Whiplash

A Technical Report submitted to the Department of Electrical and Computer Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Max McCullough

Spring, 2023

Technical Project Team Members

Leonardo Anselmo

Uriel Gomez Ibarra

John Lilly

Davis Lydon

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Harry Powell Department of Electrical and Computer Engineering

Statement of Work

Leonardo

My primary responsibility throughout the project has been the mechanical design of the device, while my secondary responsibility was the DSP (digital signal processing) algorithm. At the beginning of the project, there was a learning curve to 3D modelling since I had no prior experience; however, I do own a 3D printer, so the printing aspect was familiar to me. I started by learning to design simple objects in FreeCAD, then expanding my knowledge to fit the mechanical parameters of our specific device. The first model included a single drum arm with no interface to servos. The second model included a gear system to provide more speed to the drumstick hit; however, the third model removed the gear system since we had found quicker servos. The final model included the 3D representation of the servos within the model so I could plan the exact dimensions of the device around the real servos. I then collaborated with the team members designing the PCB to design a housing that would fit seamlessly under the main body. I collaborated with each team member to take their input and design specifications to make the device as professional as possible. Finally, I printed the device, purchased mounting hardware, and mounted the servos and drumsticks into the device.

For the DSP element of the project, I scheduled meetings with Professor Dudley White for Davis and me to understand how the DSP works. While Davis experimented with the code, I worked on the conceptual understanding so our code could run with as few lines as possible, as quickly as possible. My overall contributions to the project allowed us to have a streamline physical body that could house all elements of our project. It also allowed the project to give a professional feel, since all the electrical components are embedded and hidden away in the body.

Uriel

My primary responsibility throughout the project has been dealing with the physical design of the device. In the beginning stages of the project, I oversaw the overall schematic of the project as well as the design of the audio input circuit. To go about my responsibilities, I needed to learn KiCad [1] to build the schematics and the PCB (printed circuit board) layout. Towards the beginning of the project learning how to effectively use KiCad was urgent as it was the first step to start creating the top-level schematic for the whole project. Once I was familiar with KiCad I started designing the audio input circuit as well as teaching other hardware designers how to use KiCad. This contributes to the overall project by streamlining the learning curve on learning KiCad and allowed people to have the ability to work on the schematic as needed, thus eliminating the bottleneck effect of only having one person that can work with the schematic. The audio input circuit was then designed by as well as the power circuit of the project. The power circuit was crucial to the project as it powers two servos, the microcontroller, and several other chips that would be used in different blocks of the project. The power needs to be adequately distributed as well has regulated on the component that it will power.

Once the designing portion of the project was done, my primary task was then to assembly and test the PCB. I volunteered to do the soldering of PCB. This was integral to the overall project

since the PCB needs to be populated thus I needed to solder all the components onto the board. While soldering the components onto the board I was able to realize flaws that existed in the design. Thus, this also served as a troubleshooting step to ensure the best design possible. Then after the different board populations I oversaw the physical testing of the board. Making sure that all the connections are properly connected while also checking that the power supplies are functioning correctly, supplying the correct voltages to the correct components, and troubleshooting any connection or power issues along the way. Once the basic testing was done then I collaborated with the embedding coding side of the project to make sure that the signals from the microcontroller are correct and are to the correct pins on the PCB. Which is vital for the project to work and have a proper demonstration.

John

My primary responsibility throughout the project has been dealing with the physical design of the device, specifically the circuitry. At the beginning of the project, I was tasked with selecting the servo motors that we were going to use as well as developing the circuitry to drive the servo motors. To realize the design, I had to learn how to use KiCad software including how to import footprints and symbols as well as properly organize hierarchal schematic sheets in coordination with our other circuit designer Uriel. Once I was familiarized with the software, I swiftly designed the servo driver circuit to be above specification because the cost of the associated parts was low, and the circuit was simple in layout but needed to be robust in case of any potential issues with the servo motors. Once I finished the servo driver circuitry I then shifted to the design of the Digital to Analog Converter (DAC) circuitry that is needed to retrieve the audio output from the microcontroller. Having no previous experience with designing DAC circuits from previous course material, I had to do abundant research as well as meet with Professor Powell on several occasions to design the circuit. Overall, the accompanying circuitry was designed using previous course knowledge from the FUN (fundamentals of electrical engineering) series and Introduction to Embedded. I selected the specific DAC chip to match the bit rate of the ADC implementation and with consideration of the detailed embedded coding implementation information within the datasheet that would be useful for Max and Davis's programming considerations.

After the design portion of the project was finished, I observed and helped Uriel incorporate the DAC circuitry onto the PCB layout in KiCad to ensure proper connectivity and hardware spacing considerations. I also helped Leonardo out with some minor considerations for the mechanical housing of the device such as types of screws to be used and the removal of the center support for the drumsticks. My last major contribution to the project was the formatting and writing of the content for this report.

Davis

My primary contribution to the final project was the initial implementation and further integration of digital signal processing (DSP) into our application. Starting with a general outline

of beat detection, I was using Python through the Google Colab [2] online integrated development environment (IDE). This allowed me to pull a variety of test resources, specifically the Librosa [3] audio development library. I developed a simple ten-line beat detection algorithm from there. Further application of this code worked to analyze the beat detection algorithms being used in both this context and in other algorithms, specifically those used in other microcontrollers. As the code being used in the final project was either going to be C or C++, translation of code from the test environment to embedded language was necessary. To speed up this process, an online C/C++ IDE called OnlineGDB [4] was used for both a coding environment and a debugger for our program. During this process, the final part of the algorithm, the actual beat detection from a converted audio signal was finalized into a non-frequency domain semi-peak detection. After this was ensured to work with test data, the code was imported into the microcontroller to ensure success with the hardware. Further work was needed with both the algorithm and calling of the DSP code at this point, such as delay study in relation to sampling periods. The algorithm ends up calling multiple times to ensure proper working order over a given period of time. The final portion of my work was working with Max to ensure the output from my code could properly be sent to the hardware for processing. Throughout this process, I was also meeting with Professor Dudley White to study and discuss possible signal processing

Max

My primary contribution to the project was planning and writing the code for the input and output of the microcontroller. This consisted of three main parts: the servo control code, the analog to digital code for audio input, and the digital to analog code for audio output. The first step in this process was setting up the software development kit for the CC3220SF [2] and ensuring it worked with a simple program. I then figured out how to control the servos by looking over the data sheets to figure out exactly what kind of signal they required. I used the Texas Instruments PWM drivers [6] to simply the process so that I could focus more on the logic behind controlling speed and switching servos. One of the most important techniques that I used was I/O abstraction. I had to do this so if any of the output pins ever need to be changed it could be done effortlessly. I also took advantage of this by outputting the servo control code to LEDs on the microcontroller to ensure that it was working correctly without access to the actual servos. I used the Texas Instruments ADC drivers [6] to take the audio signal and store it into an array using 16-bit integers use our limited memory as effectively as possible. To test this part of the code, I inputted DC values into the microcontroller to verify that they were correct values. The final part of the software that I designed was the code to interface with the digital to analog converter. This required the use of SPI, which I implemented myself apart from the use of Texas Instruments GPIO drivers [6]. This code required the control oof values of each of the three required pins. Those pins included, chip select, data, and clock. During final integration phase I worked with Davis to make sure that our code was able work effectively together.

Table of Contents

Table of Figures

Table of Tables

Abstract

This project, titled Whiplash, is a device that autonomously plays a physical drum to the beat of an input song. In addition to accurately detecting and playing to the beat of the song, the strength of each drum hit will also depend on the current energy of the song. The system includes a microcontroller that performs digital signal processing on a given input song and powers two servo motors that rotate drumsticks to hit a drum. A mechanical body was designed and 3D printed, ensuring that the structure is sturdy. The device can play a physical drum which will be the basis of our experimentation. We were given the constraint of $500, any tools or parts we could harvest from previous projects, ABET guidelines, and a deadline of December 12th, 2022. Given the constraints, we were able to successfully satisfy many of the conditions outlined in our proposal. The device can play to any audio played through the aux cable connection.

Background

Our initial consideration for choosing this project was the idea of building a completely autonomous and non-human member of a band. Finding a drummer is typically a difficult task, especially when it comes to playing sets, damaging expensive drum equipment, and transporting the materials around. We wanted to scale down this project for a proof of concept, as many of us are musicians. We also wanted to be able to combine our creativity and engineering skills into one project.

There currently exist several working models of robotic drum-playing machines. One of the most prominent robotic-humanoid drummers is “Stickboy” [7]. Stickboy is a six-armed robot that has been on tour since 2007. Stickboy must be programmed to play, as it has no internal algorithm to play autonomously to a specific song. This means that while Stickboy may be much more capable of playing than any drummer, it is also incapable of creating its own music. On the other end of the spectrum, the Polyend Perc [8] is an entirely functional device capable of playing an entire drum set. The device involves aluminum enclosures that contain small cylinders to “beat” the drum, rather than strike it. The Perc also has no algorithm to automatically track music and play along, as it must be entirely programmed to the user’s specifications. Below in **Figure 1** are the two devices.

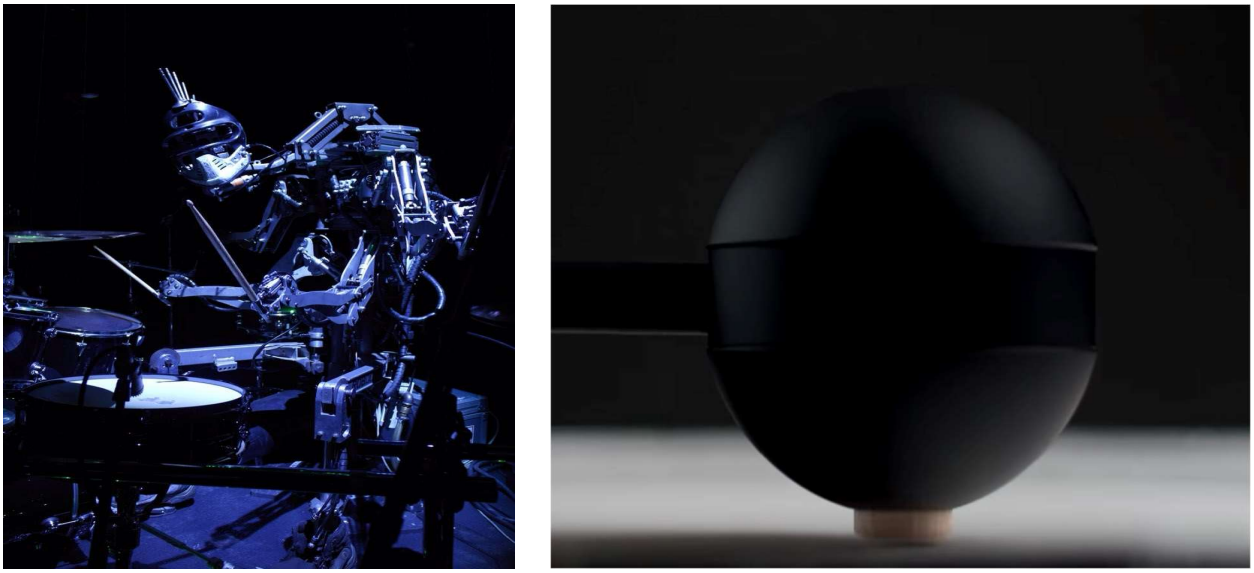

Figure 1: (From Left to Right) Stickboy and Polyend Perc

While both Stickboy and Polyend Perc fill their niche of being able to display drumming capability, neither can play without being programmed to a song's specifications. Our device, Whiplash, utilizes a digital signal processing (DSP) element to autonomously track the beat of a song and play a physical drum to that beat. Rather than hard code each drum hit as the current technology does, we can focus on making our project generalizable to any musical audio using a temporal algorithm. Temporal algorithms, or software that focuses on time precision, have been used previously to train robotic musicians to play music [9], though Whiplash does not utilize any reinforcement training.

Previous coursework that was used came from Fundamentals I, II, and III (ECE 2630, ECE 2660, and ECE 3750), Intro to Embedded Design (ECE 3430), CS classes involving Python [10] and C [11] programming (CS 1111, CS 1112, CS 2110, CS 2150), Music Theory (MUSI 4582), Electromagnetic Fields (ECE 3209), and some Digital Logic Design (ECE 2330). From the Fundamentals series, the most important topics that we used were circuit design theory and software to create a board that controls the device, power supplies, and audio analysis. Because we required a microcontroller, we used our knowledge from embedded design to program a Texas Instruments (TI) Mixed Signal Processor (MSP) [12] in Code Composer Studio [13]. We also drew on some music theory problems involving rhythms and beats per minute (BPM). As far as electromagnetic fields and digital logic design, we had to pay attention to specific theory when creating the board or building components, watching for electromagnetic field (EMF) interference or logic errors.

Physical Constraints

Design Constraints

When it came to designing the PCB there were physical limitations that we had to work around. The first of which would be the size of the final PCB, which was sixty square inches. That was the constraint that was given for the team to be able order the PCB using the student ordering that was given to the university. Another physical constraint that we had to work with was the availability of physical components and parts making sure to choose that parts that were available when we needed them and making sure that they were available in large quantities. As well has dealing with tolerances within the electrical components. Another constraint that we had to work with is our power supply in which we decided to work with a wall transformer that would allow us the be powered off a standard U.S. wall receptacle.

Given our choice to use the CC 3220SF launchpad limited use with limitations of the CC3220SF. Given that this is a board provided by Texas Instruments, we worked only in the Code Composer Studio (CCS) to program and debug all of the code that was going to be used in the project. The CPU (Central Processing Unit) has an 80MHz core, 256KB of RAM, and 1MB of executable flash [5].

Cost Constraints

Our primary cost constraint was the budget of $500. The bulk of the expense was with servos, components, and PCB, with availability being the largest concern in this facet. Availability constraints prevented us from purchasing several components and having to plan our project more carefully around when materials would be available. The servos were about 20% of the overall budget. The PCB manufacturing and components were about 50% of the budget. This is because we ordered two PCBs and had to end up replacing/adding many components, such as better heat sinks for the voltage regulators. Mechanical costs were about 15% of the budget since our only overhead was mounting hardware and filament. The remaining 15% of the budget was kept for emergencies and ended up being miscellaneous purchases such as a drum and upgraded components.

Tools Employed

Code Composer Studio (CCS) [13]: This program was used to program the MSP432 microcontroller. We were already familiar with CCS since we had used it previously in our Intro to Embedded course here at UVA. We were able to greatly improve our embedded programming skills since our only microcontroller was the launchpad we chose.

Texas Instruments Filter Design Tool [14]: This program is an online tool that is able to assist in designing several filter types. We had no previous experience with the designer, so we had to learn to use it initially. The tool was then used to design our anti-aliasing filter for both the ADC (Analog to Digital Converter) and the DAC (Digital to Analog Converter).

KiCAD [1]: This program was used for all of our primary circuit design and creation. We created the circuit schematics and PCB layout using it. We didn't have much experience with it before this class so there was a learning curve to get it started. We also used GitHub to upload our circuit files so we could share them across devices.

FreeCAD [15]: This program was used for all of the 3D CAD (Computer Aided Design) design. We were able to import models of our servos and build the design around them. We had no knowledge of 3D design or any software beforehand, so we had to learn to use the entire program.

PrusaSlicer [16]: This program was our slicing software for our 3D printer. Slicing software breaks up the 3D model into layers that will be printed on a 3D printer. We had plenty of knowledge of this program beforehand, so using it was relatively simple.

C [11]: We used the C programming language for our embedded code since the MSP432 uses the same language. We had plenty of coding experience in C previously, so we had to learn how to apply it to our specific project. The DSP code, servo code, ADC code, and DAC code were all written in C.

C++ [17]: We used the C++ programming language as a precursor to writing C code. It was much simpler to write several of the DSP functions in C++ first and then convert them to C. We had knowledge of C++ previously, so applying it to our device was not much of a learning curve.

Python [10]: We used the Python programming language to conceptualize our code before converting it to C. It was easier to write it this way because Python has plenty of libraries that allow us to create a beat detection algorithm; however, the language was not used in the final deliverable. We had plenty of experience with Python previously so using it was easy.

Multisim [18]: We used Multisim to simulate the anti-aliasing filter. We had experience with Multisim previously, so we were able to get the circuit built and tested quickly.

Visual Studio Code [19]: This program was the code editor we used to test our functions for the DSP, ADC, and DAC. We were familiar with this program, so we were able to test functions in it before applying them to our microcontroller.

Online GDB [4]: This program was used to debug simple code online. We didn't use it much beyond the initial coding phase, but we had some experience with it previously because it is easy to use and runs quickly for small programs.

FreeDFM [20]: This service was used to check our PCB and ensure that it was able to be printed and manufactured. We had experience with FreeDFM from previous classes, so converting our files and sending them in was simple.

Google Colab [2]: This service was used primarily for testing code to be shared between team members. It is useful for mid-sized programs. We had some experience with it before so there was little new learning involved.

Prusa MK3S+ [21]: This is the printer that was used to print our device. We have had plenty of experience with this specific model previously, so using it to print was much more efficient than it would have been with other printers. We were able to adjust the settings and use everything we needed specific to our project.

GitHub [22]: This service was used for sharing all of our code and transferring it between devices. We created a repository specifically for our project. We all have had experience with GitHub previously, so we were able to improve our skills and become more efficient.

Societal Impact Constraints

Environmental Impact

The environmental impact of Whiplash is expected to be moderate. The vast majority of the components used in our design are RoHS or RoHS3 compliant, meaning that they have below a specified maximum level of 6 or 10 restricted substances as defined by the EU [23]. However, the production of the printed circuit boards (PCBs) needed for the design of Whiplash have been shown to produce wastewater that is heavily contaminated with copper, iron, and organic matter [24]. Despite the seemingly multiplicative negative effect of occupying PCBs with numerous electrical components, methods of recycling waste PCBs have been shown to have economic and environmental viability [25]. Additionally, many of the components that populate the PCB can be harvested and reused for future applications. Furthermore, there is no presence of solar powered devices or lithium batteries within our design which are two major causes for environmental concern in electronics [26]. Lastly, the carbon footprint of powering Whiplash is relatively low, about the same as a curling iron [27].

Sustainability

For the mechanical model, printing and recycling is a sustainable process. The physical design is made of PLA filament. PLA is a “bioplastic”, meaning it is non-toxic, biodegradable, and recyclable [28]. The primary concern with the printing process is the energy usage of the 3D printer itself; however, the usage is between 0.05kWh to 0.15kWh [29]. The PCB portion of the device is unfortunately not as sustainable. The components are able to be recycled and reused [30]; however, the PCB itself cannot. PCB manufacturing is also a high-emission and high-energy process involving copper, resin, glass, and water. There is an estimated two million tons of wasted PCB material currently on earth [31]. Overall, Whiplash is not sustainable for the sheer fact that not all processes to create it are sustainable. We can help Whiplash become more sustainable by limiting the size of the PCB, using quality components, and increasing the life of the product as much as possible before it becomes waste.

Health and Safety

In terms of safety, one of the larger concerns associated with Whiplash is the potential of injury from the strong rotations of the drumsticks affixed to the servo motors. Although we have designed our servo motors to only rotate at a speed that is necessary for a clear audible sound, there is a possibility that if our power regulator circuit experiences an irregularity, our servo motors could drive the drum stick to turn harder and faster than it normally would. However, we are confident that even if this event were to happen, the maximum speed of the drumstick would not be enough to seriously injure the user. Another concern that we have addressed is the presence of circuit components that do contain toxic materials, even though they are RoHS certified. Although the capability of poisoning is there, it would require significant human intervention as the circuit components were designed to be contained within the housing of the device that can only be accessed by removing screws. Therefore, we conclude that the risk of injury to the responsible user is very low with responsible usage [23].

Ethical, Social, and Economic Concerns

Whiplash helps in solving several problems by providing a musical companion to practice with. The user will receive the benefits of physical feedback while being able to practice with a partner that cannot tire. The user's only responsibility is to choose their desired song and Whiplash will do the rest. One may argue that Whiplash is no different than a metronome. While Whiplash only plays to the beat of a song, it also adapts to any shift in tempo and beat that may occur during the song. A metronome is only capable of playing a constant beat, so Whiplash enables a musician to seamlessly practice an entire piece with no pauses to adapt the device to each part of a song.

The main ethical consideration that exists around Whiplash is the potential for it to take away employment opportunities from drummers. Much like any new robot that can do part or all of someone's profession there is a concern here. In this form, Whiplash will by no means be a replacement for a professional drummer entirely, but it could remove the need to hire a drummer in select situations. For example, if someone wants to hire a drummer to play relatively simple drums over a prerecorded song. If Whiplash were to be developed beyond what is mentioned in this proposal it could potentially take more opportunities away from drummers, limiting their opportunities for employment.

External Considerations

External Standards

For safety concerns, the only ones we were directly concerned with is NFPA constraints [32]. Our product is not being marketed towards children, used with a medical functionality, operated at a high voltage near water, or uses any toxic or otherwise harmful substances. In addition, our device does not use any radio or microwave frequency, with the only communication standards being present are those found on the MSP, with its Universal Serial Command Interface operating with both multiple serial formats and asynchronous ones like SPI, I2C, UART, etc. [5]. To ensure the quality of the printed circuit boards that will be used on this device we must adhere to manufacturers' specifications. For the design of the board Advanced Circuits' FreeDFM [20] specifications will be followed. Their online checker will also need to be used to verify that everything complies. Since the board was assembled by us, the minimum requirements for PCB Assembly Documentation will be followed so that the boards are produced in the expected manner. The AC (Alternating Current) to DC (Direct Current) transformer connecting the various parts of this system were also in accordance with NFPA's Nation Electrical Code [33]. Another standard that was followed when designing the PCB was conforming with the standards set by IPC, such as standards on solderability, repair, storage, handling and electrical testing [34].

Intellectual Property Issues

This project is likely patentable based on other patents that exist and contain conceptual overlap with our design, but do not function identically or for the same purpose as our design. Three of the patented designs that have functional similarities to our design are detailed below:

The first patent details a "Music instruction system" [35]. The main independent claim of this patent asserts "receiving a user selection of a musical piece; providing performance cues to a user to perform musical events on a musical instrument" as well as "providing real-time or near real-time audio feedback." Our project also takes in an audio input from the user; however, the difference is in the input method. In this patent, the audio input type is described as "a musical instrument may produce an electrical signal to be used as an audio input signal," conversely, our device is designed around taking in a song and parsing out beat and tempo. Additionally, while our device does also provide near real-time feedback, the feedback is physically reproduced and is audible as a result. Due to these fundamental differences in functionality, our device is still patentable despite the claims from this patent.

The next patent details an "Interactive, expressive music accompaniment system" [36]. The main independent claim of this patent details a "sound-signal-capturing device;" a "signal analyzer;" and an "electronic sound-producing component that produces a rhythm section accompaniment." Additionally, another dependent claim presented by this patent describes the functionality of voice recognition for timing the rhythm section accompaniment. Considering these claims, our project would still be patentable because this patent uses a microphone to capture the audio

signal whereas ours uses an auxiliary connection to pass our audio signal. Additionally, our project does not accept or use any feedback from the user other than the initial song input, whereas this patented design does.

The last patent details a “System and method for automatic drumming” [37]. The main independent claim of this patent details a “system for automatic drumming... output signals according to said MIDI serial data....at least one actuator configured to hit at least one drum according to signals outputted.” Although functionally similar, our device uses a microcontroller with SPI interfacing to create an output signal to our servo motor whereas this patent details a MIDI decoder to produce the output signals to drive the linear actuators. Furthermore, a dependent claim in this paper details a PWM (Pulse Width Modulation) duty cycle or output signal duration that is user-defined. In contrast, the PWM signal duration and duty cycle within our project is determined by the characteristics of the beat detection algorithm. While functionally being similar in design to this patent, our project is also intended for use as an educational device whereas that is not specified for the design of this patent. Consequently, with the clear differences in design choice and intended purpose, we can conclude that our project is patentable.

Detailed Technical Description of Project

Background

Whiplash, named after the movie of the same title [38], is a device that is able to detect the beat of an audio input and play a physical drum along to the beat of a song.

The chipset we have chosen is a Texas Instruments (TI) mixed signal processing 432 (MSP432) chip. We are also using a TI CC3220SF launchpad [5] that enables us to access each input/output of the chip easily. The user plays audio from their preferred device (phone, computer, etc.) and the signal is passed to the microcontroller. The microcontroller then performs the DSP to parse out the song's beat. The beat signals are then transformed internally into pulse-width modulated (PWM) signals that can be read by servos, specifically HS-805BB servos [39]. The generated PWM signals are sent to a servo driver printed circuit board (PCB) created to protect the MSP432 and servos from overcurrent. The servo driver amplifies the signals and sends them off to the servos, enabling them to spin the drumsticks and hit the drum. Below in **Figure 2** is the block diagram for the device.

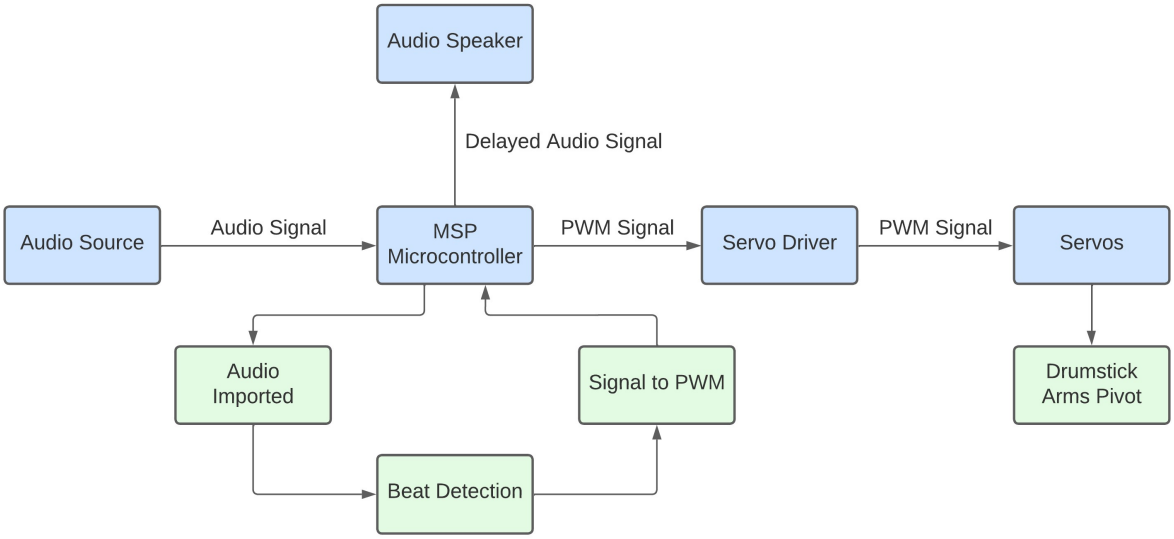

Figure 2: Whiplash Block Diagram

The device will have several other features. The strength of the drumstick hit will depend on how loud the song is playing at the moment of that hit. The device will also have an output to a speaker, so the user is able to hear the music live. The output speaker will have a delay when playing audio so the device can hit the drum exactly when it is expected to. The physical body of the device was designed using FreeCAD [15] and printed using a 3D printer. The body has mounts for the servos, while the drumsticks will be screwed into a mounting plate on the servos themselves. A model of the 3D design is provided in **Figure 3** below.

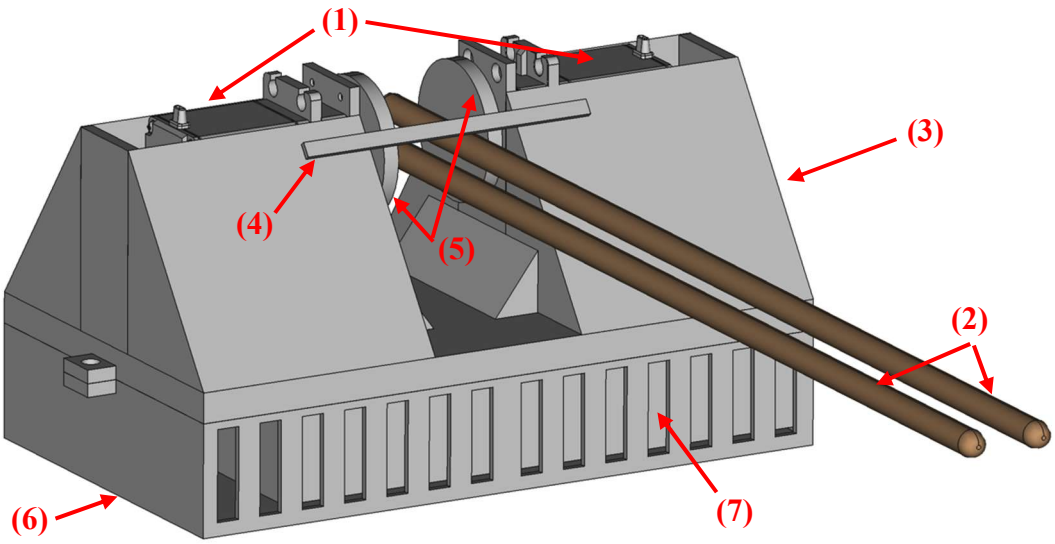

Figure 3: Whiplash 3D Model

(1) Servos: The HS-805BB servos are the main driver for the drumsticks to move. They are housed directly in the body and attached via machine screws to the main body.

(2) Drumsticks: The drumsticks are made of wood and attached to the servos via the servo mounting plates and wood screws. When the servos turn, they turn and hit the drum.

(3) Main Body: The main body is made of PLA filament and houses all the equipment, supports, and mounting pieces. The servos are directly attached to the main body.

(4) Stop Bar: The stop bar is made of PLA filament and stops the drumsticks from overshooting their return position. This saves some strain on the servos and ensures that our drumsticks will be ready for the next beat hit.

(5) Servo Mounting Plates: The servo mounting plates are made of plastic and fit directly into the rotary portion of the servo. The plates fit via matching indentations in the mounting plates and grooves on the servo rotary shaft.

(6) PCB Housing: The PCB housing is made of PLA filament and houses the PCB along with all necessary components and wires. The PCB housing attaches to the main body via machine screws through attachments that protrude on the sides of the main body and PCB housing.

(7) PCB: Inside the PCB housing is the PCB. The PCB is mounted via standoffs that sit on the housing and are screwed directly into the PCB and PCB housing using machine screws.

Components Used

First off general resistors, capacitors, and general connections to connect to the CPU. Then starting from the power circuit we will use a 12V 3A AC to DC wall transformer, that goes into a DC jack [40]. Then there are three different voltage regulators. Two of the voltage regulators are 6V 1.5A regulators (NJM7806FA-ND [41]). They have heatsinks connected to them. Then there is a 3.3V regulator (REG104GA-3.3 [42]). Then there are two sets of servo driver circuits; these are powered by the two different 6V regulators. Both circuits are identical and comprised of a transistor (FQP30N06L [43]), a set of resistors, and the servo (HS-805BB [39]) itself. Then there is the initial audio filter that is comprised of resistors, capacitors, an op-amp (TLV272IP [44]), and an audio jack. From there the signal then goes into an anti-aliasing filter comprised of more resistor, capacitors, and different op-amps (LMC6035 [45]). From there the signal goes into the CPU, from the MSP signals go into the servo drivers as well as the Digital to Analog Converter (DAC) (TLV5618ACD [46]). Which then goes into another set of anti-aliasing filter using different op-amps (TLV2170IDGKT [47]) then going into an audio amplifier (NE5532ADR [48]) to end at an audio jack.

Design Decisions and Tradeoffs

One of our first decisions was that to achieve proper attenuation for the ADC on the CPU we had to design the anti-aliasing filter. The anti-aliasing was designed to be an 8^{th} order Butterworth filter with a stopband frequency of 22kHz with -100dB attenuation. The T.I. filter designer toll was used to get the proper values of the capacitors of the filter alongside performing a Monte Carlo simulation on the components. The initial audio filter is a standard low pass filter using a summing amplifier to change the signal from two channels to a singular channel, with a corner frequency of 20Hz. We had to trade changing the signal from stereo to mono because we must work with an ADC that can only take in a single audio signal and cannot do multiple inputs. Due to the power draw of the servos 6V 1.5A regulators were used to power each servo. Overall, a 12V 3A AC to DC wall transformer was used to power the whole circuit. For the servo driver circuit, a transistor was used to simplify the components that would be needed to complete the servo driver circuit. Another tradeoff that we had to make was the fact that our CPU did not have a DAC included and thus we had to design a DAC circuit, which also required us to design another anti-aliasing Butterworth filter. From there, the signal was sent to an audio amplifier, however once again the tradeoff of having only a mono signal came into play that meant that the ending audio output is mono as well.

Schematics

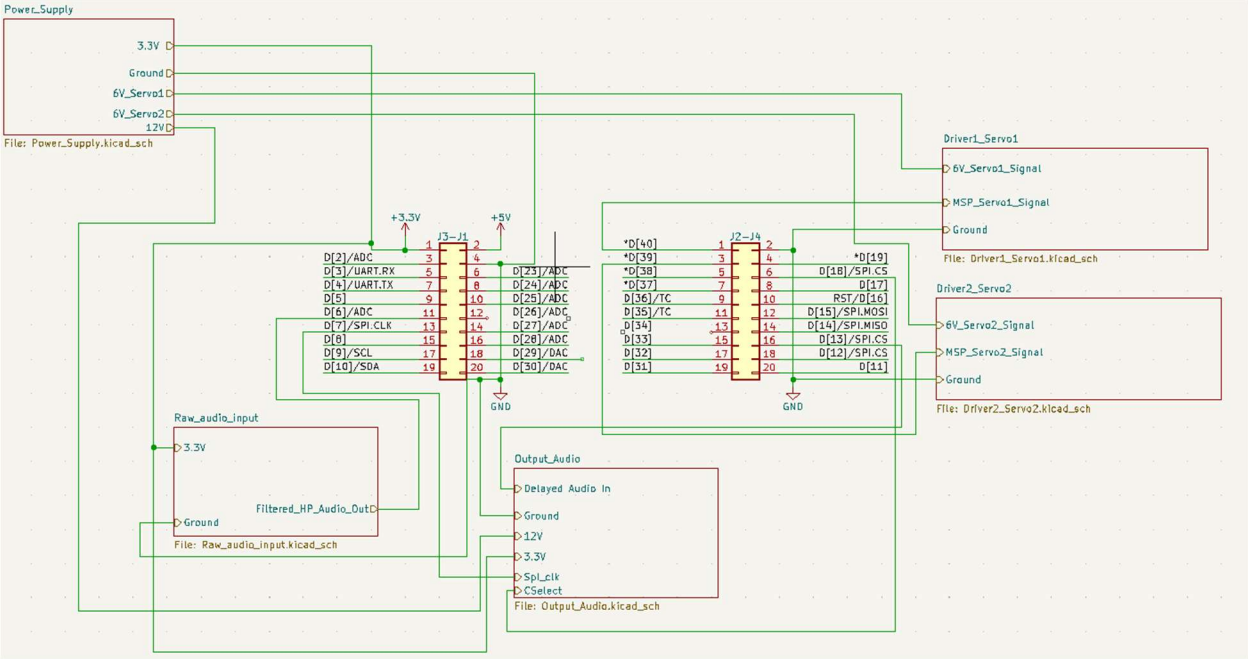

Figure 4: Top Level Schematic

The top-level schematic in **Figure 4** shows the how the different block connect with each other and shows the connections that will be done between the CPU and the different blocks as well as how the power circuit interacts with the different blocks to make sure that everything get powers. The main connections present are the Voltages coming from the power circuit going into the different blocks, as well as the CPU connections. Having the CPU send PWM signals to the servo drivers, having the CPU accept the input from the audio input circuit, and having the CPU send a digital signal to the DAC to the audio output circuit.

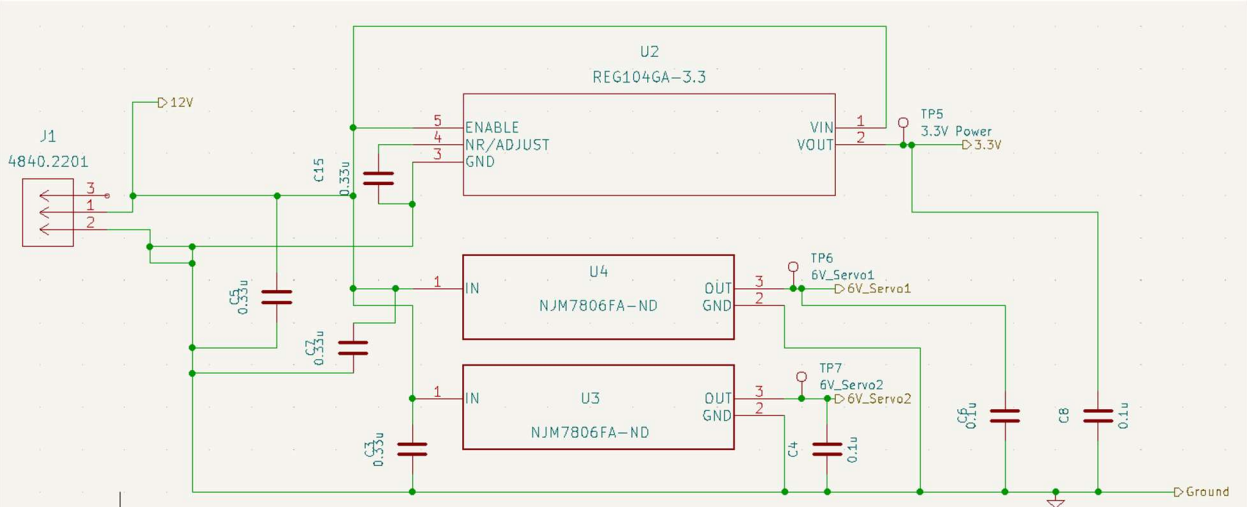

Figure 5: Power Supply Circuit Schematic

Here in **Figure 5** we see that J1 is the DC power jack, U2 is the 3.3V regulator, U3 and U4 are the 6V regulators, and various amount of bypass capacitors. There are test points after each regulator to ensure that each voltage regulator is outputting the correct voltage. The capacitors are there to then limit any AC voltage that would be going into the circuit.

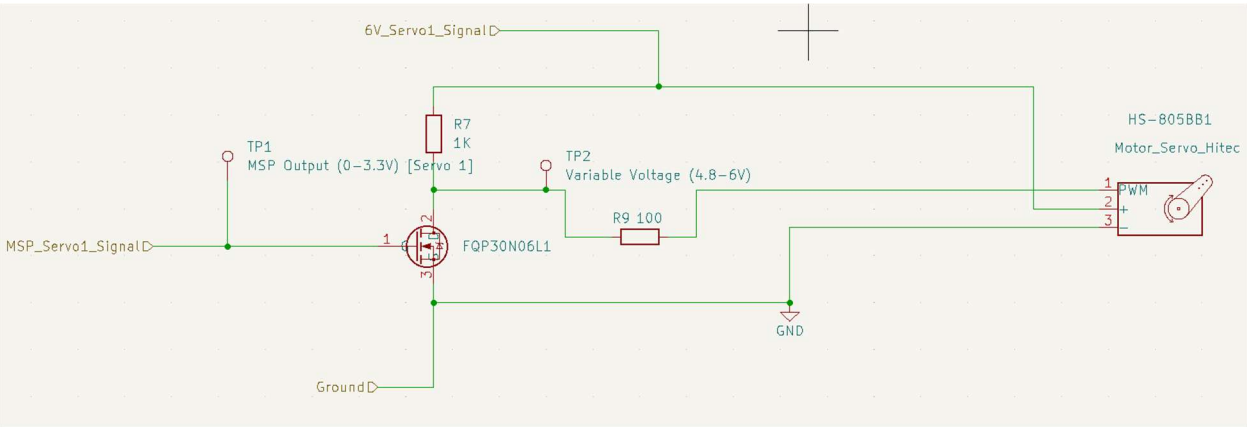

Figure 6: Servo Driver 1 Circuit Schematic

The servo drivers in **Figure 6** and **Figure 7** were designed to amplify the PWM coming from the CPU to be a 6V PWM signal in order to properly move the servos. This circuit also protects the CPU from the 6V of the 6V regulator, and thus allowing for seamless use between the CPU and the servo driver, the same is true for the other servo driver.

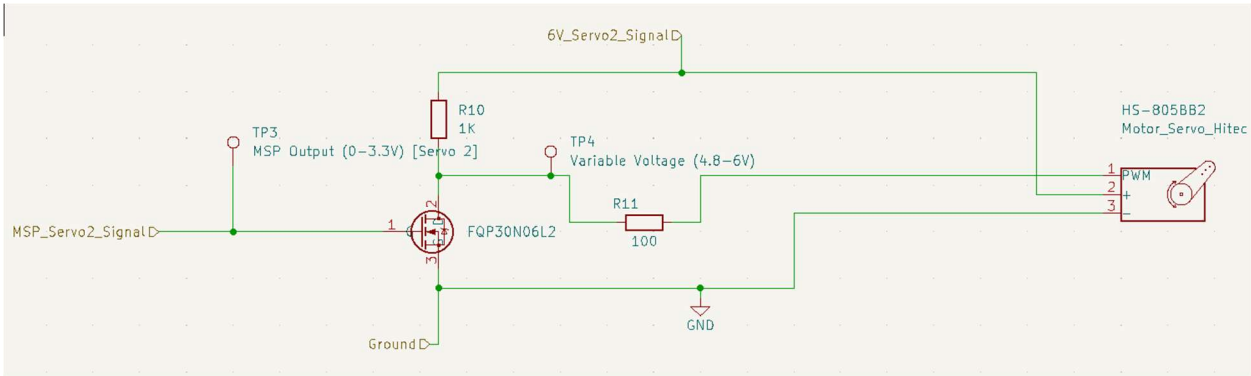

Figure 7: Servo Driver 2 Circuit Schematic

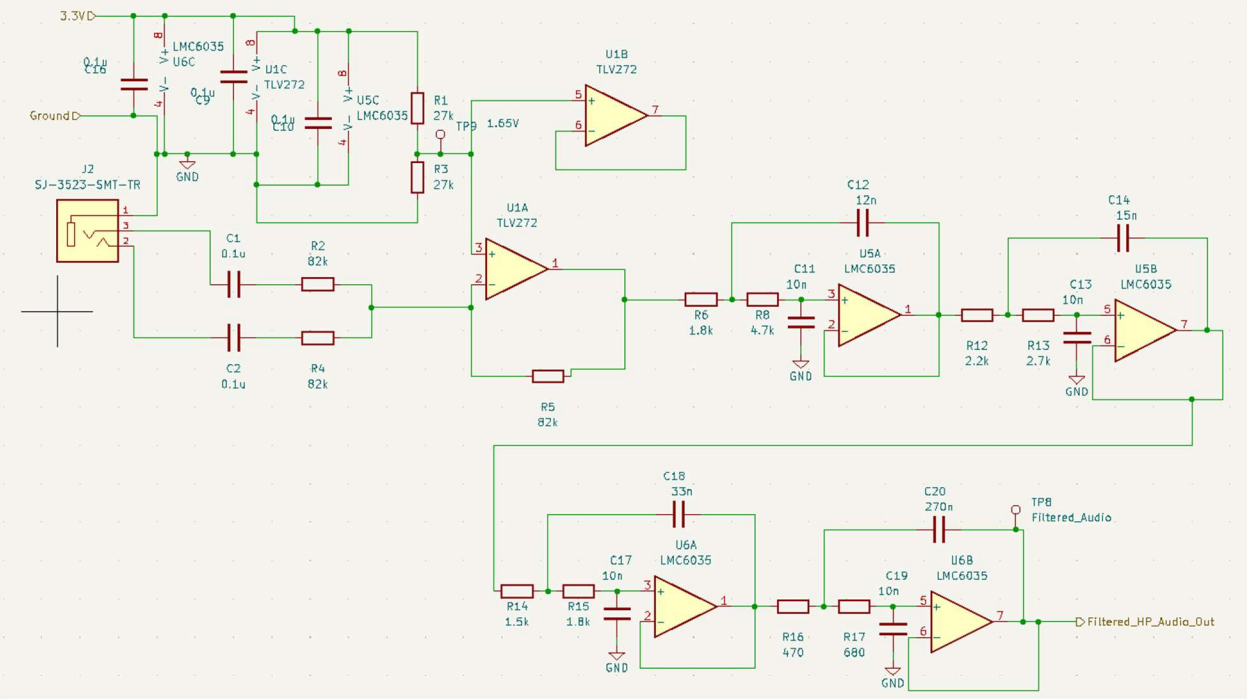

Figure 8: Audio Input Circuit Schematic

The audio input circuit in **Figure 8** works by having an audio signal being inputted into an audio jack and then it goes through a low-pass filter, into a summing amplifier, to then go into an anti-aliasing filter. The anti-aliasing was designed to be an 8th order Butterworth filter with a stopband frequency of 22kHz with -100dB attenuation. From the anti-aliasing filter, the signal goes into the CPU's ADC.

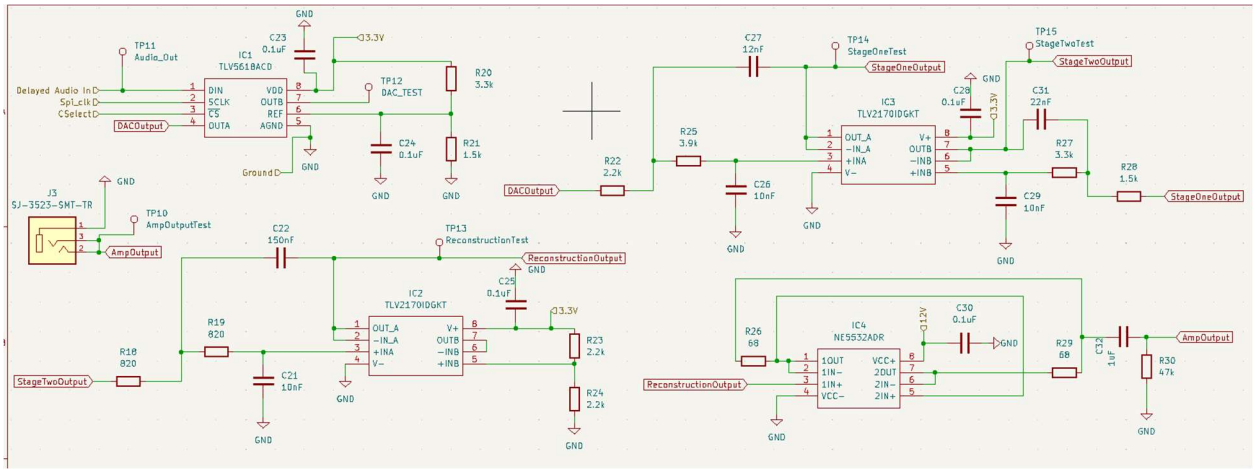

Figure 9: Audio Output Circuit Schematic

The audio output circuit in **Figure 9** takes a digital signal and then runs it through the DAC and then from the DAC the signal goes into another anti-aliasing filter to then go into an audio amplifier, to then go into an audio jack for mono output audio.

Board Layouts

The final board layout is provided below in **Figure 10**.

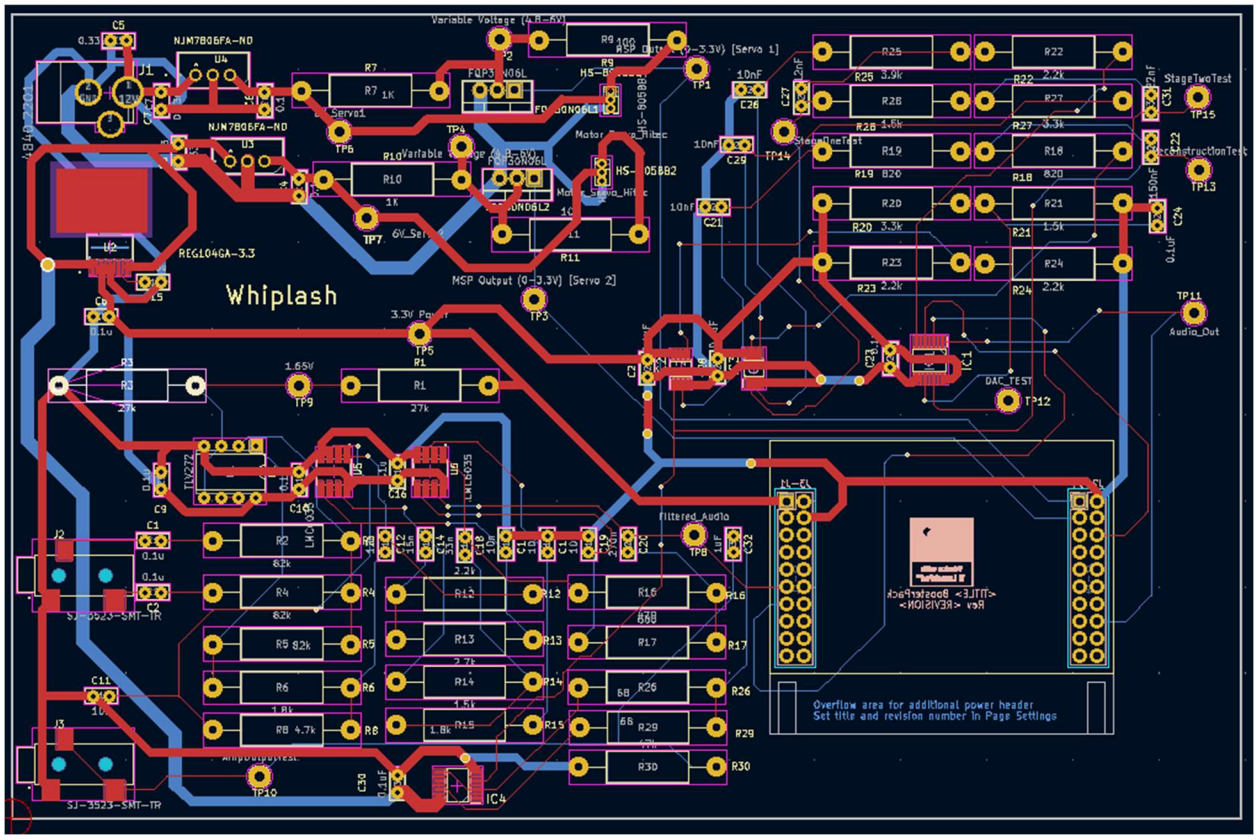

Figure 10: PCB Schematic

First thing that we wanted was to make sure that the different audio jacks and the DC power jack would be on the same side of the board and that it would be accessible to the user to be able to insert the power jack and the audio cables smoothly and at command, to when the user desires. The servo connections are put in the middle since they will be connected by cables that will be feed through the internal raceways that will connect to the board from underneath the stand. The placement of the CPU was chosen so that there would be no overhang from the PCB so that the overall design can be sleek and pleasant to the eye. The first anti-aliasing was placed close to the first audio jack and the second anti-aliasing filter was placed near the DAC, one thing that could have been redesign would be the placement of the audio filter to the end of the anti-aliasing filter coming out of the DAC.

Problems and Design Modifications

Software Design

One of the first issues that we ran into to was that there are not instantly available libraries that make DSP simple in CCS. Thus, we tried to find make code in python that would conduct the DSP necessary and then use the methods and algorithms learned to be able conduct DSP on the C-based CPU. Another problem that at one point we were in C++ for dynamically allocated data vector however we were not able to use that in the Texas Instrument hardware. Therefore, we needed to change the code to C to be able to get beat detection to work on the Texas Instruments hardware, so in the end we did not need to use the dynamically allocated data vectors and thus did not need to use C++. In the end there our beat detection algorithm does not go into the frequency domain as that would have required more time and computing power. There is also the issue that there is latency and noise in the signal when we put in the signal into the code and thus different precautions in the code for adding ranges for values to determine the input voltage with the noise were taken.

Electrical Design

One of the biggest problems that we had to deal with was the fact that our servos were not rated correctly. The datasheet stated that the servos are rated for a max of 800mA; however, when we did test for the current of the servo we saw that the servos drive about 1.2A instead of the rated 800mA. This was a roadblock in our design given that the design was set for a total of about 1A on the servo voltage regulator, but the servo drew 1.2A which caused a lot of heat to be drawn into the circuit. Another roadblock that we ran into that we decided to introduce a reset button towards the end of the project, and we did not have the traces on the PCB to complete the button design, thus we needed to use physical jumpers to properly connect the button to power and to the CPU to toggle. The other thing that we had to design was a heat sink pad for our 3.3V voltage regulator and connect it to ground. Given the design of the regulator was not a straightforward design to just add a heatsink via bolt on, we had to create a solder pad to create a heat sink for the regulator. There was also no connection on the device for the heat sink to be grounded on the PCB thus I needed to connect a wire to properly connect the heat sink to ground. Another issue is that there was a lot of noise in the 3.3V regular system and thus the team had to add another bypass capacitor to reduce the noise that was going into the regulator. Lastly more heat sinks were used than what was calculated for the servo regulators this was since the servos drew more current that what the datasheet said it would draw, therefore we needed to add more heatsinks to keep the regulators from overheating too much.

Physical Model

The model started with a rotary shaft and clip to hold the drumsticks when attaching the servo mechanism. Since we found out we could mount the drumsticks directly to the servos we no longer had use for the shaft or shaft clip. The physical base also underwent several iterations to improve both function and form. We were originally going to support the drumsticks with an additional peg that would slide through the center but decided that was not required for robustness. There were problems with several designs where the servos would fit too tightly, or the drumsticks were too close together and would not turn. When mounting the hardware, we also had to purchase many types of screws before finding the ones that were flat enough to fit in the project. The design was changed to contain the servos, the wiring, the PCB, and the MSP432 all in one unit. We printed standoffs to mount the PCB in place in the container and screwed machine screws through the bottom of the container, into the standoffs, and through the PCB. We lastly included a “stop-bar” to have the drumsticks always return to the same position and prevent the servos from overshooting beyond the original starting point. This allows us to provide a quicker and more reliable movement of the drumsticks while saving some power from the servos. One more thing that was needed, we had a small little fan on the outside of our project to help with the overall cooling of the system. Given that the heatsinks that were chosen were not enough to cool it down sufficiently in free air.

Project Timeline

The initial and final Gantt charts can be seen below in **Figure 11** and **Figure 12**, respectively.

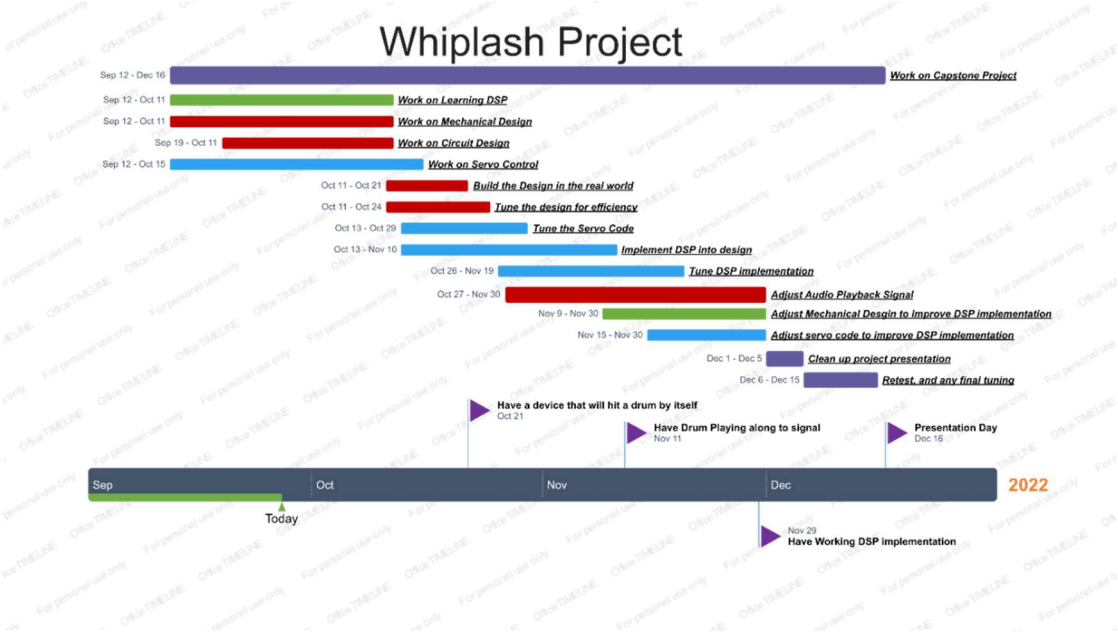

Figure 11: Initial Gantt Chart

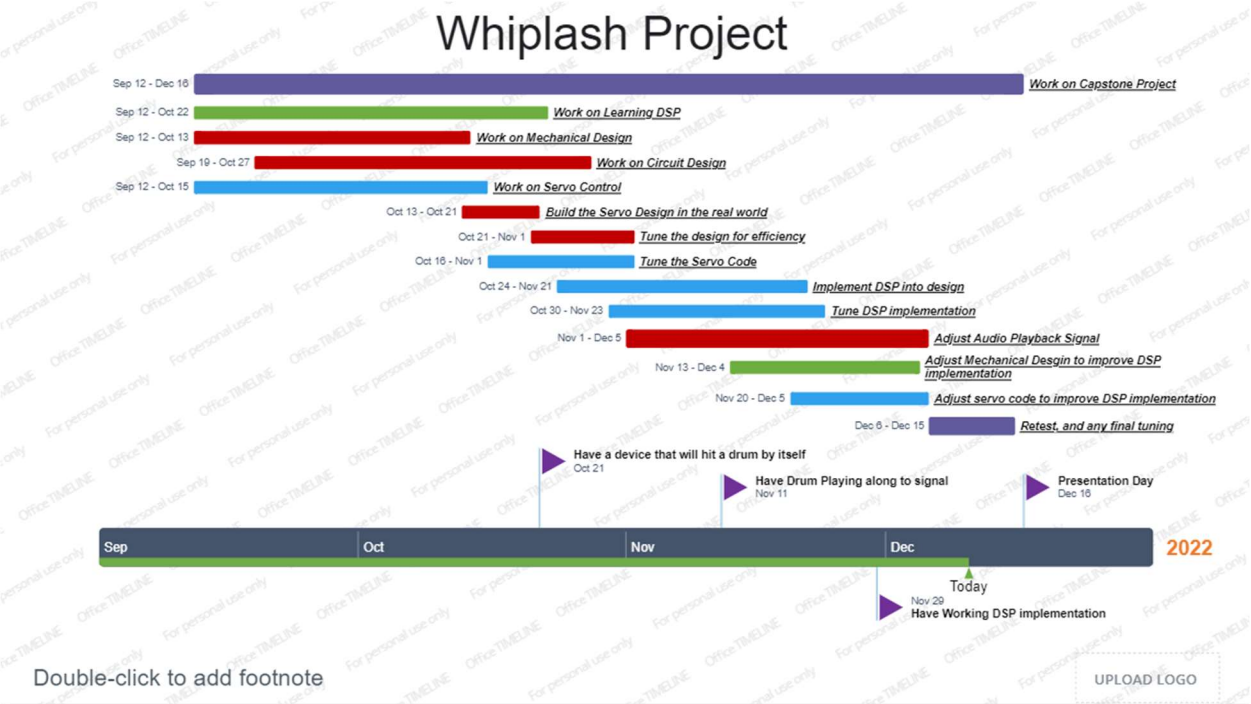

Figure 12: Final Gantt Chart

The most recent Gantt chart is shown above. Compared to the initial Gannt chart there were no major additions. The main differences from the initial Gantt chart to the most recent Gantt chart is the fact that more of the tasks were extended as some of the design portions were taking longer than what we were expecting during the initial markup of the Gantt chart.

Tasks that could have been done in parallel were done in parallel, starting from the beginning, the learning of the DSP and how to possibly implement it into our central processing unit CPU were done in parallel. Also, while those steps were taking place, the mechanical design for how the servo motors will interact with the drumsticks to move the drumsticks was done. All the while, the printed circuit board PCB was designed. The schematic that was used to represent the different subsections of the circuit was designed as well. All the previously mentioned tasks were done before the PCB was sent out to be manufactured as this allowed the team to stay on task. The code was tested on a CPU unit before the PCB came back to ensure that the beat detection and the PWM were working accordingly.

Once the initial PCB design came back then the implementation of all the physical components was implemented with the coded sections of work. During this step, the interactions between the PWM from the CPU were gauged to determine how to properly serve the servo to get the most accurate drumstick hit. This will in turn also test the power that is being distributed to the CPU and the servo motors themselves. This step also saw the testing and reconfiguration of the audio signal circuits. While the input audio signal does not require reconfiguration the output audio signal circuit will need to be reconfigured. This is due to the error and delay between when the code will be able to successfully determine a beat from any given input and when that signal will be sent over to the servo motor. Thus, if any major redesigns needed to be done on the PCB, then the team will be able to identify the problems and redesign the hardware as needed to support all the different sub circuits on the PCB.

Each team member had their own tasks that needed to be completed. Leonardo's primary tasks included overseeing the design of the mechanical system that connected both the servo and the drumstick to ensure that the servo will be able to hit the drum in a quick and safe manner as to not damage the servo. His secondary tasks were to help with the design of the servo part of the circuit, since he will be focused on the real implementation of the servo, he will be communicating what signals and power signals, servo drivers, should be delivered to the servo to ensure successful servo operation. John's primary task was working on the servo driver as well as selecting the servo itself. Once the servo driver and servos are determined then the information should be relayed to the person in charge of the power distribution to incorporate the ratings for the drivers and the servos into the power design. Uriel oversaw designing the main layout of the PCB. Alongside laying out the PCB, he designed the power distribution circuit for the entire PCB supplying accurate voltage to the different components on the board. He also oversaw designing the audio input circuit into the main CPU as well as designing the output audio circuit.

Davis oversaw learning the DSP required to implement the beat detection application onto our CPU. This learning required the use of existing libraries to be able to accurately detect the beat to any given input signal. This detection code was able to provide a signal that will have an

amplitude associated with it to determine how hard the drumstick will need to hit, to determine which beat gets accented. Max oversaw implementing the PWM system into the CPU, to ensure that the CPU will be outputting the correct PWM signal depending on the beat detection system that will be received by the beat detection that will happen after the input audio signal.

Test Plan

The test plan started with making sure that each individual block on the schematic was working. In relation to when we got the PCB, we made sure that all the connections were tested correctly and that we were not missing connections nor that connections were improperly made, ensuring that the only connections present on the board are connections that were on the team's submitted schematic. The input audio circuit would be tested by making sure that when the circuit is being 3.3V, that the audio filter was being attenuated before the initial 20Hz corner frequency of the initial lowpass filter. Then to test that there was about 100dB attenuation to the ADC of the CPU. That then concluded the testing of the audio input circuit. Then the testing of the servo drivers was done to sign the initial sending of the PWM signal is correct as well. The circuit was testing with the 12V jack being powered by the AC to DC power adapter and then connecting them to the 6V regulator to ensure that the servos were receiving the 6V that are necessary to correctly be powered by the 12V adapter alongside the 6V regulators. This test was conducted again to ensure that the PWM signals were being amplified by the MOSFET. Then the DAC and audio output circuit was tested. The test was done by sending a digital signal from the CPU to the DAC and then plugging in the output to an audio jack and ensuring that there was sound coming from the DAC, and that the signal from the DAC sounded like the song that was being inputted into the audio input system. This was done by powering the DAC on the 3.3V supply on the Virtual Bench. Once the DAC provided an output we would then connect everything together and then power the entire system to then test the power system. Testing that all the voltage regulators were tested to be supplying the correct voltage and that all of the different components were receiving the correct voltage.

On the software side of testing, we tested the algorithm in different languages just to make sure that the algorithm process is correct and that the only things that were being coded into the CPU were necessary and that no other code was needed to save memory on the CPU. Once that was tested we put that code into the CPU and then the audio input was inputting audio in the CPU. To make sure that the beat detection code was working we gave it a signal when no music was being played to make sure that the device knew when music was not being played so that the servos were not playing. Then once that was tested we then put in a song and measured what the ADC was reading to ensure that the ADC was reading a different signal to audio not being played. Once we were able to verify that the ADC was working, we sent the signal from the ACD directly to the DAC to ensure that DAC would be able to output audio. Then we wanted to test this on while the CPU was connected to the PCB. This broke our CPU, thus we needed to change our testing methods as we no longer had a DAC and will no longer output audio. We still implement the DSP code by reading the ADC when it was given a signal and making sure that the algorithm was working by seeing what intervals, notation that we used to name the beat, were

being output by the different songs. This was done by sending a metronome signal to ensure that the interval determined by a song with 120BPM (beats per minute) was like the interval determined by the metronome at 120BPM. From here the interval was then translated into a PWM signal that would then be sent into the servos. With the audio off, the timing of the servos hitting the drum was timed to make sure that the servos were hitting at a constant beat. This test was done at the end to verify the model is working.

Final Results

The team produced a device that could receive an identifiable piece of music through an auxiliary jack, parsing out the beat of the music, and reproducing the beat at either full-time or half-time tempo. However, depending on the song that was input into the device, at times the beat that the device would play was difficult to match up to what the song was outputting due to latency issues and the varying amount of time it took for the device to get enough of a sample to start playing along, despite being correct. Additionally, there was some occasional unpredictability of the behavior of the servo motors since they did not meet the specifications from the accompanying datasheet for which we had designed. All things equal, this rarely detracted from the presentation and did not prohibit us from accomplishing what we had intended as well as a few extra additions such as a stop and start button and a from scratch DAC (Digital to Analog Converter) solution to output the music. The key objectives that we had set for our device at the time of our proposal are as follows:

- Using DSP, be able to filter out a beat from a song
- Ability to hit a drum at varying rhythms
- Ability to hit a drum at varying pressures
- Ability to interpret and perform output from DSP filtering with different beat types
- Ability to provide a satisfying and intentional sound
- Consists of a clean and intentional presentation

Firstly, our design was demonstrated to be able to receive a requested song, perform digital signal processing on the song, and output the beat of the inputted song at either full-time or half-time tempo. We can confidently say that this meets the criteria of our first two proposed objectives. Unfortunately, in our final design we observed that our servos were able to hit with varying pressures, however this was not by intentional programming but rather a side-effect of how the hardware was behaving with the software. Consequently, we cannot take credit for the third objective as it was not realized by intention. In terms of different music types with different beats, our design could recreate an audible, clear, and satisfying beat regardless of whether the inputted music was jazz, pop, rock, hip-hop, or even electronic music. Considering this, the criteria for the fourth and fifth proposed objectives were met. Lastly, the design of our project consisted of a small footprint and effectively highlighted intentional design choices such as an all-enclosed PCB and power circuitry solution, a metronome inspired 3D design, and a noticeably small number of external wires that all contributed to a satisfying and professional

look and feel to the presentation of our device. Overall, according to the proposed grading scheme in **Table 1**, the letter grade we earned with the result of our device is an A.

Table 1: Proposed Grading Scheme

Grade Received	Number of Objectives Accomplished
A+	6
A	5
B	4
C	3
D	2 or less

The final project can be seen below in **Figure 13**.

Figure 13: Whiplash Final Design

Costs

The cost to produce our design prototype was notably lower than what the actual manufacturing of a single unit of Whiplash would be due to our group possessing certain components and production techniques on hand that would have to be accounted for when calculating production cost. Included in **Table 2** below is a breakdown of the associated costs of producing Whiplash both in single unit and mass-scale quantities of 10,000 units or more.

Table 2: Cost Breakdown

Component	Single Unit Cost	Mass Production Unit Cost
CC3220SF-LAUNCHXL Microcontroller	$59.99	$59.99
Resistors and Capacitors	$24.48	$7.45
IC Chips	$23.11	$13.71
Connectors, Headers and Heatsinks	$17.20	$7.85
PCB Manufacturing	$33.00	$33.00
Structural Supplies	$48.59	$48.59
Servo Motors	$85.90	$85.90
Power Delivery Components	$31.41	$23.81
Total	**$323.68**	**$280.30**

The total cost of producing our prototype was $402.37. However, this does not include the cost of a microcontroller, drumsticks, or the cost of assembly. Additionally, the price could have been even lower if we had not ordered more components than was necessary for the design. Mass production would give a hardware discount of around 14% but would raise the overall cost per unit due to considerations for injection molding, infrastructure, and logistics. A detailed breakdown of all the costs associated with this project can be found in the Appendix.

Future Work

Whiplash is a bridge technology that may eventually expand to devices with onboard AI, capable of generating their own algorithmic compositions. The device currently is nowhere near capable of fully replacing a human musician; however, if the project were to be expanded and researched further, it may very well be. The device can be reformatted to parse out individual drums in a song or “taught” by machine learning algorithms to generate its own drum track. This would allow the device to interface with a full-scale drum set and create its own music as a human would.

There were plenty of difficulties that we did not foresee in the project. Our MSP had an ADC but not a DAC, so we had to design our PCB with our own DAC chip. We also had to create an aliasing filter since we were sampling audio directly from the source. Understanding what must go on the PCB earlier would allow building and testing to be much simpler. For the physical design, we had to measure the exact dimensions of all objects and download 3D model files from online. Wrong measurements lead to multiple reprints of the body, so in the future it would be better to collaborate and ask the other members what they need on the mechanical end. The DSP element must be written custom to the project. Once the conceptual aspect is understood, the DSP can be written in C quickly.

Advice for pitfalls would be to understand that the device is not going to be perfect. The audio capabilities of the MSP will not be producing extremely high-quality audio; however, it will be enough for a proof-of-concept device. It would also help to look for quicker servos or possibly design a gear system so that the drumsticks can move quicker and hit the drum with more strength. Overall, the PCB could be remade to be slightly nicer, the code could be more efficient, and the physical design could be more streamlined. If done again, each task would only take a fraction of the time.

References

[1] Jean-Pierre Charras, “KiCAD.” 1992. [Windows, macOS, Linux]. Available: https://www.kicad.org/

[2] Jupyter, “Google Colab.” Google, 2015. [Windows, macOS, Linux]. Available: https://colab.research.google.com/

[3] Librosa, “Librosa.” [Online]. Available: https://librosa.org/

[4] GNU Projects, “Online GDB.” 1982. [Windows, macOS, Linux]. Available: https://www.onlinegdb.com/

[5] Texas Instruments, “SimpleLink™ Wi-Fi® CC3220SF wireless microcontroller LaunchPad™ development kit,” *Texas Instruments*, Dec. 02, 2018. https://www.ti.com/tool/CC3220SF-LAUNCHXL (accessed Oct. 22, 222AD).

[6] Texas Instruments, “Texas Instruments Drivers.” [Online]. Available: https://www.ti.com/tool/download/SIMPLELINK-CC32XX-SDK/6.10.00.05

[7] F. Barnes, “Stickboy - Robocross Machines,” *The Drummer*, 2007. https://robocross-machines.com/robots/stickboy/ (accessed Oct. 21, 2022).

[8] Polyend, “Polyend Perc,” *Polyend*, 2015. https://polyend.com/legacy/polyend-perc/ (accessed Oct. 21, 2022).

[9] T. Fryen, M. Eppe, P. D. H. Nguyen, T. Gerkmann, and S. Wermter, “Reinforcement Learning with Time-dependent Goals for Robotic Musicians.” arXiv, Nov. 11, 2020. Accessed: Oct. 21, 2022. [Online]. Available: http://arxiv.org/abs/2011.05715

[10] G. van Rossum, “Python v3.” Python Software Foundation, Scotts Valley, CA, Feb. 20, 1991. [Windows, macOS, Linux]. Available: https://www.python.org/

[11] D. Ritchie, “C.” 1972. [Windows, macOS, Linux]. Available: https://www.open-std.org/jtc1/sc22/wg14/

[12] “MSP430G2553 | MSP430G2x/i2x | MSP430 ultra-low-power MCUs | Description & parametrics.” http://www.ti.com/product/MSP430G2553 (accessed Dec. 06, 2016).

[13] Texas Instruments, “Code Composer Studio.” Texas Instruments, 1999. [Windows, macOS, Linux]. Available: https://www.ti.com/tool/CCSTUDIO

[14] Texas Instruments, “Filter Design Tool.” Texas Instruments. [Online]. Available: https://webench.ti.com/filter-design-tool/filter-type

[15] J. Riegel, “FreeCAD.” FreeCAD, Oct. 29, 2002. Accessed: Oct. 22, 2022. [Windows, Linux, macOS]. Available: https://www.freecadweb.org/

[16] Josef Prusa, “PrusaSlicer.”

[17] Bjarne Stroustrup, “C++.” ISO/IEC JTC 1, 1985. [Online]. Available: https://cplusplus.com/

[18] National Instruments, “Multisim.” National Instruments, 1999. [Windows]. Available: https://www.multisim.com/

[19] Microsoft, “Visual Studio Code.” Microsoft, 2015. [Windows, macOS, Linux]. Available: https://code.visualstudio.com/

[20] Advanced Circuits, “FreeDFM.” [Online]. Available: https://www.my4pcb.com/net35/FreeDFMNet/FreeDFMHome.aspx#_ga=2.105431171.1623434166.1664073003-1456432603.1663029310&_gac=1.18299211.1664073003.Cj0KCQjw1bqZBhDXARIsANTjCPKnYbgP7CLX7S6DRnRHge15B9v8y_VUFzcdwjFg-_FUwAJyExKIs90aAqTcEALw_wcB

[21] Josef Prusa, “Prusa MK3S+.” 2019. [Online]. Available: https://www.prusa3d.com/category/original-prusa-i3-mk3s/
[22] Linus Torvalds, “GitHub.” San Francisco, CA, 2005. [Online]. Available: https://github.com/
[23] European Commission, “Restriction of Hazardous Substances in Electrical and Electronic Equipment (RoHS).” 2015. [Online]. Available: https://environment.ec.europa.eu/topics/waste-and-recycling/rohs-directive_en
[24] M. Gerić, G. Gajski, V. Oreščanin, A.-M. Domijan, R. Kollar, and V. Garaj-Vrhovac, “Environmental risk assessment of wastewaters from printed circuit board production: A multibiomarker approach using human cells,” *Chemosphere*, vol. 168, pp. 1075–1081, Feb. 2017, doi: 10.1016/j.chemosphere.2016.10.101.
[25] I. D’Adamo, F. Ferella, M. Gastaldi, F. Maggiore, P. Rosa, and S. Terzi, “Towards sustainable recycling processes: Wasted printed circuit boards as a source of economic opportunities,” *Resour. Conserv. Recycl.*, vol. 149, pp. 455–467, Oct. 2019, doi: 10.1016/j.resconrec.2019.06.012.
[26] F. G. Üçtuğ and A. Azapagic, “Environmental impacts of small-scale hybrid energy systems: Coupling solar photovoltaics and lithium-ion batteries,” *Sci. Total Environ.*, vol. 643, pp. 1579–1589, Dec. 2018, doi: 10.1016/j.scitotenv.2018.06.290.
[27] DaftLogic, “List of the Power Consumption of Typical Household Appliances,” *Appliance Power Consumption*. https://www.daftlogic.com/information-appliance-power-consumption.htm
[28] BioPak Team, “What is PLA?,” 2019. https://www.biopak.com/au/resources/what-is-pla
[29] Tom Bardwell, “How Much Electricity Does A 3D Printer Use?,” *3D Sourced*, 2022. https://www.3dsourced.com/guides/how-much-electricity-does-a-3d-printer-use/
[30] ACDi, “Recycling Printed Circuit Boards,” 2017. https://www.acdi.com/recycling-printed-circuit-boards/
[31] Cadence PCB Solutions, “Eco-Friendly Printed Circuit Boards: Present and Future Manufacturability,” 2019. https://resources.pcb.cadence.com/blog/2020-eco-friendly-printed-circuit-boards-present-and-future-manufacturability
[32] National Fire Protection Association, “Codes & Standards.” Dec. 08, 2021. [Online]. Available: https://www.nfpa.org/codes-and-standards/all-codes-and-standards/list-of-codes-and-standards/detail?code=70
[33] National Fire Protection Association, “NFPA 70,” 2022.
[34] X. Street, “IPC China — Suzhou Office,” p. 28.
[35] Harold Lee and Roger B. Dannenberg, “Music Instruction System” [Online]. Available: https://patents.google.com/patent/US9333418B2/en?q=music+playing+drums&status=GRANT&language=ENGLISH&type=PATENT
[36] Jonas Braasch, Nikhil Deshpande, Pauline Oliveros, and Selmer Bringsjord, “Interactive, expressive music accompaniment system,” US10032443B2 [Online]. Available: https://patents.google.com/patent/US10032443B2/en?q=music+playing+drums&status=GRANT&language=ENGLISH&type=PATENT&page=3
[37] Michael Bryan Burns, Jr., “System and method for automatic drumming,” US11404034B1 [Online]. Available: https://patents.google.com/patent/US11404034B1/en?q=drum+player&q=G10F1%2f08&page=2

[38] *Whiplash*, (2014). Accessed: Oct. 21, 2022. [Online Video]. Available: https://letterboxd.com/film/whiplash-2014/genres/
[39] HiTEC, "HS-805BB Mega Giant Scale Servo," *HiTEC*, Feb. 24, 2007. https://hitecrcd.com/products/servos/giant-servos/analog-giant-servos/hs-805bb/product (accessed Oct. 22, 2022).
[40] Memory Protection Devices, "EJ508A." [Online]. Available: https://www.digikey.com/en/products/detail/mpd-memory-protection-devices/EJ508A/2439547
[41] Nisshinbo Micro Devices Inc., "NJM7806FA." [Online]. Available: https://www.digikey.com/en/products/detail/nisshinbo-micro-devices-inc/NJM7806FA/805764
[42] Texas Instruments, "REG104GA-3.3." [Online]. Available: https://www.ti.com/product/REG104/part-details/REG104GA-3.3
[43] onsemi, "FQP30N06L." [Online]. Available: https://www.digikey.com/en/products/detail/onsemi/FQP30N06L/1055122
[44] Texas Instruments, "TLV272IP." [Online]. Available: https://www.digikey.com/en/products/detail/texas-instruments/TLV272IP/454247?s=N4IgTCBcDaICoBkBqYDsYCSAFEBdAvkA
[45] Texas Instruments, "LMC6035IM-ND." [Online]. Available: https://www.digikey.com/en/products/detail/texas-instruments/LMC6035IM/3695358
[46] Texas Instruments, "TLV5618ACD." [Online]. Available: https://www.digikey.com/en/products/detail/texas-instruments/TLV5618ACD/306176
[47] Texas Instruments, "TLV2170IDGKT." [Online]. Available: https://www.digikey.com/en/products/detail/texas-instruments/TLV2170IDGKT/6615389?s=N4IgTCBcDaICoBkBqYCMB2ADASQCIHEBpOEAXQF8g
[48] Texas Instruments, "NE5532ADR." [Online]. Available: https://www.digikey.com/en/products/detail/texas-instruments/NE5532ADR/499497

Appendix

Table 3: Total Cost Breakdown

Order #	Index	Manufacturer Part #	Item Description	Digikey Part #	Mouser Part #	Sparkfun Part #	Qty in Stock	Qty Ordered	Qty Used	Per Unit Price	Cost	No Extras Cost	Team Name		Per 10,000 Unit Price	10,000 Unit Cost
1	1	HS-80BB	6V Servo Motor			ROB-11881	29	2	2	$42.95	$85.90	$85.90	Whiplash		$42.95	$85.90
	2	RFP30N06LE	30A, 60V N-Channel Mosfet			COM-10213	>35	2	2	$1.25	$2.50	$2.50	Whiplash		$1.13	$2.26
2	3	TLV561BACD	12 Bit, 2.7-5.5V DAC IC		595-TLV561BACD		71	1	1	$14.83	$14.83	$14.83	Whiplash		$14.83	$14.83
3	4	5006	PC Test Point Compact Black	36-5006-ND			307,790	20	16	$0.42	$8.40	$6.72	Whiplash		$0.17	$2.73
	5	87914-2016	CONN HEADER VERT 20POS 2.54MM	WM20570-ND			1,155	4	2	$1.44	$5.76	$2.88	Whiplash		$0.67	$1.33
	6	22284035	CONN HEADER VERT 3POS 2.54MM	23-0022284035-ND			16,186	6	2	$0.27	$1.62	$0.54	Whiplash		$0.10	$0.21
	7	VEL36US120-US-JA	AC/DC WALL MOUNT ADAPTER 12V 36W	1470-3644-ND			2,523	1	1	$18.91	$18.91	$18.91	Whiplash		$15.89	$15.89
	8	NJM7806FA	IC REG LINEAR 6V 1.5A TO220F	2129-NJM7806FA-ND			2,661	3	2	$0.88	$2.64	$1.76	Whiplash		$0.38	$0.76
	9	REG104GA-3.3	IC REG LINEAR 3.3V 1A SOT223-6	REG104GA-3.3-ND			770	3	3	$8.24	$24.72	$24.72	Whiplash		$4.90	$14.71
	10	EJ508A	CONN PWR JACK 2.1X5.5MM SOLDER	EJ508A-ND			17,451	1	1	$1.38	$1.38	$1.38	Whiplash		$0.69	$0.69
	11	SJ-3523-SMT-TR	CONN JACK STEREO 3.5MM SMD R/A	CP-3523SJCT-ND			92,116	3	3	$1.01	$3.03	$3.03	Whiplash		$0.46	$1.39
	12	LMC6035IM/NOPB	IC CMOS 2 CIRCUIT 8SOIC	LMC6035IM/NOPB-ND			3450	4	2	$2.21	$8.84	$4.42	Whiplash		$1.06	$2.12
	13	C333C334K5R5TA	CAP CER 0.33UF 5% 50V X7R RADIAL	399-13993-ND			1617	10	4	$0.55	$5.50	$2.20	Whiplash		$0.14	$0.57
	14	C322C104K5R5TA	CAP CER 0.1UF 50V X7R RADIAL	399-4329-ND			110713	15	13	$0.22	$3.30	$2.86	Whiplash		$0.05	$0.61
	15	C320C101J5R5TA	CAP CER 100DPF 50V X7R RADIAL	399-9771-ND			6867	5	3	$0.40	$2.00	$1.20	Whiplash		$0.10	$0.29
	16	RCE5C1H123J1K1H03B	CAP CER 0.012UF 50V NPO RADIAL	490-14498-ND			375	4	1	$0.73	$2.92	$0.73	Whiplash		$0.21	$0.21
	17	C315C153J1H5TA9170	CAP CER 0.015UF 100V X8R RADIAL	399-17275-ND			1150	4	1	$0.64	$2.56	$0.64	Whiplash		$0.17	$0.17
	18	FA14C0G1H333JNU00	CAP CER 0.033UF 50V C0G RADIAL	445-181083-ND			2028	4	1	$0.65	$2.60	$0.65	Whiplash		$0.19	$0.19
	19	C322C274K5R5TA	CAP CER 0.27UF 50V X7R RADIAL	399-9806-ND			1142	4	1	$0.70	$2.80	$0.70	Whiplash		$0.18	$0.18
	20	A 08-LC-TT	CONN IC DIP SOCKET 8POS TIN	AE9986-ND			77557	8	1	$0.19	$1.52	$0.19	Whiplash		$0.08	$0.08
4	21	507302B00000G	HEATSINK TO-220 2.5W LOW PROFILE	HS115-ND			27,632	6	4	$0.27	$1.62	$1.08	Whiplash		$0.18	$0.73
	22	OF332KE	RES 3.3K OHM 10% 1/2W AXIAL	273-OF332KE-ND			1,000	4	2	$0.95	$3.80	$1.90	Whiplash		$0.35	$0.70
	23	OF152KE	RES 1.5K OHM 10% 1/2W AXIAL	273-OF152KE-ND			1,000	4	2	$0.95	$3.80	$1.90	Whiplash		$0.35	$0.70
	24	OF222KE	RES 2.2K OHM 10% 1/2W AXIAL	273-OF222KE-ND			996	6	3	$0.95	$5.70	$2.85	Whiplash		$0.35	$1.04
	25	OX392KE	RES 3.9K OHM 10% 1W AXIAL	OX392KE-ND			445	3	1	$2.10	$6.30	$2.10	Whiplash		$0.85	$0.85
	26	OF821KE	RES 820 OHM 10% 1/2W AXIAL	273-OF821KE-ND			789	4	2	$0.95	$3.80	$1.90	Whiplash		$0.35	$0.70
	27	PR02F5020680 9KR500	RES 68 OHM 10% 2W AXIAL	BC3364CT-ND			35,649	4	2	$0.60	$2.40	$1.20	Whiplash		$0.09	$0.18
	28	OF473KE	RES 47K OHM 10% 1/2W AXIAL	273-OF473KE-ND			961	2	1	$0.95	$1.90	$0.95	Whiplash		$0.35	$0.35
	29	C322C104K5R5TA7301	CAP CER 0.1UF 50V X7R RADIAL	399-9877-1-ND			188,445	10	3	$0.15	$1.52	$0.46	Whiplash		$0.05	$0.14
	30	C330C105K5R5TA7301	CAP CER 1UF 50V X7R RADIAL	399-9886-1-ND			195,866	3	1	$0.48	$1.44	$0.48	Whiplash		$0.12	$0.12
	31	C322C123K5R5TA	CAP CER 0.012UF 50V X7R RADIAL	1,809			1,809	3	1	$0.33	$0.99	$0.33	Whiplash		$0.08	$0.08
	32	C315C103K5R5TA	CAP CER 10000PF 50V X7R RADIAL	399-4148-ND			186,292	6	2	$0.23	$1.38	$0.46	Whiplash		$0.05	$0.10
	33	C320C154K5R5TA	CAP CER 0.15UF 50V X7R RADIAL	399-4276-ND			13,232	2	1	$0.45	$0.90	$0.45	Whiplash		$0.12	$0.12
	34	C315C223K5R5TA	CAP CER 0.022UF 50V X7R RADIAL	399-4169-ND			23,615	2	1	$0.33	$0.66	$0.33	Whiplash		$0.07	$0.07
	35	TLV5618ACD	IC DAC 12BIT V-OUT 8SOIC	296-2303-5-ND			466	2	1	$14.84	$29.68	$14.84	Whiplash		$10.00	$10.00
	36	TLV2170IDGKT	IC OPAMP GP 2 CIRCUIT 8VSSOP	296-43257-1-ND			22,435	2	2	$1.60	$3.20	$3.20	Whiplash		$0.69	$1.37
	37	NE5532ADR	IC OPAMP GP 2 CIRCUIT 8SOIC	296-13621-1-ND			55,153	2	1	$0.65	$1.30	$0.65	Whiplash		$0.22	$0.22
	38	SJ-3523-SMT-TR	CONN JACK STEREO 3.5MM SMD R/A	CP-3523SJCT-ND			84,142	2	0	$1.01	$2.02	$0.00	Whiplash		$0.46	$0.00
	39	REG104GA-3.3	IC REG LINEAR 3.3V 1A SOT223-6	REG104GA-3.3-ND			652	3	3	$8.24	$24.72	$24.72	Whiplash		$4.90	$14.71
	40	V5274A-T	HEATSINK ALUM ANOD	AE10811-ND			1,876	5	0	$0.67	$3.35	$0.00	Whiplash		$0.45	$0.00
	41	C320C103J5R5TA	CAP CER 10000PF 50V X7R RADIAL	399-9771-ND			6,852	10	3	$0.28	$2.80	$0.84	Whiplash		$0.10	$0.29
	42	NJM7806FA	IC REG LINEAR 6V 1.5A TO220F	2129-NJM7806FA-ND			4,108	3	0	$0.88	$2.64	$0.00	Whiplash		$0.38	$0.00
	43	EJ508A	CONN PWR JACK 2.1X5.5MM SOLDER	EJ508A-ND			19,391	1	1	$1.38	$1.38	$1.38	Whiplash		$0.69	$0.69
5	44		PCB Manufacturing - WWW					2	2	$33.00	$66.00	$66.00	Whiplash		$33.00	$66.00
6	45		Black 3D Printer Filament and metal screws - Amazon					1	1	$27.34	$27.34	$27.34	Whiplash		$27.34	$27.34
	46		Drum + drumsticks - Amazon						1	$21.25	$0.00	$21.25				
	47	CC3220SF-LAUNCHXL	LAUNCHPAD DEV BOARD FOR CC3220SF	296-45388-ND			33		1	$59.99	$0.00	$59.99			$59.99	$59.99
							Total Cost Prototype:	$402.37		**No Extras Total C.P.U:**	$413.36		**Mass Production Total C.P.U:**		$331.55	

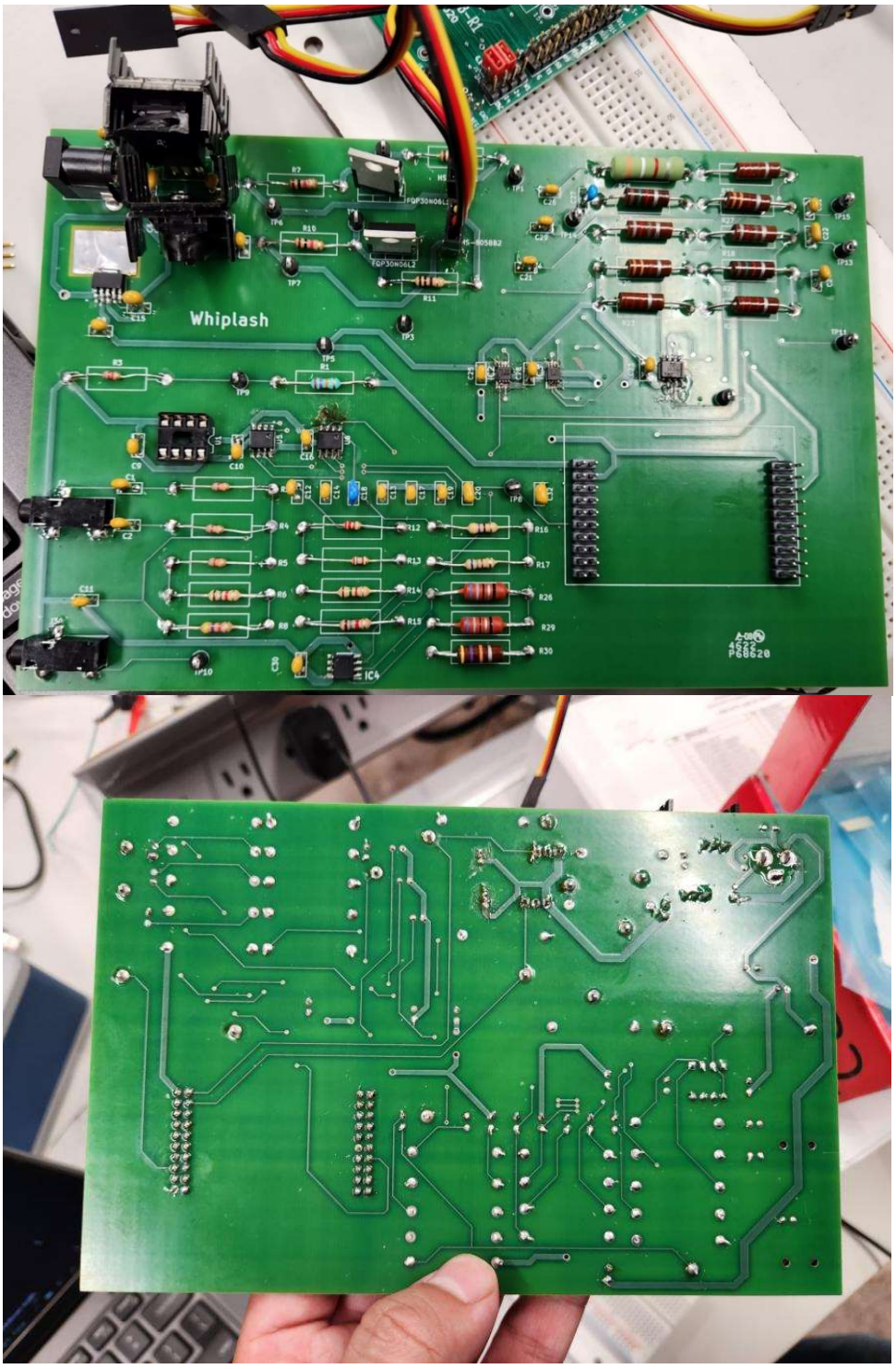

Figure 14: Front and Back of Physical PCB