OmniDrum
UCF Senior Design 1 - Divide & Conquer

Group 41 Members
Wylde Simpson - Electrical Engineering
Soufiane Louiba - Electrical Engineering
Jameson Palmer - Computer Engineering

Review Committee
Dr. Borowczak - ECE
Person 2 - Info
Person 3 - Info

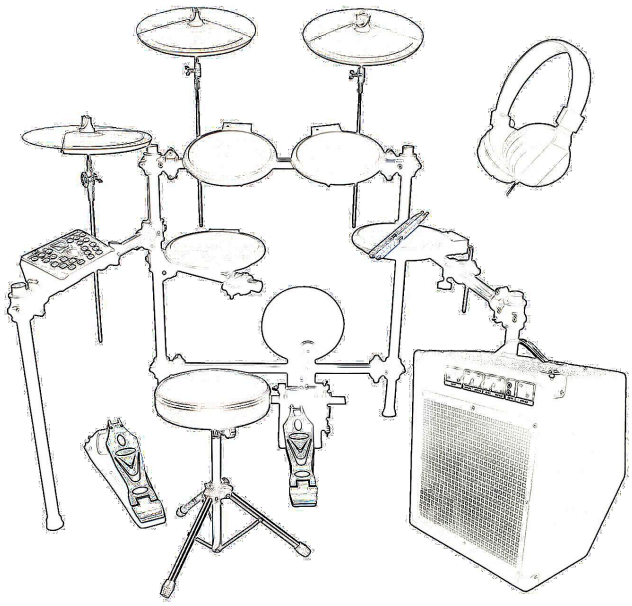


Figure 1.1 *Project Mockup*
by Jameson Palmer

1. Project Narrative

1.1 Project Background & Context

Electric drum kits have been commercially available since the 80's and are chosen by many musicians for a number of reasons. Electric drum kits are composed of various pads that imitate conventional drums in shape and size. These pads are sensitive to being struck with a drumstick and will send a signal to a central unit that plays a drum sound according to which component is being struck. These pads can also be sensitive to the velocity of the strike and change the dynamics of the sound to match the playing style of the user.

Musicians choose electric drum kits for a variety of reasons. Electric drum kits are much quieter than conventional drums which is useful during practice in settings such as apartment buildings or other locations where creating loud sounds may not be acceptable. Another reason someone might choose an electric drum kit is because of their versatility. The sounds produced by the kit can be changed and usually electric drum kits will come with a variety of sounds built in which allows the user to experience the sounds of many different types of drums without having to own each one individually.

Conventional drum kits can consist of instruments that wear with use or can have serious damage if misused, mishandled, or damaged in transport. These instruments can combine to a significant financial cost. This cost can be an increased liability should anything get damaged. Digitizing the sounds each instrument makes not only allows someone to play these recordings back, they can be preserved through years and decades with no worry of sound degradation unlike their acoustic counterparts. Cost of replacing damaged components with this electric drum kit is reduced by the components used and modularity in the design.

Electric drum kits are also useful for recording and composition purposes. Generally with a conventional drum kit, recording someone playing the drums can be complicated and expensive. It may require buying multiple high quality microphones, mixing equipment, and even acoustically treating the room being used. The costs here quickly add up and may be above the budget of many players. Electric drum kits can be plugged directly into recording equipment without the need for complicated setup which reduces the entry cost to recording drums. Overall there are many reasons that musicians may want to purchase an electric drum kit as opposed to a conventional one.

1.2 Project Motivation

Quality electric drum kits can be very expensive to buy which is why some companies have made budget electric drum kits. These kits - while much lower in price compared to high quality kits - still cost a lot of money to the average young musician and are generally of lesser quality and may have less functionality. Our goal is to make an electric drum kit with a low enough cost that it can be bought

by someone on a restrictive budget, yet still has the functionality required to perform up to amateur standards. Many less experienced musicians are younger and do not necessarily have the money to purchase expensive equipment. Instruments are also cumbersome and heavy. The total weight of all drums, bells, and symbols would be drastically reduced. This project boasts a smaller footprint which would bring greater accessibility to players who might find it difficult to make an area for practicing.

Nowadays as music production software becomes available to an increasing market of people, many younger and less experienced musicians are getting access to the tools required to make their own music. Programs such as Digital Audio Workstations (DAWs) can allow beginner musicians to make professional-sounding music with many of their built in tools and recording functionalities. However even with music production tools becoming cheaper and more widespread, live drum recording tools are still harder for beginners to access due to budget constraints. Many of these people are only able to use sequencers built into their music production software, which generally are not as capable of making complex and interesting rhythms. Our project aims to provide an option that both allows for beginners to develop their skills and knowledge with drums to practice on, as well as providing a tool that can actually be used alongside music productions software to allow for players to incorporate their own drumming into music they want to create.

1.3 Project Functionality

Users of **OmniDrum** will be able to attach a number of sensors to different surfaces of their liking which will correspond to a certain component of the drum set. These sensors will detect vibrations and send a signal to a central hub which will include a power supply board, microcontroller for processing information, speaker for standalone practice, amplifier board for the speaker, an optional headphones port, and a MIDI Out port.

The central hub will connect to each of the sensors and play sounds out of the speaker corresponding to which sensors receive input. The sensors may also have the function to detect the magnitude of the vibrations and thus the velocity of the strike. This will allow the hub to adjust the volume and sound according to how hard the surface was struck. The user will also be able to choose between a variety of built-in libraries with different styles of drum kits to adjust the sounds to their liking. By default the drum sounds will be played out of an onboard speaker with a knob for volume adjustment. This allows the device to be used as a standalone drum kit allowing the user to practice without the need for a separate computer. There will also be a jack to connect a pair of headphones to the device in case the user needs to be silent or prefers to listen through their own device. When the headphone jack is being used, the onboard speaker will be muted. The headphone jack can also be used to connect the **OmniDrum** to an audio interface and allow its sound to be recorded.

Another main function of the device will be its ability to output MIDI information based on the player's inputs. MIDI (Musical Instrument Digital

Interface) is a ubiquitous communication standard used to allow musical instruments to communicate with computers and other devices. MIDI allows these devices to transmit what notes are being pressed, the velocity of the notes, and more information regarding adjustments made to parameters. Most common DAWs today are heavily reliant on the MIDI standard for creating a seamless bridge between hardware and software. Certain MIDI note values are commonly used to represent different drum hits and sounds. The **OmniDrum** will be able to communicate what drums are being hit and with what velocity to a DAW allowing the MIDI information to be recorded. It should be noted that while MIDI transmits note values and velocity, it does not transmit audio. This allows the user to choose any drum sound they have on their computer and assign it to each MIDI note being transmitted from the **OmniDrum**, making it an even more versatile product. The MIDI capability allows the **OmniDrum** to be used in the workflow of an amateur musician to produce complex and interesting drum rhythms that can be directly incorporated into their music. This is also especially helpful for traditional drummers to get their ideas out of their heads into their projects in a more familiar and natural way, as opposed to clicking around on a sequencer with limited capabilities.

2. Objectives & Requirements

2.1 Goals and Objectives

Basic Goals:

1. The device will be able to detect input from sensors and play a corresponding sound
2. The device should be able to play at least 3 sounds at once
3. The device should be able to output MIDI signals corresponding to which sensors are triggered
4. Sensors should be able to be placed on most hard surfaces and detect a hit

Advanced Goals:

1. The device should be able to detect the velocity of each hit and play a different sound depending on the velocity
2. Sensitivity of the sensors should be adjustable through a potentiometer
3. The device should have a headphones/line out jack, which mutes the onboard speaker when in use
4. Integrate an amplifier for headphone usage listed above. The amplifier may be disabled as to not interfere when connected to another sound system to not add unnecessary gain.
5. The device should have multiple sets of built-in sounds

Stretch Goals:

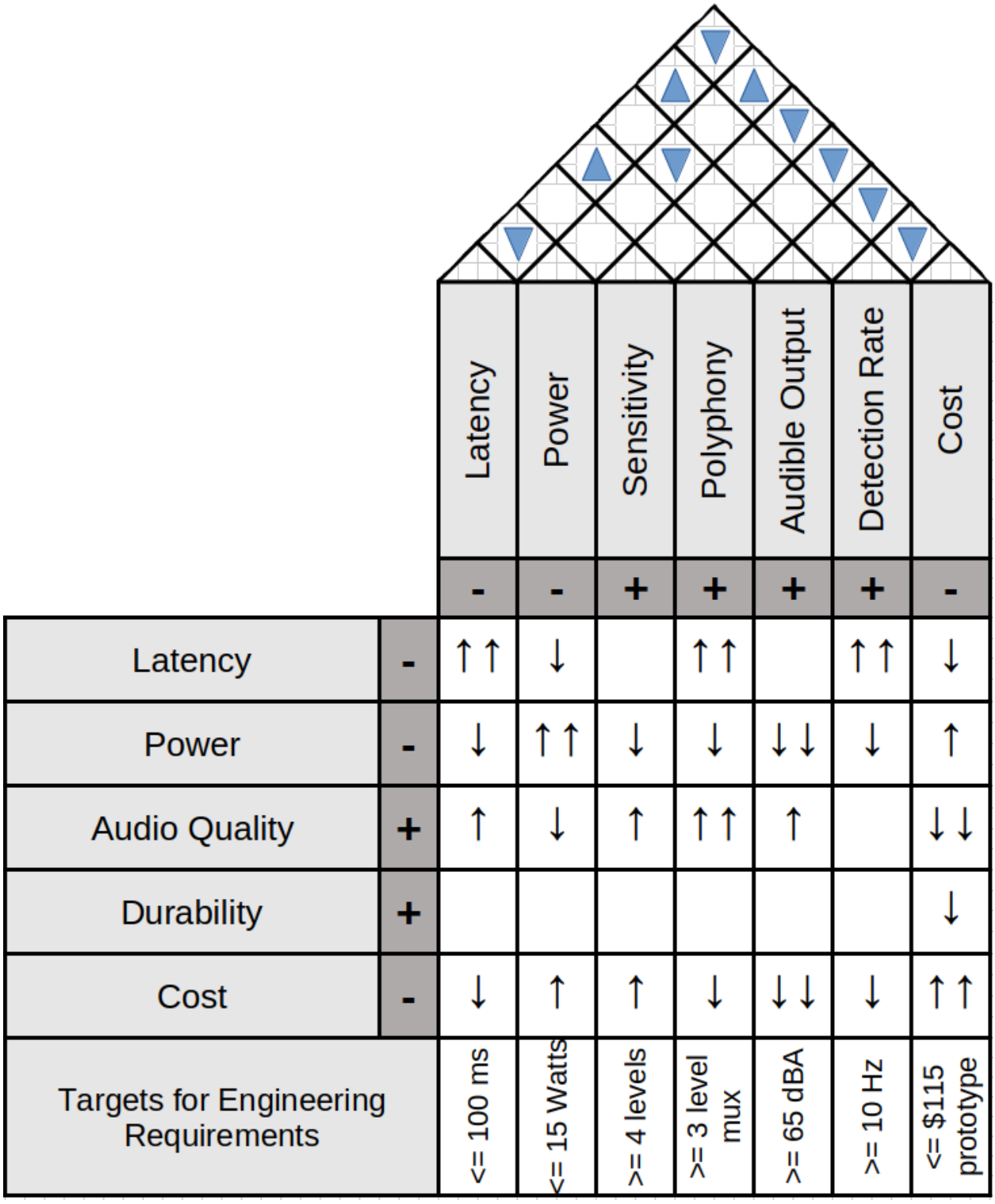
1. Stylization such as waveform or frequency spectrum display
2. Modularity such that an instrument may be added/removed with ease.

3. **Bump/crosstalk detection** to prevent sensors from picking up vibrations on the frame from accidental knocks or another struck sensor.
4. **Demo mode** where sample drum routines could play on their own while lighting up the instrument that is played. A user may follow along the routine by striking an instrument when it flashes.

2.2 Requirements & Constraints

Requirement & Constraints	Description
Polyphony	Must be able to play at least 3 sounds at once
MIDI Output	Must output correct MIDI information notes 35 through 45
Vel. Sensitivity	Must be able to detect 3 levels of velocity
Detection Rate	Must be able to detect hits that are at least 100ms apart
Low Latency	Latency must be no more than 100ms
Low Power Draw	Target power for circuit board should be less than 15 Watts
Audible Output	External speaker should achieve at least 65 dBA

House of Quality Table:



		Latency	Power	Sensitivity	Polyphony	Audible Output	Detection Rate	Cost
		-	-	+	+	+	+	-
Latency	-	↑↑	↓		↑↑		↑↑	↓
Power	-	↓	↑↑	↓	↓	↓↓	↓	↑
Audio Quality	+	↑	↓	↑	↑↑	↑		↓↓
Durability	+							↓
Cost	-	↓	↑	↑	↓	↓↓	↓	↑↑
Targets for Engineering Requirements		<= 100 ms	<= 15 Watts	>= 4 levels	>= 3 level mux	>= 65 dBA	>= 10 Hz	<= $115 prototype

Figure 2.3.1 *House of Quality*
by Jameson Palmer

2.3 Block Diagrams

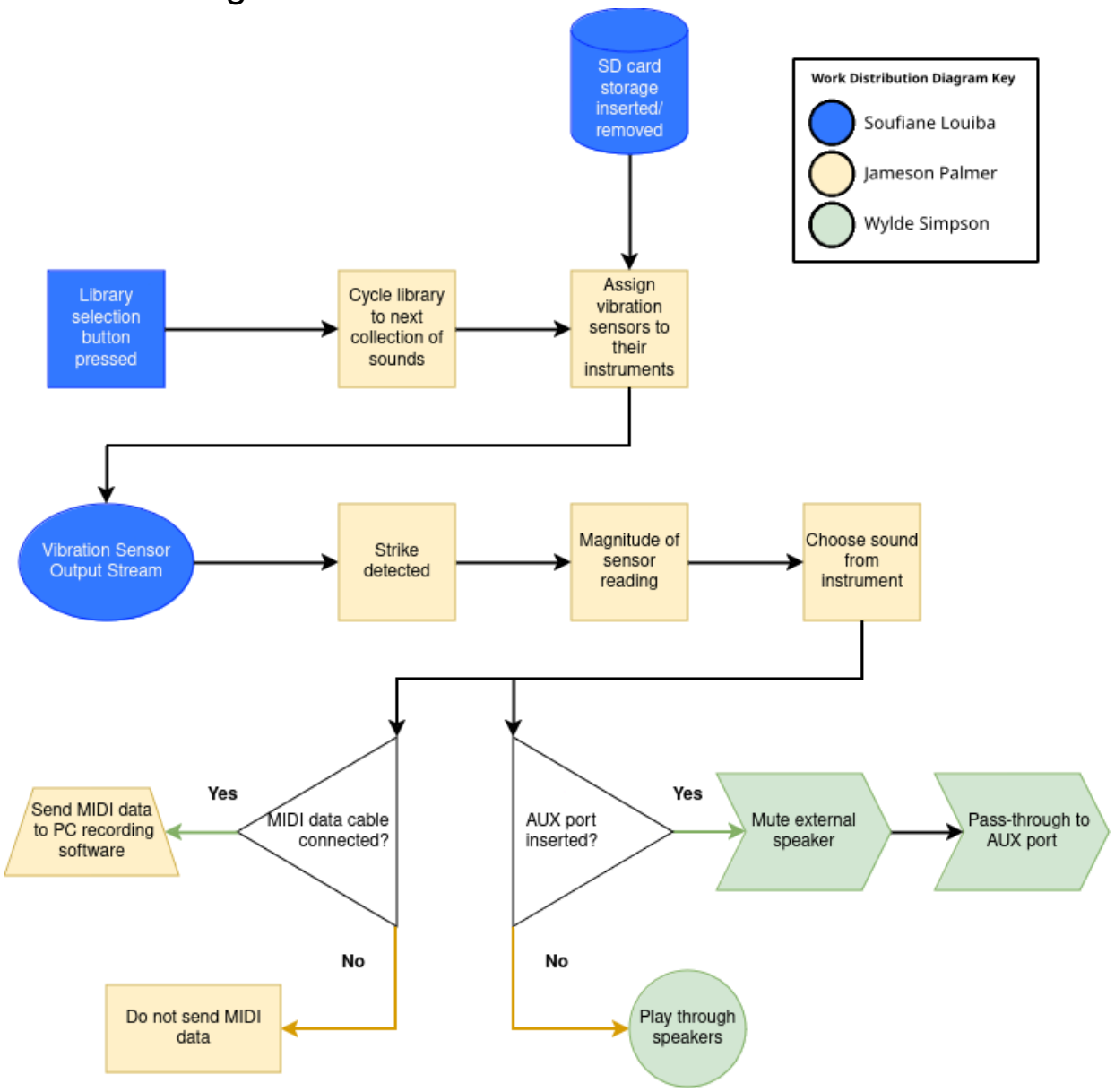


Figure 2.3.2 *Software Flowchart*
by Jameson Palmer

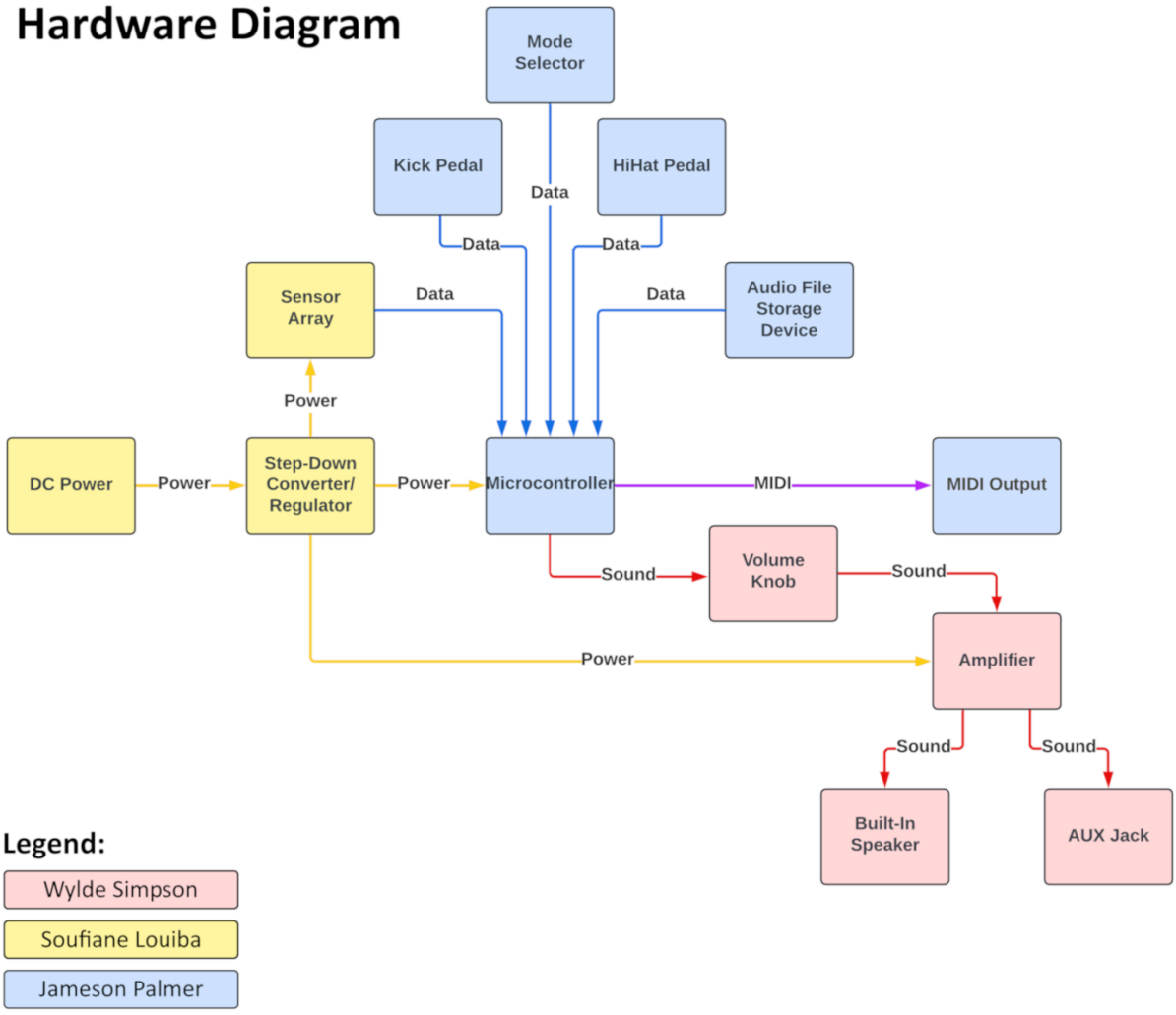


Figure 2.3.3 *Hardware Flowchart*
by Wylde Simpson

Certain tasks require collaboration by different members of the team to complete. The tasks listed here generally depict who is in charge of what, but some tasks will be worked on by multiple people in order to ensure the design works cohesively and properly.

3. Research and Part Selection

3.1 Research

3.1.1 MCU

3.1.2.1 Purpose

For this project we will be using a microcontroller to both detect input from the user as well as play the desired files at the correct time on an audio output device. The MCU will also have the job of outputting MIDI information so that the user may use the device to record the rhythms that they play onto an 3rd party piece of software capable of receiving and storing MIDI information. In order to accomplish this we need to find a microcontroller that can read audio files from a mass storage device and then transmit that data as an analog signal from an internal digital-to-analog converter, or transmit the file as a digital signal to an external digital-to-analog converter so that it may be amplified and played out loud.

Our MCU needs to be able to output multiple audio files at the same time. While playing, the user will be hitting multiple components of the drum kit at the same time and will need audio files to be able to play simultaneously. The microcontroller will be reading these files from an external storage device that will be discussed in a later section. It will also need to be able to cycle through different ‘profiles’ which will serve as separate collections of drum sounds that a user may choose from in order to play different types of drum kits.

MIDI and its role in the musical field will be discussed in a later section. Most microcontrollers should be able to transmit MIDI data over a GPIO pin. MIDI is an asynchronous digital communication standard consisting of basic 8-bit strings to transmit information on what notes are being played on an instrument and with what velocity, among other things. The MIDI standard is used ubiquitously within the world of digital music production.

3.1.2 Audio Output Device

3.1.2.1 Types of Output Devices

The goal of this component of the system is to convert the electrical signal output from the MCU into auditory feedback for the user of the device. Converting an electrical signal into a mechanical sound can be accomplished using different types of devices including electrodynamic speakers, piezoelectric buzzers, or even outside audio devices like headphones or in-ear monitors.

Electrodynamic speakers are a well known method for turning electrical energy into acoustic energy through the use of electrodynamic transducers. These generally consist of three parts: the motor, diaphragm, and suspension. An electrical signal that has been appropriately amplified is applied to the motor

which causes it to ‘push’ and ‘pull’ on the diaphragm. The purpose of the diaphragm is to help radiate and project the vibrations into the air in such a way that they are both loud enough and provide an accurate frequency response. The shape of the cone used has a significance on the frequency response of the output and thus the quality of the sound. The concept of frequency response curves is discussed in the next section. The final part of an electrodynamic transducer is the suspension, the purpose of which is to support the diaphragm and attach it to the rest of the speaker assembly while still allowing it to flex and vibrate in the desired fashion.

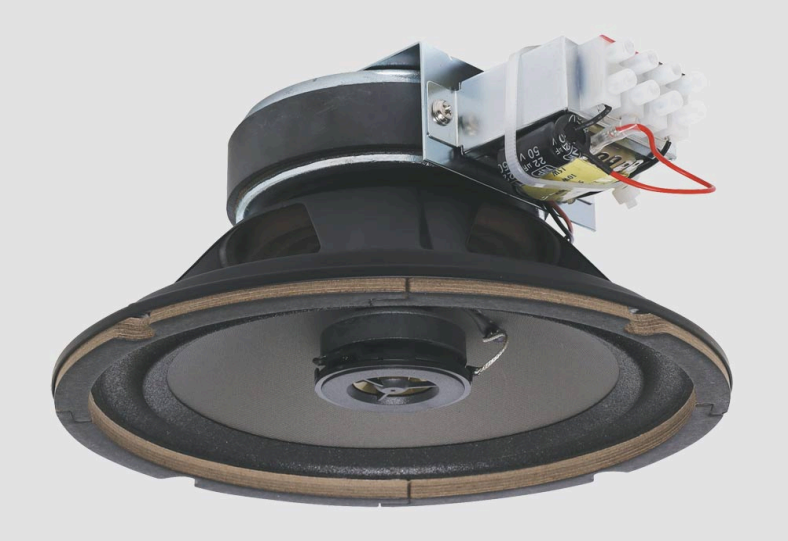


Figure 3.X *Example of an Electrodyanmic Speaker*
https://www.redbackaudio.com.au/wp-content/uploads/2016/09/C-2120B.jpg
(PENDING PERMISSION)

Piezoelectric buzzers function somewhat differently than electrodynamic speakers. To produce sound piezoelectric buzzers have a membrane that flexes based on the voltage applied to it. An advantage of piezoelectric buzzers is their low cost. They are relatively small and may require an added diaphragm to project their output. The most common use for piezoelectric buzzers is for alert tones and other applications that require only a single tone. They usually receive a digital signal in the form of a square wave at the desired frequency. These are not too useful for outputting analog audio with more harmonic content but they can be used for that purpose. Even though these buzzers generally receive a 1-bit digital signal, we can use pulse width modulation (PWM) to modulate the input signal based on the frequency of the desired signal. This way a 1-bit buzzer can be used to output varying frequencies. While 1-bit buzzers do have this capability, the output is usually plagued with much harmonic distortion and has a very limited frequency range.

Figure 3.X *Example of a Piezoelectric Buzzer*
https://squishycircuits.com/cdn/shop/products/Squishy-Circuits_Piezzo-Buzzer.jpg?v=1525723149 (PENDING PERMISSION)

Other devices such as headphones and in-ear monitors are very similar in that they employ transducers except much closer to the ears of the users and at much quieter volumes. While we do intend to allow the user to use these devices in our final design, they will not be provided to the user and are not intended as the default listening method. We intend to use some form of a loudspeaker so that the sound from the device will be audible to the user within the room that they are in; at the same time we intend to offer a connector jack for the user to plug in their own peripheral audio device if they choose to.

For our project, the best option would be to use an electrodynamic speaker due to it having a preferable full-spectrum frequency response as compared to the other options. It also gives us the ability to have the sound fill the room so that the user and other people may hear what they are playing without having to provide their own listening device.

3.1.2.2 Frequency Response

The term "Frequency Response Curve" refers to the curve derived from relating the average amplitude of a signal and the frequency of that signal. A desirable frequency response is one that is as flat as possible such that it outputs all frequencies with the same average amplitude and in turn provides the most accurate recreation of the sound. That being said there are many cases where a flat frequency response curve may not be the most desirable to the user. While flat frequency responses are the most useful for accurate audio replication, recreational listeners often prefer a more ‘V-shaped’ curve that has an emphasis

on lower bass frequencies as well as higher frequencies such as cymbals and sibilance, along with less emphasis on mid-range frequencies. This is in part due to the fact that human ears do not have a flat response curve for which they pick up different frequencies and usually emphasize mid-range frequencies the most. In general, all people's ears have slightly different frequency response curves based on the physical structure of the outer ear, inner ear, and even head shape.

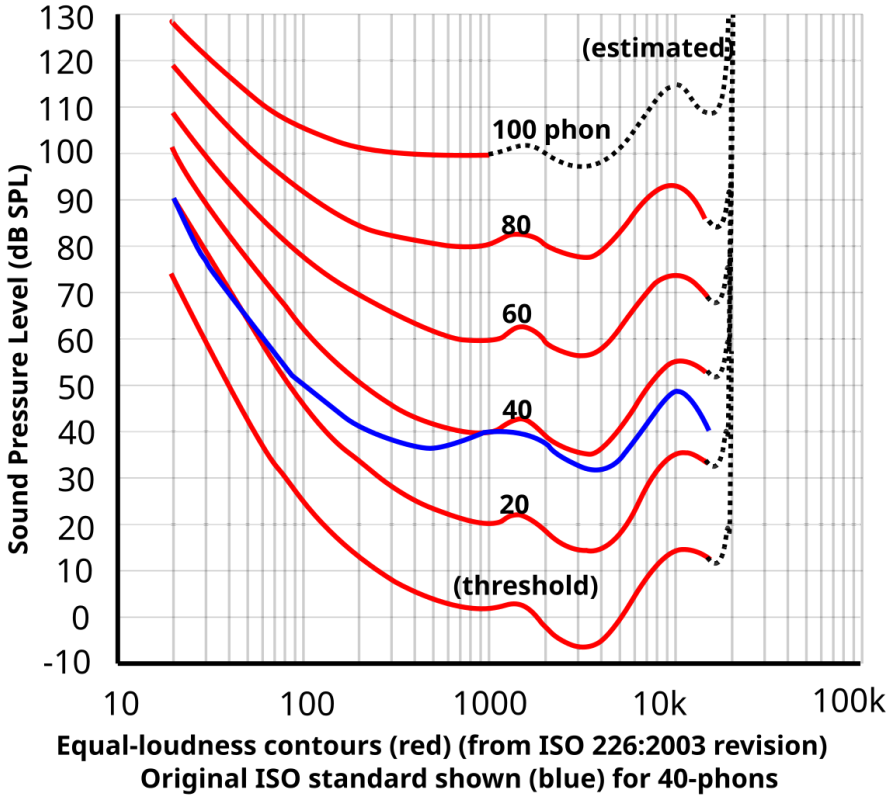


Figure 3.X *Graphic Representation of the Equal Loudness Contour* https://www.google.com/url?sa=i&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FEqual-loudness_contour&psig=AOvVaw0QD8qBgm87e3FZdWvBeVOH&ust=1729979379997000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCIiapcXBqokDFQAAAAAdAAAAABAE *(PERNISSION PENDING)*

The image below shows an example of a frequency response curve measured from a certain pair of studio headphones. The target curve in this case is not 'flat', but represents what the source believes would be the most desirable response curve to compensate for the natural variations in sensitivity of human ears.

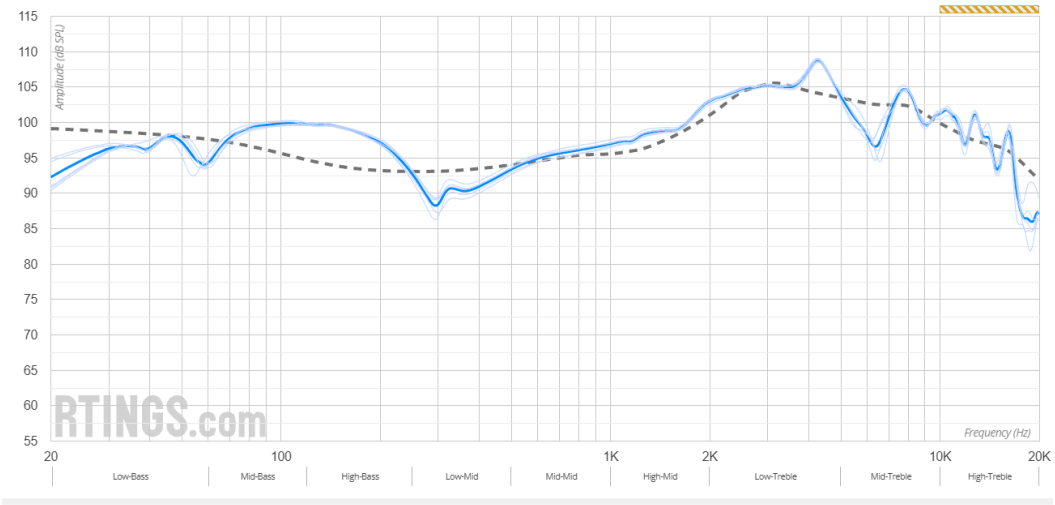


Figure 3.X *Raw FR L Graph for Audio-Technica ATH-M50x Headphones by Image Source: https://www.rtings.com/headphones/1-8/graph/25500/raw-fr-l/audio-technica-ath-m50x/295*

It should be noted that this description is a simplification of a very complicated topic. Which specific response curves sound good and in which application varies greatly between different situations, and acoustic measurements can be taken in a variety of ways. For our purposes the frequency response curve allows us to get a general idea of the sound quality of the type of speaker we choose. The goal is to use a speaker that provides decent replication of drum sounds so that the device may be used without the user needing to provide their own playback device. While an excellent frequency response and high sound quality are preferable, that is not the primary focus of the project.

3.1.2.3 Speaker Wattage and Impedance

When choosing a speaker to use in our project, there are certain specifications that must be considered. We need to consider the wattage of the speaker we want to use as well as the impedance. The wattage of the speaker, while not solely responsible for the overall loudness of the speaker, is a good estimate for how loud the speaker will be. The wattage listed on speakers is usually the peak power output, rather than the continuous power output. This is important for understanding how much power a speaker can be expected to use on average. It is harder to estimate the exact average power of a speaker since it changes with the volume and harmonic content of the output, but generally the RMS wattage is a fraction of the peak wattage. For the purposes of this project, a speaker around 1-5 watts would be appropriate. Speakers of this range are able to fill an entire room while requiring a modest amount of power. While this is not an appropriate amount of power for a live performance, the user will be able to plug in an external playback device to the output port and amplify the sound

further if needed. 1-5 Watts is an appropriate amount of power for a practice setting.

The impedance rating of a speaker also has an effect on the sound output as well as the specifications of the amplifier that will need to be paired with it. Common impedance values for speakers are 4Ω, 6Ω, or 8Ω, though there are more possible values. Lower impedance is generally regarded as being better because it means that the speaker is able to get louder with less distortion than a speaker of higher impedance, however it can be more expensive as well because the amplifier needs to be able to provide higher amounts of power. For our purposes the impedance of the speaker will not matter too much so long as we can match it with a proper amplifier. If the speaker is within the power constraints of the project it should serve the purpose adequately.

3.1.3 Amplifier

3.1.3.1 Why Amplify?

In our design, the MCU will pull files from a microSD card and transmit them to a Digital-to-Analog converter. After the digital signal is converted to analog, it needs to be amplified to an appropriate level to be handled by a speaker. The required amount of amplification that the circuit will need depends on the specifications of the speaker. The amplifier used cannot be too powerful or it may damage the speaker and possibly even more of the circuit. Amplifiers need to be matched to speakers depending on both wattage and impedance. The exact specifications of the amplifier will depend on the choice of speaker and will be covered in a later section.

3.1.3.X Low-Level & High-Level Signals

Low-Level and High-Level are terms used to generally describe the strength and/or type of signal passing through a device or cable in the world of audio. A low-level signal is the type of signal that will come out of the Digital-to-Analog converter in our circuit. Low-level signals are signals that have not been amplified such as signals coming out of electric guitars, microphones, and unpowered mixers. High-level signals are signals that have been amplified and would go into loudspeakers or PA systems. The distinction between these types of signals is important because plugging a low-level signal into a device that is expecting a high-level signal will not work due to the fact that the low-level signal is simply not strong enough to drive the device. High-level signals going into a device expecting a low-level signal have the capability to damage the equipment because of the fact that they are much more powerful than the device can handle.

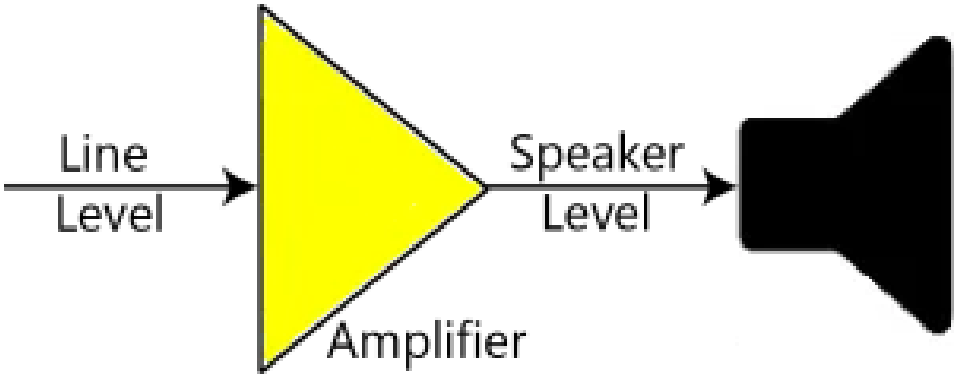


Figure 3.X *Understanding Audio Level, Line Level, Speaker Level. Image Source:* https://www.rtings.com/headphones/1-8/graph/25500/raw-fr-l/audio-technica-ath-m50x/295

While the term 'high-level' describes a signal that has been amplified, it is not specific about how powerful the signal is. A signal coming out of a 30 watt amplifier is much different than the signal coming out of a 1000 watt amplifier. These terms are context dependent and have different meanings based on what the situation is. High-level in our project may be a signal amplified to drive a 3 watt speaker. In the context of car A/V however, this might mean the signal is amplified to drive 200 watt speakers.

3.1.4 Auxiliary Output Jack

3.1.4.1 Audio Jack Form Factors

There are many different form factors for audio output jacks. We will need to decide on a specific one to mount to the main housing of the device. The purpose of the AUX jack is to provide the user with the ability to connect their own personal playback devices. These can range from headphones to external speakers to even connecting to a USB audio interface or other recording device. This can be especially useful if the user would like to play the instrument quietly and without using the internal speaker, or in a situation where there is too much ambient noise for the internal speaker can be useful. It is also useful if the user would like to directly record what they are playing into a software program or some other device. Some examples of audio jack types are the well known 3.5mm jack, the ¼" jack, the XLR connector, as well as RCA cables.

The 3.5mm jack, commonly referred to as the 'Headphone Jack' on many modern devices, is one of the most ubiquitous audio jacks found on consumer products. Some examples of the male 3.5mm connector are shown below, as well as examples of the female 3.5mm jack. The 3.5mm jack can be found on computers, mobile devices, toys, instruments, and certain peripherals. Its widespread use makes it a very accessible type of connector to use in our design

as most people would have some type of listening device that incorporates a 3.5mm audio connection.

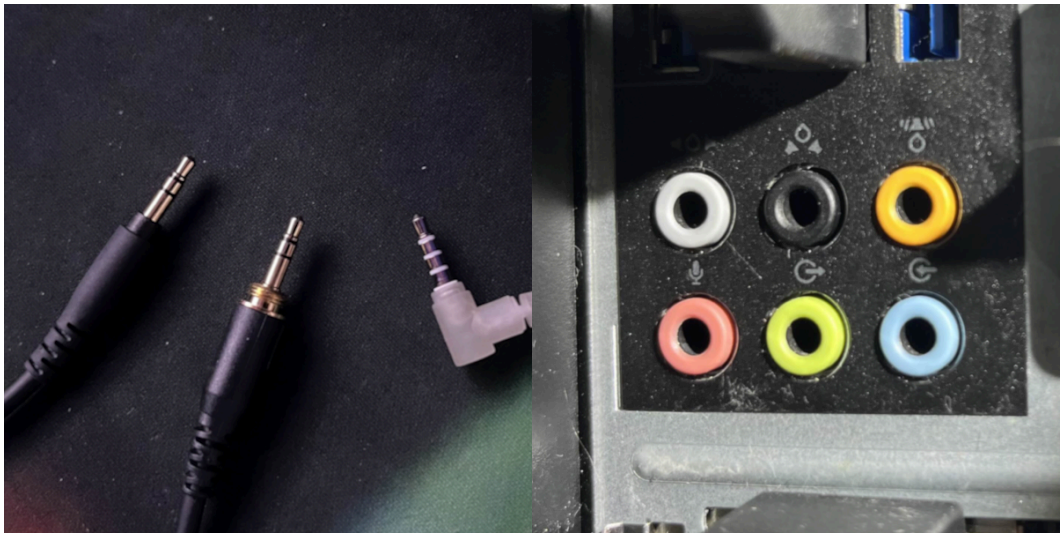


Figure 3.X *Male & Female 3.5mm audio connectors. Image Source: Author*

Another form factor is the ¼” jack; cables for this kind of jack are colloquially referred to as “Guitar Cables”, “Instrument Cables”, or sometimes “Line Cables” as they are commonly used to carry high-level or ‘Line’ level signals. However, ¼” cables are also very common for other uses with sound technology. The ¼” is primarily prevalent in the sound/music space as it is used to connect guitars to amps, instruments to mixers, headphones to interfaces and monitoring equipment as well as patch signals in modular synthesizers and Effects Racks. The ¼” connector and jack appears as an oversized 3.5mm connector and jack, and it functions in practically the same way. The bigger jack is preferred in these situations due to it being more rugged and able to handle higher currents than 3.5mm cables. Most studio headphones used for more professional audio work have this type of connectors as they are expected to be plugged into a mixer or USB audio interface where ¼” jacks are the norm. It is possible that a user of the drum kit would be more likely to have listening devices that use this connector assuming they have other audio equipment, but also the 3.5mm connector cannot be ruled out since the product is aimed at a low price point and a user buying equipment on a budget is more likely to be listening with simple headphones or earbuds. This must be taken into consideration when choosing which connector type to implement into our design.

Figure 3.X *Male & Female ¼" audio connectors (with male 3.5mm for reference). Image Source: Author*

The XLR connector is a form factor commonly used in stage settings. This connector type most commonly contains 3 pins, though a 5 pin version does exist but is much less common in audio-related situations and is found more often in lighting control scenarios for transmitting DMX information. XLR audio cables are generally used for connecting microphones to other devices such as mixers or PA speakers. The use of pins for XLR audio transmission is very similar to that of balanced 3.5mm and ¼" connectors such that there is a ground pin, a hot pin for carrying the audio signal, and a cold pin used for noise reduction. While XLR audio connections are useful in the world of sound, they generally do not fit our specific purposes and are unlikely to be useful.

Figure 3.X *Male & Female XLR audio connectors. Image Source: Author*

The final form factor researched is the RCA connectors and jacks. RCA connectors have been around for a very long time and get their name from the Radio Corporation of America which invented the design in the 1930s. The design consists of a central protruding conductor surrounded by and separated from an outer barrel-like conductor. The central conductor carries the analog signal while the outer conductor acts as ground. These connectors have been used for many audio related purposes in the past such as connecting to radios, record players, and tape decks, as well as connecting TV sets and early home computers to external audio devices. They were also used to transfer video over both Composite video connections and later Component video connections before being succeeded by HDMI. Use of RCA cables has dissolved over the past few decades aside from specialized uses today such as Tape In/Tape Out recording on mixers as well as use in connecting car stereo systems to amplifiers and other DSP related components.

Figure 3.X *Male & Female RCA audio connectors. Image Source: Author*

3.1.4.2 Tip, Ring, and Sleeve

It can be seen in **Figure 3.X** that the number of separate contacts on the male connectors of the 3.5mm and ¼" jacks varies. These contact areas are referred to as the Tip, Ring, and Sleeve. 3.5mm and ¼" connectors can be set up in different configurations such as TS, TRS, TRRS, and even the very uncommon TRRRS. The number of contact areas is representative of the number of individual wires running through the cable. The sleeve portion of the cable is dedicated to carrying ground. A cable set up in the TS configuration is used for carrying a mono signal, or a single channel of audio, where the tip carries the audio signal and the sleeve carries ground. This is the most basic configuration and lacks the advantages of more advanced configurations.

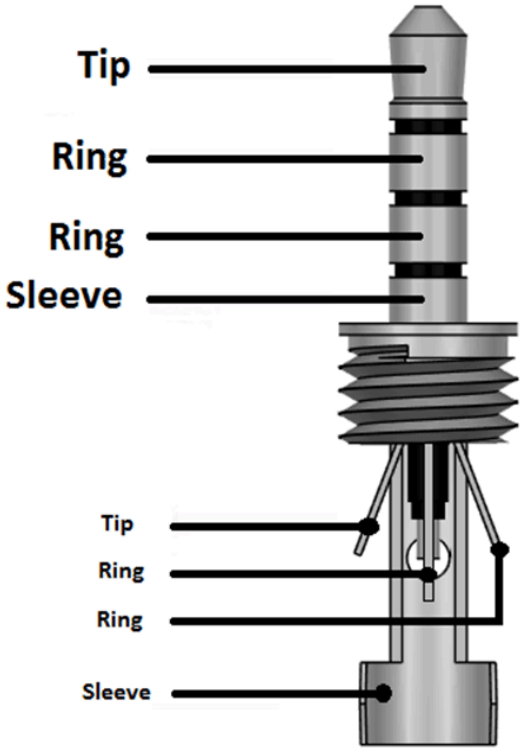


Figure 3.X *3.5mm Audio Jack, Tip Ring Sleeve Example Image. Source: https://www.google.com/url?sa=i&url=https%3A%2F%2Fcomponents101.com%2Fconnectors%2F35mm-audio-jack&psig=AOvVaw2Ihe2ZSpAII_UP7lRyZaIo&ust=1729979590256000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCMC1xanCqokDFQAAAAAdAAAAABAf (PERMISSION PENDING)*

Cables with the TRS configuration have three conductors and can be used for different purposes. One option is to use it for carrying a stereo signal, where the tip carries the left channel while the ring carries the right channel. Another use for TRS cables is carrying a single ‘balanced’ mono channel. In this case the tip carries the ‘hot’ audio signal while the ring carries a ‘cold’ neutral channel. As interference is picked up by cable, it has an effect on both the hot and cold conductors. At the other end of the cable, the polarity of the signal on the cold channel containing the isolated noise is inverted and the inverted noise is then added to the hot signal to phase cancel the noise in the hot signal. This technique is exceptionally effective at reducing interference noise in applications where the length of the cable is significant enough to pick up interference from the environment.

Further configurations such as the TRRS configuration are common on earbuds or headsets which include a microphone. In the case the tip carries the left audio channel, the ring adjacent to the tip carries the right audio channel, the ring adjacent to the sleeve carries the outgoing signal transmitted from the microphone, and the sleeve acts as ground. Further configurations such as

TRRRS are very niche and usually only applicable in specialized applications. A cable with this configuration carries 4 conductors as well as ground and could be used to carry 4 unbalanced mono channels, 2 balanced mono channels, 2 pairs of stereo channels, or a number of other combinations of signals.

A useful feature of this form factor is that it allows for some intercompatibility between different cable configurations. For example, many computers contain a TRRS headphone jack as well as a separate microphone jack to account for the many different types of listening devices that could be connected. A pair of stereo headphones with a TRS connector can still be used normally even when plugged into the TRRS jack because the contact for the ring adjacent to the sleeve on the female connector would connect to the sleeve contact of the male connector on the headphones. This would appear to the computer as being shorted to ground when the device is plugged in and tell the computer that the listening device connected does not have a TRRS microphone connection, without transmitting any digital data between the devices. A mono listening device with a TS connector would function similarly in this scenario and only play the left channel from the computer, which isn't preferable but still allows for the listening device to function. However, it should be noted that certain combinations, such as this, can result in damage to devices if they are old or badly designed. Most devices today are designed to protect against possible damage from shorting certain connections, but not all devices are, so it is best to use the correct connectors when possible and when you know that the combination will be accepted by the device.

3.1.4.3 Headphone Amplifier

For the auxiliary output to function correctly, it needs to be amplified to the required level to drive headphones. Headphones generally have much higher impedance than loudspeakers and will require a different type of amplifier to be able to drive them. This amplifier will be a separate amplifier than the one used to drive the loudspeaker.

3.1.4.4 Muting the Internal Speaker

A common feature of many musical devices is the ability to mute the external loudspeaker when a device is plugged into the auxiliary jack, without the need for the user to flip a switch. We would like to incorporate this feature in our design to streamline the user experience.

3.1.5 Sensors

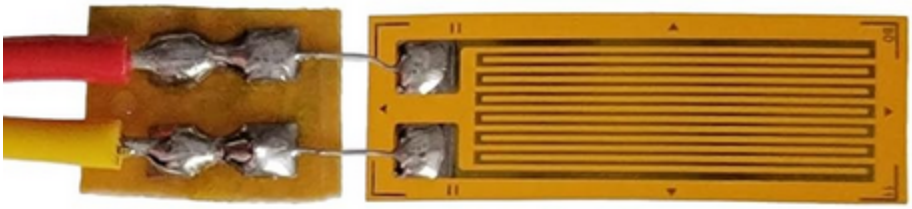
3.1.5.1 Applicable Sensors

The user interface for our drum kit will consist of several modules such as the cymbal, hi-hat, snare, and bass. The designs for each module will incorporate a sensor to detect a user's strike on the instrument. Translating a strike into

digital communication with the mainboard will be the duty of whichever sensor best applies. What can be detected from a strike on the instrument? The plane of each instrument will encounter a vibration. Acoustic instruments are fundamentally designed for vibration frequencies to create music so a vibration sensor would seem to be a good candidate for our design. Each strike on a surface could have its force measured as well. The pressure or force placed on the surface area could be connected to a force sensor which can measure many different levels of force exerted. Another alternative would be to consider the strike surface being connected to an actuator such that it may depress from a hit. At greater force on the instrument the actuator would depress at a greater velocity. Measuring the velocity could then provide us enough information to determine the force of a strike at many different levels.

3.1.5.2 Force Sensor

A force sensor, also known as a pressure sensor, measures the amount of force applied to it. Many of these devices work by converting the mechanical energy applied to them into electrical energy. Some subtypes of this kind of device include strain gauges and piezoelectric sensors. Strain gauges are often made of metal foil wrapped by wire which encases a plastic or ceramic cylinder. Under stress the electrical changes with the deformation of its shape. For piezoelectric sensors, they are often made of a crystalline structure such as quartz. Ceramics can also be made in a molecular structure that has a piezoelectric characteristic. The piezoelectric effect is an electric charge generated by a force on this material which will bend the crystalline structure. The deviation of these molecules will shift positive and negative charge along the crystal. The amount of charge generated is proportional to the force applied to the material and this charge can be measured through an ADC. Measuring the charge through an ADC can then allow us to measure the amount of force. Both of the described sensors would need power supplied to them. Although the piezoelectric effect creates an electric signal, many of the available sensors require a power source for the added circuitry. This should be considered when wiring our instrument modules to the controller board as there should be wires for power as well as data.



An example of a strain gauge is the BF120-10AA 120 Ohm resistance strain gauge. Supplied voltage is 10 Volts and has a variant BF120-3AA with supplied voltage of 3 Volts.

3.1.5.3 Velocity Sensor

These devices can observe displacement over time. Some implementations of velocity sensors include optical sensors which measure light as it is deflected off of a target. Light can be directed back to a photodiode which generates electric current when subject to photons. A dedicated beam of light is integrated with this type of sensor. When in motion, the displacement changes the refraction of this light beam on the photodiode so a rate of change can be used to calculate the velocity of our target. For our purposes a velocity sensor would use a linear encoder which can detect velocity along one direction. When an instrument is struck the sensor can observe a piston as it is depressed inward. The target of our sensor would be the end of the piston. Our implementation would need to have a low latency response so detecting velocity with a light beam could minimize the time for a response from our instruments. Due to the need for an integrated light source, this type of sensor would need active power delivery to operate. Implementation of this device should consider wires to dedicate a power source as well as wires to receive measurement data.

3.1.5.4 Vibration Sensor

These sensors could be used to measure the shaking of a strike surface on our drum kit. These can implement piezoelectric sensing technology or accelerometers. The sensor could be attached to the strike surface which is our target to be measured. It would detect the changes of motion of the target over time. Piezoelectric materials would have stress placed on their shape to create a charge across the material. Accelerometer-based implementations could detect the rate of motion over time to measure a frequency of a strike. These accelerometers can also use the piezoelectric materials.

3.1.6 MIDI

3.1.6.1 What is MIDI?

MIDI, which stands for Musical Instrument Digital Interface, is a communications standard that was invented in 1983 for use with synthesizers and other digital instruments. The MIDI standard allows for musicians to connect instruments to computers or other instruments in order to communicate what notes are being played, with what velocity, and other information such as Control Change codes and Program Change codes on different individual channels. Over the years MIDI has become extremely successful and is almost everywhere in the world of digital music.

In our specific case, we will be using MIDI to allow the user to communicate which drums they are playing with a computer program they may own. Digital audio workstations (or DAWs) have support for communications with MIDI hosts. If the user chooses to, they will be able to connect to a DAW and record which drums they hit in real time so that they may play them back later or edit them.

It should be noted that MIDI does not transmit audio data; it transmits purely digital information. Different notes in the MIDI standard are communicated with values from 0 to 127. The reason MIDI is useful in our scenario is that it will let the user record when they play certain drums at certain times so that they can record various rhythms they may come up with. Each component of the drum set will be assigned a ‘note number’ so that the computer can display when each individual drum was played. The user can then assign a drum sound file on their computer to each ‘note’ such that when they hit that specific drum, a chosen sound plays in the computer. This allows the user to endlessly customize the sounds of their entire drum kit, record it into the computer, and adjust the mixing and effects on the drum kit to eventually use it in a musical project in conjunction with other instruments they may record. The user will have to provide their own computer and DAW in order to utilize this feature, adding MIDI capability to our device only gives it the ability to connect to this kind of system in the first place.

3.1.6.2 How does MIDI communication work?

MIDI communication is done asynchronously at a baud rate of 31250 Hz. Information is sent in 8-bit packets containing note values, velocity values, and further information allowing a musician to change parameters on different devices remotely. MIDI cables generally consist of two 5-pin male ends which plug into female jacks at either end. The instructables.com website contains an example image of how an MCU would be connected to a MIDI jack in order to enable MIDI communications

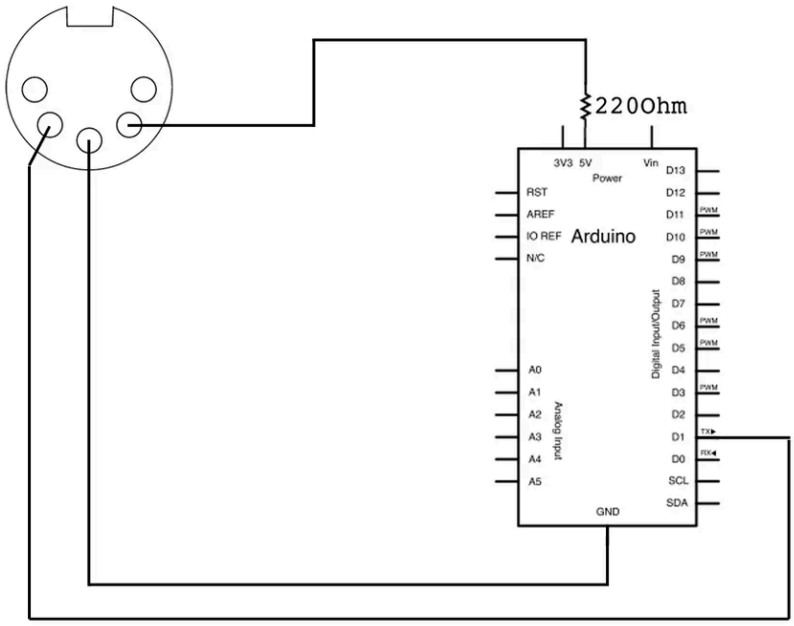


Figure 3.X *Arduino MIDI Connection Diagram. Image Source: https://www.instructables.com/Send-and-Receive-MIDI-with-Arduino/ (PERMISSION PENDING)*

The image shows that the MIDI connections are quite simple and wouldn't require a particularly advanced microcontroller to transmit. The constraints posed by the requirement for MIDI capability are minimal and we should be able to easily find a microcontroller that supports this.

3.1.6.3 What Note Values to Use?

MIDI note values range from 0 - 127 which gives us a lot of options for choosing which notes should correspond to each component of the drum kit. Luckily, the music production community has decided common note values for different types of drum sounds. Although these are not strictly the rule, many brand name products stick to these values as their ubiquity allows for widespread compatibility with other products. According to zendrum.com the commonly used MIDI drum values are as follows

35	Acoustic Bass Drum
36	Bass Drum 1
37	Side Stick
38	Acoustic Snare
39	Hand Clap
40	Electric Snare
41	Low Floor Tom
42	Closed Hi-Hat
43	High Floor Tom
44	Pedal Hi-Hat
45	Low Tom
46	Open Hi-Hat
47	Low-Mid Tom
48	Hi-Mid Tom
49	Crash Cymbal 1
50	High Tom
51	Ride Cymbal 1

Figure 3.X *Common MIDI Drum Values. Image Source:* *https://www.zendrum.com/resource-site/drumnotes.htm*

These values encompass the different components of the drum set that we plan to implement for the user. This should allow the user to be able to easily use our drum kit with drum kit presets that they may have in their DAW, as well as customize the sounds of it to their liking, enabling more versatile use cases of the drum kit.

4. Design Constraints and Standard

4.x Software

4.x.1 Sound Library

4.x.1.1 Directory Structure

Sound files will be saved in external flash storage. The directory structure for these files should not use spaces in the directory paths or file naming. Each instrument should map to the proper sound file on this device. Some devices may be mapped to the same sound profile such as two cymbal instruments or multiple snare drums. We will be implementing a directory structure such that a profile has a sound file mapped to each instrument installed. There may be

multiple profiles saved to flash storage to cycle through using the controller board buttons. The modules of our drum kit consist of four distinct instrument types:

1. Cymbal
2. Hi-hat
3. Snare
4. Base

For each of the four instrument types there will be several strike levels which can have their own sound file. Depending on how hard an instrument is hit the level of the strike will be determined by the sensor's output. Within each profile will be directories for each instrument which themselves will have the required sounds for each level of strike.

```
PROJECT/
├── PROFILE_1/
│   ├── BASE/
│   │   ├── 1_HIGH
│   │   ├── 2_MID
│   │   ├── 3_LOW
│   │   └── ...
│   ├── CYMBAL/
│   ├── HI-HAT/
│   ├── SNARE/
│   └── ...
├── PROFILE_2/
├── PROFILE_3/
└── ...
```

The PROJECT folder will be in the top directory location of the device meaning it is not within any other folder on the device. All profiles reside within this PROJECT folder. Only profile folders should exist in this directory. If there are folders that contain anything unrelated to the sound library here the folder may cause unforeseeable errors. The number of profiles that exist here should be no greater than 16 or 32 as the user cycling through greater amounts may have a poor experience. Naming convention of any custom profile folder should not need to adhere to any convention although it can be greatly helpful to include a number either at the beginning or the end to have it arranged in a specific order in the selection display. The numbers should be unique starting from 1 up to the maximum profiles that are chosen to implement.

Within each profile folder the default configuration can assign automatically to the instruments. Default configurations include instrument names BASE, CYMBAL, HI-HAT, and SNARE. Absence of any of these folders may leave the correlating instrument unassigned unless other configuration data can be found such as the profile.conf settings file. This configuration file will reside

within their respective profile's folder. The sound files within each instrument folder need to be numbered starting from 1. The number for sound files should be at the beginning followed by a label which can be anything. The standard number of files to each instrument should be three although as few as one file is necessary for operation. Greater numbers of files can be included and will equally divide strike levels amongst them. The loudest strike should always be numbered 1 and goes softer as the count increases 2, 3, 4, etc.

4.x.1.2 Filesystem

The format that can be most suitable for our integrated computer would be the FAT32 filesystem. Nearly every device can use this format so a user may change the contents stored on the external storage. One of the biggest restrictions to this format is the maximum partition size of 2TB and file size of 4GB. These restrictions are well beyond the needs of our storage device requirements. Contents in this storage device will only be read from. Because we are using this as read only memory (ROM) the implementation of FAT32 support in software should be greatly simplified. Aside from these reasons, the legacy factor of FAT32 also provides a plethora of documentation and resources. Source code for other project implementations of this file format is also widely available online if we wish to use a free software solution.

4.x.1.3 File Format

Performance of our project can be significantly impacted by the choices of which file format and codecs we choose to use. There are several hardware constraints that will impact the decision for using compressed audio over uncompressed audio formats. The time to read from the flash storage will greatly increase if we choose to use uncompressed data while on the inverse side if a compressed codec requires too much processing from the integrated computer our project could behave poorly. Codecs that implement multiple compression level schemes introduce greater complexities to write software for.

The contents of any given sound file should consist of just near a second of audio to play the sound of an instrument. For uncompressed audio this duration would result in a small amount of data. With this in mind it is considered to use the lossless and uncompressed audio codec of wav. The relatively small sizes of our instrument sound files and minimal hardware requirements fit the constraints of our design. The drum kit hardware will have minimal compute cycles and I/O available to handle decompressing other formats. Software requirements to implement this wav format will be simplified for these same reasons.

Audio buffering can assist in lowering I/O to the external storage by caching files from the current selected profile closer to the processing cores. When a profile is active we can precache the relatively small sound files in memory and with uncompressed audio there should not be a larger memory footprint from caching both the compressed data and the decompressed audio.

The wav format has minimal characteristics for how data is encoded. These are details such as the sample rate, channel count, and bit depth. A common configuration for wav files includes 44.1 kHz sample rate, 16-bit depth, and stereo channel audio. For a one second sound file in this format the file size is approximately 90kB. The operating system for the controller board should be able to implement this common case at a minimum. The software can include broad support to wav format as development continues or through the use of freely available libraries to have greater support of the format.

4.x.2 Controller Board

4.x.2.1 Instrument Assignment

For each of the instrument modules there should be a dedicated port. The instrument assignment can be set statically in software to these input ports on the controller board. Statically assigning the instrument to these ports can help in simplifying coherence between the hardware layout and the operating system software. Implementing dynamic instrument assignment based on whichever module is plugged in to a given port can be challenging. There would need to be a way to identify between each module.

An alternative to having the operating system detect which module is connected would be to allow the user to have control over which input port has a particular module. Hardware requirements for this implementation would include a display that shows the configurations of each port, a button to select and cycle through these ports, and a button to cycle through one of the four instrument types to assign. Improper configuration could cause issues between the types of modules as the sensors for each module may have different calibration levels. The values that get read from one instrument module may be different from another instrument such as cymbals and snares. Matching each port correctly should be simple from a user perspective regardless of these decisions.

Not every port needs to be used. In the case that some ports are left unused the instrument assignment of these ports can still be set but they are nullified in software. There should be no signals received from an unused port. Even when a port is unused we can implement software to not take any value until an instrument module is inserted. While the software is running a user may plug and unplug these modules. The operating system should be aware of which plugs are active and which are disabled. Allowing each port to still have an instrument assigned can give plug-and-play functionality. The operating system should detect when a port changes between unused and used.

```
PROJECT/
├─ PROFILE_1/
│   ├─ BASE/
│   ├─ CIMBAL/
│   ├─ HI-HAT/
│   ├─ SNARE/
```

```
│  ├─ ...
```

Expanding the number of instruments in a profile may be as simple as adding another directory in the profile. Although some modules can be assigned in a way that they share the same sound file, there can also be a way to assign more unique combinations. If a user wishes to change the sound of one of their snare modules they may do so using the selection controls on the controller board. The snare module can be assigned to one of the other available instruments sound files or an added instrument they placed in the profile. The display will give information of which port is selected and can cycle through the directory names inside the active profile.

4.x.2.2 Profile Selection

A standard profile with four distinct instruments with three levels of strike detection can be approximately 1MB in size if using one second wav formatted files. This could allow for many profiles to be loaded onto our external storage device although we should have the number of profiles that can be cycled limited to some 16 or 32 unique profiles. The controller board will have a display and button controls for navigating the available selections of profiles. There should be a way to cycle all available directories within the PROFILE folder. This cycle should work with a forward and back button. Although some profiles may be customized to include greater or fewer instrument and sound files a user should have all instrument assignments made when switching profiles. This can be difficult for when a user creates a profile that deviates from the standard file structure as some custom sound levels and instruments may be present.

```
PROJECT/
├─ PROFILE_1/
├─ PROFILE_2/
├─ PROFILE_3/
├─ ...
```

In the case that a profile does not follow the standard file structure the operating system should keep a profile's settings when it is cycled back. A persistent method of allowing profiles to keep their assignment settings would be to include a profile.conf inside any given profile. Between power cycles this persistent file may load the custom profile in the desired configuration. The user experience is important when considering how customizable any given profile may be.

In the absence of such configurations, the operating system should make a best effort to assign a profile to a default configuration. A default configuration should be set to PROFILE_1 with the standard file structure. If any instrument folder is missing from the standard profile structure the system should not

reassign automatically but rather leave the instrument unassigned for the user to choose.

4.x.2.3 Strike Level Selection

This project will implement sensors to detect between several levels of strikes. The standard configuration should distinguish three levels of a strike and a level for no strike. Each of the four instrument modules may implement different sensing technologies that send out different values for the same forces applied to them. Calibration of these sensors will provide us with the threshold levels to divide our sound files.

Our lowest threshold should be between the point of the lightest strike detected and the lowest sensor reading. This boundary will be our dampening value to differentiate the lowest strike between bumps and knocks. No audio should play from these bounds. Each sensor should be hardened to crosstalk and bumps through software. The highest threshold should be just below the sensor values for the hardest strike on an instrument. When a sensor reads beyond this threshold the sound file for the high strike should play.

```
PROJECT/
├─ PROFILE_1/
│  ├─ BASE/
│  │  ├─ 1_HIGH
│  │  ├─ 2_MID
│  │  ├─ 3_LOW
│  │  ├─ ...
```

With both of these thresholds determined we can assign any number of strike levels that are available for the assigned instrument. Profile customization can add or reduce the number of strike levels present in their directory. In the case of only one strike level present we may use the lowest threshold as the value our sensor can read to play that sound. If two levels are present we may use these two existing thresholds. As more levels of strike detection are present the system may evenly split between these two boundaries as it adds more levels of strike detection. The limitation for how many divisions can be done would be determined by the range of values a sensor may provide.

The operating system should be able to track how many levels of strike an instrument includes by the number of files in a directory. The file naming structure would need to follow standard conventions for the correct strike level to be used.

4.x.2.4 Operating System

Our integrated computer hardware will be a limiting factor for the operating system. With minimal hardware specifications the software that runs on the

controller board should be minimal and light on compute cycles. Other requirements for the software include responsiveness and highly parallelized communication for input and output streams. This system will be monitoring the array of input ports for sensor data, reading data from external storage, reading audio codecs, multiplexing audio playback, and managing output data streams to midi and speakers.

The drum kit will need to have operations done in real time or give the behavior of real time responsiveness. Creating a real-time operating system from scratch can be challenging. Some freely available operating systems that can be used for the base of our system include FreeRTOS. Support for this lightweight embedded operating system is present across many popular microcontrollers. Using this for the base of our software will provide more focus on the custom software of our project.

At startup our software will boot into the default profile, PROFILE_1. Precaching can then bring sound files from external storage into RAM. The software will identify the directory structure of the active profile and bind instruments to the relevant inputs. Scanning through a profile should have case handling for any deviations from the standard structure, any added or missing files or folders for instance.

The input signals will be individually buffered to memory where their values can be used for branching instructions. The operating system scheduler should give a higher priority to the task which monitors these buffer values. Because we are buffering the sensor readings we can retain the values of multiple sensors and read back all buffers at once. This scheduled task is considered to have a soft time deadline. In the event of a soft deadline passing for a real time operating system the value may not be wasted but at a degradation of performance. If the task were to miss the target deadline to read the buffer we may return back to this task and take the value this buffer still contains. Buffers should be overwritten when their contents are read from this task or if a new input value is detected from the sensor.

Multiple instruments must be able to play over each other. When a number of instruments are struck in series the sounds should layer over each other. To enable continuous playback for each sound file we consider a multiplexer configured for stereo channel output. All source files should be downmixed to stereo. Audio reencoding may be implemented to transform audio with more than two channels before sending it to the multiplexer. If a source is mono audio then the single channel should be sent to both channels to retain consistent playback from both channels. As more sounds are directed to the multiplexer their signals will be added over each other to give this layered audio effect.

Even though FreeRTOS is a complete operating system there are still edge cases that can cause errors for our project. Error handling should be dealt with gracefully while the system is still running. During software development we will use a detailed software flow chart explaining each case and branching direction as the software runs. A completed state machine diagram will help to minimize edge cases and reduce errors while the drum kit is running.

4.x.2.5 MIDI Data Output

5.Power Supply:

In this chapter we will discuss about the main components of the Omnidrum,which parts will suit better our project demands regarding the power consumption and efficiency

5.1. Power Specifications (Voltage and Current)

Our goal is to maintain the Omnidrum efficiency while keeping the total power consumption below 15 watts, to do that we need to focus on both the power supply and the design of the circuit

Our group will determine the voltage and current that are required for our OmniDrum system. In most of the electronic drum kits, the power requirements are usually between 9V and 12V DC, while the current supply will depend on the number of components (drum pads, sound module,displays,microcontroller, and amplifier). A calculation or estimation of the total current consumption for our system will need to be performed according to these components. The power consumption for Omnidrum could be broken down as follow

5.1.1. *Microcontroller:*

The choice of a microcontroller will be based on our project objectives of efficiency along with the low power consumption also maintaining a low failure rate, for our OmniDrum we're considering two types of microcontrollers.

5.1.1.1. *MSP430:*

Because we used this microcontroller in the past classes like embedded systems and Junior design 1 this might be a good choice for our project, but let's look at the power consumption of the MSP430, the operating input voltage of the MSP430 ranges from 3.6V to 15V.

The power requirements for each MSP430 family are listed below in Tables 3. The power given is based on the amount of current the core consumes per megahertz (MHz). The Analog IMAX column indicates the amount of current added if the additional functional blocks are used.

Table 3. MSP430x2xx Family Power Requirements[1]

DEVICE FAMILY	PIN NAME	VOLTAGE (V)		CPU I_{MAX} [2] (μA/MHz)	ANALOG I_{MAX} (μA)	COMMENTS
		MIN	MAX			
F20xx	V_{CC}	1.8	3.6	370	Comp_A+ = 60 ADC10 = 1200, ADC10_I_{REF} = 400 SD16_A + I_{REF} = 1700 RefBuffer = 600	20x1: Comp_A+ 20x2: ADC10 20x3: SD16_A
F21x1	V_{CC}	1.8	3.6	410	Comp_A+ = 60	
F21x2	A_{VCC}, D_{VCC}	1.8	3.6	350	Comp_A+ = 60 ADC10 = 1200, I_{REF} = 400	
F22xx	A_{VCC}, D_{VCC} [3]	1.8	3.6	550	ADC12 = 1200, I_{REF} = 400 OA = 290	22x2: ADC10 22x4: ADC10, 2 OAs OA currents for a single amplifier
F23x0	A_{VCC}, D_{VCC} [3]	1.8	3.6	550	Comp_A + = 60	
F23x, 24x[1], 2410	A_{VCC}, D_{VCC} [3]	1.8	3.6	445	Comp_A + = 60, ADC12 = 1000, I_{REF} = 700	224x1: Comp_A+ 23x, 24x, 2410: Comp_A+, ADC12
F241x, 261x	A_{VCC}, D_{VCC} [3]	1.8	3.6	560	Comp_A + = 60, ADC12 = 1000, I_{REF} = 700 DAC12 = 1500	241x: Comp_A+, ADC12 261x: Comp_A+, ADC12, two DAC12s DAC12 outputs not loaded; DAC12 currents for a single DAC

Figure *5.1.1.1.1* MSP460x2xx family power consumption
(https://www.ti.com/lit/an/slva335c/slva335c.pdf?ts=1729720208179&ref_url=https%253A%252F%252Fwww.google.com%252F)

5.1.1.1.2 *Linear Voltage Regulator (* TPS6212x 15-V, 75-mA Highly Efficient Buck Converter)

Since the MSP430 requires only 3.3V input Voltage we will need to use a step down voltage regulator between the power source which could be an AC adapter or a battery and the MCU. The TPS62122 is a highly efficient solution (up to 96% efficiency) capable of driving75-mA loads. 11-µA quiescent current makes the TPS62122 an ideal choice in systems concerned over battery life. The TPS62122 could be purchased for about 5 dollars, in addition to the low cost of this device, it provides additional protection to the MSP430 during the shutdown process. When VOUT has reached regulation (3.3V in this example), the TPS62120 signals the MSP430 through the PG (power good) pin. The TPS62120's SGND pin provides additional protection during the shutdown process. During shutdown mode, SGND provides a discharge path from the output capacitor to ground. This feature prevents lingering output voltages from discharging through the MSP430.

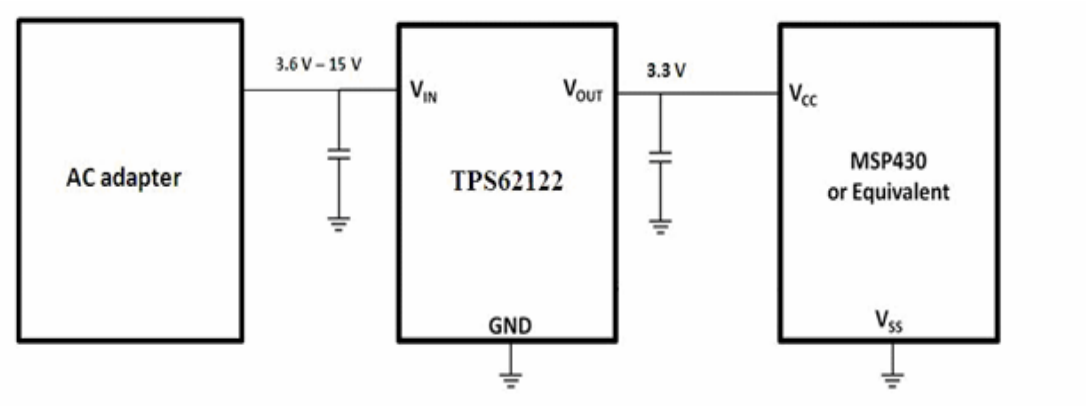


Figure 1. TPS62120 and MSP430 Simplified Block Diagram

Figure 5.1.1.1.3. TPS62120 and MSP430SImplified Block Diadrams **source(**https://www.ti.com/lit/an/slva335c/slva335c.pdf?ts=1729720208179&ref_url=https%253A%252F%252Fwww.google.com%252F**)**

Reference . (TPS62122 Datasheet– 15-V, 75-mA, 96% Efficiency Step-Down Converter (SLVSAD5))

5.1.1.1.3. *Linear and low dropout (LDO) regulator with reverse current protection*

The TPS709-Q1 low dropout voltage regulator has low voltage input ranging 2.7Volts to 30 volts which cost only about a dollar, it provides a regulated output voltage ranging 1.2volts to 5 volts. The LDO is capable of providing a steady output voltage without being impacted by the change in the input voltage which minimizes the noise making a good candidate for our design. Since the power consumption plays a big role in our project, using this device might be an issue for our design because of the heat dissipation due to the large difference between the input and the output voltages
The power dissipation of an LDO P(loss) is:

P(loss) = (Vin - Vout) Iout

Efficiency = Vout/Vin

This creates a reduction of efficiency by Vout/Vin
So, if our input voltage is 12V and output voltage which will be the voltage required by the MCU about 3.3V for the MSP430

Efficiency reduction = 3.3/12 = 0.275 = 27%

The excessive of the power dissipation could also damage the LDO

The LDO works better when the output Voltage is closer in value to the input Voltage or if the power consumption is not top priority for the design.

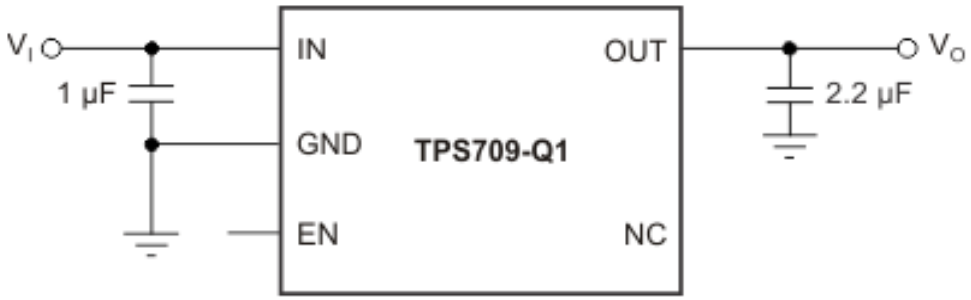


Figure *5.1.1.1.4* *. Low dropout voltage regulator (LDO) TPS709-Q1 source(TPS709-Q1 data sheet, product information and support | TI.com)*

5.1.1.2. *Arduino*

When it comes to choosing the right microcontroller, the size can play a big factor in reducing power consumption because each additional electrical device in the PCB consumes power to operate, therefore we need to look at the smaller one that fits our project. Other methods used to lower the power consumption involve reducing the clock speed and enabling the low power mode. The table below shows the power consumption of different Arduino microcontrollers with different sizes using the three methods mentioned above.

Microcontroller	Reference 9V	Reduce Clock Speed 9V		Reduce Clock Speed and Operation Voltage 3.3V		Enable Low Power Mode 3.3V		Enable Low Power Mode 9V	
Result of measurement	Absolute	Absolute	Relative	Absolute	Relative	Absolute	Relative	Absolute	Relative
Arduino Nano	22.1 mA	18.5 mA	-16%	3.4 mA	-85%	3.4 mA	-84%	4.8 mA	-78%
Arduino Pro Mini 5V	14.6 mA	10.0 mA	-32%	3.7 mA	-75%	1.6 mA	-89%	3.2 mA	-78%
Arduino Pro Mini 3.3V	5.1 mA	3.8 mA	-25%	3.7 mA	-27%	1.6 mA	-69%	3.2 mA	-38%
Arduino Uno	98.4 mA	42.8 mA	-57%	11.6 mA	-88%	11.5 mA	-88%	27.9 mA	-72%
Arduino Mega	73.2 mA	61.8 mA	-16%	16.7 mA	-77%	11.9 mA	-84%	26.9 mA	-63%
Average Reduction			-29%		-70%		-83%		-66%

Figure 5.1.1.2.1:power consumption of different Arduino microcontrollers
source:(Guide to reduce the Arduino Power Consumption - DIYI0)

From these results we can see that the Arduino Pro mini is the most power efficient but the only problem with it is that it only has 6 analog inputs, so if we were to pick this for our project then we will have to add a multiplexer to expand the number of the analog input that will suit the number of the drum pads that we will use for the OmniDrum. One example MUX that can be added to the microcontroller isTexas Instruments CD74HC4067 16-Channel multiplexer which costs only $1.05.
Source (16 Channel Multiplexer - COM-00299 - SparkFun Electronics)

5.1.2 *Sound Module Power:*

The sound module is the heart of the electric drum,which is responsible for processing the signals from the drum pads. Our group will check the voltage and current specifications of the module that we will be choosing; in most cases the sound modules are powered by DC, from about 9V to 12V.

5.2. *Power Supply Source*

The user of the OmniDrum will be able to plug the kit directly to the power outlet for indoor use with an AC power adapter or use batteries for outdoors use which gives the user the flexibility of using the kits anywhere.

5.2.1 *Battery Power:*

the choice of a batteries will be determined by the to suit our requirement which is staying with power budges, so for power consumption of 15 watts, a 12V battery will be suitable with a current draw of no more than 1.25A
P = V x I = 15W
I =15/12 = 1.25A max
So to have a power lasting 5 hours and needing to draw 1.25A from the system we will need a 5h x 1.25A =6.25Ah battery.

5.2.2 *AC Power Adapter:*

A more stable solution would be to use an AC to DC adapter, which converts the high voltage AC down to the level of DC voltage required by the OmniDrum. The AC adapter must have the same voltage and current specifications required for the OmniDrum kit with a clean supply to prevent electrical noise for a good sound quality.

5.3. *Power Distribution*

5.3.1. *Internal Power Regulation:*

Some components of the OmniDrum may require different voltages. In such cases, a voltage regulator circuit is needed. For example, where portions of a system (sensors or controllers) require a voltage of 5V while the main supply is at a voltage of 12V, a step-down voltage regulator can be used to convert voltage to the required level (a set ratio).

5.4. *Protection and Safety:*

Fuses will be added in our power circuit to prevent excessive current from damaging the circuit in case of shorting or an overcurrent situation. The fuse will be just a little larger than our total expected current rating of the system. We will also use diodes in the power supply which can save some of the sensitive electronics from getting damaged due to reverse polarity.

5.5. *Energy Efficiency:*

Low Power Components: If your products are battery-powered, then you would want to use components that consume less power to prolong battery life.

Using efficient amplifiers or LED lights that don't suck too much power can minimize energy use. Sleep Modes: If your drum kit has a microcontroller, then putting your peripheral unit(s) into the sleep mode whenever it isn't used helps to minimize power consumption on the drum.

5.6. *Real-life examples of powering electric drums Commercial Drums:*

Most commercial electric drum kits, like Roland and Yamaha, use 9V or 12V common external power adapters which have certain current ratings (usually 500mA-2A), which forms a yardstick of reference to design your own.

5.7. *A few DIY*

electric drum projects do use Arduino or Raspberry Pi as sound processors.Yet it is mostly microcontrollers, for example, the Arduino or the Raspberry Pi, which process the signals and are powered by either a 5V USB supply or a 9V battery pack with a voltage step-down regulator.

6. Instrument sensor and housing design:

6.1*. Drum Pads:*

6.2. *Sensor design:*

Piezoelectric sensors are one of the most important components of our project; these sensors constitute an end product combining functionality with cost-effectiveness. The term “Piezo" originates from the Greek word, meaning, press. This sensor consists of two defined zones on a 27mm-brass-disc diameter: the inner circle contains a slice of quartz crystal that generates a negative output voltage, while the outer circle generates a positive output voltage.

Piezoelectric occurs when a mechanical pressure is applied on the christal; in our case, when the drum pad is struck with the drumsticks. The hit causes the piezo sensor to activate, producing an alternating current (AC) at the output, which runs through the cable to the module (or “brain”) and triggers it to produce a sound, every sound in the OmniDrum has its own sample which is different from a pad to another, the user then will have the option to play the sound back through either speaker or headphones. The strength of the electrical signal is directly proportional to the force of the hit, making the sensor highly responsive to varying levels of pressure. A light tap produces a weaker signal, resulting in a softer sound, while a harder hit generates a stronger signal, leading

to a louder sound. This makes the sensor highly responsive to different playing dynamics, letting the drummer control the sound's volume and intensity naturally.

One of the stand out reasons for choosing this sensor is its effectiveness. At less than $7 for a pack of 50, it provides a high-performance, low-cost solution, making it ideal for our project. This combination of affordability and precision makes the Piezoelectric sensor an excellent choice for applications where sensing force and pressure are critical.

In a typical circuit with piezoelectric sensors, which generate small voltages when deformed, a pull-down resistor connected between the sensor's output and ground ensures that when the sensor is not generating voltage, the output is reliably pulled down to 0V (or the low logic level). This allows for cleaner and more consistent voltage readings when the sensor is triggered, reducing noise or false readings.

In our case, we will use 1M Ohm resistors as pull-down resistors in parallel with the piezoelectric elements. The purpose of it is to stabilize the voltage readings from the piezo elements. Without the pull-down resistors, the output from the piezo sensors could float when inactive, leading to unreliable or noisy readings.

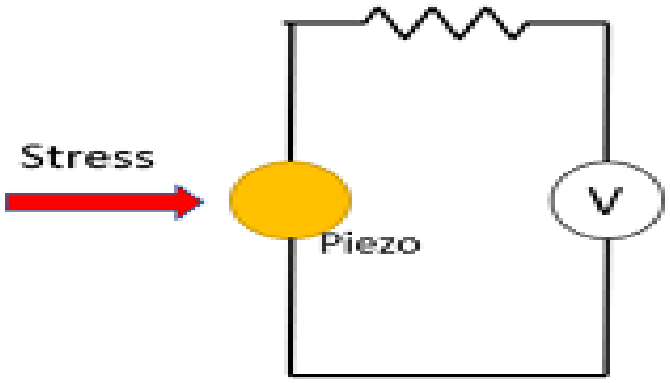


Figure 6-2.1: Piezoelectric sensor
Source (fyp_CS_2018_CMH_-_1506502.pdf (utar.edu.my)

6-3 *System Flow*

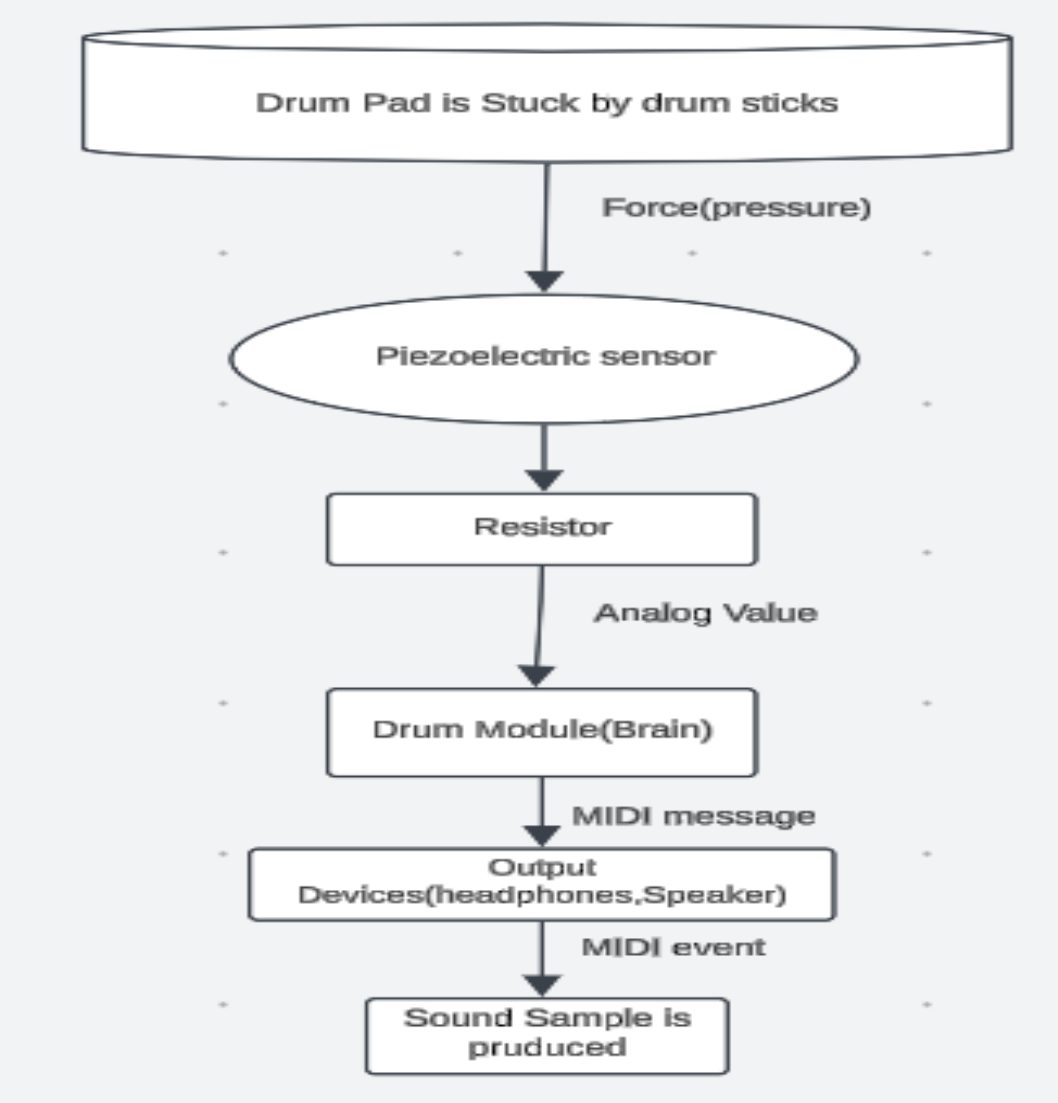


Figure 6.3.1 System Flow design

6.4 *Sensor setup:*

There are many types of Drum Pads, with varying applicability depending on the usefulness of the different playing techniques. Hence, the number of piezo sensors will differ from the most basic pad to the ones that contain multiple zones. They can be broken down into the following types of electronic drum pads with their respective piezo sensor requirements:

6.4.1. *Single-Zone Drum Pad*

These are the most basic electronic pads, designed to identify strikes in just a single zone. These pads are destined for toms, kicks, and other drums that do not demand more than one striking zone.
requires 1 sensor to be placed under the Center of the pad to detect any strike on the pad surface

Examples: Such as kick drum pads and tom pads in basic kits.

6.4.2. *Dual-Zone Drum Pad*

Such pads are made to detect hits in two different areas, usually the drumhead (center) and the rim, which could serve well for snare drums.
Two sensors are required to be placed one: Head sensor placed under the center of the pad, and the Rim sensor placed at the edge to register Rim shots.

example:Dual-zone snare drum pads, dual-zone tom pads.

6.4.3. *Triple-Zone Drum Pad*

Such pads are made to detect hits from three separate zones; these pads can find application in more complex playing techniques, much like those on a ride cymbal. The sensors are usually placed under the center of the pad(the drumhead), rim, and either bell or edge.

The table below shows different types of drum pads that produce one to three different sounds, therefore, our group is going to follow this table to place the sensors in their designated location.

Pad Type	Number of sensors	Pad placement
Single-Zone bass drum/kick Drum)	1	Under the Center of the Pad
Dual zone Drum Stool/Throne	2	Center of the and Rim of the Pad
Triple Zone Hit-Hats	3	Center, Rim, and Bell of the Pad
Single-Zone Cymbal Pad	1	Center of the pad
Dual zone Cymbal Pad	2	Bow and Edge
Triple Zone Cymbal Pad	3	Bow,Edge, and Bell

High Hat Pad (Single Zone)	1	Bow
High Hat Pad (dual Zone)	2	Bow and Edge
Kick Drum Pads	1	One Zone
Percussion Multipads	1 per pad	Single zone

Figure 6.4.3.1.: sensor set up for different types of drum pads

6.5. *Sensor connection to the Control board:*

A single zone pads will have one sensor, the positive lead will be connected to the analog input pin of the microcontroller and the negative will be connected to the ground on the MCU, a 1M ohm resistor will be placed in parallel to the piezo sensor as pull down resistor to dampen the voltage spikes and bring it down the safe levels to ensure a good read of the voltage produced by the sensor. The piezo sensor and the resistor can be soldered together to ensure a solid connection of the two, this will keep the analog voltage reading at zero volts until then until the piezo sensor sends a signal (voltage) basically preventing a false triggering. After adding a resistor we will connect the positive side to the microcontroller analog input, let say A0 and connect the negative side to the ground(GND).

We are also considering adding a diode like a Schottky diode to protect the microcontroller from a high voltage spike generated by the Piezo connected in parallel with the piezo sensor, the diode's cathode will be connected to the positive lead of the piezo sensor and the anode connected to the ground, if necessary we will add a small resistor of about 10K ohms connected in series with the piezo sensor to limit the current flowing into the analog input for to protect the microcontroller.

For the double and triple zone pads that contain more than one piezoelectric sensor we will follow the same way of connecting the single zone pad. A pad that has two sensors, each sensor will be wired to a different analog input in the control board.

A final circuit design of connecting the OmniDrum piezo sensors to the microcontroller using one diode and one resistor for the piezo sensor will look like the figure below adding more sensors.

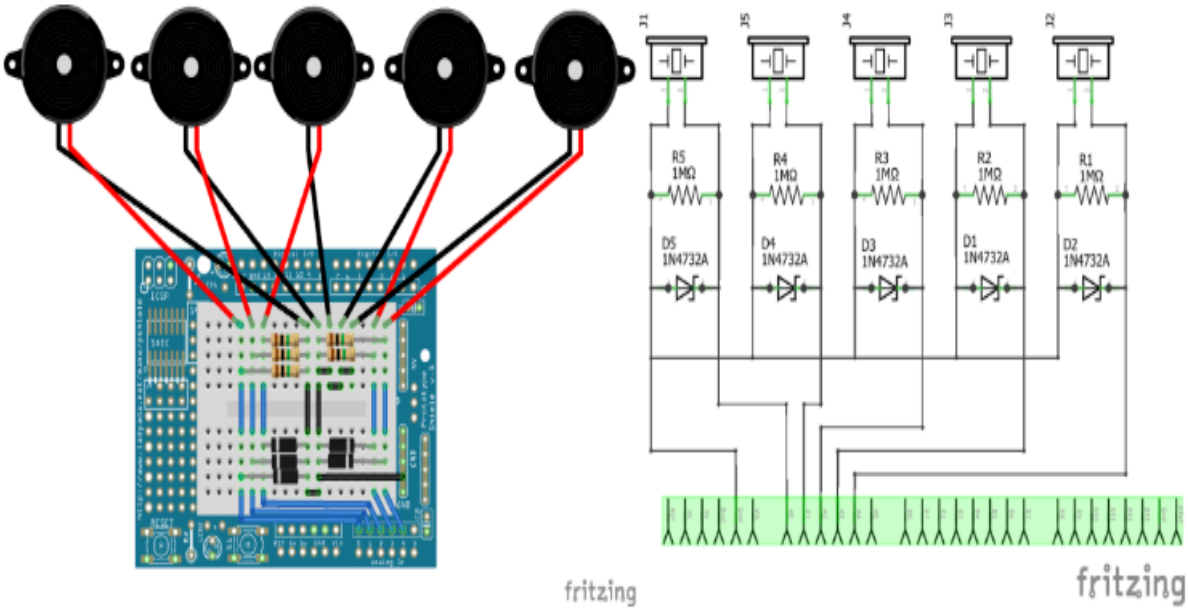


Figure 6.5.1 *circuit design of piezo sensor to the arduino MCU*
Figure :source(Arduino Piezo MIDI Controller – Simple DIY Electronic Music Projects)

5. Comparison of ChatGPT or Similar
6. Design
7. Overall Integration
8. Administration

8.1 Budget Estimates

This project is being funded by the group members involved in designing and making it. The table below provides estimates for the cost of the materials required to complete the project. The project aims to create a low cost solution to the problem of creating a functional entry-level electric drum kit.

Part	Estimated Cost
PCB	$30
Microcontroller	$3

IO Jacks	$3
Main Housing	$20
Sensors	$10
Pedals	$15
Speaker	$3
Amplifier Circuitry	$3
Power Delivery Circuitry	$5
Connecting Cables	$15
Other Components (Indicators, Buttons)	$3
Storage Device	$5
Total	**$115**

8.2 BOM

8.3 Milestones

In order to stay on track our team has set certain milestones in accordance with what is due throughout the course. Designing and testing is hard to foresee accurately and will probably vary in reality compared to what is listed in this table. Hard deadlines should stay the same unless changed by the SD1 coordinators. These milestones cover only SD1 and up to the end of the Fall 2024 semester.

Time	Milestone
Week 3	Finish D&C document
Week 4	Meet with SD coordinators and begin Draft Paper
Week 5	Research specific parts, refine theoretical design
Week 6	Work on draft paper, select final components and begin design
Week 7-9	Testing components, rough PCB design, draft paper
Week 10	Submit first draft paper, continue testing/designing
Week 11-12	Submit draft paper revision
Week 13-14	Submit final draft
Week 15	Submit mini demo video

8.4 Work Distributions

Task	Primary	Secondary
Kit Assembly and Housing Design	Wylde	Soufiane
Instrument Sensor and Housing Desig	Soufiane	Wylde
Sound Library Storage	Soufiane	Jameson
Connection between Sensors and Cor Board	Soufiane	Wylde
Strike Level Detection	Jameson	Wylde
Sound Library Instrument Selection	Jameson	Soufiane
Sensor Instrument Assignment	Wylde	Jameson
Sound Library Mode Switching	Jameson	Soufiane
Auxiliary port passthrough	Wylde	Soufiane
MIDI cable data path	Jameson	Wylde
Control Board PCB Layout	Wylde	Jameson
Power Supply	Soufiane	Jameson
Control Board Software	Jameson	Wylde

Declaration: We hereby declare that we have not copied more than 7 pages from the Large Language Model (LLM). We have not utilized LLM for any purpose at this time.

9. Conclusion

Appendix