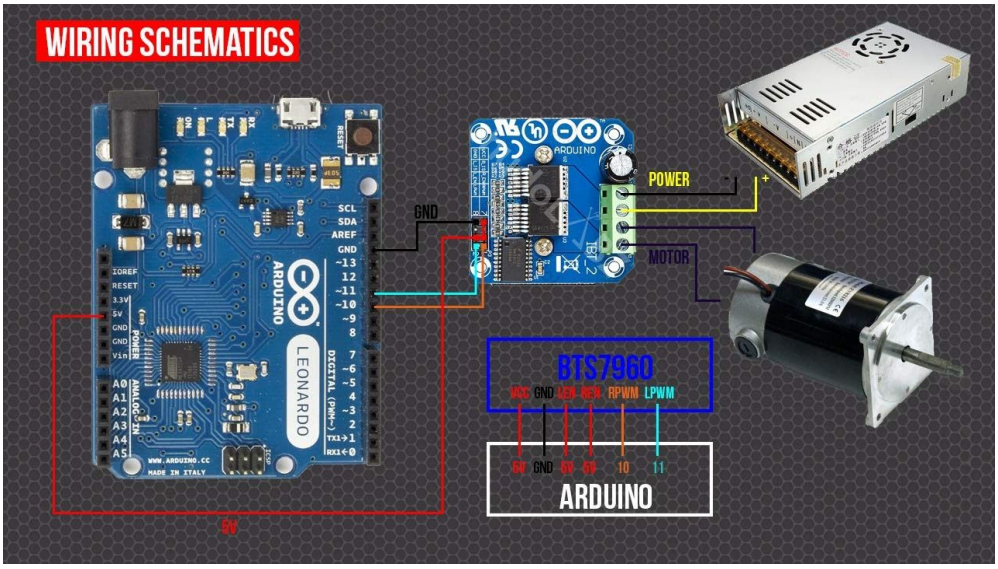


Figure 32 : H-bridge module schematic

An H-bridge is a circuit which switches the polarity of an applied voltage. They are often used in applications to allow DC motors to run forwards and backwards. A basic representation of an H-bridge is provided in Figure 14. When switch 1 and 4 are closed, the current flows through the motor from left to right, and the motor will start spinning in one direction. If switch 2 and 3 are closed instead, current will flow through the motor from right to left and the motor will spin in the opposite direction.

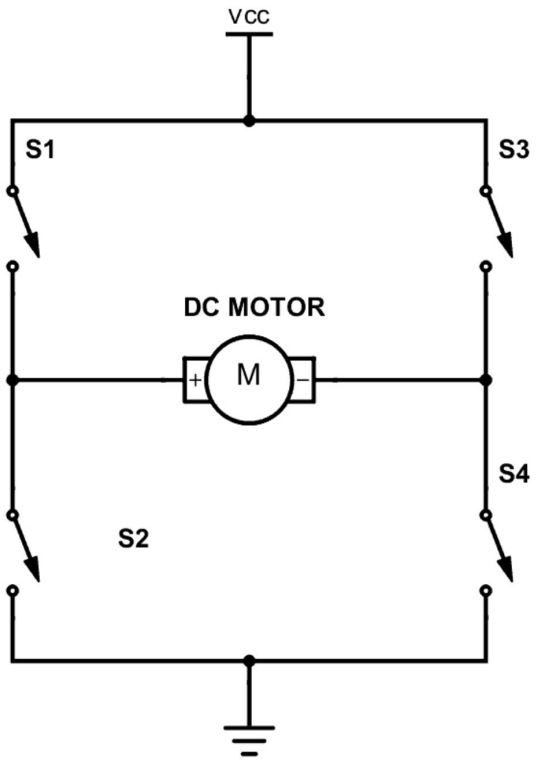


Figure 33: A basic H-bridge representation

The duty cycle of the PWM signal determines the speed of the motor, but we leave this at 50% duty cycle because we are not concerned with speed as we are determining the amount of tuning by delays within the code and want the speed to stay constant. The code which the arduino uses to control the motor is simple. If the frequency is lower than desired, the arduino will send signals to the h-bridge which will cause the motor to turn forwards. If the frequency is higher than desired, the arduino will send signals to the h-bridge which will cause the motor to turn backwards (see Appendix B for preliminary source code).

4.3.6 Additional Design Features

Mechanical Design - The mechanical design of the project aimed to create an authentic experience, allowing users to feel as in touch with the original instrument. For this reason an actual bass guitar body and neck were used, in addition to primarily using as much wood in construction as possible. Wood was also the ideal choice in the construction of the assistive guitar as it allowed the team to easily construct the device and manipulate the design according to our needs.

Platform - In order to host the electronics necessary for creating the assistive bass guitar, a solid foundation for the bass and supporting components was necessary. A quality piece of hardwood was used as a base for the entire system, allowing for proper mounting of the bass and solenoids. Each solenoid received its own pedestal carefully spread out over the length of the neck of the guitar. The solenoids were screwed using their proper mounting holes into the 1x4" pedestals, which were subsequently attached to the base using small L-brackets.

Bass - The bass itself was attached to the base carefully by using a combination of cut construction lumber, L-brackets, and screws. Two primary arms were attached to the hardwood base. These acted as a guitar stand in which the bass guitar was permanently connected to. Larger L-brackets were used between the arms and the base in order to ensure that the heavy weight of the instrument would not result in any shifting or possible damage due to moving the instrument. The only true modifications to the chosen bass guitar were the two sets of pilot holes drilled in the back of the body, and the removal of the butterfly knob from the string tuning peg. Looking at the project as a whole, the design is most similar to an exoskeleton for a bass guitar, allowing a user to adapt most guitars within reason to the platform.

4.4 Summary

The section written above describes the approach and mindset of the team in designing and creating the assistive bass guitar for those with muscular dystrophy. In our approach for the project, we assigned each team member a different aspect of the project. These general sections

were the hardware inputs and outputs, software inputs and the user interface, signal analysis and tuning hardware and mechanical design. Each member specializes in one of these sections, however all team members worked on all aspects of the projects. This sectionalizing was mainly for organizational purposes to make sure tasks and objectives were met in each part of this project. Due to this, both the hardware, software, signal analysis and mechanical aspects of this project were done in unison, as to progress the project in a timely manner.

5. Realization and Results

5.1 Finalized Design

The final design of the prototype developed in parallel with the development of each individual testing module of the project. Primarily, the design consists of a typical bass guitar mounted horizontally along a base of perforated board. Constructed to hold the guitar upright, a pine “guitar stand” is directly screwed and adhered to the project base using L-brackets and heavy weight epoxy. Each of the five solenoids is epoxied directly to pedestals of pine, which are upright and placed along the neck of the guitar. The tuning motor has a custom bracket built at a 90 degree angle which is directly screwed into the head of the guitar. Mounting for the plucking servo was completed by mounting a custom 3-D printed box to the face of the guitar with wood screws. Construction was completed using assorted hardware and heavyweight gorilla glue epoxy.

5.1.1 Variations from Concept Design

During the initial brainstorming period the envisioned design of the project quickly changed from a flat laying guitar approach to the horizontal upright project that the team pursued. Major changes to the project occurred during the development of the individual testing modules of the project. Major design changes were primarily in regards to the tuning motor and servo. Initially it was thought that the motor itself would directly connect to the string of the guitar, and there would be a direct mounting point to the base of the project. The team quickly

found that modifying a bass tuning peg was a far more realistic approach and allowed for directly mounting the motor to the head of the guitar. The servo enclosure developed when the need for a secure mounting position became a roadblock for the project, and self plucking was no longer applicable to testing the project systems. Initially the team thought that the servo would be placed on a pine pedestal such as the solenoids, but placing the enclosure directly on the guitar allowed for a far more aesthetically pleasing look, and allowed for a far more predictable and consistent plucking mechanism.

5.2 Testing Period

After the team finalized the overall design, and determined the plans of action in order to implement each module of said design, the prototyping period of the process began. This stage of the project included the building of both the electrical, mechanical and software portions of the project. Each module was split into different groups, where each team member was assigned to a different aspect of the design to help expedite the building and testing process. At every stage, testing was done separately on each system to ensure the proper working of each system. At the end of the testing period, each module was combined together in order to have a complete working prototype assistive bass guitar. In the sections that follow, each area of the project is described in depth regarding the testing and realization of the project and its coming to fruition.

5.2.1 Tuning Implementation

After much testing and configuration the guitar is now able to tune itself to a desired note. We chose the A-String for the single string to be played as it offered a large range of notes and was not too hard to tighten. As described in the methodology section, the string is plucked and passed through a low pass filter to the arduino where it runs a function to determine the main frequency of the signal. This filter can be seen in figure XXX. The filter was set to have a gain of 20 since the incoming signal of the guitar was about 150mV and we needed to make the signal much larger to read it. The signal wave can be seen in the figure below.

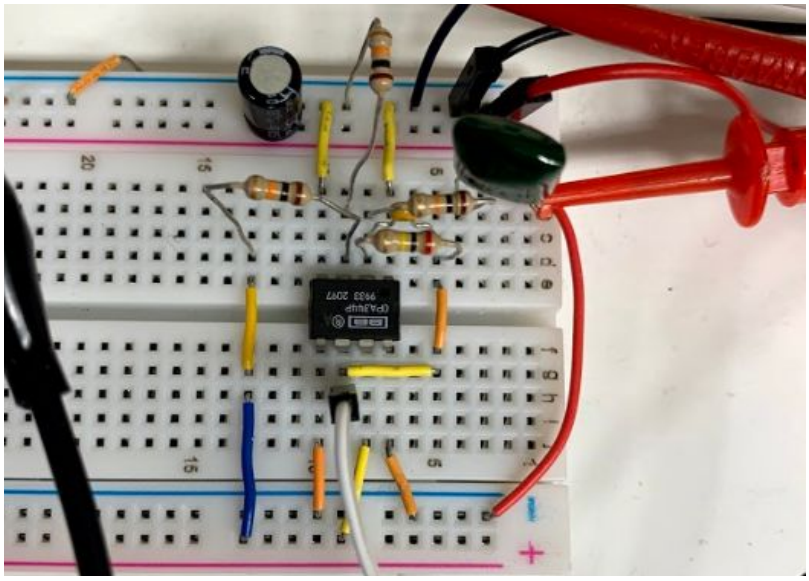


Figure 34: Low Pass Filter Circuit

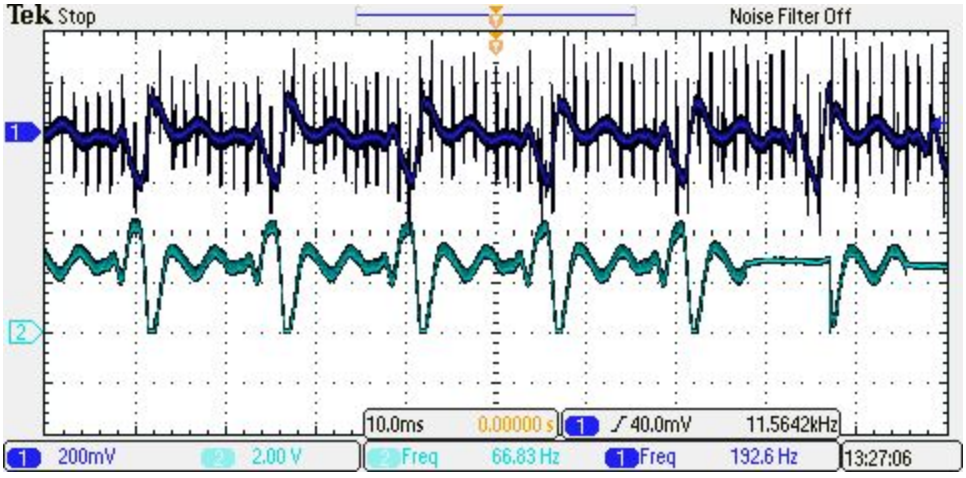


Figure 35: Input Signal vs. Output Signal

Since we are using an A-String we want it to be tuned to 55 hertz. The arduino will first detect the initial frequency. If it is out of tune, it will first determine the frequency and turn the motor attached to the tuning peg to either tighten or loosen the string. The motor was programmed using different increments of time to get the string in tune. For example, if the frequency detected is 40hz and our target frequency is 55hz, the motor will turn for 1.5 seconds counterclockwise to tighten the string. The amounts of time for each frequency were determined through multiple tests to find the correct string tension. The program is then set up to test the strings frequency again to make sure it was correctly tuned. The times set for each frequency will tune within two hertz however, some factors such as the string slipping may throw this off. Therefore, it will continue to test the string until it is in tune. If it is already within two hertz of the desired note it will not activate the motor and turn on an LED indicating that it has exited tuning mode. Since the tuning is done before playing, time was not an issue to need the motor to spin faster.

When plucking the string, we found that the signal became harder to isolate as the string lost energy. This loss in energy makes the algorithm detect larger values since it is trying to compensate for this change. Therefore, the code is set to filter out the outliers and only record the main frequency. The guitar was also altered slightly to better detect the string's pluck. The guitar's pickup was moved closer to the string and the string was lowered down so it would be detected better.

5.2.2 Mechanical Work

Solenoids

The solenoids were adhered to pedestals placed along the neck of the guitar. Due to the tapering of the guitar string from one end to the other, there is a variation of 3mm between the pedestals to allow all five solenoids to hit approximately in the center of the string. The pedestals are made of pine due to it being one of the best softwoods for construction. The solenoids are very simply epoxied to the tops of the pine blocks, and the bottoms are connected using both L-brackets and epoxy. Using L bracts ensures that the pedestals stand square to the neck of the guitar. Each solenoid has a slightly modified neck consisting of larger hardware. Increasing the

cross-sectional area of the solenoids contact point allows for more even pressure being applied to the string, and more tolerance in regards to the placement of the solenoids.

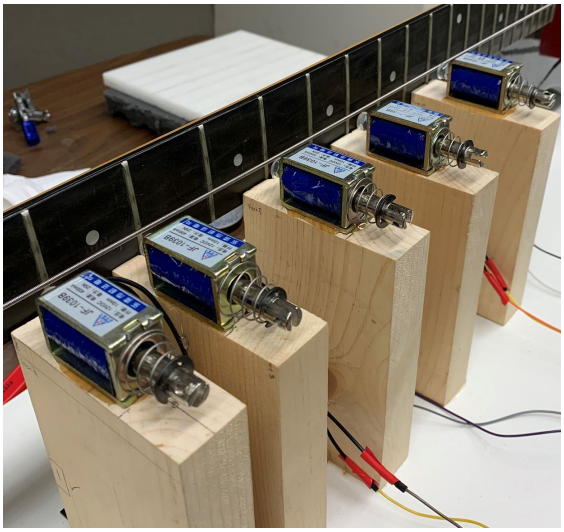


Figure 36: Solenoid Placement on Mounted Pedestals

Motor

The DC geared tuning motor is attached to the neck of the guitar using strictly metal hardware. An L-bracket was used to create a 90 degree angle between a mounting bracket consisting of the supplied motor mount and a strip of metal used as the base. The motor itself connects to a modified bass guitar tuning peg with the butterfly knob removed. The coupler used between both shafts is a 6mm solid motor couplers. The side which directly connects to the tuning peg is milled out for 6.456mm and tapped through to accept M3 metric hardware. A lathe was used for all milling to ensure that the solid coupler would provide a straight connection between both shafts.

Figure 37: DC Geared Motor Mount

Servo

The metal geared servo required two major considerations, a mounting point and necessary noise damping. An enclosure was custom made to allow for direct mounting to the bass guitar. The enclosure was designed in solidworks, exported as an STL file and printed with support material on a 3D printer. The design allows for 1/4in gaps on four of five sides of the servo which were filled with high density sound damping foam. The 5th side had an 1/8 in gap filled with foam. The servo itself had its metal arm slightly modified to accept a guitar pick on the end. All hardware used on the servo was M3 metric hardware.

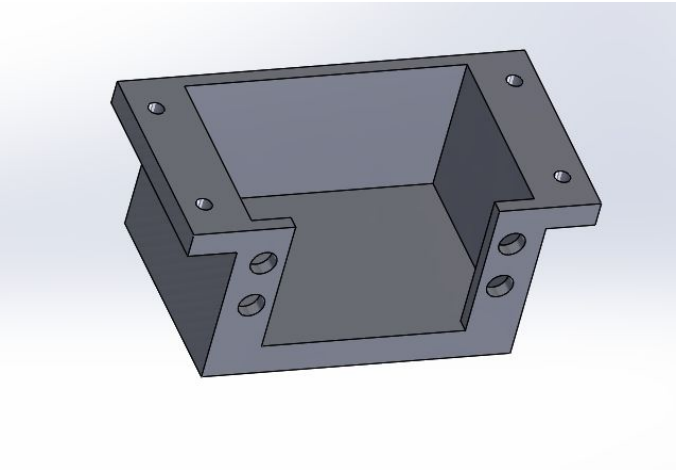


Figure 38: Solidworks File of Servo Enclosure

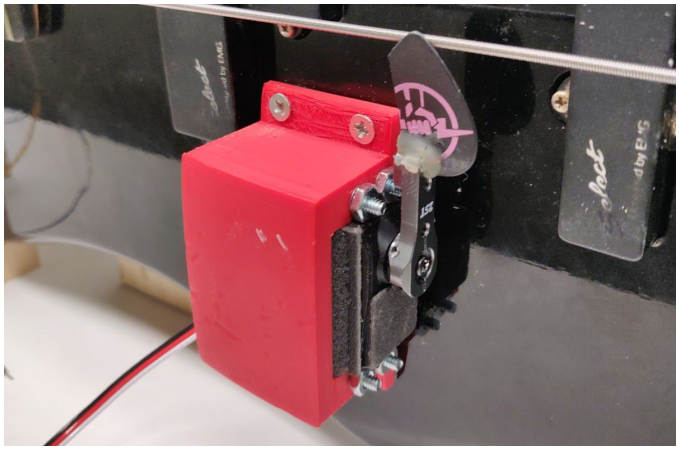


Figure 39: Printed Enclosure Mounted with Servo and Foam

Stands and Mounts

In order to mount the guitar, pine was cut to form a horizontal stand for the guitar. The tops of the stands were cut to mimic the natural curves of the guitar for aesthetic reasons. In order to mount the pine to the project base, large L-brackets were used. At the front of the guitar two small blocks of pine ensure that the guitar sits upright at exactly 90 degrees. To mount the guitar to the pine stands, 1 ¼ in wood screws were used to drill all the way through both bodies of wood.

Figure 40: Pine Guitar Mounts

5.2.3 Circuitry

The design includes six main blocks: power switching, signal processing, motor driving, bluetooth connection, solenoid driving, and servo driving. The more detailed blocks will be explained in the following sections. Below is a top-level block diagram of the circuitry of this project. It should be noted that the solenoid block in the top level diagram is multiplied by the 5 solenoids of the design.

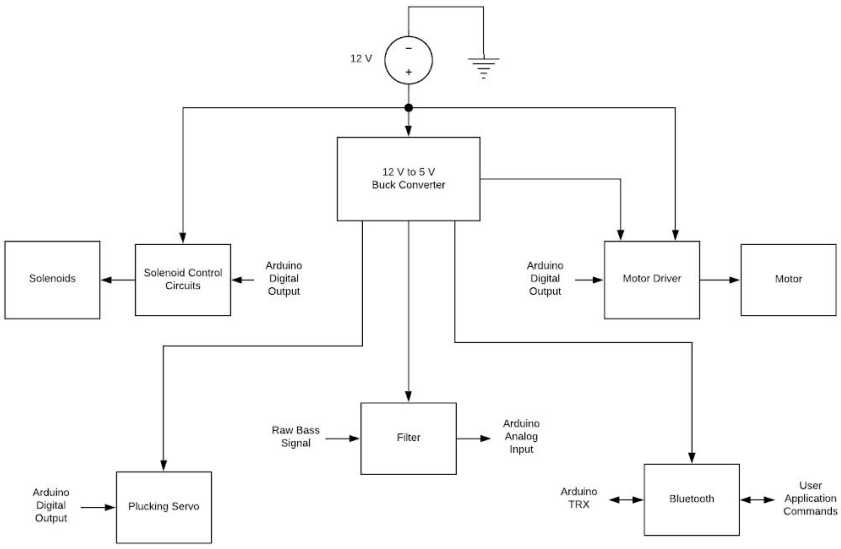


Figure 41: Top Level Block Diagram of Circuitry

Filter

The filter block of our design contains an active low-pass filter with a gain of 20 and a cut-off frequency of 170 Hz as it is intended to take a 150 mV peak input signal from the bass pick-ups and output a filtered signal ranging from 0 to 5V. A diagram of the filter circuit used is provided in the signal analysis portion of the methodology section of this paper as Figure 33. We measured the internal source resistance of the pickup and modeled it by a 10 kΩ resistor.

The operational amplifier used in this block is the OPA344P by Texas Instruments and has the following simplified block diagram.

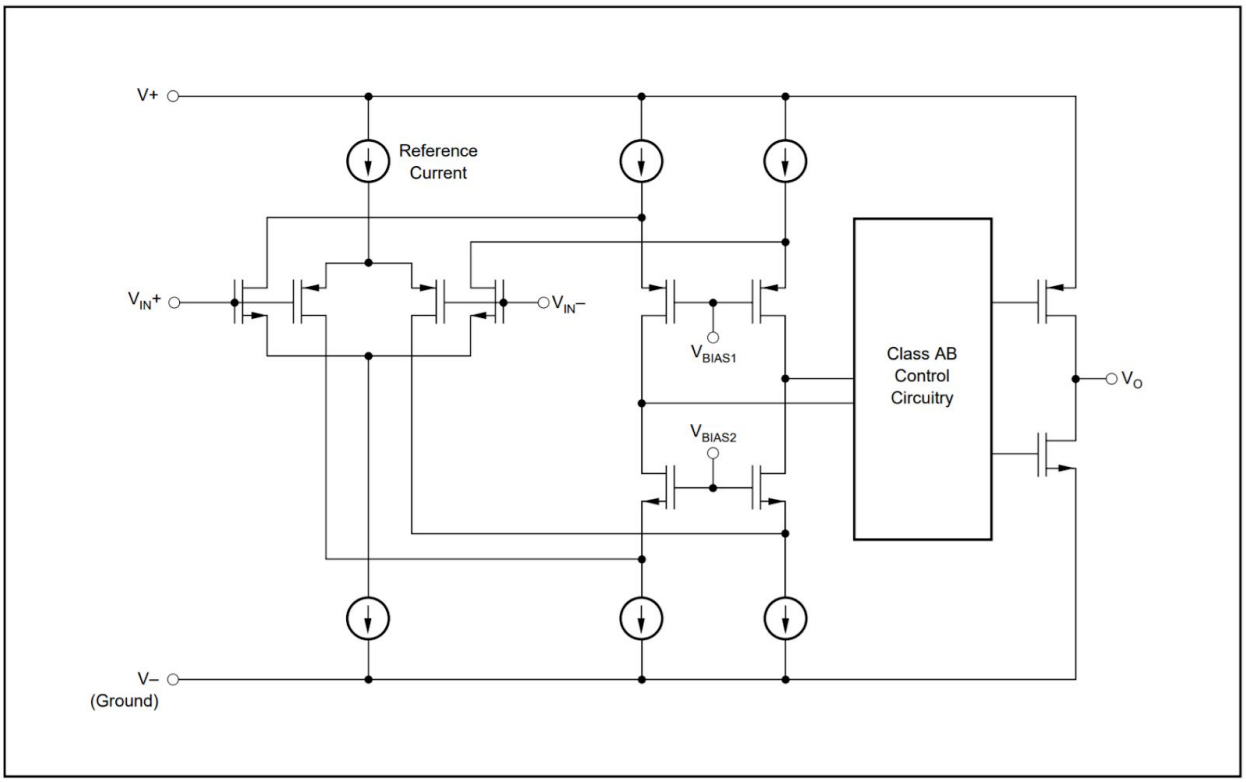


Figure 42: Operational Amplifier Block Diagram, OPA344P [33]

In the implementation of this op-amp, we use the five volt rail supplied by the buck converter module as V+ and ground as V-. Our non-inverting input is the raw signal from the pick-ups of the bass guitar and our inverting input utilizes a voltage divider to pull the output signal above ground and set the DC voltage reference to 2.5 volts. The use of the 200 kΩ and 10 kΩ resistors creates negative feedback which sets the closed loop gain of the op-amp to 20.

$$\text{Closed Loop Gain} = A_v = \frac{V_{out}}{V_{in}} = -\frac{R_f}{R_{in}} = -\frac{200\ k\Omega}{10\ k\Omega} = -20$$

We don't mind that the signal is inverted since this doesn't affect the frequency of the signal, which is the focus of our signal analysis. The 4.7 nF capacitor in the negative feedback circuit sets the cut-off frequency to 170 Hz.

$$\text{Cutoff Frequency} = f_c = \frac{1}{2\pi RC} = 170\ Hz = \frac{1}{2\pi(200\ k\Omega)\ C} \rightarrow \text{C} = \frac{1}{2\pi(200\ k\Omega)(170\ Hz)} = 4.7\ nF$$

The resulting success of this circuit is shown in Section 4.2.1.

Power Switching

The majority of our circuitry is powered by five volts, so we employ a buck-converter module to step down a twelve volt supply and create a five volt rail. The module we use is part number 106990003 from Seeed Technology Co., Ltd. The module utilizes the LM2596 Step-Down Voltage Regulator from Texas Instruments. A schematic of the LM2596 is shown below.

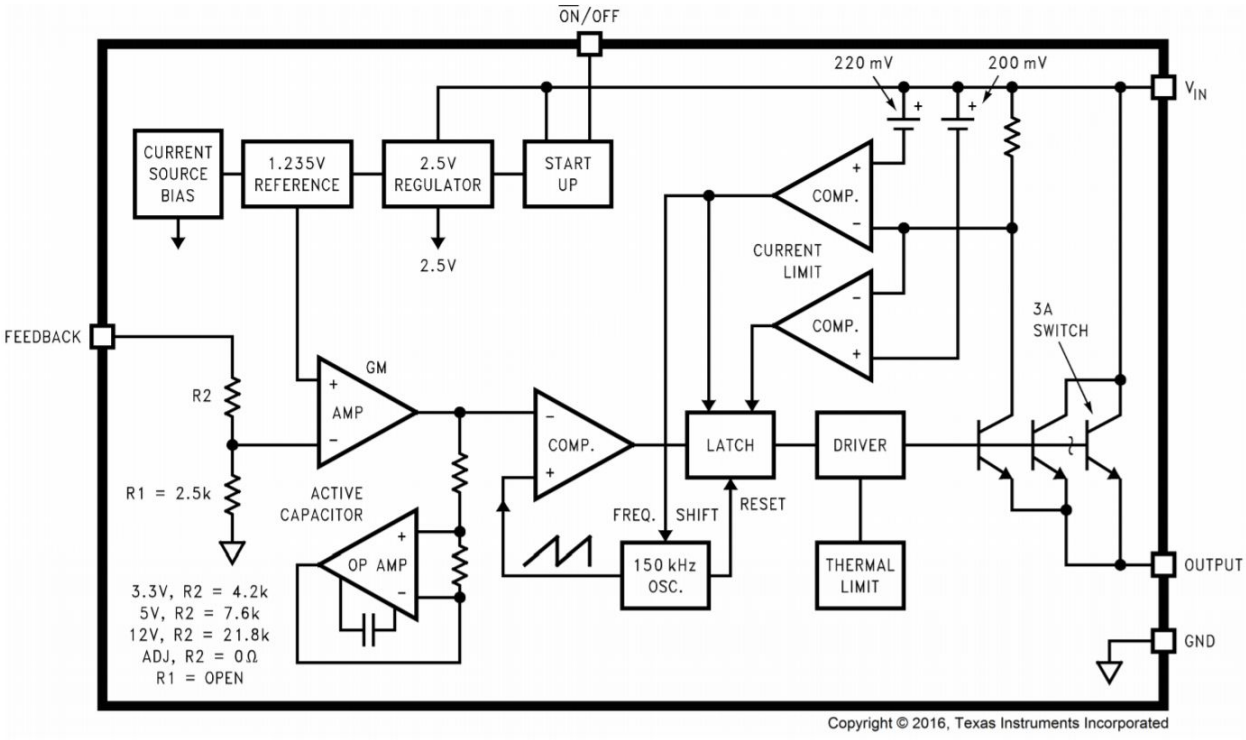


Figure 43: Step-Down Voltage Regulator Block Diagram, LM2596 [30]

The module follows the adjustable output application of this chip, as provided below by Texas Instruments. Using the potentiometer located on the module (shown below as R2), we set the output voltage value to be 5 V.

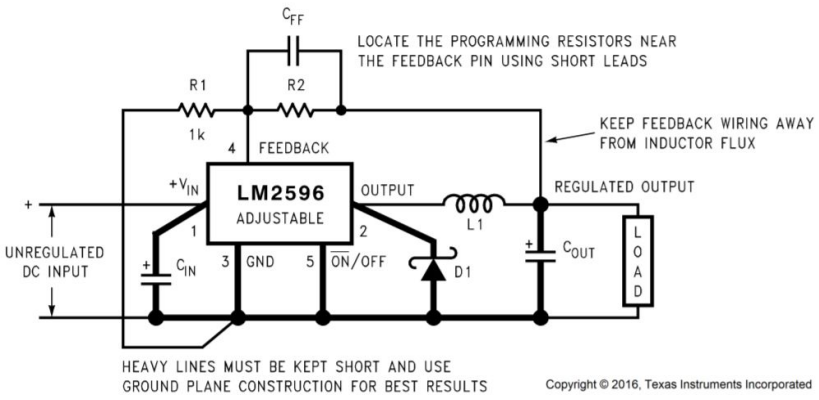


Figure 44: Step-Down Voltage Regulator, LM2596 Adjustable Output Application [30]

Motor

In order to control the direction of the DC motor in our tuning block, we utilize a motor driver module. This module, its connections to the motor, and its connections to the Arduino are provided below.

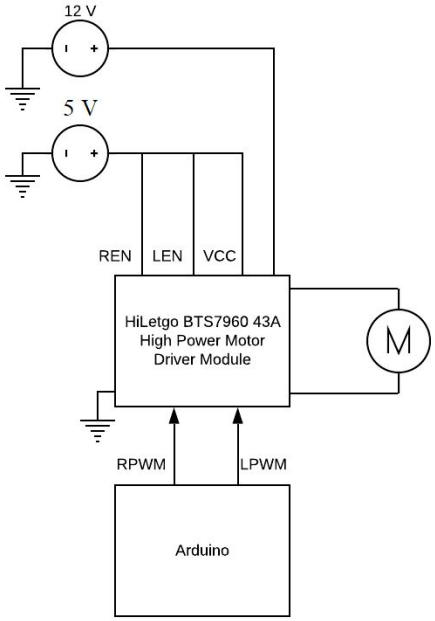


Figure 45: Motor Control Diagram

Our team utilized a HiLetgo BTS7960 motor driver module to drive the tuning motor.

The schematic of the motor driver module is provided below:

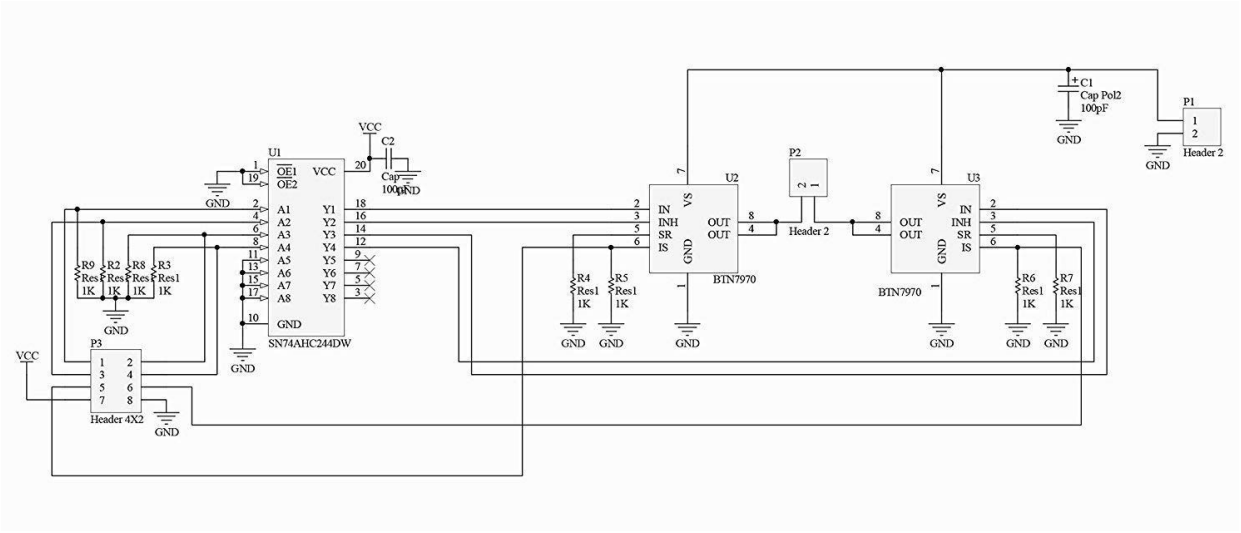


Figure 46: Motor Driver Module Schematic [31]

In the motor driver module schematic above, VCC is five volts and the second header is supplying twelve volts. The two components titled BTN7970 are each PN half bridges created by Infineon. The block diagram of one half bridge is shown below.

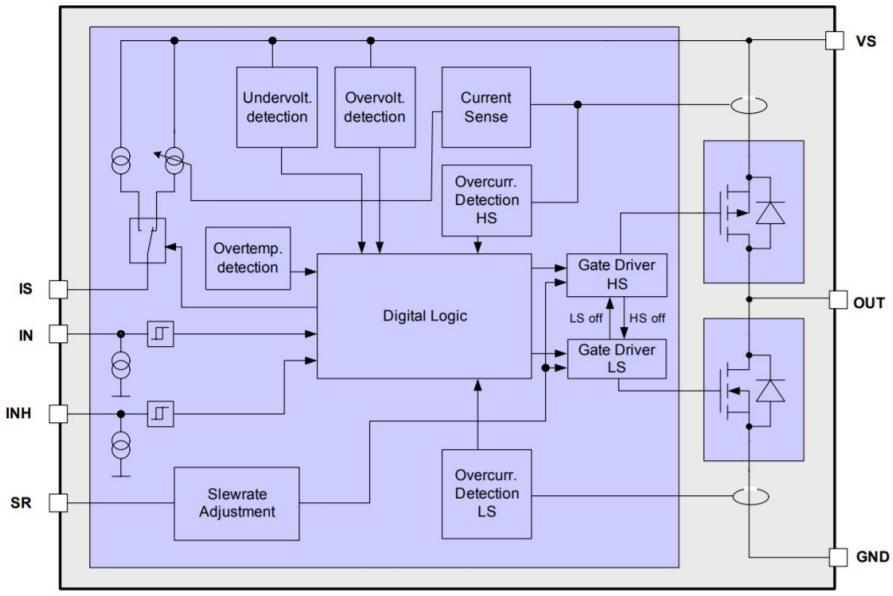


Figure 47: Infineon BTN7970 PN Half Bridge Block Diagram [32]

Bluetooth

The HC-06 bluetooth module is powered by six volts. The connection between the Arduino and the HC-06 bluetooth module is outlined below.

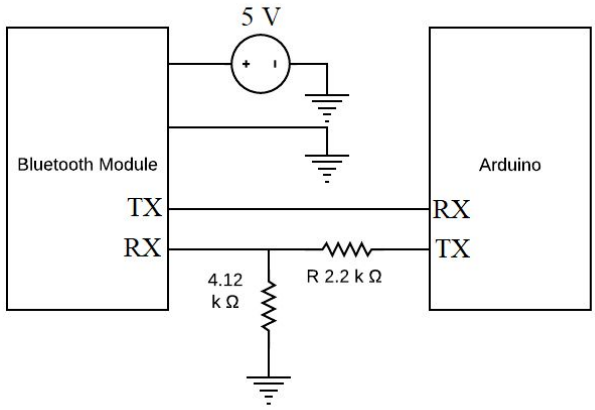


Figure 48: Bluetooth Module Schematic

The HC-06 uses 3.3 V logic, but the Arduino uses 5 V logic. Because of this, a voltage divider is used between the transmit of the Arduino and the receive of the bluetooth to step the 5 V logic down to 3.3 V.

$$V_{out} = V_{in} \frac{R_2}{R_1 + R_2} = (5\ V)\ \frac{(4.12\ k\Omega)}{(2.2\ k\Omega) + (4.12\ k\Omega)} = 3.3\ V$$

Servo

The plucking mechanism of our system consists of a servo motor which is powered by our five volt rail and is connected to the Arduino through a digital pin for controlling purposes. A schematic of this block is provided below.

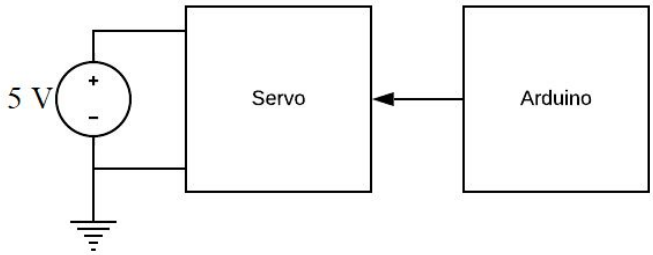


Figure 49: Servo Schematic

Solenoids

The fretting of the string is executed by five push-pull solenoids, one for each fret of our design. Each of these solenoids is controlled through the gate of a MOSFET connected to a digital pin on the Arduino with a diode connecting the source to the twelve volt supply. This ensures voltage only flows into the positive bank to account for kickback from the solenoid when it turns off. One of these circuits is outlined below, as each solenoid block is equivalent.

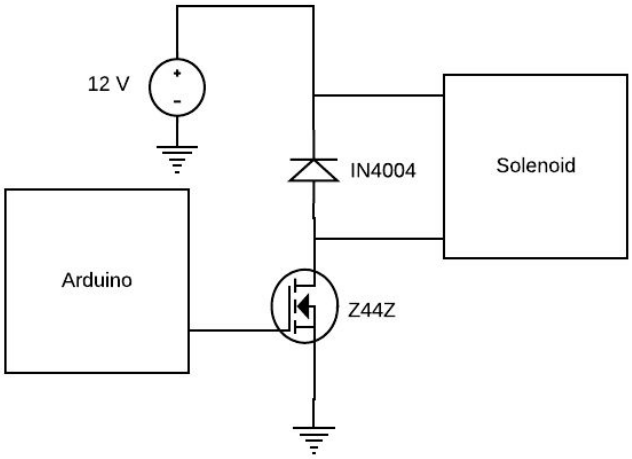


Figure 50: Single Solenoid Schematic

5.2.4 Mobile Application

As stated in the background section, we utilized MIT App inventor to design and create a mobile application to control the guitar through the use of an android tablet. The app was designed to send a string data type to the arduino via bluetooth to control a digital logic output to each of the guitar's components. The program was very versatile in that it was able to operate with the bluetooth client of our chip, matching its baud rate automatically to communicate. The app was run off of a server that transmitted data to our tablet in real time. We were able to edit the app without needing to start a new session each time. When we were satisfied with the final product it allowed us to change it into a stand alone app that would work separately from its server.

The app was set up to have multiple buttons for guitar control having a button for each of the five frets, one to play an open note and one button to tune the guitar. The implementation of the buttons was very straightforward using block like coding to program each one. The coding format for the app can be seen in figure 54. Each block was set up to send a string to the arduino. The arduino utilized the loop() function to continually check incoming strings and compare them to carry out its various functions. For example if the user wanted to press down on the first fret, they would press the "Fret1" button and the arduino would send a string value of "2on" to be compared within the code. ("2on" was used instead of "1on" since the 2 indicates the digital I/O pin number that controls the first fret) There were the additions of other buttons for testing purposes such as direct control of the motor to be able to put the string out of tune.

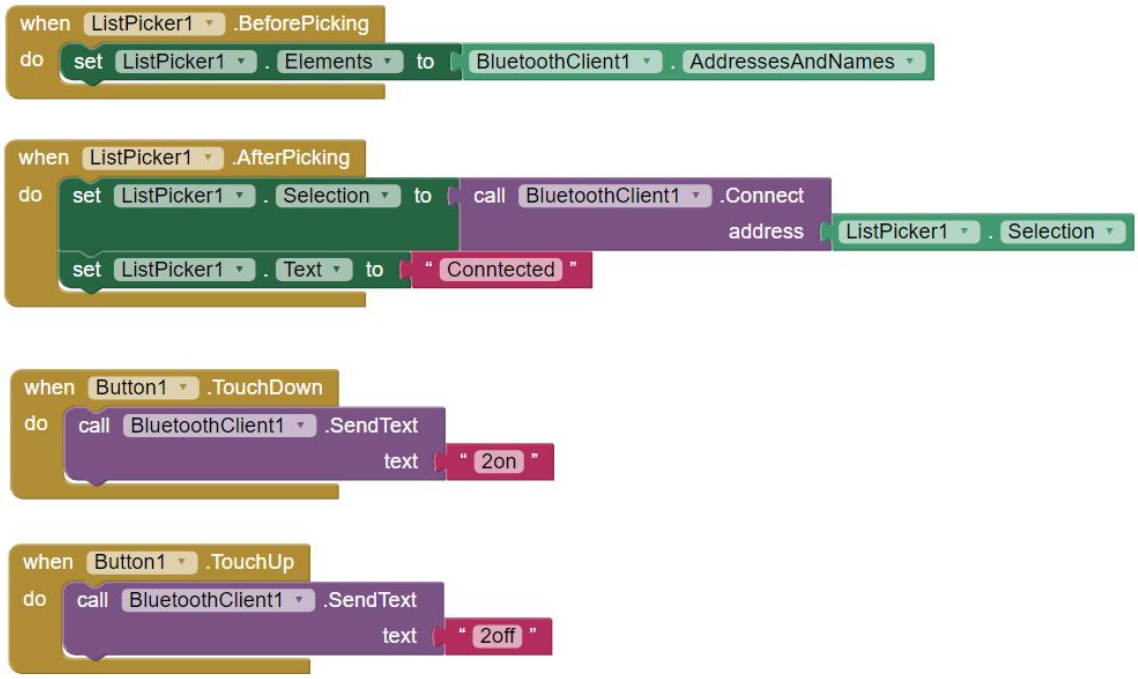


Figure 51: Bluetooth Set up and Fret Control Code

Overall MIT app inventor was very useful in the creation of a mobile application. It saved a lot of time in programming as we did not have to do larger amounts of coding by ourselves. The use of a mobile application also saved a lot of time by using digital inputs rather than

analog. Previous MQPs used force sensitive resistors and buttons for inputs. The ability to add and remove buttons on the fly rather than replacing physical components allowed for faster testing and more versatility to the functionality of our project.

5.2.5 Sound Testing

An important aspect during the testing period of the finalized design was the output sound produced from the mechanical actuators manipulating the guitar, along with the output sound from the connected amplifier. Once all the electrical and mechanical portions of the design were implemented together, the output quality began to be an important part of the testing process. This includes the control of the output noise from the bass amplifier, and the minimization of extraneous noises coming from the actuators, like that of the firing of the solenoids and the turning of the plucking servo.

5.2.5.1 Sound Dampening

In order to reduce the amount of mechanical noise from the added electrical components on the bass guitar, research and testing periods were conducted in the field of material science, where certain mechanical properties of materials were researched. In general, this project requires materials that have good acoustical dampening properties, where the material handles and stores acoustical vibrations well, which means the sound cannot travel as far.

The servo module produces a lot of high-pitched noise from the turning of the small plastic gears within its enclosed gearbox. Intrinsically, small servos produce a lot of excess sound due to this, so a sound dampening method needs to be implemented in order to reduce the noise the servo produces. From this, tests were done with extra plastic and foam pieces within our NECAMSID lab, where the foam would be placed around the servo to see the effect. The team discovered that a foam cover securely placed around the servo module would cause a significant decrease in the high-pitch noise. From this, a 3D printed module of a servo case was created using Solidworks to allow foam to be placed around the servo, all while the servo is mounted to the guitar itself. The Solidworks design is shown above in figure 40

After this was accomplished, multiple tests were conducted to see if the sound emitted from the new encompassed servo was reduced. The results were very positive, as the servo gear sounds were reduced, however vibrations still were allowed to be transferred through mechanical connections to the guitar itself. Due to this, a sound-dampening vinyl, a very dense material, was attached to the servo itself to help reduce this transfer of vibrations. After this, the servo was re-attached to the bass guitar and the testing period began for the final servo testing. After the sound-dampening material of both the foam and the dense vinyl material were added to the servo module, along with the 3D printed enclosed servo case, the high-pitched noise from the servo was successfully reduced.

Figure 52: Sound-dampening foam material

Another area where sound-dampening was needed is with the solenoids. The activation of the solenoid itself is relatively quiet, however the metal solenoid tip striking the bass string created a loud noise that needed to be reduced. The striking of the solenoid on the bass string also causes a vibration in the string that was unforeseen and needed to be addressed. When the solenoid retracts, there is a metallic striking sound that is caused by the retraction of the metal plunger of the solenoid into its case.

To help reduce the excess noise interference created by the solenoids, many intuitive and simple methods were first tested and employed. First, the string of the bass guitar itself was raised higher from the neck of the bass guitar. This was done by using an Allen key set to raise the height of the string, which can be manually changed with a mechanism on the bottom of the

bass guitar. This simple change helped to reduce the unwanted vibration after contact as the string was no longer vibrating of the lower frets. The next simple fix was adding a small rubber o-ring to the very end of the solenoid. This does two things. The first is the small o-ring limits the retraction of the solenoid by a very small length. Second, the rubber piece acts as an insulator between the two metal contact points which absorbs some of the vibration and sound that would otherwise cause a loud metallic clicking noise. Once these two simple additions were made, the solenoids already operated at a quieter level.

However, the striking of the metal solenoid tip onto the bass string still needed further testing. To do this, the team used the material science application CES EduPack, which is a database of a wide variety of materials that allows the user to test for differing material properties to see which material is best suited for a certain application. For our application, research went into the type of material needed that would best mimic a human finger stringing a bass string. For this, the material properties of Compression Strength, Hardness and Fracture Toughness were found to be the most important. Compression Strength is the yield strength in compression, meaning the elasticity of the material. For our purposes, a lower limit is ideal to have a more elastic material. Hardness is a material's overall strength. Our material does not have to be extremely strong, however it needs to be pliable enough to absorb sound vibrations. Lastly, Fracture Toughness is how durable the material is over time. This is an important quality, as this specific material will be utilized repeatedly on multiple solenoid strikes. With this information, values were inserted into Stage 1 of the program which sets parameters for selection of materials. This sorts and limits certain materials from passing the selection process. Next, a graph was created of a Price per density versus hardness to visualize the materials that fit the qualifications. The graph is shown below.

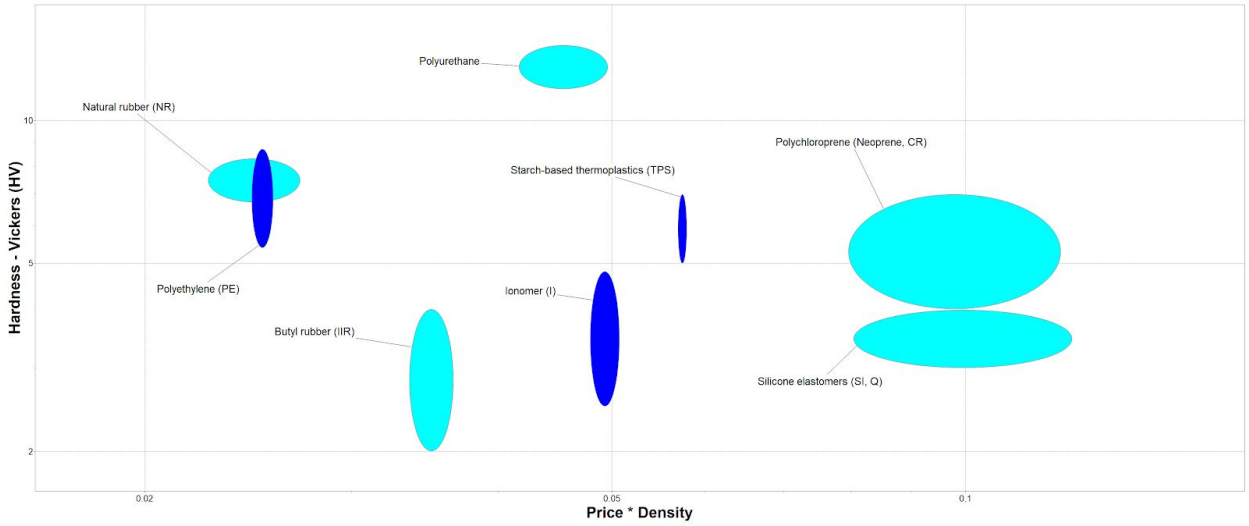


Figure 53: Price/Density vs. Hardness Limiting Graph

From the graph created, eight (8) different materials were limitedely selected to fit our material qualifications. The oval shapes represent a certain material, with the color representing the sub-group material type. The teal color group represents Elastomers, and the Blue color group represents Polymers. From here we tested multiple materials based on how available the product would be to use, with price and usability being the major factors. From this, were tested multiple materials that were readily available for usage.

At the end of testing, the team decided to use Polyurethane, which has the highest hardness out of all the selected materials. We chose this due to its low price point, along with its availability. The material chosen is shown below.

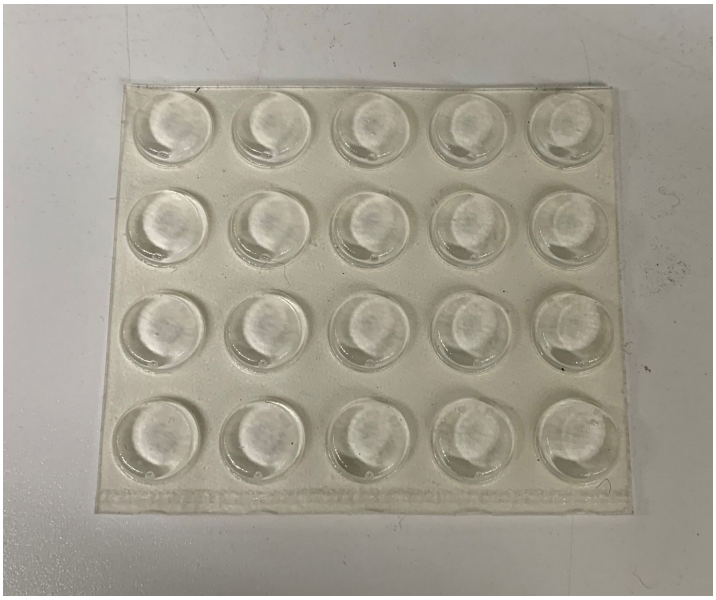


Figure 54: Polyurethane material bumpers for Solenoid Tips

The material shown above was then added to the solenoids and tested with the overall system. The solenoids were each individually fired onto the bass string, testing whether the sound was dampened and reduced due to the added material. The implementation onto the solenoid is shown below:

Figure 55: Polyurethane material bumpers for Solenoid Tips

Looking at the above figure, the Polyurethane elastomer material was added to the top of the solenoid. To increase the surface area of the striking material onto the solenoid to accommodate the material, two washer pieces were added to the solenoid piston, with the material placed on top. This allows an increased area for the solenoid to hit the bass string, which ensures a proper hold onto the string itself to help reduce vibrations. As can be seen as well, the o-ring mentioned previously can be seen in this picture, which helps to reduce the retraction noise of the solenoid. With all of these additions and sound-dampening techniques tested, the solenoids actuated more quietly, successfully ending the testing period of the sound-proffing period of the project.

5.2.5.2 Sound Amplification

The final output sound of the bass guitar was the last testing step in the process of combining the overall assistive bass module. This was due to the fact that minimal work was needed to be done in order to properly amplify the output sound of the bass guitar to produce the desired sound output. To acquire the required sound, the same output from the guitar that is inputted into the active low pass filter circuit is also inputted into a bass amplifier. This was accomplished by splicing the bass wire to allow a connection to both the low-pass filter and the bass amplifier simultaneously. The team had access to a spare bass amplifier that was provided by one of the team members, and also had spare bass amplifier wires that were purchased earlier within the project period. From here, the first test was to make sure the desired sound came from the bass amplifier with normal playing of the bass. From there, the overall system was tested with the added assistive components. This testing period went smoothly and as expected, as all other modules were previously tested. Lastly, a bass amplifier was purchased as the final portion of the project period. This bass amplifier is shown below.

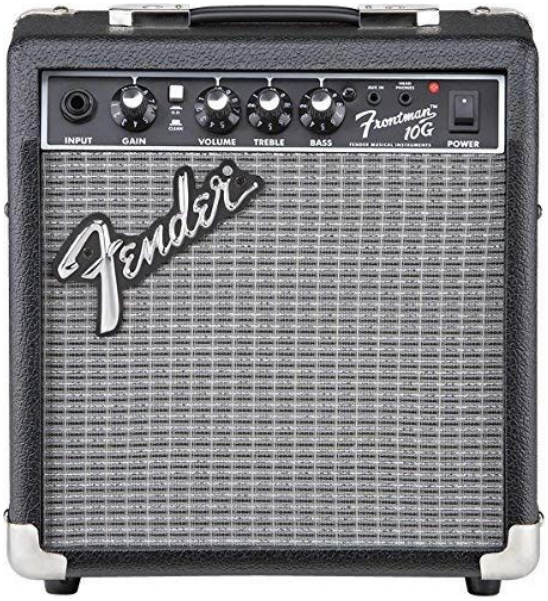


Figure 56: Purchased Bass Amplifier

5.2.6 Combined Module Testing

In order to test each block of our design as we made changes and worked towards our final design, we breadboarded each block and combined them to create a working testing system. The breadboarded test setups described in this section follow the circuit schematics outlined in Section 4.2.3.

Filter

The filter test setup is shown below. The top rail is the twelve volt supply rail while the bottom rail is the five volt supply rail. The active low pass filter utilizes a combination of resistor and capacitor values to achieve a cutt-off frequency of 170 Hz.

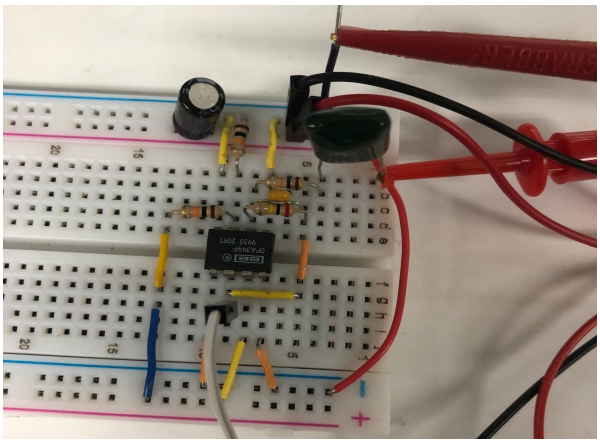


Figure 57: Filter test setup

Power Switching

An image of the buck converter module and its test setup is provided below. In this image, the top rail is connected to the twelve volt supply while the bottom rail is the new five volt supply created by the buck. This buck module allows the breadboard to supply two different voltage levels for the device. The 12V supply (top rail) powers the motor, solenoid and Arduino Mega, while the 5V supply (bottom rail) powers the bluetooth module and H-bridge module for the motor.

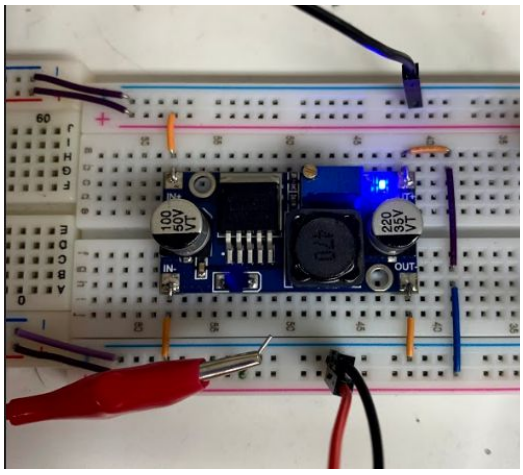


Figure 58: Buck Converter Test Setup

Motor

The breadboard test setup for the motor driver module is shown below. The motor is connected to the H-bridge module which is digitally controlled by the Arduino.

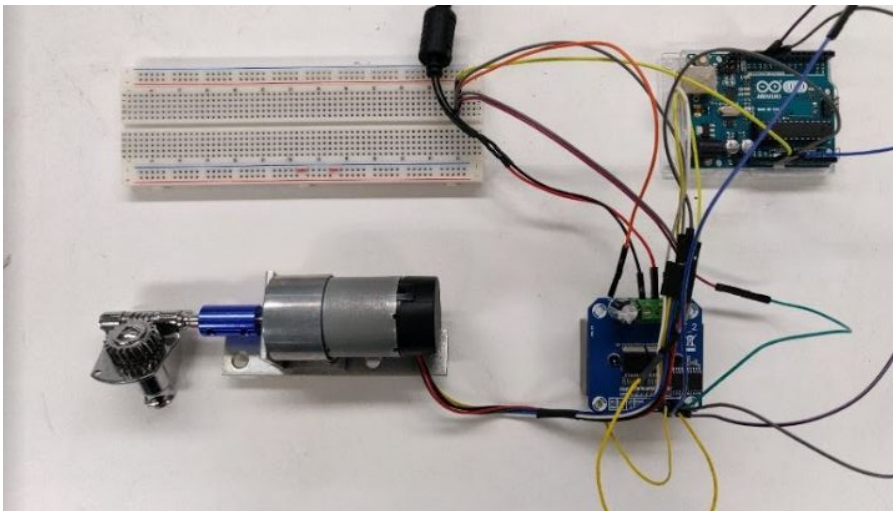


Figure 59: Motor driver test setup

Bluetooth

The breadboard test setup for the bluetooth module is provided below. The blue wires connect from the transmit and receive pins of the bluetooth module to the receive and transmit pins of the Arduino (not shown). A simple voltage divider using that utilizes a ⅔ ratio divides the 5 V from the digital pins of the Arduino to a 3.3 V operating voltage for the Bluetooth HC-06 module.

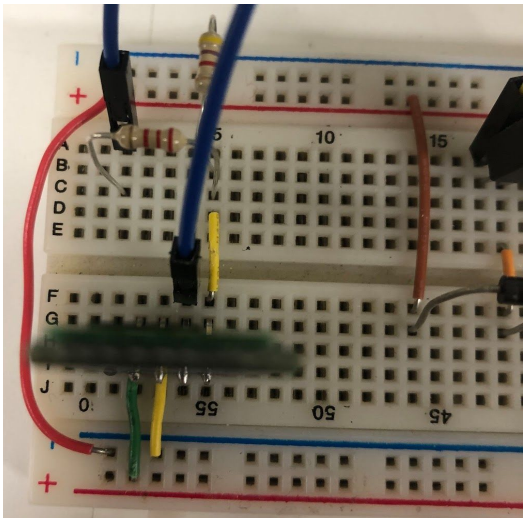


Figure 60: Bluetooth Breadboard Test Setup

Solenoids

Lasty, the five solenoid testing blocks are breadboarded as shown below. The LEDs were used in place of the solenoids for easy visualization of progress during testing. When a test button on the working mobile application is pressed, for example if "LED 1" is pressed, then a signal of "1on" is sent to the arduino, which sends a digital high signal to turn the first LED on.

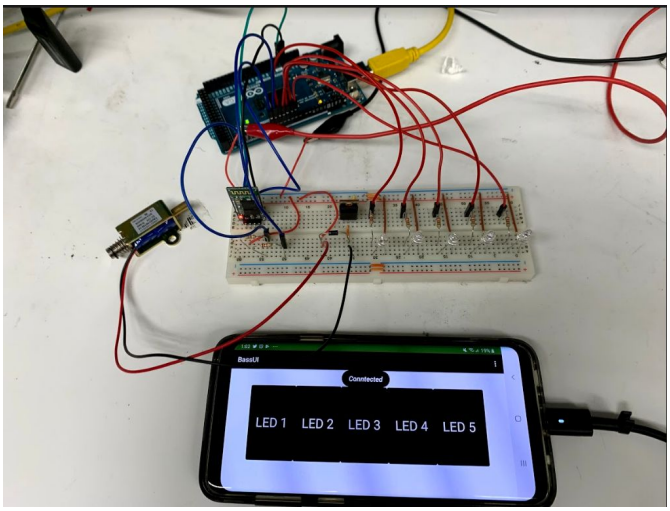


Figure 61: Solenoid Test Circuit using LEDs

PCB

Once our breadboard setup was functioning, we created a printed circuit board (PCB) using the Altium Designer PCB design software and had it printed at OSHPARK. The PCB is a two-layer board with a ground layer and another layer for both the five and twelve volt traces. Most footprints were downloaded from UltraLibrarian, except for the buck converter which was discontinued. Some footprints required simple edits, for example the MOSFET footprints found on UltraLibrarian had incorrect order of the terminals. Other than these few issues, the schematic and footprint creation portion of the PCB design was simple. As for the traces, the five volt traces do not require much current to flow through them, so we made them 40 mil wide. The only part which required a significant amount of current was the input header for the twelve volt supply, the traces between the twelve volt supply and the buck converter, and the twelve volt traces powering the solenoids. As for the input header, we wanted to use the same header pins for everything for a simple design, so we doubled the header pins for the twelve volt supply in order to double the current capability from three amps to six amps. These header pins have a pitch of 2.54 mm to match the pitch of the Arduino header pins. The traces carrying the twelve volt signal have a width of 140 mil to ensure proper current capabilities for the design. An image of the PCB layout in Altium is provided below.

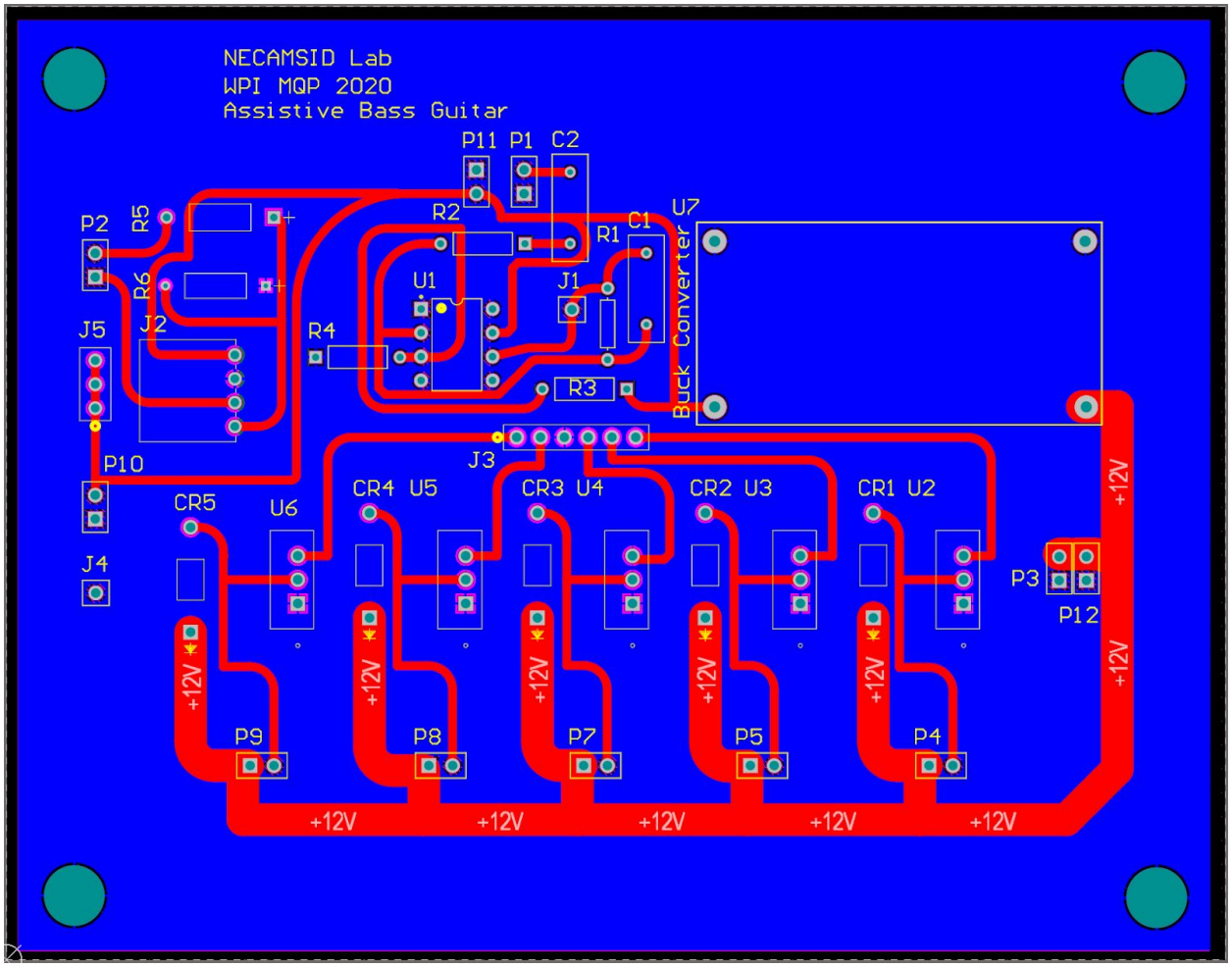


Figure 62: PCB Layout in Altium Designer

An image of the final soldered PCB is provided below.

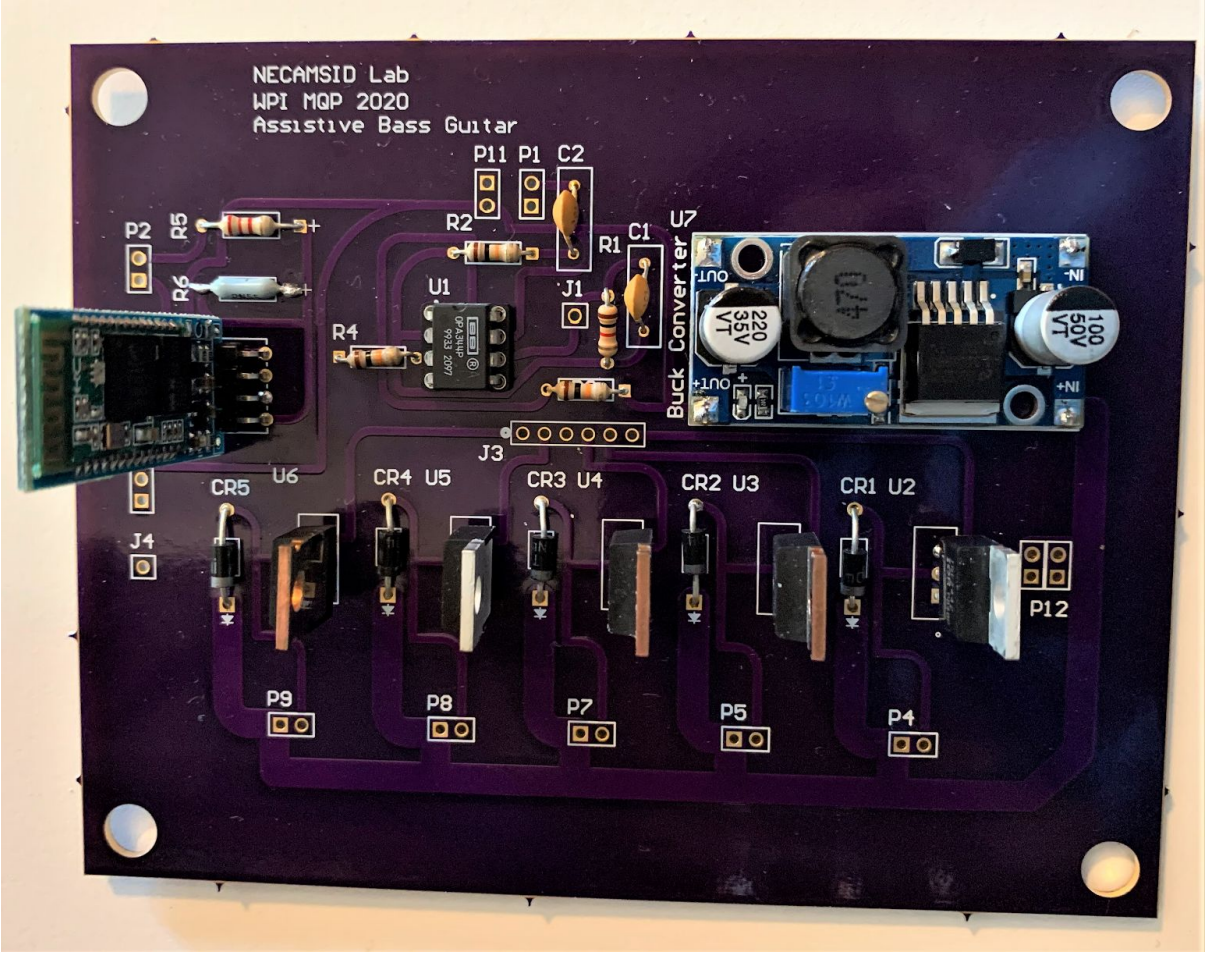


Figure 63: Final Soldered PCB

5.2.7 Overall Implementation Overview

For the project, testing was accomplished for each module separately to allow for multiple stages to be tested simultaneously. For example, testing on the MOSFET power switching to turn on the Solenoids was done in tandem with the motor mounting portion of the encoded motor to the head of the neck of the bass guitar. Due to this, the final portion of the design came together piece by piece, with the end of the testing period allowing the merger of all the modules together to create the overall prototyped design module.

The overall implementation mainly required the combination of the electrical portion with the mechanical portion. This entailed connecting the electrical circuits of the power

switching, filtering and bluetooth connections, to that of the physical mechanical structure of the bass guitar. Once this was accomplished, the final prototype for the assistive bass guitar was completed. Our design consisted of a bass guitar mounted upright onto a wood frame. Wood connectors were used and screwed into both the bass guitar itself and the bottom wood frame to keep the bass guitar upright and sturdy. The encoded motor had a mount specially fabricated to allow the connection of the motor, through a coupler, to the worm gear of the tuning pegs of the bass guitar. The servo was attached with the enclosed sound-proof case to the base of the guitar to allow the plucking of the bass strings. Solenoid mounts were created out of wood blocks, which were measured to take into account the height at which the solenoids needed to be in order to strike the bass string. Epoxy was then used to secure the solenoids to their respective mounts.

Each one of these mounted components has an electrical wire connection to another module, all of which have been fastened to the wood base. The created printed circuit board (PCB), was fastened to the wood base of the mount, to which the solenoid and servo connections route. The motor is connected to an H-bridge module, which is fastened next to the PCB. Lastly, the Arduino Mega microprocessor is also fastened next to the PCB in a way to reduce wire connections between the bass guitar and the electrical components. A bass amplifier has a connection from the output of the bass guitar that allows the final output sound to be played. All of these modules were created and tested separately in order to facilitate processes, and combined post-testing to create the finalized design. To use and play the assistive bass guitar, an android tablet is used to connect to the assistive bass guitar through the use of an app called “Bitar’s Guitar” downloaded on the tablet device.

6. Discussion

6.1 Achievements

At the conclusion of the project, the team was successful in bringing to fruition the design concept detailed at the start of the prototyping process. From this, the team based the amount of success achieved in the completion of the project to the amount of delivered design objectives and goals. The objectives and goals for this project are outlined in the Methodology section in sections prior in this research paper, and outlined the goals that the project intended to fulfill. This objectives were as follows:

- *Wireless Application*
- *Automatic Tuning*
- *Clean Output Sound and Signal*
- *Ease of use*

At the project's conclusion, the team has successfully implemented all the objectives that were outlined, and all the intended goals were achieved. The wireless application was implemented with the use of a bluetooth enabled module connected to the Arduino Mega, which allowed the communication between the user interface and the assistive bass guitar. The user interface to play the bass guitar was implemented with the application builder MIT App Inventor. The user interface provides touchscreen buttons that correlate with playing frets and notes on the bass guitar, and also allows the user to tune the device before initial usage.

The automatic tuning provides the user the ability to adjust the bass string in order to have the string be in tune. The single string design of the device uses the ‘A’ string, which is considered in tune at 55 Hz. The tuning software created, along with the mechanically mounted encoded motor, work together in order to tune the string within +- 2 Hz of this ideal frequency.

The achievement in a clean output sound did not just involve the addition of an amplifier to the output of the bass guitar, but also with the success in sound-dampening materials and designs implemented by the team which allowed for the reduction of extraneous noise from the added electro-mechanical components. Research was done on materials and material properties in order to find suitable methods to help absorb excess sound vibrations from the solenoids, sero and motor. Without the addition of these sound-proofing methods, unwanted vibrations would have caused a rather noisy output sound.

The last objective that the team set for the completion of the project was the usability of the device for those with Muscular Dystrophy. In order to accomplish this, the successful implementation of a touch screen tablet design was used in order to place the buttons in a way to easily mimic that of a person's hand placement. This allows those, not only with MD, but all to easily play the assistive bass guitar.

6.2 Shortcomings

6.2.1 Mobile App

One shortcoming of the app was the inability to transmit the serial monitor of the arduino to the app screen. We wanted to be able to see any transmitted information from the arduino on the screen of the tablet without the need of a computer. We were unable to send the information correctly as the data would not transmit correctly and resulted in errors. We determined that this was not necessary for the project and focused efforts elsewhere to keep on schedule.

The other shortcoming of this app was that it was very difficult to design a nice UI. It had a rigid autoplace grid and was non intuitive when placing buttons and text boxes. We were able to make a decent UI but were disappointed in its lack of designability.

6.2.2 Solenoids

A shortcoming of the solenoids is that it can only be used in a binary sense. This means that it can only be in two states, extended or retracted. We wanted to mimic a human finger

pressing down on the string as closely as possible. However, the solenoid is a little more forceful than a human finger, firing off instantly rather than a slower press. The instantaneous contact with the string would cause it to vibrate making it harder to get a clean sound from the guitar. We were able to negate some of this by using rubber tips on the ends to dampen the vibration.

The solenoids also presented the challenge of not allowing the use of every fret. Due to budget constraints and design time we were only able to apply solenoids to five of the twenty four frets on the neck of the guitar. We would have liked to be able to play a larger range of notes, however the five we chose offer a wide enough range to play. Though it is not as good as a regular finger it still works well as an alternative.

6.2.3 Tuning Implementation

Some shortcomings of the tuner is that we were unable to move the motor using units other than time. This meant that we would need to write several switch case statements to account for the full range of the string. Each case accounted for a range of two hertz. Overall it worked very well to get within two hertz of the targeted fifty five hertz. We would have liked to correlate each frequency to a bit value to more accurately and reliably tune to a specific frequency. Unfortunately, we could not account for small amounts of motor and string slippage while tuning, instead using time to turn the motor proved very effective in the end and we are happy with the result.

6.3 Impact for Assistive Technology

The project had a focus to allow those with Muscular Dystrophy the opportunity to continue their love for arts with an assistive device. This device is that of an assistive bass guitar that will allow those with limited movement in their hands to play a guitar-type acoustical instrument. While the main focus for the team was specifically MD affected people, the device has the overreaching potential to a broader array of people that have afflictions causing limited motion within their hands. For example, stroke victims have a similar loss of muscular function within their extremities, meaning this device has the potential to allow stroke victims the

opportunity as well to continue with their desire to play instruments. This project has the potential to be expanded for more strings and a larger variety of instruments.

6.4 Manufacturing and Costs Evaluation

The prototype designed throughout this MQP relied heavily on custom manufacturing techniques and manual design work that is relatively hard to quantify. The major area of concern is costs involved in custom woodworking, and materials used. Realistically, the team understands that the full production of the project would most likely require material changes. While wood was used for the primary construction of the project, in order to feasibly pursue production, a more suitable material like metal or acrylic would make the most sense from a manufacturing standpoint. Major manufacturing challenges the team faced while using wood were primarily regarding tolerance. While wood is easily used for construction, wood stock is not always “perfect” in nature. Additionally, the project required multiple custom fabricated pieces, such as the collar between the tuning motor and bass tuning peg. While parts of this nature were easily manufactured with time and persistence using the teams resources at WPI, quantifying what their cost to build in an industrial setting is difficult without further research into the manufacturing requirements. The quantifiable costs of the project prototype can be seen below in the overall bill of materials.

Part	Price Per Unit	No. of Units	Total Costs	Total Project Cost
1/4 W Resistors	$0.02	5	$0.10	**$318.64**
1N4004 Diode	$0.19	5	$0.95	
Arduino Mega	$35.00	1	$35.00	
Bluetooth Module	$7.99	1	$7.99	
Buck Boost Converter	$1.50	1	$1.50	
Capacitor	$0.29	1	$0.29	
Guitar Cord	$3.99	1	$3.99	
H-Bridge	$13.49	1	$13.49	
IRFZ44Z Mosfet	$0.91	10	$9.10	
Metal Gear Servo	$17.95	1	$17.95	
Misc Hardware and Wood	$20.00	1	$20.00	
Motor shaft coupler	$4.25	1	$4.25	
OP 344P Op-Amp	$1.50	1	$1.50	
Rubber Solenoid Tips	$0.09	5	$0.45	
Servo Enclosure	$4.30	1	$4.30	
Solenoids	$18.95	5	$94.75	
Sound Damping Foam	$9.99	1	$9.99	
Tuning Motor	$32.00	1	$32.00	
Used Bass Guitar	$40.00	1	$40.00	
Vinyl	$12.55	1	$12.55	
Wires	$8.49	1	$8.49	

Figure 64: Bill of Materials

Major areas of consideration for the further production of the project would require a standardized guitar of use. This would simplify the large scale manufacturing of the supporting mechanical design and create a more realistic approximation of the total costs involved with this project. The 3D printed servo enclosure would be supplemented with a suitable ABS plastic

alternative produced in large quantities, bringing the overall manufacturing of the part to cents rather than dollars.

6.5 Limitations

Overall, the design proposed by this report was successful in meeting its objectives but there is room for improvement in certain areas. The team encountered a few limitations in the design and actualization of this project in areas such as app-development, mechanical design, and resource shortages.

Our team lacked members with strong experience in computer science and mechanical engineering, as our team was composed of all electrical and computer engineering students. If we had more of a background in computer science, our application design would have been able to follow a more advanced design. Also, if we had more of a background in mechanical engineering, aspects such as sound dampening and structural design could have improved.

Another limitation on the team in the design of this project was a shortage of time and financial resources. With more time, the team could have invested more attention on the improvement of all aspects of the design including the user application and the aesthetics of the final product. With access to more financial resources, more solenoids could be purchased for a wider range of available notes.

6.6 Future Work

6.6.1 Real-time Tuning

Originally, the team wanted to implement a method to allow the string to automatically tune the string in real time. This would entail that the output signal from the bass guitar would need to be constantly checked through the peak-detection software, which would send an encoded position for the motor to move to in order to re-tune the string to the proper frequency. The problem with this design is that it requires an extreme amount of mechanical movement from the motor, which would be extremely loud and unreasonable to implement. Initial attempts

were made at implementing this design, however many issues arised with latency and mechanical error. The motor could not tune fast enough in order to play a melody with only one strum.

During the design period of the project, the team met with Professor Scott Barton of arts, communications and humanities at Worcester Polytechnic Institute. Barton has experience in working with self-playing instruments and self-playing guitars, and provided the team with insight into methods that would be beneficial to the assistive bass guitar. He mentioned that the issue of real time tuning has been an ongoing problem, and is extremely complicated to implement. For this reason, and our issues at implementation, the team decided to instead tune the bass guitar prior to playing, and use multiple frets and solenoids to implement note playing. A future implementation of this project has the potential to focus just on this problem, as its implementation is a complex problem enough in itself to be a long-term project.

6.6.2 Professional Mobile Application

The team implemented its mobile application using the software MIT App Inventor. This application is a very basic app builder that does not allow much manipulation of the interface, making the interface not as usable as initially desired. The team lacked experience in the field of Computer Science and app development, meaning we had to resort to using a simpler app builder environment in order to deliver a mobile experience. In future implementations of this project, a new application for the user interface between the user and the guitar would allow a better overall experience for the user. The team recommends that Computer Science majors, or those with a more in-depth computer science background, implement an application for the user interface that allows for a more complex design. This would allow for an interface that would make the user more readily able to touch the different buttons that correlate to a note being played on the guitar. Also, a more in-depth application would allow for a more professional looking application as well, where more artistic freedom can be implemented.

6.6.3 Hammering and Strumming Actuators

Another area where the team believes there is room for improvement is the way in which the bass guitar is played. The team decided, through research and conversations with an experienced professor in the field of robotic acoustical instruments, that a combination of solenoids and servos was the best course of action in order to play the bass guitar remotely. The solenoids are best used for the hammering operation as they provide a linear force directly onto the string, while also being relatively quiet and fast. A servo, while having a high-pitched noise due to the gearing, provides an effective plucking mechanic that is easy to program. However, the team ran across some issues with using these devices to play the bass guitar.

With the solenoids, the team originally had intentions of using smaller 10 N rated solenoids, however after multiple tests realized that the force required to push down on the string is much greater. This caused the team to pivot, and purchase much larger solenoids that have a 25 N rating force. These solenoids are much louder as well, making the noise interference problem much harder. A similar issue occured with the servo, as original intentions were to use a smaller servo that would fit more easily on the bass guitar. Instead, a much larger servo needed to be purchased to have enough force to pluck the bass string. Due to these issues, future renditions of this project have the potential to research and test new methods of playing the bass guitar. One idea that the team has is to use a stepper motor in place of the servo. A stepper motor would provide a more accurate and timely pluck of the string, while also providing quiet operation. Also, instead of using multiple solenoids mounted on the base of the device, a track system could be implemented to have only one type of actuator push down on the string. These and other ideas have the potential to improve the functionality and play style of the assistive bass guitar.

7. Conclusion

The design of the Assistive Bass Guitar was an overall success. The four objectives outlined in Section 4.1 were met. The design is wireless, using a bluetooth module, so that the user can comfortably control the instrument from the tablet. The design also succeeds in automatic tuning through the use of an algorithm which employs autocorrelation. The third objective, that the design produce a clean output sound, was met through the use of sound dampening materials in the solenoids and servo motor. Lastly, the tablet application is easy to use and allows all available notes to be reached. Although the objectives were met, the latter objective (ease of use) could be improved by physically testing the design with people affected by Muscular Dystrophy. Also with a higher budget, more solenoids could be used to reach more frets. Overall, as a proof of concept, this design is a success and with some future work, it could have a positive impact.

Appendix A

```
#include <Servo.h>
#define LENGTH 512

//solenoid pin selection
int led1 = 2;
int led2 = 3;
int led3 = 4;
int led4 = 7;
int led5 = 6;
int tune_led = 44;

//servo pin selection
int servo1 = 9;
//integer for servo starting position out of 360 degrees
int servpos = 40;
Servo myservo;
int tx = 1;
int rx = 0;
char inSerial[15];

//frequency detection declarations
byte rawData[LENGTH];
int count;

//motor control
int mleft = 10;
int mright = 13;

// Sample Frequency in kHz
const float sample_freq = 8919;
int Afreq = 55;
int freqSum;

int len = sizeof(rawData);
int i, k, a;
long sum, sum_old;
int thresh = 0;
```

```
float freq_per = 0;
float avg_freq = 0;
int nums = 0;
float main_freq = 0;
byte pd_state = 0;


void setup() {
  Serial.begin(9600);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(tune_led, OUTPUT);

  pinMode(tx, OUTPUT);
  pinMode(rx, INPUT);

  pinMode(mright, OUTPUT);  //setup H-bridge control pins at outputs
  pinMode(mleft, OUTPUT);

  pinMode(servo1, OUTPUT);
  myservo.write(servpos);



  analogRead(A1);
  count = 0;
  allpinslow();



}
void loop() {
  int i = 0;
  int m = 0;
  delay(60);
  if (Serial.available() > 0) {
    while (Serial.available() > 0) {
      inSerial[i] = Serial.read();
      i++;
```

```
    }
    inSerial[i] = '\0';
    Check_Protocol(inSerial);
  }
}


void allpinslow()
{
  digitalWrite(led1, HIGH);
  digitalWrite(led1, LOW);
  digitalWrite(led2, HIGH);
  digitalWrite(led2, LOW);
  digitalWrite(led3, HIGH);
  digitalWrite(led3, LOW);
  digitalWrite(led4, HIGH);
  digitalWrite(led4, LOW);
  digitalWrite(led5, HIGH);
  digitalWrite(led5, LOW);

}

void damp() {

  digitalWrite(led3, HIGH);
  delay(2000);
  digitalWrite(led3, LOW);

}
void servo(){
  myservo.attach(servo1);
  servpos = servpos + (servpos * -2);
  myservo.write(servpos);
  delay(100);
  myservo.detach();
}

void detect_frequency()
{
  Serial.begin(115200);
```

```
 for (a = 0; a < 5000; a++) {
   //Serial.println(a);

   if (count < LENGTH) {

     count++;
     rawData[count] = analogRead(A1) >> 2; //shifts datat by 2 to fit 10bit ADC rading into 8 bit
byte
     if (a == 1000) {

     servo();
     }
   }
   else {

     sum = 0;
     pd_state = 0;
     int period = 0;
     for (i = 0; i < len; i++)
     {
       // Autocorrelation
       sum_old = sum;
       sum = 0;
       for (k = 0; k < len - i; k++)
         sum += (rawData[k] - 128) * (rawData[k + i] - 128) / 256;

       // Serial.println(sum);

       // Peak Detect State Machine

       //pd_state = 2
       if (pd_state == 2 && (sum - sum_old) <= 0)
       {
         period = i;
         pd_state = 3;
       }
       //pd_state = 1
       if (pd_state == 1 && (sum > thresh) && (sum - sum_old) > 0) pd_state = 2;

       //pd_state = 0
       if (!i) {
```

```
        thresh = sum * 0.5;
        pd_state = 1;
       }
     }
     // for(i=0; i < len; i++) Serial.println(rawData[i]);

     // Frequency identified in Hz
     //pd_state = 3
     if (thresh > 100) {
       freq_per = sample_freq / period;
       if ((period != 0) && (freq_per < 80)) {

         main_freq = freq_per;
         delay(10);
         Serial.println(main_freq);


       }

     }
     count = 0;
   }
  }
  //main_freq = avg_freq/nums;
}

void tune_string() {
  freqSum = Afreq + main_freq;
  //Serial.println(freqSum);

  switch (freqSum) {
    case 90 ... 94:          //frequency detected is between 35 and 39 Hz
      digitalWrite(mleft, HIGH);
      digitalWrite(mright, LOW);
      delay(2000);            //turn motor clockwise to increase frequency for 5 s
      digitalWrite(mright, LOW);
      digitalWrite(mleft, LOW);
      damp();
      freqSum = 0;
      break;
    case 95 ... 97:          //frequency detected is between 40 and 42 Hz
      digitalWrite(mleft, HIGH);
```

```
  digitalWrite(mright, LOW);
  delay(1500);           //turn motor clockwise to increase frequency for 5 s
  digitalWrite(mright, LOW);
  digitalWrite(mleft, LOW);
  damp();
  freqSum = 0;
  break;
 case 98 ... 100:         //frequency detected is between 43 and 45 Hz
  digitalWrite(mleft, HIGH);
  digitalWrite(mright, LOW);
  delay(1250);           //turn motor clockwise to increase frequency for 5 s
  digitalWrite(mright, LOW);
  digitalWrite(mleft, LOW);
  damp();
  freqSum = 0;
  break;
 case 101 ... 103:         //frequency detected is between 46 and 48 Hz
  digitalWrite(mleft, HIGH);
  digitalWrite(mright, LOW);
  delay(1000);           //turn motor clockwise to increase frequency for 5 s
  digitalWrite(mright, LOW);
  digitalWrite(mleft, LOW);
  damp();
  freqSum = 0;
  break;
 case 104 ... 107:        //frequency detected is between 49 and 52 Hz
  digitalWrite(mleft, HIGH);
  digitalWrite(mright, LOW);
  delay(750);           //turn motor clockwise to increase frequency for 1 s
  digitalWrite(mright, LOW);
  digitalWrite(mleft, LOW);
  damp();
  freqSum = 0;
  break;
 case 108 ... 112:        //frequency detected is between 53 and 57 Hz
  digitalWrite(mright, LOW);
  digitalWrite(mleft, LOW);
  damp();
  freqSum = 0;
  break;
 case 113 ... 115:        //frequency detected is between 58 and 60 Hz
  digitalWrite(mleft, LOW);
```

```
    digitalWrite(mright, HIGH);
    delay(500);           //turn motor counterclockwise to decrease frequency for 1 s
    digitalWrite(mright, LOW);
    digitalWrite(mleft, LOW);
    damp();
    freqSum = 0;
    break;
   case 116 ... 118:        //frequency detected is between 58 and 60 Hz
    digitalWrite(mleft, LOW);
    digitalWrite(mright, HIGH);
    delay(750);           //turn motor counterclockwise to decrease frequency for 1 s
    digitalWrite(mright, LOW);
    digitalWrite(mleft, LOW);
    damp();
    freqSum = 0;
    break;
   case 119 ... 121:        //frequency detected is between 61 and 70 Hz
    digitalWrite(mleft, LOW);
    digitalWrite(mright, HIGH);
    delay(1000);           //turn motor counterclockwise to decrease frequency for 5 s
    digitalWrite(mright, LOW);
    digitalWrite(mleft, LOW);
    damp();
    freqSum = 0;
    break;
   case 122 ... 125:        //frequency detected is between 58 and 60 Hz
    digitalWrite(mleft, LOW);
    digitalWrite(mright, HIGH);
    delay(1250);           //turn motor counterclockwise to decrease frequency for 1 s
    digitalWrite(mright, LOW);
    digitalWrite(mleft, LOW);
    damp();
    freqSum = 0;
    break;
   default:                //assuming frequency will not be outside range of 40 to 70 Hz: frequency
detected is between 54 and 56 Hz
    digitalWrite(mright, LOW);
    digitalWrite(mleft, LOW);  //do not turn motor
    break;
  }
}
```

```
void Check_Protocol(char inStr[]) {
  //int i = 0;
  int m = 0;

  if (!strcmp(inStr, "demo")) {



  }

  if (!strcmp(inStr, "tuneron")) {
    delay(100);
    detect_frequency();
    delay(100);
    tune_string();

    delay(10);

    while ((abs(main_freq - Afreq) > 2) && (main_freq > 0)) {

      main_freq = 0;
      delay(100);
      detect_frequency();
      delay(5000);
      tune_string();
      //Serial.println(main_freq);
      delay(10);

      if (abs(main_freq - Afreq) <= 2) {
        break;

      }

    }

    damp();
    main_freq = 0;
    Serial.println("tuneroff");
    digitalWrite(tune_led, HIGH);
    delay(2000);
    digitalWrite(tune_led, LOW);
```

```
  delay(10);
  Serial.begin(9600);

}


if (!strcmp(inStr, "righton")) {
  digitalWrite(mright, HIGH);
  digitalWrite(mleft, LOW);
}

if (!strcmp(inStr, "rightoff")) {
  digitalWrite(mright, LOW);
  digitalWrite(mleft, LOW);
}

if (!strcmp(inStr, "lefton")) {
  digitalWrite(mleft, HIGH);
  digitalWrite(mright, LOW);
}
if (!strcmp(inStr, "leftoff")) {
  digitalWrite(mleft, LOW);
  digitalWrite(mright, LOW);
}

if (!strcmp(inStr, "servo")) { //Led Off
  servo();
}

if (!strcmp(inStr, "2off")) { //Led Off
  delay(50);
  allpinslow();
  digitalWrite(led1, LOW);
  //    for (m = 0; m < 11; m++) {
  //      inStr[m] = 0;
  //    }
  i = 0;
}
//Serial.println(inStr);
if (!strcmp(inStr, "2on")) { //Led on
  allpinslow();
  digitalWrite(led1, HIGH);
```

```
  //delay(50);
  servo();
  //    for (m = 0; m < 11; m++) {
  //      inStr[m] = 0;
  //    }
  i = 0;
}

if (!strcmp(inStr, "3off")) { //Led Off
  allpinslow();
  digitalWrite(led2, LOW);
  //    for (m = 0; m < 11; m++) {
  //      inStr[m] = 0;
  //    }
  i = 0;
}

if (!strcmp(inStr, "3on")) { //Led on
  allpinslow();
  digitalWrite(led2, HIGH);
  //delay(50);
  servo();
  //    for (m = 0; m < 11; m++) {
  //      inStr[m] = 0;
  //    }
  i = 0;
}
if (!strcmp(inStr, "4off")) { //Led Off
  allpinslow();
  digitalWrite(led3, LOW);
  //    for (m = 0; m < 11; m++) {
  //      inStr[m] = 0;
  //    }
  i = 0;
}

if (!strcmp(inStr, "4on")) { //Led on
  allpinslow();
  digitalWrite(led3, HIGH);
  //delay(50);
  servo();
  //    for (m = 0; m < 11; m++) {
```

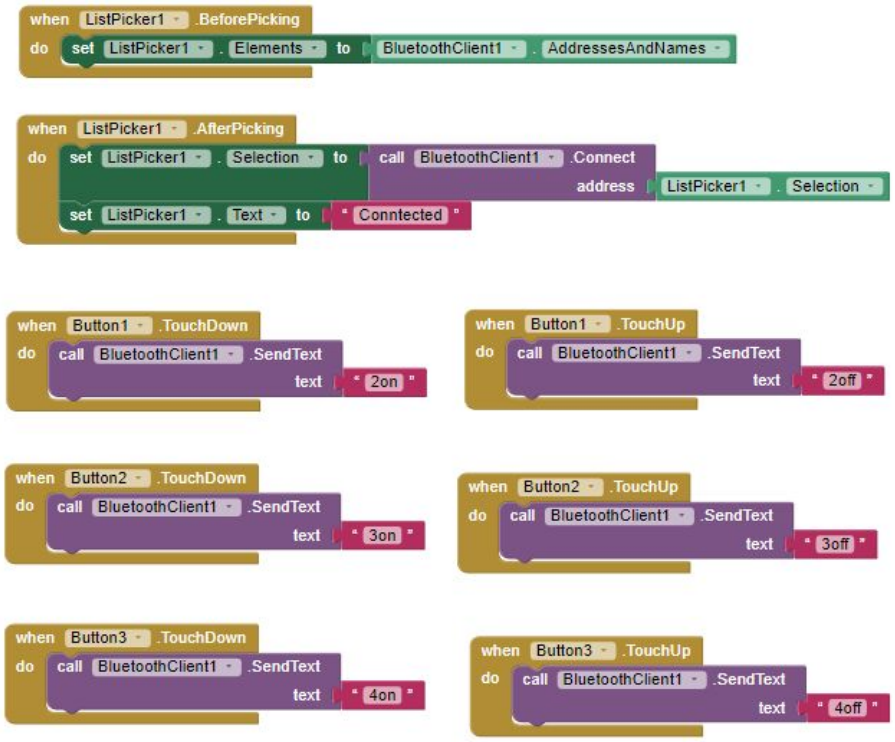
```
  //      inStr[m] = 0;
  //    }
  i = 0;
}
if (!strcmp(inStr, "5off")) { //Led Off
  allpinslow();
  digitalWrite(led4, LOW);
  //    for (m = 0; m < 11; m++) {
  //      inStr[m] = 0;
  //    }
  i = 0;
}

if (!strcmp(inStr, "5on")) { //Led on
  allpinslow();
  digitalWrite(led4, HIGH);
  //delay(50);
  servo();
  for (m = 0; m < 11; m++) {
    inStr[m] = 0;
  }
  i = 0;
}
if (!strcmp(inStr, "6off")) { //Led Off
  allpinslow();
  digitalWrite(led5, LOW);
  //    for (m = 0; m < 11; m++) {
  //      inStr[m] = 0;
  //    }
  i = 0;
}

if (!strcmp(inStr, "6on")) { //Led on
  allpinslow();
  digitalWrite(led5, HIGH);
  //delay(50);
  servo();
  //    for (m = 0; m < 11; m++) {
  //      inStr[m] = 0;
  //    }
  i = 0;
}
```

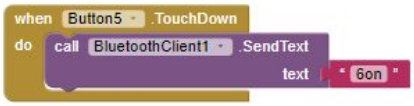
```
  else {
    for (m = 0; m < 11; m++) {
      inStr[m] = 0;
    }
    i = 0;
  }
}
```

Appendix B

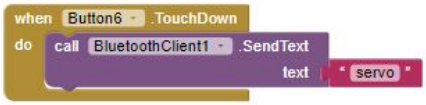

when ListPicker1 .BeforePicking
do set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
when ListPicker1 .AfterPicking
do set ListPicker1 . Selection to call BluetoothClient1 .Connect
address ListPicker1 . Selection
set ListPicker1 . Text to " Conntected "
when Button1 .TouchDown
do call BluetoothClient1 .SendText
text " 2on "
when Button1 .TouchUp
do call BluetoothClient1 .SendText
text " 2off "
when Button2 .TouchDown
do call BluetoothClient1 .SendText
text " 3on "
when Button2 .TouchUp
do call BluetoothClient1 .SendText
text " 3off "
when Button3 .TouchDown
do call BluetoothClient1 .SendText
text " 4on "
when Button3 .TouchUp
do call BluetoothClient1 .SendText
text " 4off "

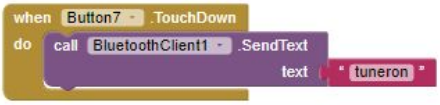

when Button4 .TouchDown
do call BluetoothClient1 .SendText
text " 5on "


when Button4 .TouchUp
do call BluetoothClient1 .SendText
text " 5off "

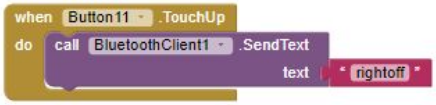

when Button5 .TouchDown
do call BluetoothClient1 .SendText
text " 6on "

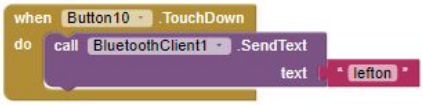

when Button5 .TouchUp
do call BluetoothClient1 .SendText
text " 6off "

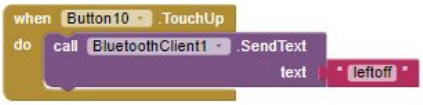

when Button6 .TouchDown
do call BluetoothClient1 .SendText
text " servo "

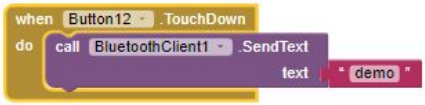

when Button7 .TouchDown
do call BluetoothClient1 .SendText
text " tuneron "


when Button11 .TouchDown
do call BluetoothClient1 .SendText
text " righton "


when Button11 .TouchUp
do call BluetoothClient1 .SendText
text " rightoff "


when Button10 .TouchDown
do call BluetoothClient1 .SendText
text " lefton "


when Button10 .TouchUp
do call BluetoothClient1 .SendText
text " leftoff "


when Button12 .TouchDown
do call BluetoothClient1 .SendText
text " demo "

References

[1] J. Denoncour, “Art for the Disabled’ February-2019.

[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] "Search: Magicmakingmusic." Search: Magicmakingmusic. Web. 13 Apr. 2016.

[4] Adafruit Industries. “Standard Servo - TowerPro SG-5010.” *Adafruit Industries Blog RSS*, https://www.adafruit.com/product/155.

[5] Adafruit Industries. “Small Push-Pull Solenoid - 12VDC.” *Adafruit Industries Blog RSS*, https://www.adafruit.com/product/412.

[6] Aidan. “Controlling a Solenoid with an Arduino - Tutorial.” *Core Electronics*, 22 Nov. 2018, https://core-electronics.com.au/tutorials/solenoid-control-with-arduino.html.

[7] *Arduino Mega 2560 Rev3*, https://store.arduino.cc/usa/mega-2560-r3.

[8] Kim, Vadim. *How to Design 10 KHz Filter. (Using Butterworth Filter Design)*. Msu.edu, 2011.https://www.egr.msu.edu/classes/ece480/capstone/fall11/group02/web/Documents/How%20to%20Design%2010%20kHz%20filter-Vadim.pdf

[9] “Metal DC Geared Motor w/Encoder - 12V 251RPM 18Kg.Cm.” *DFRobot*, https://www.dfrobot.com/product-634.html#.UWE4PNb8n5-5.4.

[10] Ada, Lady. “Force Sensitive Resistor (FSR).” *Adafruit Learning System*, https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr.

[11] Duchenne Muscular Dystrophy (DMD) | Muscular Dystrophy Association. (2019). Retrieved 8 October 2019, from https://www.mda.org/disease/duchenne-muscular-dystrophy

[12] Roederer, J. G. (2009). *The Physics and Psychophysics of Music an Introduction*. New York, NY: Springer US.

[13] Hartmann, W. (1997). *Signals, sound, and sensation* . Woodbury, N.Y: American Institute of Physics.

[14] Fitch, W., & Rosenfeld, A. (2007). Perception and Production of Syncopated Rhythms. *Music Perception: An Interdisciplinary Journal, 25*(1), 43-58. doi:10.1525/mp.2007.25.1.43

[15] Physics of Music - Notes. (n.d.). Retrieved from https://pages.mtu.edu/~suits/notefreqs.html.

[16] Blecha, Peter. "Audiovox #736: The World's First Electric Bass Guitar!". *Vintage Guitar*. Slog, John J.; Coryat, Karl (1999). *The Bass Player Book: Equipment, Technique, Styles and Artists*. Backbeat Books. ISBN 0-87930-573-8.

[17] Imtiaz F, S. (2015). A Classical Case of Duchenne Muscular Dystrophy. Hereditary Genetics, 04(01). doi: 10.4172/2161-1041.1000139

[18] Walter, M., & Reilich, P. (2017). Recent developments in Duchenne muscular dystrophy: facts and numbers. *Journal Of Cachexia, Sarcopenia And Muscle*, *8*(5), 681-685. doi: 10.1002/jcsm.12245

[19] Hunnekens, M., Huijben, J., & de Groot, I. (2017). Hand function in boys and men with Duchenne muscular dystrophy (DMD). *Neuromuscular Disorders*, *27*, S234. doi: 10.1016/j.nmd.2017.06.499

[20] Hiller LB, Wade CK. Upper extremity functional assessment scales in children with Duchenne muscular dystrophy: a comparison. Arch Phys Med Rehabil 1992;73:527-34. Lord JP, Portwood MM, Lieberman JS,

[21] Sulewski, J., Boeltzig, H., & Hasnain, R. (2012). Art and Disability: Intersecting Identities Among Young Artists with Disabilities. *Disability Studies Quarterly*, *32*(1). doi: 10.18061/dsq.v32i1.3034

[22] 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.

[23] Taylor, M. (2005). Self-identity and the arts—Education of disabled young people. *Disability & Society, 20*(7), 763-778.

[24] Caspers Conway, K., Mathews, K., Paramsothy, P., Oleszek, J., Trout, C., Zhang, Y., & Romitti, P. (2015). Neurobehavioral Concerns Among Males with Dystrophinopathy Using Population-Based Surveillance Data from the Muscular Dystrophy Surveillance, Tracking, and Research Network. *Journal Of Developmental & Behavioral Pediatrics*, *36*(6), 455-463. doi: 10.1097/dbp.0000000000000177

[25] The New England Center for Analog and Mixed Signal Design atWPI. (2019). Retrieved 8 October 2019, from http://users.wpi.edu/~mcneill/analog/center.html

[26] How To Build Custom Android App for your Arduino Project using MIT App Inventor - HowToMechatronics. (2019). Retrieved 8 October 2019, from https://howtomechatronics.com/tutorials/arduino/how-to-build-custom-android-app-for-your-arduino-project-using-mit-app-inventor/

[27] Frequency and Pitch. (n.d.). Retrieved from https://www.nde-ed.org/EducationResources/HighSchool/Sound/frequencypitch.htm.

[28] Vibrating String. (n.d.). Retrieved from http://hyperphysics.phy-astr.gsu.edu/hbase/Waves/string.html.

[29] Reliable Frequency Detection Using DSP Techniques. (2017, February 4). Retrieved from http://www.akellyirl.com/reliable-frequency-detection-using-dsp-techniques/.

[30] SIMPLE SWITCHER® Power Converter 150-kHz, 3-A Step-Down Voltage Regulator, LM2596, Rev. E, Texas Instruments, 2020. [Online]. Available: http://www.ti.com/lit/ds/symlink/lm2596.pdf

[31] BTS7960 43A High Power Motor Driver Module, HiLetgo, 2018. [Online]. Available: http://www.hiletgo.com/ProductDetail/1958385.html

[32] High Current PN Half Bridge NovalithIC™, BTN7970, Rev. 1.1, Infineon, 2007. [Online]. Available: https://www.mouser.com/datasheet/2/196/BTN7970_DS_11-255465.pdf

[33] LOW POWER, SINGLE-SUPPLY, RAIL-TO-RAIL OPERATIONAL AMPLIFIERS MicroAmplifier ™ Series, OPA344P, Texas Instruments, 2008. [Online]. Available: https://www.ti.com/lit/ds/symlink/opa344.pdf