



DESIGN DOCUMENT FOR MODULAR AUDIO MIXER

EE 491 | DEC1503

ABSTRACT

A technical outline of the modular audio mixer for our client, Jay Becker, with the help of our advisor, Josh Bertram. This document covers the System Design, the Detailed Design, and the Design Documents used in order to construct and convey our concept solution.

Team Leader: Clayton Hawken
Team Concept Holder: Chad Stobbie
Team Webmaster: Brian West
Team Communication Leader: Deborah Baeder

Table of Contents

Introduction.....	2
<i>Problem Statement</i>	2
<i>Proposed Solution</i>	2
<i>Semester Deliverables</i>	2
System Design.....	3
<i>System Requirements</i>	3
<i>Functional Decomposition and System Analysis</i>	3
Figure 1: Block Diagram	3
<i>Projections</i>	3
Figure 2: Improvements	3
Detailed Design Specifications.....	4
<i>User Interface and Applications</i>	4
Figure 3: Enclosure Specifications	3
<i>Software</i>	4
<i>Hardware</i>	5
<i>Mixer</i>	5
Figure 4: Mixer	5
<i>Power Supply</i>	6
Figure 5: Power Supply	6
<i>Relay Drivers</i>	6
Figure 6: Relay Circuit	7
<i>Simulation</i>	7
Figure 7: Relay Model	8
Figure 8: Relay Simulation	9
<i>Challenges</i>	9
<i>Testing</i>	10
Design Documents.....	10
<i>Component List</i>	10
Figure 8: Components and Cost	10
<i>Raspberry Pi B+ Pin Map</i>	11
<i>Digital Potentiometers and Rotary Encoders</i>	11

Figure 9: DigPot Pinout Map	11
Figure 10: Encoder Pinout Map	11
<i>LCD Screen</i>	11
Figure 11: LCD Pinout Map	11
Raspberry Pi B+.....	12
Figure 12: Raspberry Pi B+ Pinout Map	12
<i>Mechanical CAD</i>	13
<i>Autodesk Inventor Dimension Views</i>	13
Figure 13: Enclosure Presentation	13
<i>Autodesk Inventor Assembly View</i>	13
Figure 14: Top Right View	13
<i>Enclosure Fabrication</i>	14
<i>PCB Schematic</i>	14
<i>Software Design</i>	15
Figure 15: Interrupt Flow Chart	15
<i>Standards</i>	16
Conclusion.....	16

|| INTRODUCTION

Problem Statement

Our client, Jay Becker, has many different audio sources that he listens to. He only has one set of speakers by his computer, and needs a way to connect all of these audio sources with the speakers. His current setup consists of a 3.5mm audio source going into his computer, and the combined output goes out through his sound card to the speakers, a “daisy chain” configuration. This is not ideal because it requires that the computer to be on in order to use the audio from the 3.5mm source. The problems facing this situation are unique, and as such, they will require a unique solution.

Problem Solution and Deliverables

The hardware will consist of a PCB that will be divided into 3 main circuits. The mixer will be the circuit that mixes the audio signals together into a single output. The power supply will provide power to all the ICs on the PCB, as well as the raspberry pi. The relay driving circuit will provide power for energizing the relays. Since the raspberry pi GPIO is incapable of providing enough current, we need this auxiliary circuit. These circuits will be fabricated onto a printed circuit board, and then have specific components linked to the exterior. The mixer’s interface has dials on the top left of the enclosure that allow the user to adjust the volume of each channel, and then view the respective channel level on the LCD screen. A power toggle switch activates the board, as well as a single red LED above the toggle itself. Furthermore, our audio mixer will include a single C program file to handle user inputs and adjust the output accordingly.

May 2015 Deliverables: Have one functioning, prototype mixer, along with the results of comprehensive testing, and improvement ideas for the final build.

December 2015 Deliverables: Fully-functioning prototype that meets all requirements (including Bluetooth accessible and easy assembly for creating additional mixers).

SYSTEM LEVEL DESIGN

Functional Decomposition and Systems Analysis

Our mixer must be capable of mixing three analog 3.5 mm inputs, and outputting the mixed signal to a single analog 3.5 mm analog output. The mixer must be able to handle a stereo signal. It also must be capable of driving headphones. The mixer must contain the ability to be controlled by remote, either by a smartphone app or a desktop application. The mixer must be cost-effective and easily manufactured.

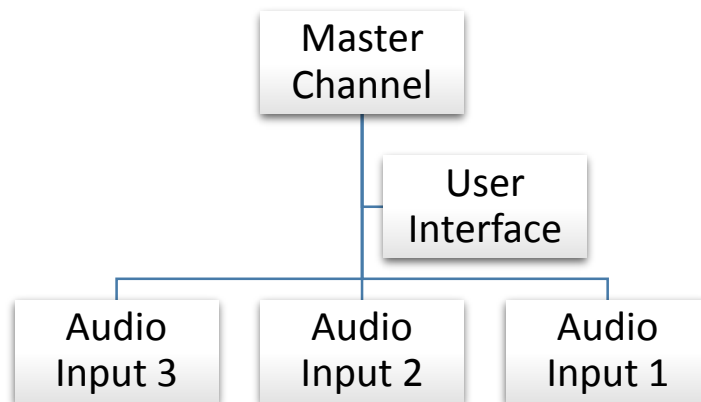


Figure 1: Block Diagram

Improvement Projections

We expect to improve our first prototype once the basic functionality has been validated through testing. The following Version Table demonstrates the potential updates.

Version 1.0	Version 2.0
White on Blue LCD screen	RGB LCD screen
3.5 mm Audio Jacks I/O	Bluetooth, Audio Jacks, and/or USB I/O
Single material enclosure	Translucent/Neopixel-boarder enclosure
CC/CCW Rotary Encoders	CC/CCW Rotary Encoders w/ mute
Mix three input channels, output two-channel stereo, and drive an amplifier or headphones.	Surround Sound

Figure 2: Improvements

DETAILED DESIGN SPECIFICATIONS

User Interface

Multiple sources must be routed to a master channel. Each input channel is given an analog audio jack on the left side of the enclosure. One master channel dial can control all three input channels to one output channel. The LCD screen has visual indicators, as well as a numerical indicators for levels from 0 to 126. The dials rotate both clockwise, and counterclockwise. This behavior will directly affect the amplitude of the analog sound signal, causing an increase or a decrease. The Raspberry Pi will store the latest level and remember upon shut down. The dials have a muting function intended to disregard any noise traveling through an inactive input. The LCD screen will allow for four character lines of information. The dimensions of the screen are about 4x3 inches, and will have white text on a blue background. The screen will be placed center-top of the enclosure. The LCD screen may have the ability of dimmable intensity after prolonged use. The power switch toggle is a simple up/down device that illuminates a small indicator LED on/off dependent upon power status.

Mechanical Dimensions

Component	#	Height	Outside the enclosure Pop Out	Length	Width	Allows for error in fabrication Diameter	From nearby parts Separation
Enclosure	1	3.0 inches front 4.5 inches back		13.0 inches	8.0 inches	-	-
Audio Jacks	4	1.25 inches bottom	Flush	2.5 inches back	-	0.297 inches	1.5 inches others
LCD	1	-	0.384 inches	4.024 inches	1.76 inches	-	-
Rotary Encoders	4	-	0.225 inches	2.5 inches back	-	0.3 inches	1.5 inches others
LED Indicator	1	-	0.118 inches	2.5 inches back	-	0.4 inches	1.5 inches switch
Power Switch	1	-	< 0.839 inches	4.0 inches back	-	0.4 inches	1.5 inches LED
Base	1			13.0 inches	8.0 inches		

Figure 3: Enclosure Specifications

Website or Mobile Applications

The client suggested that the mixer potentially have the ability to connect to the internet, and be controlled through a website interface. The site's layout would be identical in respect to location on the physical interface. After more research, the program MAX MSP could potentially help our client by employing a Graphic User Interface through MIDI channel technology.

Software

The use of the Raspberry Pi allows us more functionality in our project so that we can continue to add different inputs and features to our mixer without much change to the project

overall. Since the Raspberry Pi supports a full Linux operating system, the programming for this project must be different compared to a microcontroller with no operating system. Running our file on startup, our program will need to be run whenever the power is cycled on the mixer. This is more difficult than a microcontroller since our program must be run when the operating system is booting up.

Hardware

Mixer

This circuit will mix the audio signals together. It employs digital potentiometers to attenuate the input signals. There are digital potentiometers in total: one for each input channel and one for the master volume. Each potentiometer IC is actually a double potentiometer, because the mixer uses 2 channel stereo audio. The values of the potentiometer will be dictated by the raspberry pi. The potentiometers will communicate via I2C with the raspberry pi. The circuit contains two voltage buffers to ensure that there is no signal loss between the mixing stage and the master volume stage. There are two additional voltage buffers connected to the output so that the mixer is capable of driving low impedance loads, such as headphones. Each voltage buffer consists of a simple 8 pin operational amplifier IC with the output connected to the inverting input. A schematic of the circuit is shown below.

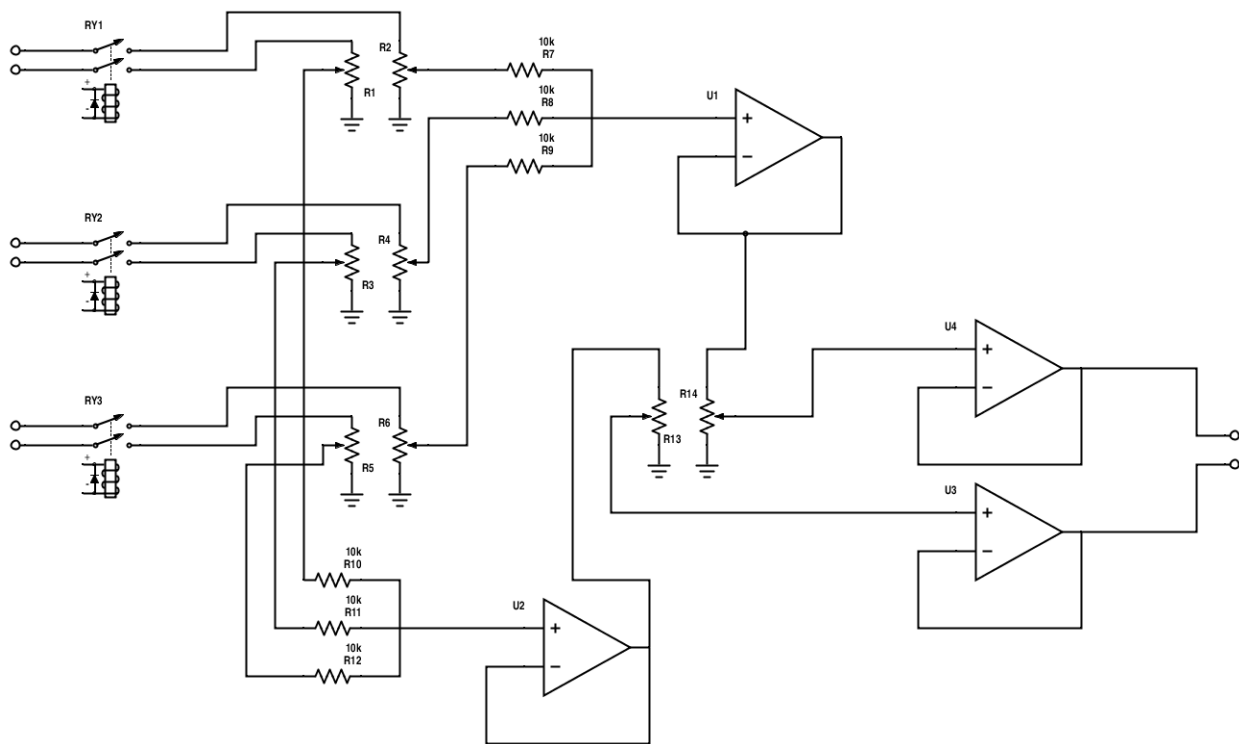


Figure 4: Mixer

Power Supply

Since we are creating an active mixer, there will be several components that need a power supply in order to operate. As such, the PCB will contain an on-board power supply that will convert 120VAC mains electricity into low voltage DC that will work with the active components. The raspberry pi requires a stable 5V power supply to operate. The LCD, rotary encoders, and digital potentiometers also operate on 5V. The op amp ICs require both a positive and negative supply rail, but the exact value of voltage is up to the user. It makes the most sense to use 5V for the positive rail on the op amps, since all other components also use 5V. Thus, we will use -5V for the negative rail for symmetry purposes. This will provide more than enough headroom for the input signal, to make sure that there is no saturation. The input from any 3.5 mm source should be no larger than 1.5V or so. A schematic of the power supply is shown on the next page.

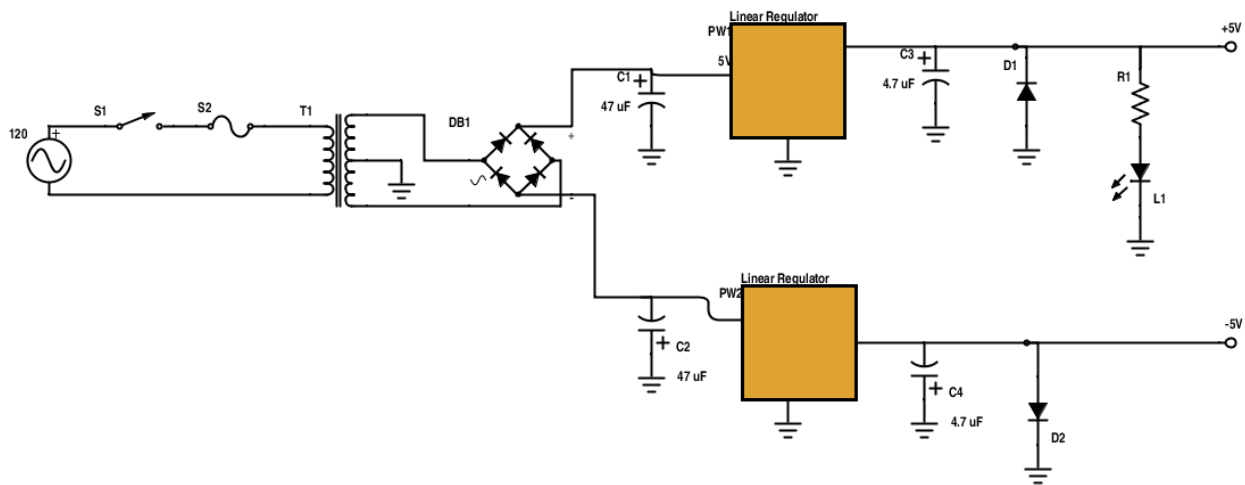


Figure 5: Power Supply

The power supply consists of a transformer, bridge rectifier, and filter capacitors. There are also two linear voltage regulators to make sure that the voltage stays constant. Since the mixer contains a microcontroller and other digital components, the supply will need to be very consistent. The supply also contains a pair of reverse biased diodes to protect the regulator ICs in case of short circuit. There is a red indicator LED mounted on the enclosure to indicate that the mixer is powered on.

Relay Drivers

The 3.3V GPIO pins on the raspberry pi are incapable of providing enough current to energize a solid state relay. Therefore, we will need to create a circuit that will switch the relays on and off. The circuit will need nine relays in total: 2 for each audio input, plus two for the master volume, and one for the LCD. We will use 5 GPIO pins in order to switch these 9 relays: one for

each audio input, one for the master volume, and one for the LCD. The inputs for the relays for the audio inputs and master volume will be in parallel, so that we can use only one GPIO pin to control both the left and the right channel at the same time. The circuit consists of a bipolar transistor biased in saturation to act as a switch, thus switching on and off the energizing current through the input of the solid state relays. The solid state relays use an LED in the input to energize the switching, so the circuit also contains some current limiting resistors to ensure that no more than 15 mA runs through the inputs of the relays. The circuit for the relay driving circuit is shown on the next page.

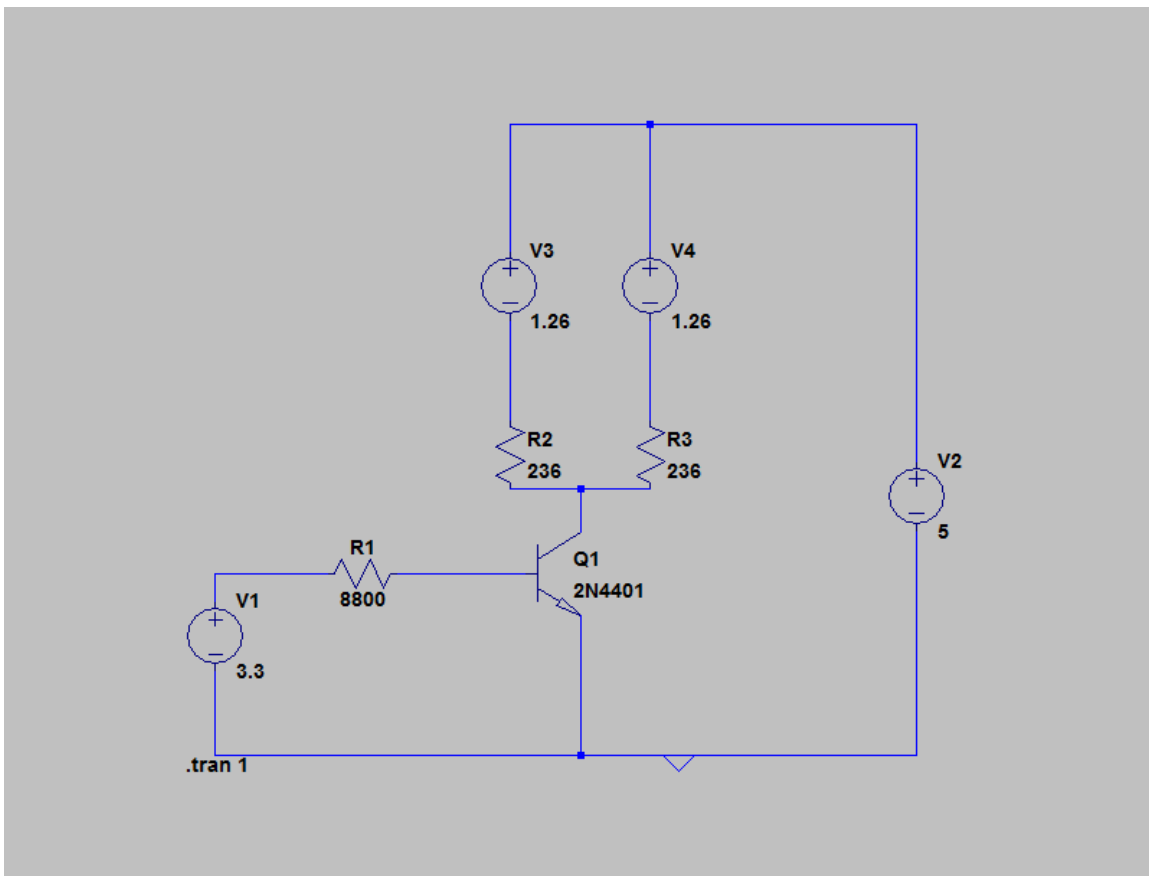


Figure 6: Relay Circuit

Simulation

By using SPICE simulation software, we can model and test our analog circuitry before we order and build it. This is very important, because it will be very costly and time consuming to modify the design after it is already built. We used LTSPICE to model the relay driving circuit, to verify that it works as designed. A screenshot of the modeled circuit is shown below.

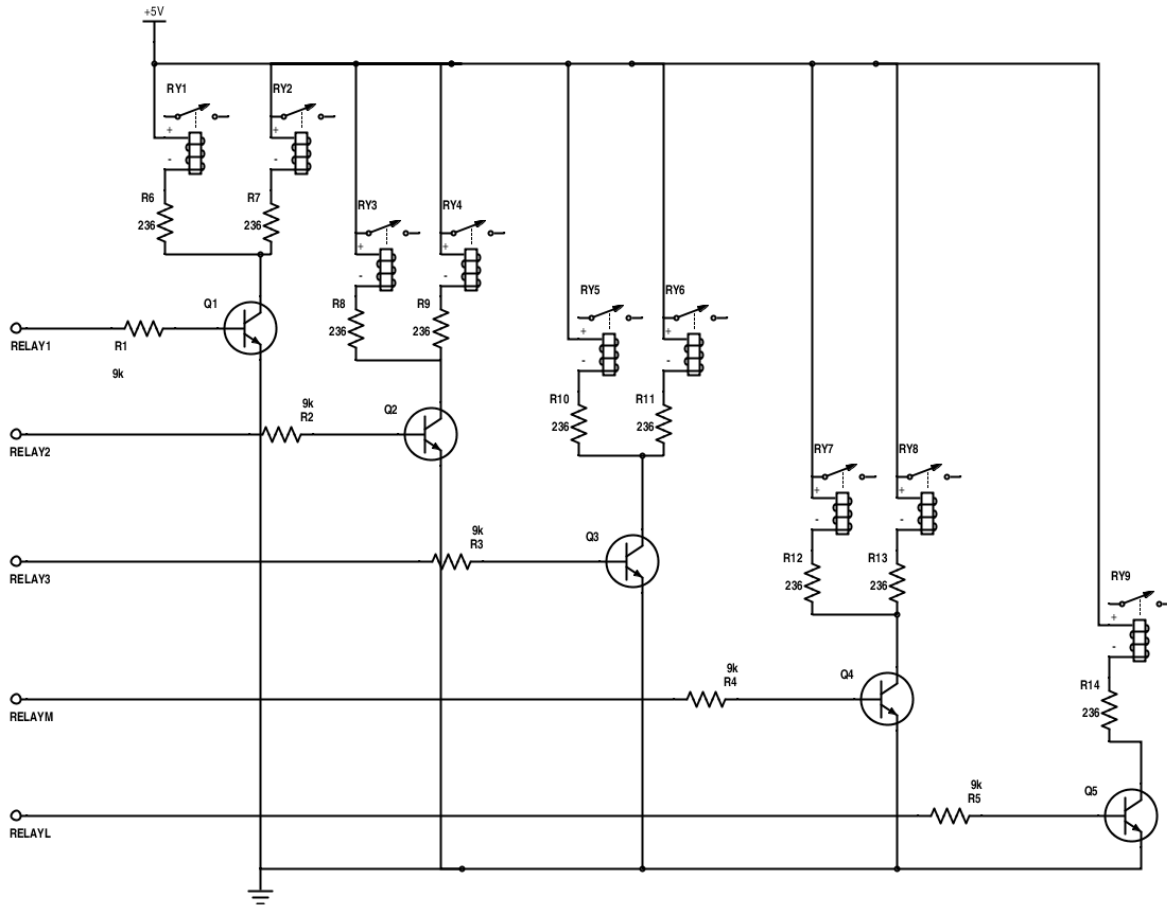


Figure 7: Relay Model

The 3.3V GPIO pin on the microcontroller is modeled as the 3.3V voltage source. The 5V output from the power supply is modeled as the ideal 5V voltage source. We will use the 2N4401 BJT to act as the switch, which is a BJT that is specifically designed to be used in switching applications. The main function of the circuit is to use a very small current from the microcontroller to switch the larger current that is necessary to energize the solid state relays. The GPIO pins on the microcontroller are not capable of providing enough current without this driving circuit. The solid state relays are modeled as a 1.26 voltage drop, because the “coils” of solid state relays are actually LEDs. On the next page is the output of the simulation.

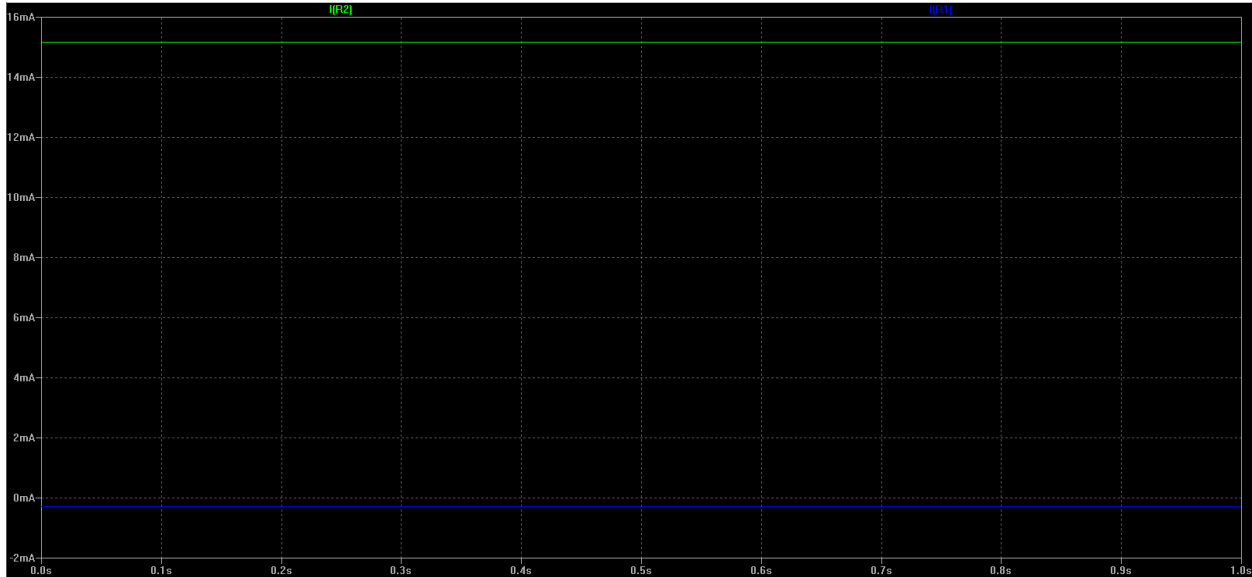


Figure 8: Relay Simulation

The green line is the current through one of the solid state relay inputs. The blue line is the current provided by the GPIO output pin on the raspberry pi. The relay input will have 15 mA flowing through it. This is enough to energize the relay, and make sure that the ON series resistance is minimal. The current outputted by the raspberry pi GPIO pin is about 0.3 mA. This is a very small current, which is well within the capabilities of the on-board raspberry pi 3.3V supply. This ensures that we don't overload the raspberry pi directly with the current needed to switch on the relays. In the next weeks, we will test the other two circuits in simulation software: the power supply and the mixer

Challenges

- Raspberry pi Polling/Interrupt response time may be inadequate for smooth user input. Additional Programming or a change in micro-controller may be needed
- Battery Backup may be required for proper state-saving in the raspberry pi in the event of an unexpected power off
- Near GPIO Limit for raspberry pi, only have a few repurpose the slots

Testing

Our primary test to ensure that the prototype audio mixer is valid will be the frequency response. Using software and equipment provided by Iowa State we can set up an efficient logarithmic sweep of the audible frequencies. We will need to use other audio equipment in order to set up an acceptable standard for signal loss, but generally we want a linear, minimal delayed response, with constant amplitude at all frequencies in the audible range. The following demonstrate possible testing situations:

- Applying various sinewave to input to measure sinewave amplitude output for each input for audible frequencies around 50 Hz to 20 kHz, while ensuring a flat response curve.
- Testing for noise of sample audio files directly through speakers vs through mixer to speakers.
- Ensuring that power cycling the device does not cause the programs to crash and lose functionality.

DESIGN DOCUMENTS

Component List

Tables below list out the intended components involved in fabrication and potential costs.

• Potentiometer	x4	Component	Estimated Cost
• Incremental Rotary Encoder	x4	Raspberry Pi	\$30-\$40
• LCD Display	x1	4 x 3.5 mm Female Jacks	\$10
• Low Power Relay/Contact for input mute	x3	2 x Analog Audio to HDMI	\$2-\$20
• Power Switch	x1	2 x Analog Audio to USB	\$2-\$20
• LED for power indicator	x1	10 x Digital Potentiometer	\$20
• Active Buffer Component for output	x1	5 x Digital Display	\$10-\$50
• Audio Female 3.5mm Jack	x4	5 x Rotary Encoder (Dial)	\$10
• 5vDC Transformer/Power Supply	x1	4 x Switch	\$5
• Linear Regulator	x2	Various Passive Circuit Components	\$5
• PCB Board	x1	Enclosure Fabrication	\$0-\$50
• Enclosure/Housing	x1	PCB Fabrication	\$10-\$50
• 40-wire jumper	x1	Raspberry Pi Power Supply	\$10
		Filtering Amplifiers/Transistors	\$1-\$20
		Total = \$125-\$200	

Figure 8: Components and Cost

Raspberry Pi B+ Pin Map

The following images outline the intended usage per Raspberry Pi B+ Pin Mapping for the digital potentiometers, the rotary encoders, the LCD screen, and the Raspberry Pi itself.

Digital Potentiometers and Rotary encoders

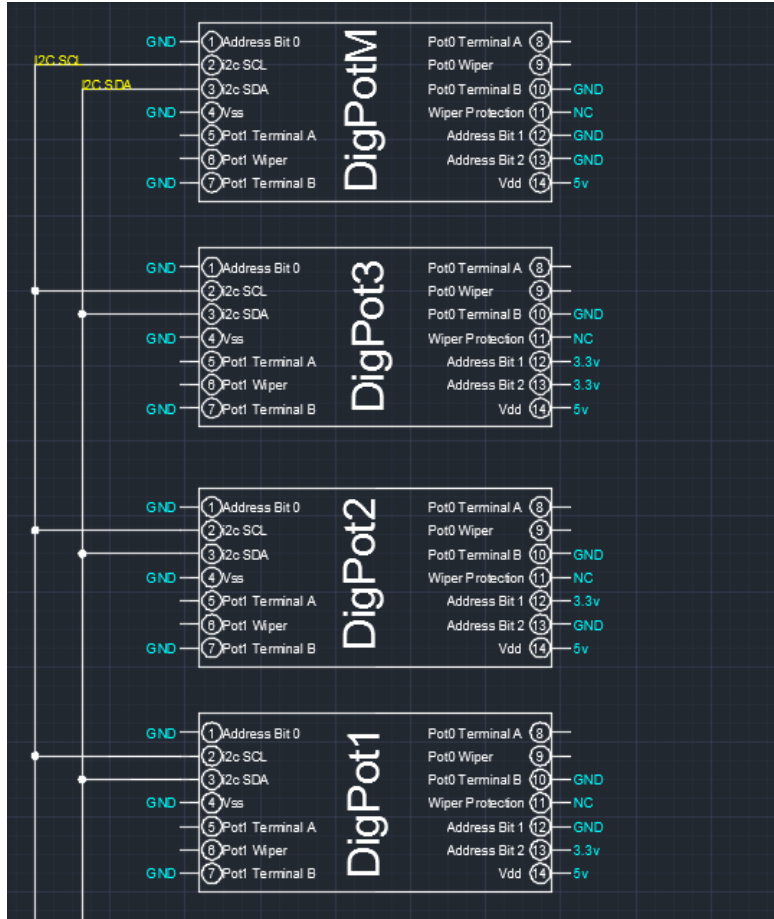


Figure 9: DigPot Pinout Map

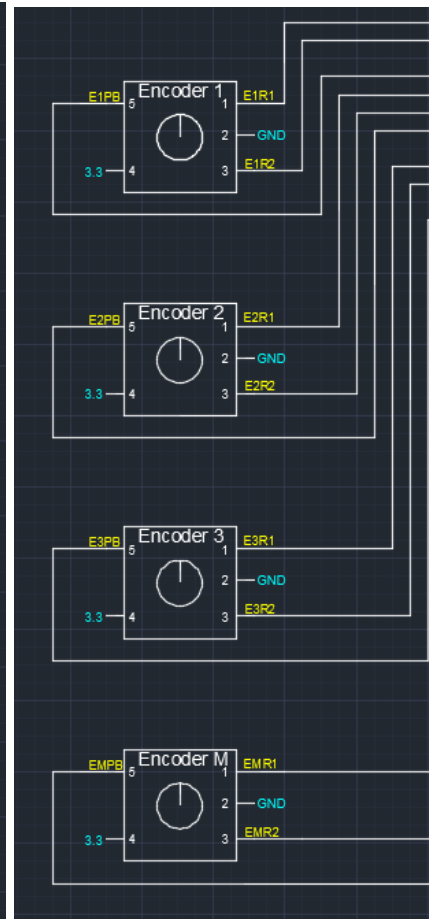


Figure 10: Encoder Pinout Map

LCD Screen

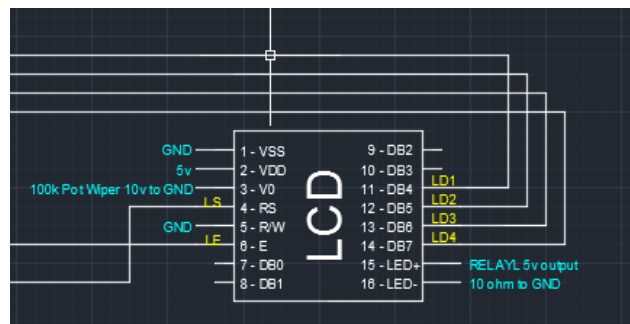


Figure 11: LCD Pinout Map

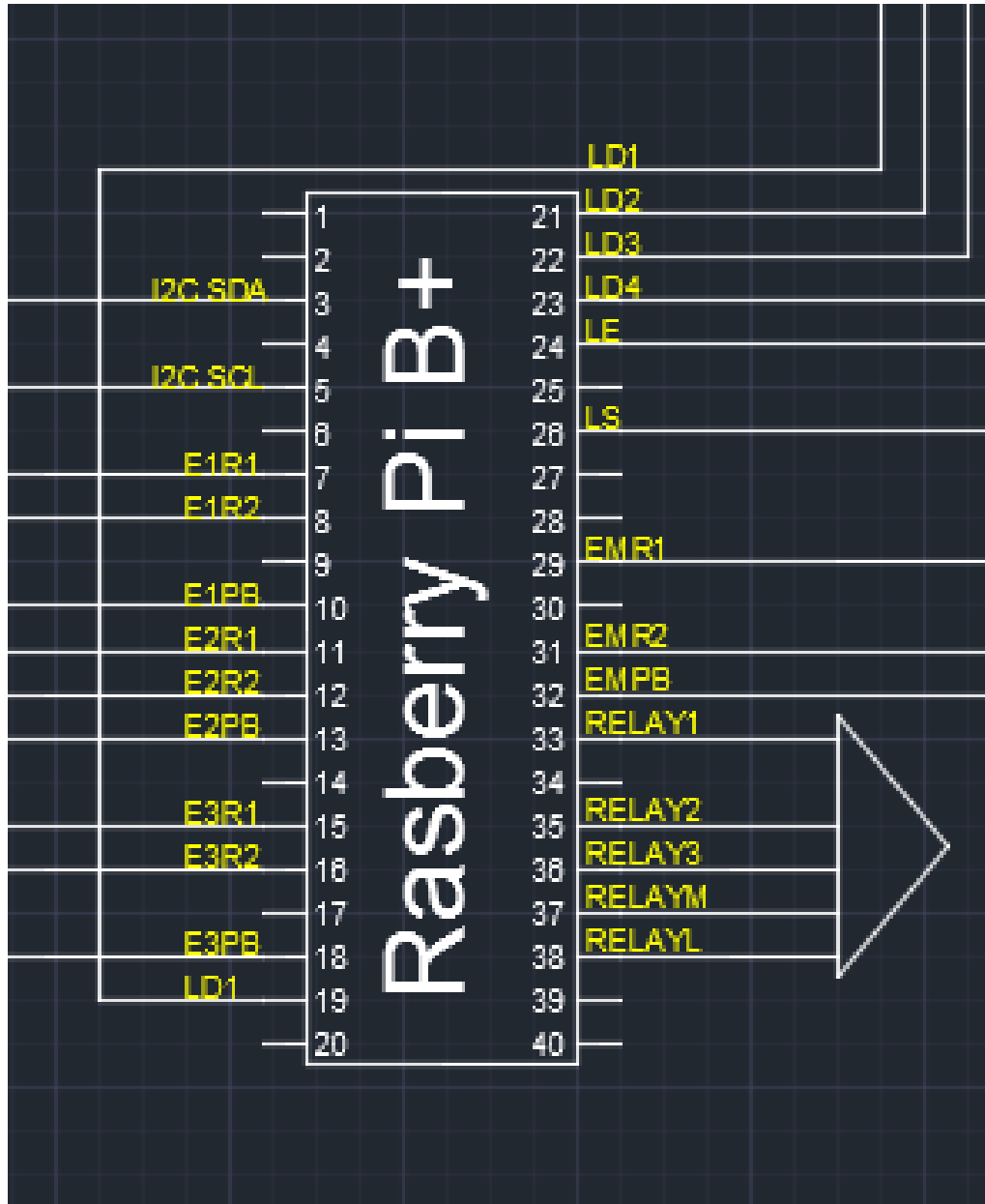


Figure 12: Raspberry Pi B+ Pinout Map

Mechanical CAD

The following images convey the exact measurements and specifications to build our first enclosure. The enclosure was drawn using Autodesk Inventor.

Autodesk Inventor Dimension Views

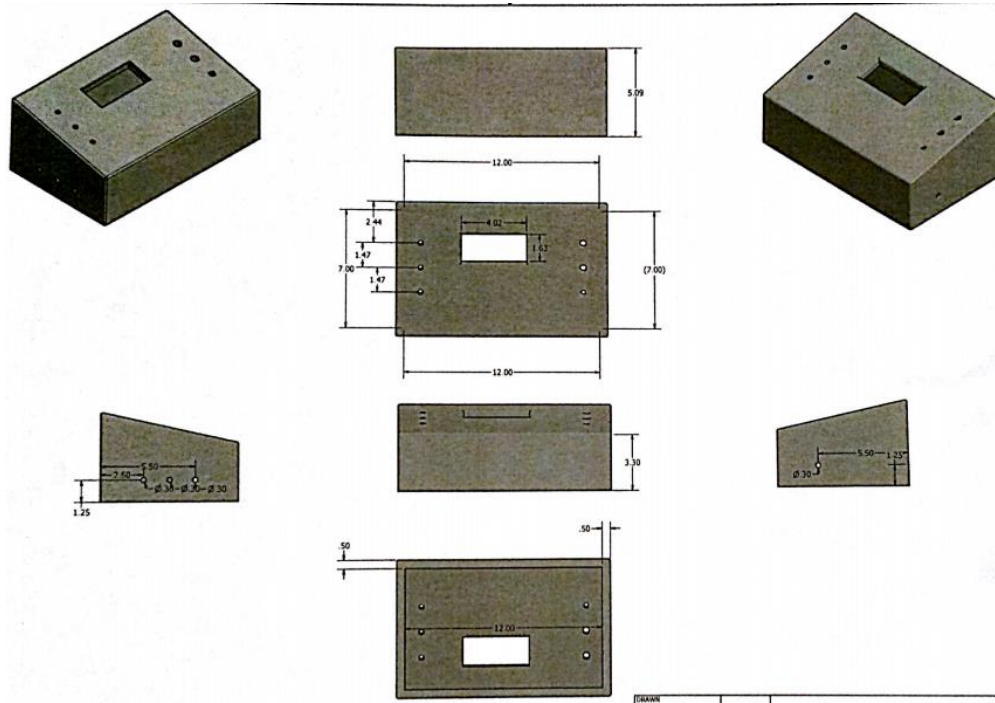


Figure 13: Enclosure Presentation

Autodesk Inventor Assembly View

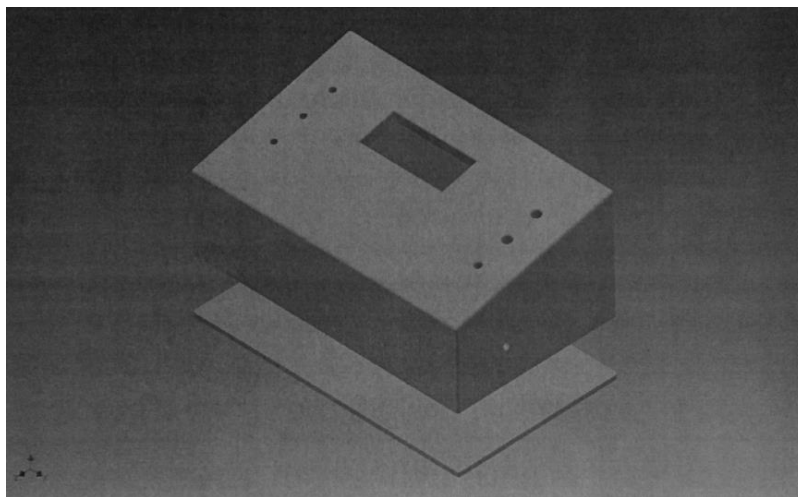


Figure 14: Top Right View

Enclosure Fabrication

After further research, the fabrication source will become more apparent. Potential ideas are: 3-D printing in Howe Hall or College of Design, utilizing the Boyd Lab in Hoover Hall, or outsourcing from Protocase.com.

PCB Schematic

To house the components for the mixer, we will create a printed circuit board. This board will combine the three analog circuits into a single assembly. The raspberry pi, however, will be on a separate board. The board will contain a mixture of through-hole and surface mount components. A wires in the mixer will be routed through the PCB. That is, the components mounted on the enclosure will have wires going to the PCB, and then there were be a PCB output interfacing with the raspberry pi. This will be much neater than having the wires go from the components directly to the raspberry pi.

We will design the board in a program called Cadsoft Eagle. This is a free program that will allow us a lot of freedom in designing the PCB. The main reason why we chose Eagle is because it has an extensive user community, with many tutorials. Since none of the members of our group has any experience in PCB design, we will need to rely heavily on internet tutorials in order to design our PCB. Once we have our PCB designed, we will pick a manufacturer in order to fabricate it. We will consult with Lee, the electronics technician, in order to determine which vendor will best suit our needs.

Software Design

The figure below demonstrates the program's logic flow of our usage of interrupts and loops.

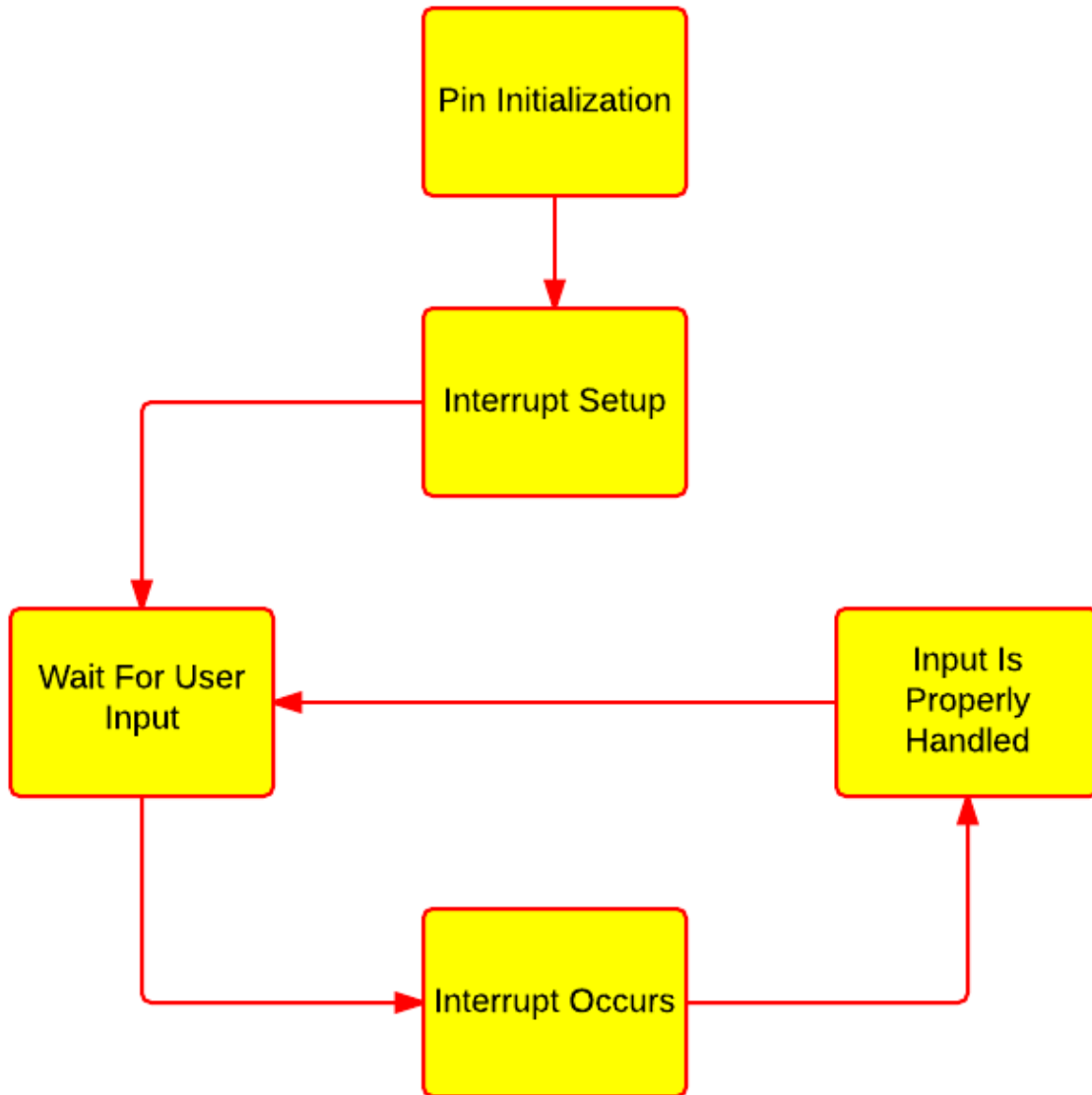


Figure 15: Interrupt Flow Chart

- 1. Naming Standards:** A naming standard must be established for our project to ensure multiple developers can understand each other's code with minimal confusion.

1.1. Variables and Constants

- 1.1.1.** All variables and constants must have meaningful names that begin with a small letter and use an underscore for a space between words. Their names must also avoid keywords that are used in the C programming language.

1.2. Functions

- 1.2.1.** Functions must have a name that identifies their purpose and conforms to the same standards of variables. Parameters in function definitions should also contain meaningful names.

1.3. Other

- 1.3.1.** Abbreviated names or names without meaning should be avoided. All names should be in English and pronounceable in case of future discussion.

- 2. Organization:** Organization of code in our project is important to make it understandable to all members of the group. Forming our code similar to the diagram allows anyone reading the code to understand what function each part of the code has.

2.1. Layout

- 2.1.1.** Our code should be laid out similar to our diagram. Global variables should be initialized at the beginning to achieve proper scope. All initializations should be done at the beginning of the code even if not used till later. Variables with limited scope must be initialized at the beginning of their scope. A fair amount of white space should be used between breaks in the code for easier reading. Indentation should be consistent throughout the code and properly show the scope of each line. A bracket should occur on the line after a function call or similar use is made. Space before and after operations must also be used.

2.2. Loops

- 2.2.1.** Loop variables should be declared just before a loop and contain the proper naming convention. The use of breaking a loop in the middle of its iterations should be avoided.

2.3. Other

2.3.1. Parentheses and brackets should be used excessively to avoid confusion in equations, function code, or in any other areas. Anything that can be done to add readability to the code should be included.

3. Comments and Documentation: Comments and documentation are important parts to removing ambiguity in our code. Utilizing both can lead to more meaningful discussions and more time saved in the long run.

3.1. Comments

3.1.1. Comments should contain meaningful info to allow any reader to understand the flow of the code. They should be easy to read and also agree with the code. Comments should contain the same indentation as the code they refer to.

3.2. Documentation

3.2.1. Documentation is also important to create clean code. Doxygen style comments and code should be performed to create documentation that can be looked at by all members of the group to achieve clarity in what is occurring in the code.

These standards will be used just as similar standards are used in other programming languages. It allows any user to be able to read the code and identify what is happening. Less ambiguity potentially allows for more member input into achieving the goal of project since some members may not be programming as much. This will help save time in the future during group discussions and also in debugging.

|| CONCLUSION

Summary

We will provide a box with simple, intuitive user interface that effectively mixes required signals for 3.5mm audio, USB, HDMI and eventually Bluetooth. Our interface will contain on/off switches, as well as volume control dials with a digital display of current volume levels. Our end goal includes a compact mixer that has minimal (Undetectable from the human ear) noise or signal loss. We also hope to keep the cost below 200\$, however this is dependent on where source PCB and enclosure fabrication. With a mostly functioning prototype delivered in May 2015, we hope to improve upon that to meet all and exceed some requirements for the December 2015 prototype. With a simple, low cost and Bluetooth capable design, we could provide benefits to a lacking area of consumer audio mixers.