Project Plan Final: Cyren

Version 2: 4/25/19

Members:
Justin Shaver, Thomas Frye
Will Pigg, Chandler Davis,
Caleb Hendrickson, Dan Bohlke

Faculty:
Chen Degang
Randall Geigar

Table of Contents

List of Figures

List of Tables

1.0: Introduction

1.1: Acknowledgement

Our client, Dr. Randall Geigar, has made this project very open ended. All of the design features, use cases, functional requirements will be up to our team to decide. The only requirement is that the target audience must be musicians and this project must utilize our skills as engineers and that we learn and grow from this experience. Our adviser, Dr. Chen Degang, has offered his support with the technical aspect of this project, specifically electronics, in the form of advice or answering questions. Dr. Geigar has also offered his assistance with any advice we may need.

1.2: Problem Statement

Musicians often require many devices in order to achieve desired sounds. In the case of a guitarist, a combination of several pedals may be required to produce a specific level of distortion to emulate a bassline. Another combination of pedals would be needed for the lead or chorus. When working with a multitude of devices, the musician will have to make sure that there are no compatibility issues and spend a great deal of time adjusting the device settings to find combinations that sound well together. Many music devices do not have features to save or load settings, which requires the musician to manually adjust their equipment each time they switch between sounds or effects. The musician is also required to store their equipment which may pose a problem under tight constraints. In the case of live performances, especially those that will require travel, having the space required in a vehicle may be quite the challenge along with ensuring that the equipment remains safe and undamaged. Also, working with many devices while on stage may require frequent breaks between songs to switch between equipment or adjust settings. When performing in a band, each musician will often have their own equipment for their instrument which even further emphasizes the problem with requiring breaks during performances along with storage and transportation. The need to have several devices for music production presents problems with storing and transporting the equipment, ensuring compatibility between devices, and seamless live performances.

1.3: Operating Environment

The end product will be designed to appeal specifically to musicians and to function properly in any environment in which they wish to produce music. This will include casual use at home, in an open space for jam session with band mates, or on stage during a live performance. For the design to maintain functionality in these conditions, the product must be durable. The device will often be used on the floor where it could potentially collect dust and frequent scuffs and scratches. It will be important that the device chassis is stable and made of a robust material. This will also be important for interacting with the device as it will have several stomp buttons and pedals. The product will also be expected to withstand frequent travel, especially for users that will be using the device for live performances or on tour. In addition, it is vital that the device can interface with existing music equipment, whether that be instruments, speakers, MIDI controllers, etc. This will require the use of pre-existing standardized connections for the I/O.

1.4: Intended Users and Intended Uses

Intended Users

As stated before, the target audience for this product is musicians. To appeal to the widest range of musicians possible, the design will be compatible with any instrument that can interface electronically. The device will be able to interface with multiple instruments at the same time, appealing to bands or even the one man band type. The end product could be used in an educational settings as well, possibly by teacher for demonstration purposes.

Intended Uses

The design is intended to be used by a musician, alone or in a group, to interface with one or more instruments and produce music with a large and diverse selection of effects and layering options that are easy and user friendly to implement or modify while keeping all the sound processing on one device. The design will feature the ability to save and load effect combinations and layers allowing for seamless transitions between desired sounds and recording those sounds.

Use Case Diagram

Below is a visual representation of the intended uses of our design in the form of a use case diagram.

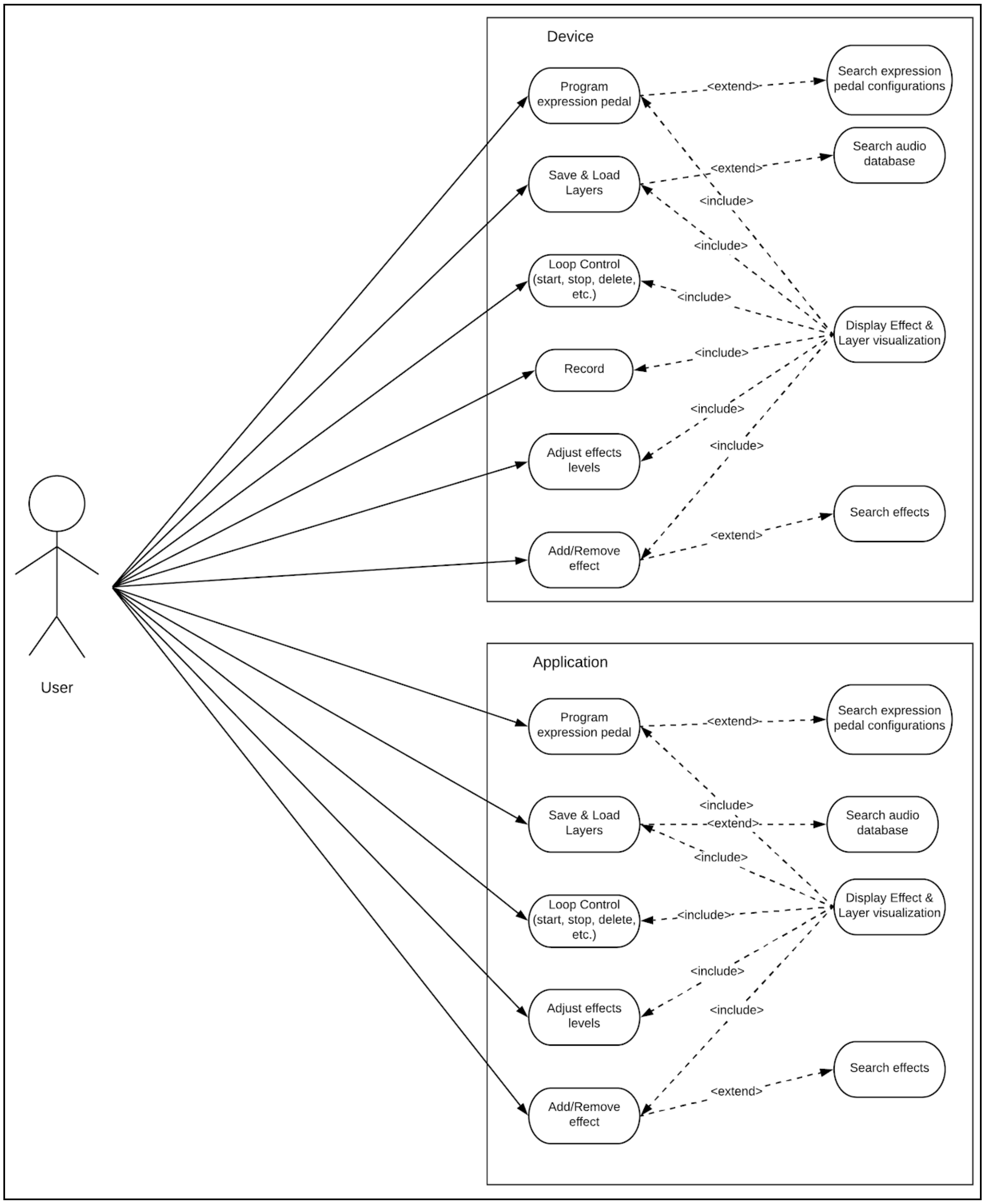


Figure 1: Use Case Diagram

1.5: Assumptions and Limitations

Assumptions

- The maximum of simultaneous users will be the amount of people playing instruments that are currently interfacing with the device along with, optionally, a person controlling the device via the phone/desktop application.
- The primary user will be the person that has the device at their feet, will likely be a guitarist or pianist, but could be any other type of musician.
- While the musician is using the device with a instrument, it is intended that the user will interact primarily with the device itself rather than the phone/desktop application.
- The phone/desktop application will be used primarily to load preset effects or layers to the device before the musician intends to perform or play music.

Limitations

- Device must be able to be powered from a standard wall outlet.
- Device must be compatible with standard audio connections.
- Device controller must have adequate hardware resources to handle raw audio processing with minimal latency.
- Device chassis must be durable to withstand stomping for pedal and button functionality as well as wear and tear from travel.
- Device must have bluetooth connectivity to interface with phone application and USB connectivity to interface with desktop.

1.6: Project Goals and Deliverables

Given the results of our meeting with our client, we discovered they would prefer to have a hands-off style approach. Our initial meeting showed us that they had many ideas, many of them not being completely related. By using some of their ideas and combining them with our own, we were able to construct our desired project goals and some itemized deliverables.

Project Goals

Our defined goals for our project are:

- Provide a looping and sound effect device for both live musicians and music producers.
- Ensure that our device is intuitive yet powerful at the same time.
- Must be mobile and durable.
- Take away limitations of performers with use of our product.

Required Project Deliverables

The final project (sound effect device) must:

- Be able to create, edit, and stack live/old recorded segments to be "looped"
- Allow the user to create, edit, and stack sound effects on real-time inputs and recorded loop segments
- Be able to store sound effect "profiles", share sound effect profiles, and load sound effect profiles
- Be durable enough for heavy use and provide a quality feel to the device
- Have a intuitive user interface while maximizing the potential of the device
- Allow for the following inputs: XLR, L/R ¼ inch, 3.5mm auxiliary, MIDI, tuner knobs, stomp buttons/pedals, and expression pedal
- Allow for the following outputs: ¼ inch headphone, L/R ¼ inch, MIDI
- Be able to interface with the user either hands free through the device or through phone app to provide more applications

1.7: Previous Work / Literature Review

Basic Terms

Guitar Pedals

- Effect pedals usually take the form of small metal boxes which sit on the floor in front of you. These can be switched on and off using your feet. Hence, pedals. The technology contained within these pedals is designed to alter your tone in any number of ways. One of the primary types of pedals around today are effects pedals.
- Common Guitar Pedal effects would include:
    - Delay/looping: Delay is a commonly-used effect where the pedal repeats your sound at predetermined intervals after you've played it.
    - Reverb: Reverb simulates the sound of your instrument being played in a larger physical space.
    - Drive: Overdrive is an effect that 'pushes' the guitar's signal before it reaches the amplifier.
    - Modulation: There are a few main types of modulation; chorus, phase, tremolo, wah and flange. These effects are very distinct and should be used with purpose/precision.
    - Tremolo: Making your signal subtly cut in and out of volume.

Live-Looping

- Live looping is the recording and playback of a piece of music in real-time using either dedicated hardware devices, called loopers or phrase samplers, or software running on a computer with an audio interface.
- Loopers allow you to record entire passages of play, then 'loop' them back (repeat them) whilst you play something new over the top.

- This offers the ability for a single musician to create multiple layers to their live music, resulting in a sound close to that of a "full band".

Effect Stacking (Chaining)

- Guitar Pedal “Chaining” is the technique of patching two or more of these effects pedals together so that the final output is a combination of the original recording and the effects stacked on top of it. Combining pedals together can create a very unique, distinct sound and when used correctly can sound very pleasing to the ear.
- Effect stacking can also be done through the means of a software that can apply and stack effects to a loop, such as the combination of a DAW and VST

Live-looping Hardware

Boss RC-300

- The BOSS RC-300 is a top of the line Loop Station. Record, playback, and control three separate stereo tracks, each with transport-control footswitches. RC-300 also features a master expression pedal, a 3-channel mixer, dedicated volume knobs for each channel, 16 onboard effects optimized for looping, and MIDI I/O.
- Strengths:
    - The internal memory can support up to 3 hours of continuous music to the internal memory with effects added.
    - USB connectivity, XLR input, and 2 x Instrument, 1/8" (Stereo) inputs make the device functional for many types of input.
    - USB port allows for easy downloads.
- Weaknesses:
    - Lack of a visual user interface is not friendly to beginners
    - Design only allows 3 tracks to run simultaneously
    - Additional footswitches are still required for full hands-free operation
- Takeaways for our project:
    - We may have to sacrifice functionality if we want to increase ease of use and to appeal to the non-pro player

Live-Looping Software

Ableton’s Live-looping

- Ableton’s live looper is one example of the many software based loopers. This looper is basic, but standard with all the functionalities of a loop pedal.
- Features include: recording, playback, clear track, undo, pitch shift up to 3 octaves above & below, in semitone intervals, reverse, double length, and halve length
- You are also able to quantize the loop to the current BPM, or have it set the tempo depending on your signal. It can also mute the input, so the channel only

plays back the loop, not the input signal when recording with a live instrument or midi controller.

- You can run as many loopers as you like, (depending on your CPU power) and can have them on separate tracks, or on one track inside an effects rack.
- Strengths:
  - As long as your CPU can handle it, you can run as many looper tracks with as many effects on those tracks as desired
  - Pitch Shift, Double Length, Halve length, Reverse
- Takeaways for our project:
  - We will have to scale the amount of effects we that can be applied using our device in accordance to our CPU power
  - Most loopers have a reverse functionality

Effect Stacking Software

Guitar Rig (and RigKontrol)

- Guitar rig is an amp and effects modeling software package developed by native instruments. Primarily designed for electric guitar and bass, the software uses amplifier modeling to allow real-time digital signal processing in standalone and studio (VST/DXi/RTAS/AU) environments
- Guitar Rig Pro offers 54 modeled stomp boxes and effects from foot pedals to complex studio tools. These effects can be used on guitars, vocals, synths, drums, etc.
- The Guitar Rig environment is a modular system, providing capabilities for multiple amplifiers, effects pedals and rack mounted hardware. The software simulates a number of devices such as preamplifiers, cabinets and microphones.
- The system allows customization of module parameters – either through manipulation of the graphical interface, use of a MIDI controller or employment of the RigKontrol foot control pedal. Settings can be saved as presets and exported and shared with other users.
- RigKontrol is a foot-operated USB and MIDI controller that is directly compatible with with the guitar rig software. It contains an audio interface and Direct box, allowing integration with live sound environments.
- The RigKontrol device can operate Guitar Rig using its' eight switches and an expression pedal.
- Strengths:
  - Effects can be applied to any instrument
  - Modular design
  - Can be operated in many different environments
  - 54 modeled stomp boxes
  - Preset browser
- Weaknesses:
  - Quality of effects might be questionable, may not have the same authentic sound as hardware
- Takeaways for our project:

    - The modular design allows for Guitar Rig to be operated solely by the RigKontrol controller if desired.
    - The audio interface and DI (Direct Box) allows for integration with live sound environments.
    - The preset browser is also something to take note of, this would increase ease of use for beginners.

Looping and effect stacking hardware

Boomerang III Phrase Sampler

- Boomerang III is a guitar looper and effects pedal that features four loop tracks which can be played in tandem with many effects as well as unlimited stacking. Boomerang III is made for solo performers who want complex sound or for users wanting to create experimental new sounds with a wide array of effects.
- Strengths:
    - Great for live performance
    - Great synchronization
- Weaknesses:
    - Has limited memory, cannot run pre-recorded tracks
- Takeaways for our project:
    - Keep in mind who are target users are and implement functionalities based on what would appeal to those users. The Boomerang is a good example of a looper that works well for its intended audience.

Digitech Jamman Solo XT Looper

- Digitech Jamman Solo Looper is a pedal that is friendly toward the beginner guitar player. It offers two play modes for recording:
    - "Free Form" looping style, where the loop is set according to the timing of your pedal-presses. In "Free Form" style, you'll need no instruction, simply choose a location for the loop and tap the pedal to start looping! There's a color-coded LED light to tell you whether you're recording, playing or overdubbing.
    - "Auto-Quantized" looping, where you receive some assistance from the pedal to keep everything in time
      To activate the quantization feature, you simply have to dial in a tempo before you start playing. You can either do this by going into the menus and manually dialing in a BPM, or by simply pressing the "Tempo" button and then tapping the pedal in your desired beat. The rhythm guide will start playing, which you can turn up or down using the "Rhythm Level" knob. If you hold the footswitch down you can remove the backing, or you can leave it playing if you want a guide (there are nine different options if you want to change the sound). The process is pretty much the same as recording free-form, except that you get a one-bar count in before you record. If your timing is a little off on your pedal presses, the pedal will stretch or shorten it to keep everything in sync. The "Time-Stretching"

feature comes alongside the quantization. This allows for the user to edit a recorded a loop to play at a faster/slower speed.

- Strengths:
    - Undo and redo feature
    - Time stretching
    - Many options for playback and stopping
    - 2 types of play modes
- Weaknesses:
    - The onboard memory is good for 35 minutes of stereo looping, but there are many options out there with much more memory.
    - It's not ideal for live use due to the single pedal (different combinations of pedal presses and holds call up other functions like stopping and undoing which may increase chances of a miscue)
- Takeaways for our project:
    - The Auto-quantize feature and rhythm guide may be something worth looking into for our project. This feature would be great for beginner players.

Bluetooth enabled pedal boxes/effects software

Zoom MS-100BT MultiStomp Guitar Pedal with Bluetooth

- User interface and backlit LCD screen make programming intuitive and straightforward. Its stereo input jack accepts signal from passive and active guitars, as well as from line-level devices such as electronic keyboards and other effects processors, and its dual line-level output jacks enable you to record many sounds in stereo.
- The MS-100BT allows you to use up to 6 effects simultaneously. You can also chain effects together in any order you like—important because an overdriven chorus sounds quite different from a chorused overdrive.
- MS-100BT provides 50 areas of memory where you can store edited and chained multi-effects as patches (user-created presets).
- MS-100BT also provides an option to create a custom list of up to 26 patches for the MS-100BT to cycle through as you step on the foot switch—handy for live performance. This list can be reordered or erased at any time.
- The MS-100BT has a built-in chromatic tuner that supports all standard guitar tunings—even drop tunings of up to 3 semitones. When activated, the LCD immediately shows you whether the note you're playing is sharp, flat, or dead on. You can opt to either bypass the currently selected effect when tuning, or to mute the signal altogether, allowing you to tune in silence.
- Tempo can be set in real time by either tapping a knob or the foot switch, providing you with instant synchronicity.
- The MS-100BT Auto Save function ensures that every edit is automatically saved. Edited patches can be named, and can be stored in any memory area, allowing for convenient storage.
- Strengths:

    - Can use up to 6 effects simultaneously
    - LCD screen interface
    - Responsive tuning/ Many tuning features (chromatic tuning)
    - Auto-Save functionality
    - Ability to chain effects in specified order
    - Can save user-created presets
- Takeaways for our project:
    - Bluetooth connection
    - Tempo and chromatic tuning features of the MS-100BT are great teaching tools for helping users that are not familiar with these concepts
    - Certain effect combinations will be made useless if they are not able to be applied to the sound in the correct order

Hotone XTOMP Bluetooth Modeling Effects Pedal

- This stomp-box can be transformed into over 140 different classic, vintage and modern effects for guitar, bass and more. The XTOMP mini uses a mobile app (iOS and Android) via Bluetooth or a desktop app (Windows and macOS) via USB with an ever-expanding effects library to load your favorite tones into the pedal. Both apps are designed to easily manage your tones, including loading effects and firmware updates into the pedal via Bluetooth (mobile) or USB (desktop).
- To simulate effects, the XTOMP mini utilizes advanced Comprehensive Dynamic Circuit Modeling (CDCM) technology. CDCM, which processes various incoming signals dynamically, is unlike current modeling systems that use the same modeling circuit regardless of the signal. CDCM allows for larger and more complex modeling algorithms, which equal more realistic and natural tones. This results in authentic-sounding effects.
- Strengths:
    - Large model/effects library, and 50 original effects
    - Buffered bypass On/Off footswitch (100% pure analog signal path)
    - Mobile App
- Takeaways for our project:
    - Bluetooth connection
    - We may want to investigate modeling technologies like CDCM to do our effect modeling

2.0: Proposed Approach

2.1: Proposed Solutions

We narrowed down four possible solutions that would satisfy the intended user:

- Looper - a physical device that would be similar to existing looping hardware but with more features and a visualization of the looped audio in the form of a layered equalizer. The device would be able to loop 4 layers at a time with physical selector switches to switch between the layers likely in the form of foot pedals so that the musician can use the device and an instrument simultaneously. The device will also be able to save and load loops from an onboard storage system through user interface navigation using knobs and buttons. A possibility is to include an online database feature that will allow the user to upload and download loops from a user supported archive. Features include:
    - Loops playback recorded on start and stop along with ability to loop backwards
    - Ability to record multiple loops in parallel in the form of layers
    - Ability to save current loops and load previous saved or downloaded loops
- Super Tube Amp - a physical device that will behave like an ordinary amplifier but the idea is recreate the particular sounds of tube amps digitally which will be more cost effect and versatile. The device will have an input and output that will amplify the signal and apply the specific tube amp distortion selected by the user. The user will be able to select the desire distortion from an included replicated tube amp library through and onboard user interface. The user can also create their own distortion effects through a mobile or desktop application and upload them to their device or to the user supported library for other users to download and share. Features include:
    - Digitally replicates the distortion effect of a tube amp
    - The sound processing hardware combines analog tube amp and digital technology
    - Ability to create, upload, and download tube amp effects to or from an online database
- Special Effects Recommendation Software - desktop software or web application that can be given an audio sample and iterate through many combinations of distortion effects to identify possible combinations that listeners would enjoy. This will require machine learning from user input to be able to find what audio signals will sound appealing to humans. The software will also be able to analyze existing music from an artist’s discography or an audio sample and identify distortion effect combinations that would imitate the sound. Will require a online user base to increase the sample size that the software can learn from. Features include:
    - Iterates through many combinations of effects on sounds and identifies possible good combinations

    - Utilize machine learning to analyze existing sounds and music
    - Identify effect combinations to imitate a sound through an inputed audio sample or music library
- Effect Stacking - a physical device that would have many pedal connections and an audio input and output. The user will be able to select the order in which the audio signal passes through the pedals through an onboard user interface. The audio signal would then be passed to the output with all the added pedal effects in the selected order. The device will also allow the user to select between parallel and series combinations. Features include:
    - Connects to existing guitar effects pedals and can change effect levels
    - Change order or settings of pedals without physically reordering them
    - Ability to manipulate effects remotely and quickly

2.2: Assessment of Proposed Solutions

As a group, we took the Proposed Solutions from earlier in the document and decided to make a combination of almost all of them (The Tube Amp excluded). This can lead to all sorts of benefits but also possible detriments. For starters, a solution with such a broad scope makes it less focused on one particular aspect and as such may be inferior to a more focused solution in a particular area. However the strength in this design is that we can accomplish one of our main goals: “Making the process of looping/sound effect manipulation easier for the average consumer”. Having so much functionality means that as a group we must be careful in selecting hardware (and software) that will be capable of supporting so much functionality. This has been taken into consideration as mentioned in the “Risks and Challenges” Section in which such capacity concerns are addressed.

Our solution hopes to achieve singularity in being a device that not only allows the user to accomplish so much without the need of multiple devices, but is also easily usable by even inexperienced users. This itself will provide a road block later on as the design must be able to incorporate all of our ideas while still maintaining that simplicity we hope to achieve with a visual representation in the LCD screen. Through countless design drawings from our group, we all came to agree that this solution was optimal as it is making musicians lives easier, containing lots of varying functionality, and is still feasible.

2.3: High Level Block Diagram

Below is a block diagram detailing the interconnectivity of the environment our product will function within. Descriptions of the functions are below.

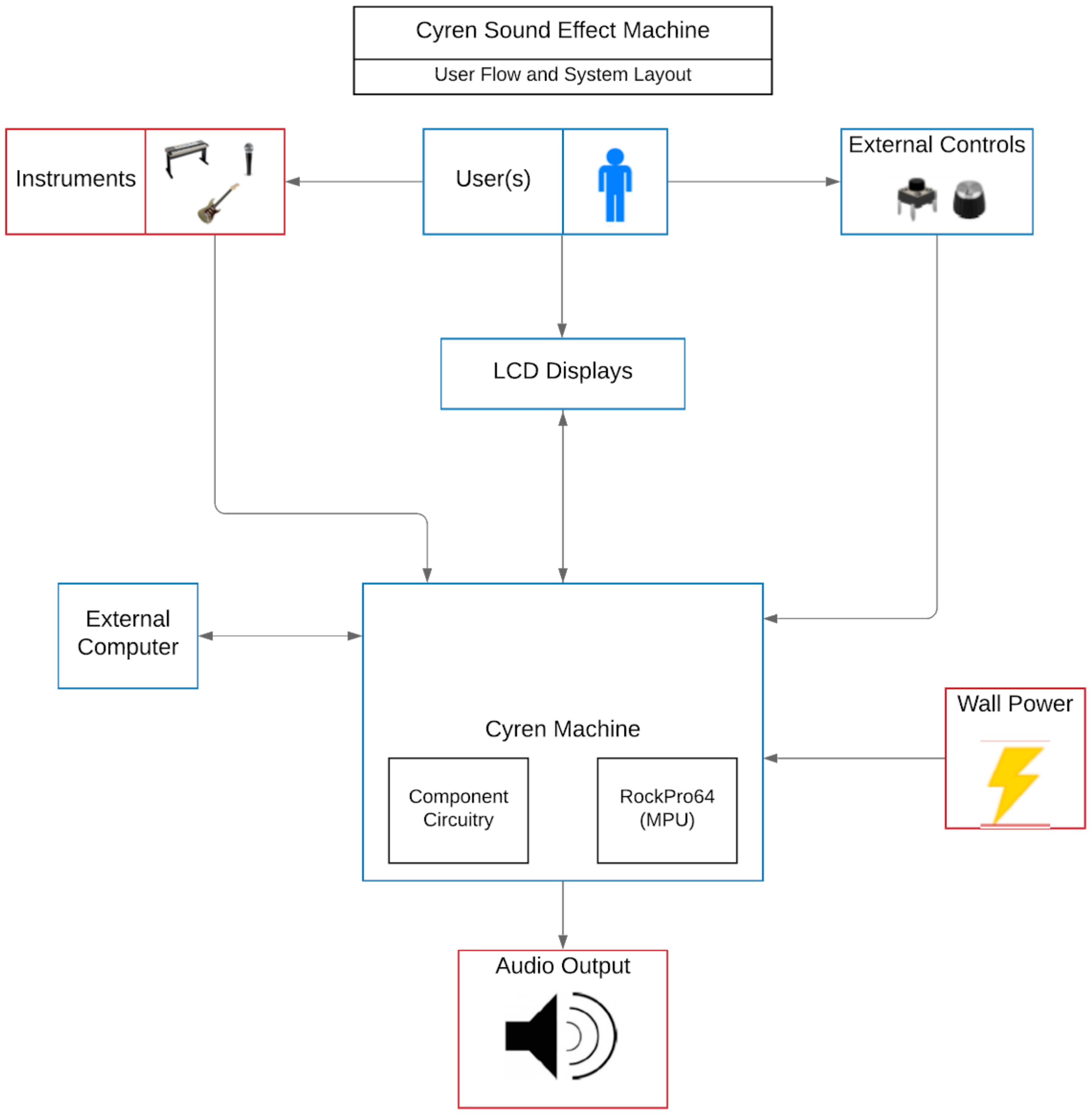


Figure 2: Block Diagram

The block diagram above is a top-level view of how our user will interact with our device. There are a few things to mention here. The *instrument* block can be interpreted as any instrument that uses a 0.25” TRS plug for music signals. This includes microphones, guitars, pianos, and possibly synthesizers. They can be plugged into the back of the machine through our input breakout boards. MIDI and XLR jacks will be implemented as well.

The user will interact with the instruments as well as the illuminated buttons and knobs on the surface of the machine. Any function that will be manipulated by the buttons and knobs will be displayed properly on a 5” LCD display, and any sound loop or sample will be displayed in real time on the right 7” LCD display. These were chosen to be readable from a standing position.

The *external components* block indicates the surface buttons, knobs, pedals and possibly switches that the user will interact with. Note, the circuitry inside of the *Cyren Machine* block will consist of wired connections for power, ground and GPIO wiring for each component. Some of our components will need peripheral circuitry such as filters, opto-isolators, and voltage dividers in order to function properly.

Lastly, the machine will communicate with a PC or other computer if the user so chooses. The unit will be powered by a prefabricated 12V, 3A power supply from wall power.

Circuit schematics will be addressed in the “Circuit Design” section.

2.4: Design Specification Overview

The design will feature the ability to save and load effect combinations and layers allowing for seamless transitions between desired sounds and recording those sounds. The design will have the capabilities of several types of music production devices. Modest size of the unit makes storing and transporting easy. In addition, it is vital that the device can interface with existing music equipment, whether that be instruments, speakers, MIDI controllers, etc. Universal inputs will ensure compatibility between devices, and benefiting seamless live performances. The design is intended to be used by a musician, alone or in a group, to interface with one or more instruments and produce music with a large and diverse selection of effects and layering options that are easy and user friendly to implement or modify while keeping all the sound processing on one device.

Functional Requirements

- Loops playback recorded on start and stop
- Ability to record multiple loops in parallel in the form of layers
- Ability to save current loops and load previous saved or downloaded loops
- Connects to existing guitar effects pedals and can change effect levels

- Able to change order or settings of pedals
- Ability to manipulate effects on the device/application quickly
- The phone/desktop application will load preset effects or layers to the device.
- Able to create, edit, and stack live/previously saved recorded segments to be "looped"
- Allow the user to create, edit, and stack sound effects on real-time inputs and recorded loop segments
- Be able to store sound effect "profiles", share sound effect profiles, and load sound effect profiles

Non-functional requirements

- Usability
    - Device will be able to be powered from a standard wall outlet.
    - Device will be compatible with standard audio connections.
    - Device controller will have adequate hardware resources to handle raw audio processing with minimal latency.
    - Device chassis will be durable to withstand stomping for pedal and button functionality as well as wear and tear from travel.
    - Device will have Bluetooth connectivity to interface with phone application and USB connectivity to interface with desktop.
    - Supplemental phone application will be available for download to enhance experience
    - LCD screens will provide user a more intuitive interface for device functions
- Security
    - Access permission for the application may only be granted by the device/application administrator account.
- Reliability
    - Device must be connected to the application via bluetooth at all times
    - If the system crashes, it will reboot to a safe startup state, losing any unsaved presets
    - Must be durable and well built in order to withstand heavy use(stomping, stage performance, travel, etc.) and provide high-quality feel
- Performance
    - Layer visualization must be displayed in real time
    - Effects levels must be displayed in real-time
    - Live-looping must be instantaneous real-time(start, stop, add, delete)
    - Playback must be in real-time with minimal latency
    - Effects must be applied within several seconds after user indication
    - Recording function must begin immediately, preceded by a 4 count "metronome"
    - Pre-saved pedal & effect configurations must be able to load within 60 seconds
    - Should be compatible with any instrument that can interface electronically

- Availability
    - System should maintain all available functions for while device connected to the application
    - System will have complete device functionality while not connected to the application
    - No maintenance period for version updates
- Scalability
    - System must be able to handle up to 4 loop layers and 20 effects simultaneously.
    - The maximum concurrent users will be the amount of instruments that are interfacing with the device along with a person controlling the device through the phone/desktop application. This will most likely be no greater than 4 users.

2.5: Technical Approach

Cyren is meant to fulfill many different needs that a musician may desire, but this is obviously a larger task to accomplish if we keep the goals vague. To avoid this, we knew we had to construct clearly itemized goals and deliverables and use them as a guide to reach our end goal. Like stated above in the document, Cyren's goal is to provide a looping/sound effect device for musicians that is both powerful and intuitive. Often times the biggest struggle with devices like these is the user's ability to understand and utilize the device's full potential. If we keep this in mind and make this our goal during development, we will be closer to achieving the desired end product. Along with being intuitive, we need to ensure that the product is durable, portable, and reliable. If this device is being used regularly by performers, being stomped on and what not, it needs to be able to withstand abuse while performing to its expected standards. By defining these goals, we then were able to move to planning.

Using the deliverables, it helped us draft up a project plan to facilitate our project. First we had to as a group understand what type of technologies would go into the product and how to get from start to finish. We split the group into the areas that they strive best. Example being, our electrical engineering undergrad is going to focus on wiring I/O devices together with the microprocessor while designing an optimal enclosure. Upon doing this, we were better able to become *experts* in the areas we were assigned, allowing us to perform proof of concepts (POC) to find the best technologies to use for our product. Once a POC was finished, the corresponding teammate would demonstrate their findings to the team and make an educated decision on what technology is best used for the task in front of them. Once demonstrated and approved by the team, we then move onto the next task's research. It is important to note, that we do *not* start implementing the new found technologies until all research is completed, by doing this we will be more likely to find compatibility issues early on rather than when they arise from implementation.

2.6: Testing Requirements Considerations

Validation and Acceptance Test

First to address the testing we must define what functionality the user expects in the product. The user expects to be able to:

1. Edit their music with many different types of sound effects
2. Mix together sound effects
3. Record music
4. Replay/Loop
5. Stack Loops on top of each other
6. Use pedal to control sound effects
7. Visualize the sound effects through a UI both in the software and on the pedal itself
8. Be compatible with standard I/Os for musical instruments
9. Be portable

Test Plan

Now we have to test each of these requirements to make sure the user is able to do these things. We will test the above requirements with the following tests:

1. Use the product in person with standard musical instruments owned by project members and confirm success and variety of sound effects.
2. Experiment with the product's sound effect mixing and assure that there aren't bugs in mixing types. Also created automated tests that run through combinations to ensure there are no bugs.
3. Test the limits of the recording process, i.e. keep recording for as long as possible, create test recordings to see how long of a recording can be stored, etc.
4. Perform replays of the recordings and listen to the audio and analyze the audio to ensure it matches the recording.
5. Once again we need to prove that we can stack loops to create diverse sounds but then create automated tests of stacking multiple loops and making other sound effects to ensure there are no hidden bugs.
6. Since almost all group members are musicians (and also have contacts that are professional musicians) we will want to use them as test subjects and record their feedback to ensure that the pedal is effective while playing music.
7. Verify the working UI corresponds correctly to each physical configuration on the pedal to ensure no bugs.
8. Test multiple standard I/Os in the musical market right now on the device and ensure proper compatibility.
9. Ensure that a single user could relatively easily transport the device without excessive assistance

Design Testing and Implementation

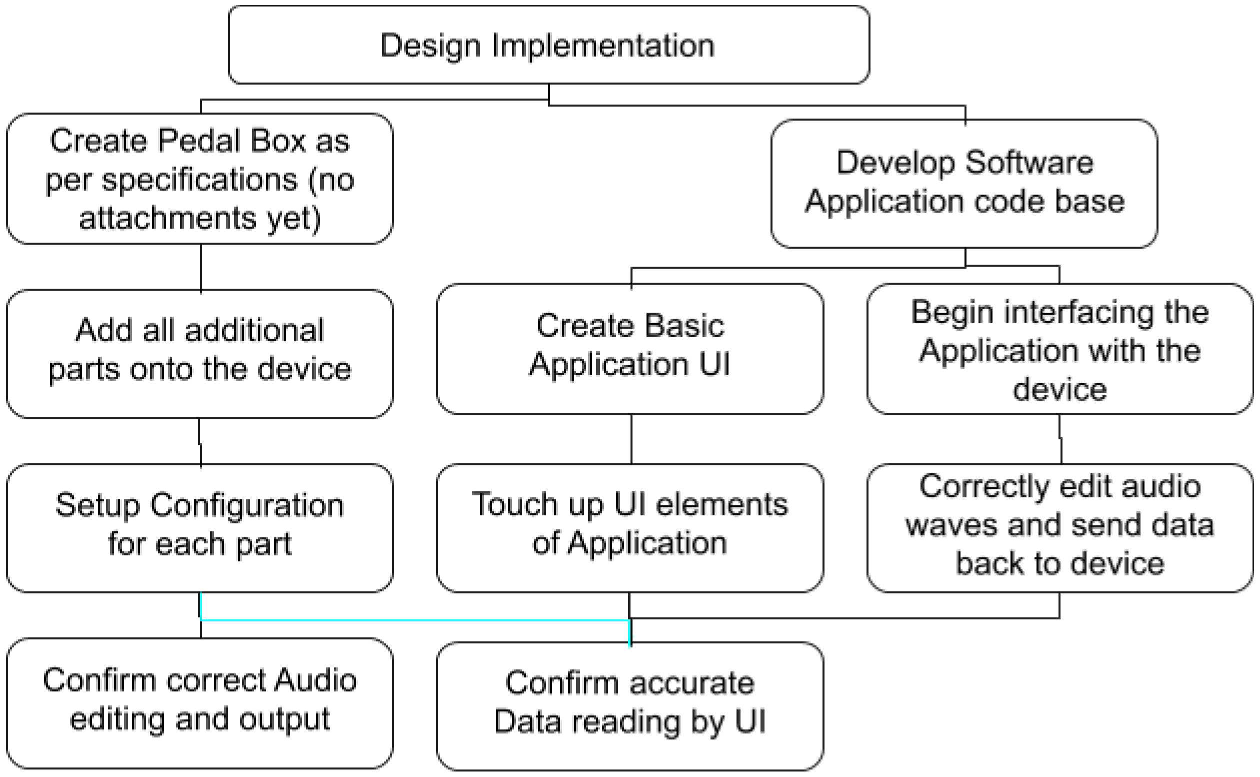

Design Implementation
Create Pedal Box as per specifications (no attachments yet)
Develop Software Application code base
Add all additional parts onto the device
Create Basic Application UI
Begin interfacing the Application with the device
Setup Configuration for each part
Touch up UI elements of Application
Correctly edit audio waves and send data back to device
Confirm correct Audio editing and output
Confirm accurate Data reading by UI

Figure 3: Design Testing Flowchart

- Create Pedal Box is a simple task point which involves constructing the basic box product with pedals attached
- Develop Software Application Code Base is to be completed at the same time as the Create Pedal Box and is quite self explanatory, things that are expected to be complete include: central c file, helper c files, and some libraries that are be required
- Add all additional parts to device, this includes adding the working knobs, LCD screens, and more
- Create Basic Application UI. This step involves having a simple console-based user interface that connects to the code base
- Begin interfacing the Application with the device. This step is mostly just testing connections to our pedal box and checking for correct reading of input and GPIO pins
- Setup Configuration for each part. This begins the new stage of the design which is now setting up each part to specifically work as the user will use them, for example the user knob being able to turn constantly without stopping versus having it begin and end at certain points.
- Touch up UI elements of Application is finally creating a true UI that displays on the LCD screen of the hardware system and looks nice to the user
- Correctly edit audio waves and send data back to device. This stage is the back-end portion of Touch up UI elements of application in which the

programmers are using the input being read into by the hardware device and making correct changes to it where methods could be named “loop()” or “replay()” and then are called by the UI later

- Confirm correct audio editing and input. Since the audio should be getting edited by this point, we want to confirm that this input is being read correctly by the device and is being changed the way it has been programmed to.
- Confirm accurate data reading by UI. Finally, as the creators we will be checking that the UI matches the input and sound editing being done by the system.

Functional Testing

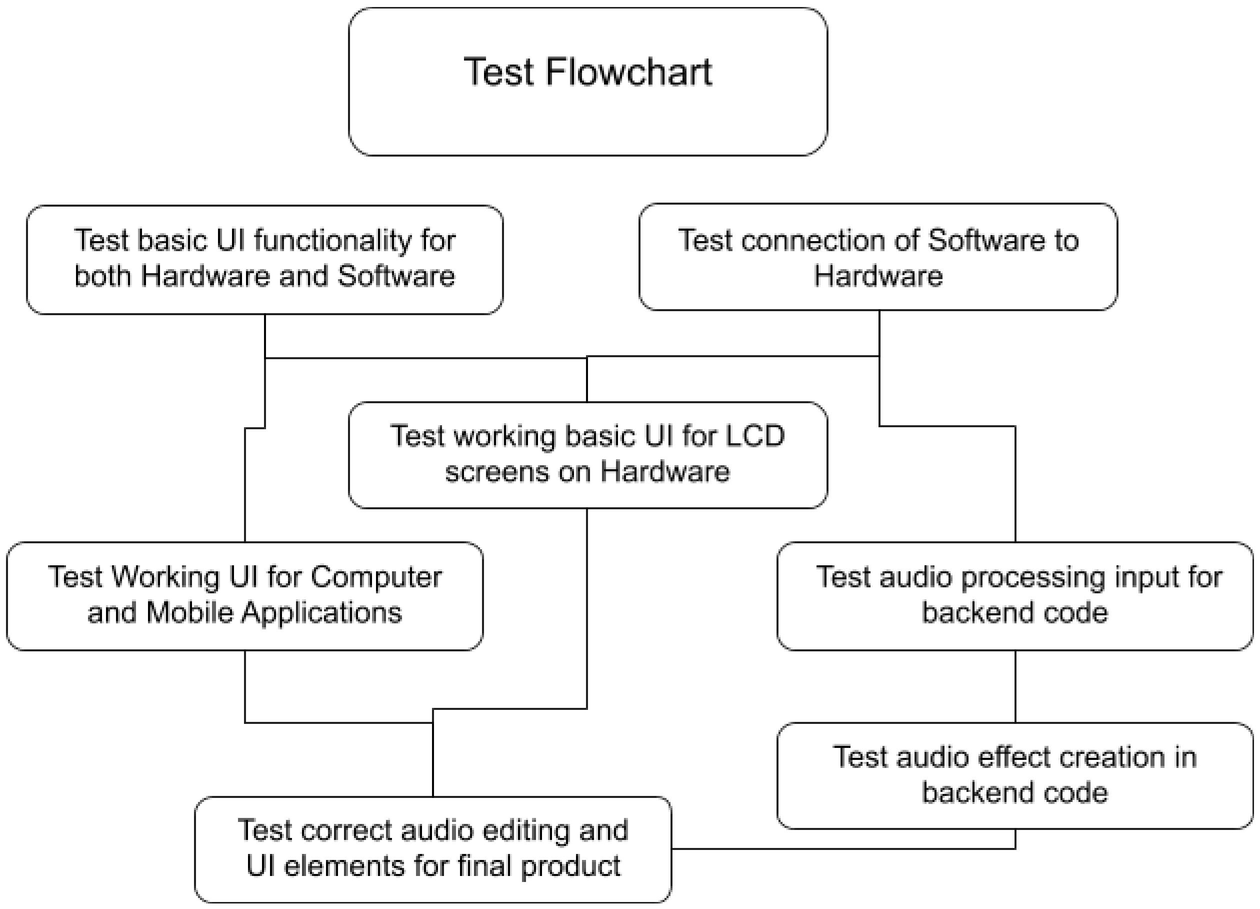

Test Flowchart
Test basic UI functionality for both Hardware and Software
Test connection of Software to Hardware
Test working basic UI for LCD screens on Hardware
Test Working UI for Computer and Mobile Applications
Test audio processing input for backend code
Test audio effect creation in backend code
Test correct audio editing and UI elements for final product

Figure 4: Functional Testing Flowchart

This flowchart shows the order in which sections of the development process is tested. The specifics of each point in this flowchart are as follows:

- Test Basic UI functionality for both Hardware in Software: This can seem rather vague but it’s more simple that it seems. This simply means we are testing to ensure that we have a code base, have a basic hardware system, and that these 2 things aren’t broken in any obvious ways and perform their most basic functionalities.

- Test connection of Hardware to Software: This step is as simple as confirming input being read by the software from the hardware (as reading input is part of the basic functionality of the hardware)
- Test working basic UI for LCD screens for Hardware: The naming of this step may be misleading, the concept is rather that there is some sort of communication with the code base to the LCD screen. This is as simple as just displaying basic Hello World text on the screen to ensure it's working properly.
- Test Working UI for Computer and Mobile Applications: It is in this step that we are testing the basic UI developed in previous steps for both Computer and Mobile apps, since we discussed that we would have both available to the user so that the main functionality could still be available even on the go.
- Test audio processing input for backend code: This step is straightforward, we are simply verifying that the code is able to accept input correctly from the hardware device and store in within the program.
- Test audio effect creation in backend code: Once the audio input is confirmed to be processed correctly, it's then extremely important that we are testing the main functionality of the software. This is the most important part of the project so it is equally important that it work correctly and that the user is able to create sound effects and edits sound.
- Test correct audio editing and UI elements for final product: This is just a one last run through test to make sure that our product meets the level of quality expected of it and it's competitors.

Non-Functional Testing

```
#include libraries (including tiny test for assertions)

int main(){

        initialize_test_variables
        test_audio_processing_speed()
        test_maximum_audio_limits()
        test_design_aspects_of_system()
        test_ease_of_system_use()
        test_max_number_of_users_at_once()
        test_musical_instrument_flexibility()
        test_saved_settings()
        test_hardware_durability()

}
```

These pseudocode methods represent what kinds of non-functional tests we are performing. More will be explained in the following list but be aware not all of these tests are done in code, some will be done in the real world, it just helps to represent them in pseudocode visually.

- test_audio_processing_speed(): This test is designed for discovering the speed of our program in terms of it's audio processing/manipulating abilities. This test is done in code using some C libraries that have methods for measuring runtimes. Some aspects of the audio processing that are tested are: audio transfer speed, audio manipulation speed, and GPIO pin manipulation speed.
- test_maximum_audio_limits(): This is one of those tests that is done in the real world and not actually in code. Our process is to raise the volume to its maximum that is possible and check that we are able to maintain the quality. The biggest challenge for this test is to find a location where performing this test isn't causing distress for others nearby.
- test_design_aspects_of_system(): This test is partially done in code and partially in the real world. As the creators of the project, we are performing a review of our design along with our faculty to determine the validity of our design and then we input the scores of each review into our tests and receive a report of generally how successful we are.
- test_ease_of_system_use(): This test is done the exact same way the previous test above is done except instead it is sample users that play instruments giving the feedback on how easy they think the system is to use. Then the data is put into the test in the code and we receive a report of how easy our system is to use.
- test_max_number_of_users_at_once(): This test is rather simple. We are creating a bunch of additional users (most likely on mobile) and just seeing how many the system can take at a time without having issues so we can establish a limit on the number of users we will be allowing to access the system.
- test_musical_instrument_flexibility(): This test is ensuring that our device is compatible with all sorts of instruments. We are doing this by making a checklist of every instrument we can think of that could realistically work with an electric audio editing device and then seeing which ones are compatible with the device and then we are putting this data into the test and receiving another report on our success or failure.
- test_saved_settings(): This may be the simplest test of them all as we are simply confirming that the settings that are saved by the user will still be available to them after the device is powered down.
- test_hardware_durability(): The arguably least-important (but my favorite) test is to test the durability, In this test we are simply trying to be rather harsh on the machine physically to see what kind of damage it can sustain without breaking. We have multiple copies of the product at this point so it isn't catastrophic if it breaks.

2.7: Possible Risks and Risk Management

Below is a risk assessment of the potential challenges we will come across as we are developing our design. The biggest hurdle that we will foreseeably have to address is the capability of the controller. The onboard controller must have the adequate hardware resources to handle multithreaded sound process and raw audio manipulation. This will require a powerful processor and ample RAM. We hope to mitigate this problem with efficient software and intelligent design.

Risk / Challenge	Risk Description	Impact	Occurence Probability
RAM Usage and Capability	We are currently looking at a Raspberry Pi as the primary controller for our device. The highest model is the 3B+, which has 1GB of RAM. Since we are planning on having four layers of effects, we will be doing four times worth of audio processing. A problem could arise where the amount of RAM is too low to handle this kind of load. It is currently not possible to add more RAM to the rPi.	High	High
Having Adequate Processing Power	Along with the RAM usage, the audio processing also puts load on the Raspberry Pi's processor. The Raspberry 3B+ touts a Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz processor. It could be the case that once we test the software component, we find that the processor is not enough to handle what we need it to.	High	Low
System Versatility with Simple User Interface	We want our device to be able to do many different tasks and act as a "One Stop Shop" for the musician user. However, we do not want a user interface / experience that is confusing or slow to use. This creates the challenge of providing an interface that is easy to use on stage, but that presents the user with a powerful, diverse toolset.	Medium	Medium
Lack of Hardware Knowledge	The realm of audio processing hardware is new to our group members. On top of that, we only have one Electrical Engineer on our team, who will carry a lot of the weight of the hardware implementation. This presents our entire group with the challenge of researching and selecting hardware that properly meets our needs.	Low	High
Cost of Materials / Budget	As this project is open to creative decisions, the budget for the project will likely shift and change throughout development. This fact combined with the hardware knowledge risk means we will only know estimated cost once we've decided on certain components that will be needed for the device to function. The cost of the project is both a risk and a challenge.	Medium	Low
Interfacing through GPIO	An issue that we have come across, and are working through now, is interfacing the software with the hardware	High	High

pins	through the use of the GPIO pins on the RockPro64 controller. This is primarily caused by the RockPro64's pins being set up differently from a Raspberry Pi's, while almost all resources online for interfacing with GPIO pins are for the Raspberry Pi. As this is one of the largest roadblocks for our team right now, it has become our primary focus until solved.		
Fitting Electrical Component	As we are finalizing components, we may find that we will require a significant amount of electrical components. With that, we run the risk of having to modify our encasement design and the layout of our physical interface to be able to fit all of our necessary components.	High	Medium

Table 1: Risk Assessment

2.8: Standards

Electrical Safety - IEEE NESC HBK-2017 - 2017 NESC(R) Handbook, Premier Edition

- Standard Details: Gives users insight into what lies behind the NESC's rules and how to apply them. The Handbook was developed for use at many levels in the electric and communication industries, including those involved in system design, construction, maintenance, inspection, standards development and worker training.

Software Development - IEEE 12207-1996 - ISO/IEC 12207, International Standard - Information Technology - Software Life Cycle Processes

- Standard Details: Provides a common framework for developing and managing software. IEEE/EIA 12207.0 consists of the clarifications, additions, and changes accepted by the Institute of Electrical and Electronics Engineers (IEEE) and the Electronic Industries Alliance (EIA) as formulated by a joint project of the two organizations. IEEE/EIA 12207.0 contains concepts and guidelines to foster better understanding and application of the standard. Thus this standard provides industry a basis for software practices that would be usable for both national and international business.

User Documentation - IEEE 26511-2012 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Requirements for managers of user documentation

- Standard Details: ISO/IEC/IEEE 26511:2012 specifies procedures for managing user documentation throughout the software life cycle. It applies to people or organizations producing suites of documentation, to those undertaking a single documentation project, and to documentation produced internally, as well as to documentation contracted to outside service organizations. It provides an overview of the software documentation and information management processes, and also presents aspects of portfolio planning and content management that user documentation managers apply. It covers management activities in starting a project, including setting up procedures and specifications, establishing infrastructure, and building a team. It includes examples of roles needed on a user documentation team. It addresses measurements and estimates needed for management control, and the use of supporting processes such as change management, schedule and cost control, resource management, and quality management and process improvement. It includes requirements for key documents produced for user documentation management, including documentation plans and documentation management plans.

Data Transmission - IEEE 599-1985 - IEEE Standard Definitions of Power Systems Data Transmission and Related Channel Terminology

- Standard Details: This standard includes definitions of those terms deemed necessary of performing or understanding measurements on channels used for data communications. terms that are applicable to audio interfaces of wireline, leased lines, and multi-channel equipments and baseband interfaces of multichannel equipment.

Bluetooth - IEEE 802.15.1-2002 - IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN - Specific Requirements - Part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)

- Standard Details: This IEEE Standards product is part of the 802 family on LAN/MAN. Abstract: The lower transport layers [(Logical Link Control and Adaptation Protocol (L2CAP), Link Manager Protocol (LMP), baseband, and radio] of the Bluetooth™ wireless technology are defined.

3.0: Statement of Work

3.1: Proposed Design

A combination of the following already existing music production tools:

Looper	Effect Stacking Device
• Loops playback recorded on start and stop along with ability to loop backwards • Ability to record multiple loops in parallel in the form of layers • Ability to save current loops and load previous saved or downloaded loo	• Connects to existing guitar effects pedals and can change effect levels • Change order or settings of pedals without physically reordering them • Ability to manipulate effects remotely and quickly

Table 2: Music Equipment Features

In our design, we took the properties of a looper and an effect stacking device and combined them into one device along with a supplemental application. This application will add many more functionalities to our product. The app will be able to save and load layers to the looper, control pedals/effects, and visualize instrument and effects layers. Our solution hopes to allow the user to perform all of these functionalities without the need of multiple devices, but is also easily usable by even inexperienced users because of the simplicity of the visual representation of the LCD screen.

Our proposed design consists of an enclosure that will be modeled in Autodesk Inventor 2019. For prototyping purposes the initial enclosure will be made of wood. The enclosure can be seen in following picture. The finished model seen, is precisely measured to accompany all components.

Note: For our prototyping phase there will be no back panel, contrary to the image below.

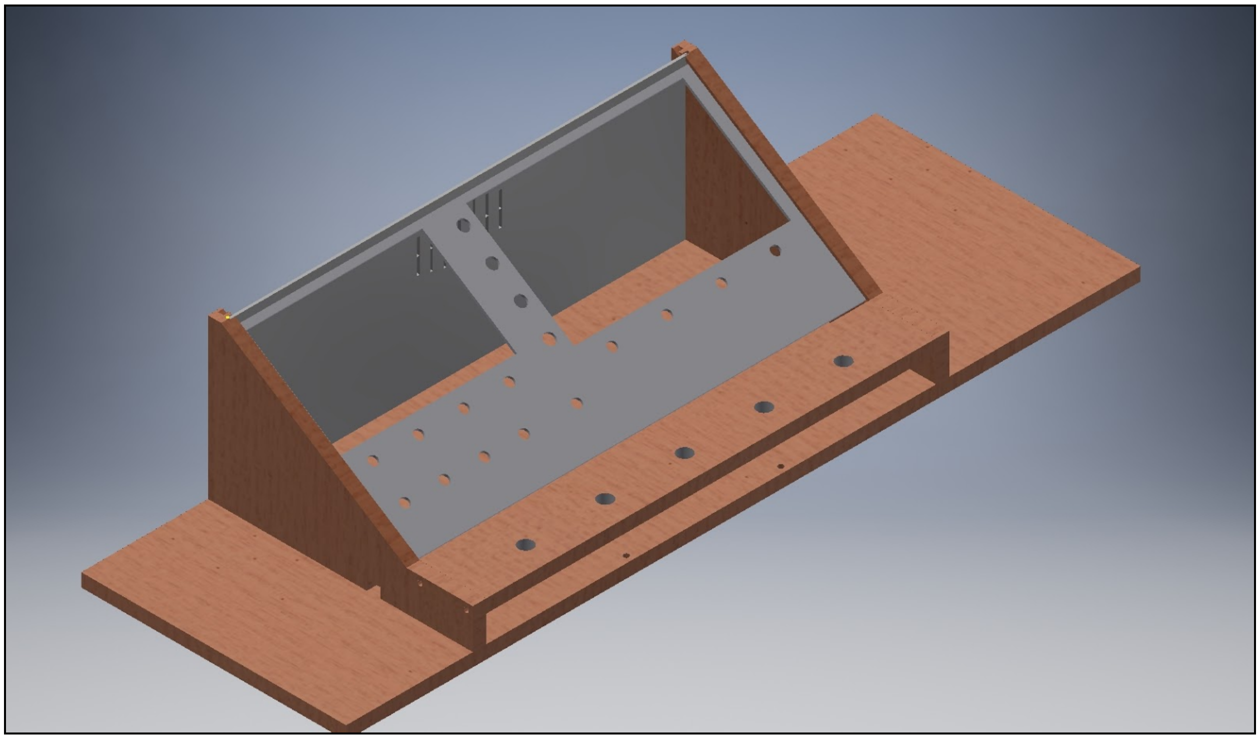


Figure 5: Pre-prototype Enclosure

Note that this enclosure is a rough prototype that functions mainly to implement the pedal functions as well as the LCD displays. This enclosure will be created in AutoCAD and milled into wooden boards.

For circuit design most components will be wired in parallel to a GPIO pin on the microprocessor. The software will then continuously read the pins to see if input is detected, and then carry out a function. The main issue at the moment is power consumption. Further research is needed to decide if a larger power supply will be needed to control the unit.

Also, further discussion is needed for the menu/GUI on the LCD screens. The current solution is to have an onslaught of menus that can be navigated through with knobs and buttons. Each knob, button, and stomp switch can be programmed through the menus. The main focus for prototyping is to have a structured unit with functioning input and output ports to the RockPro64.

As of now, the only implementation that has been done for our design is basic GUI testing. We have also ordered and received several components for our pedalbox device. These components consist of the following:

- Rotary encoders
- Stomp switches
- ADS1115 Adafruit ADC
- InGenius ¼" Audio Input Jack
- RockPro64 Microprocessors
- MIDI Input/Output Jacks
- 5" LCD Display (touch-capable)
- 7" LCD Display (touch-capable)
- Illuminated Pushbuttons
- Expression Pedal
- Secondary Pedals

The stomp switches and ADC have been implemented in lab with oscilloscopes and an Arduino code. The proof of concept for these components were to simply learn how they will work and interact with a simple MPU. A regular guitar signal was implemented through the InGenius device, into the ADC, and into an Arduino for processing. The digital signal was then graphed to prove it was working.

After further research into our RockPro64 microprocessing board, we can eliminate the use of our peripheral ADC breakout boards. The MPU board has a 4-channel ADC built in, which makes it easier to implement the InGenius analog inputs as well as the microphone inputs.

The plan for future implementation is to run a code with the RockPro64 and the rest of the system to create a simple input and output. On a grander scale, 5 stomp switches, 3 analog inputs, 8 rotary encoders, and 4 buttons will be fed into the MPU. After this point, the components will be able to be placed and fastened into the enclosure.

Further progress will implement foot pedals into the design, likely through the GPIO.

3.2: Design Architecture

For our design architecture, we have chosen a Data Flow Software Architecture.
Why we chose a Data Flow architecture:

- This architecture is used when input data to be transformed into output data through a series of computational manipulative components.
- In data flow architecture, each filter will work independently and is designed to take data input of a certain form and produces data output to the next filter of a specified form. The filters don't require any knowledge of the working of neighboring filters. This will be perfect for our design. We need the data transformation to be modular and able to service the different combinations of transformations on the data.

Data Flow Diagram

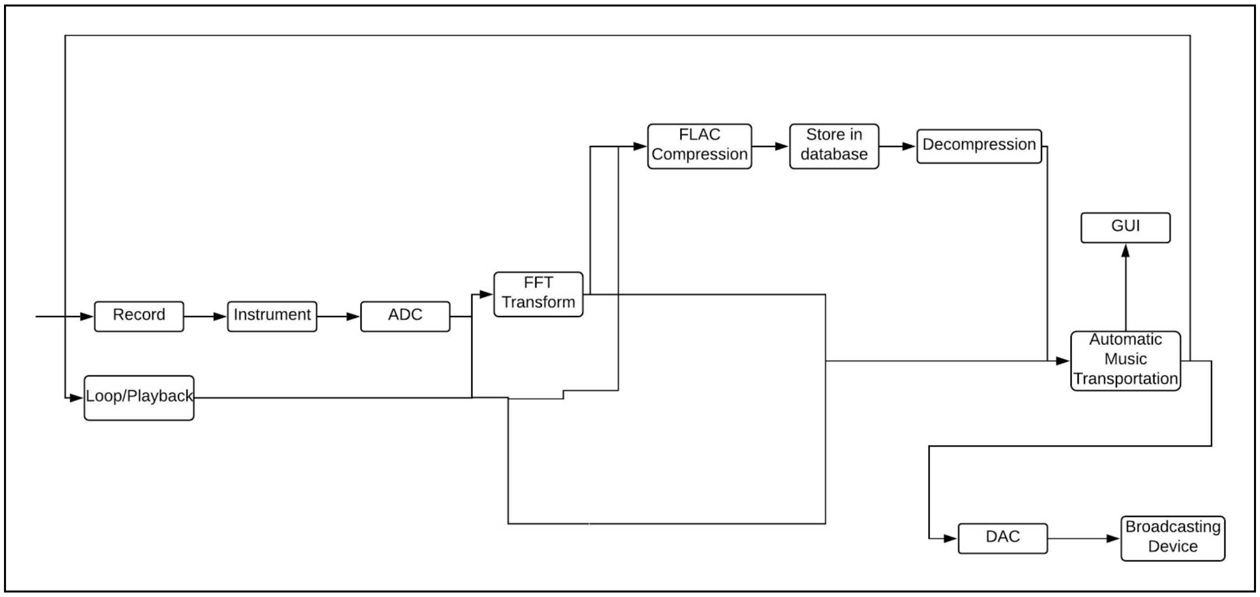


Figure 6: Data Flow Diagram

The figure represents a pipe-and-filter architecture. It uses both pipes (arrows) to transmit the data and filter components (boxes) to modify the data.

To start the data flow, the user will do one of the following simultaneously:

- Record a loop
- Playback a loop

Once the desired option has been selected, the data will be able to do any combination of the following:

- FFT Transform (Add effect(s), adjust effect(s), etc.)
- FLAC Compression (Saving layers, presets, etc.)

After the appropriate operations, the signal will be interpreted via automatic music transportation into midi notes to be displayed on the user interface. Adjustments of effects or additions on effects will also be displayed. From here, the sound will be output through a DAC and transmitted to a broadcasting device.

Use Case Software Block Diagram

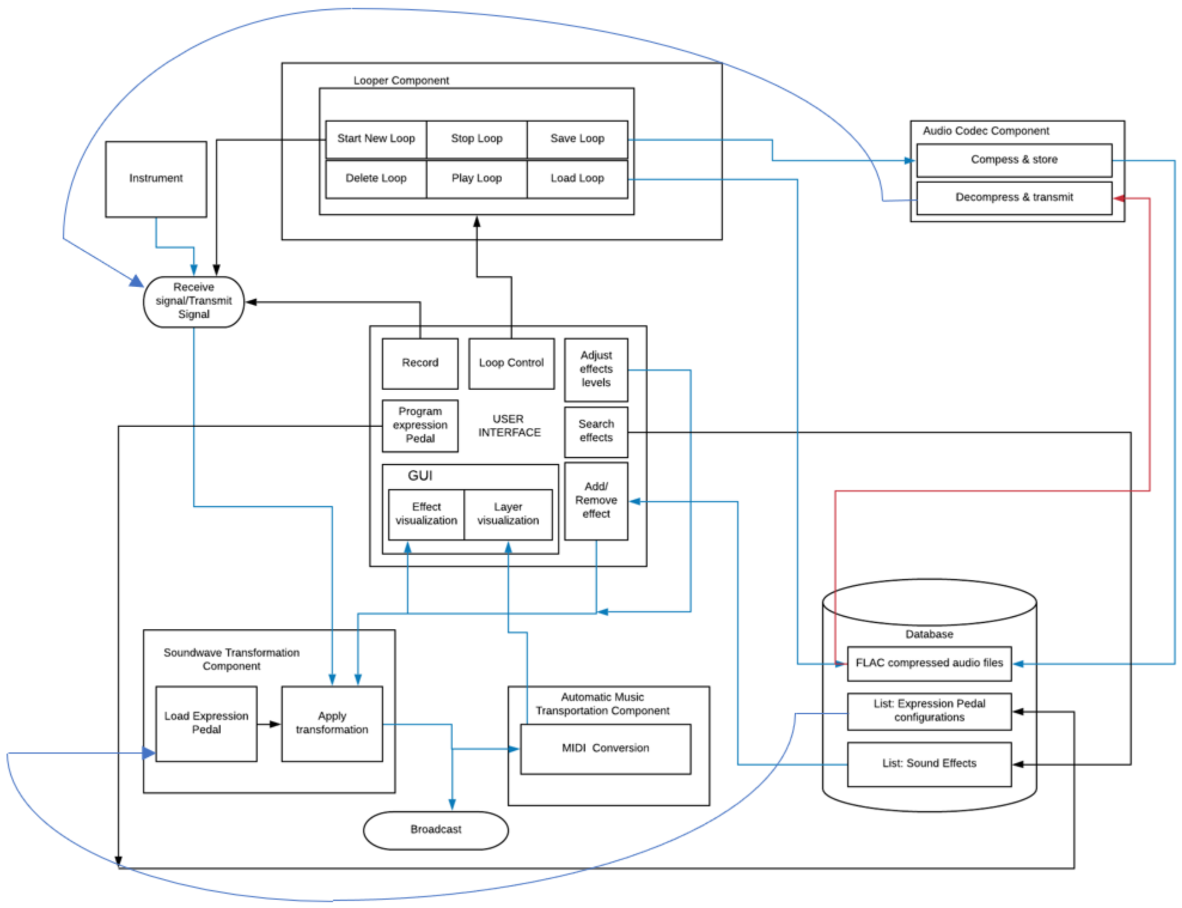


Figure 7: Use Case Software Block Diagram

As an intermediary step before designing our final software block diagram, we designed a Use Case Software Block Diagram. This diagram details all of the requirements for our design through grouping use cases to components and mapping dependencies between components.

- Use cases are depicted in rectangles(excluding the database) and implicit functions are represented in bubbles (Broadcast, Receive signal/Transmit Signal)
- Components are determined through groupings of functionally similar operations
- Edges represent dependencies between components/use cases
    - Blue edges represent transmission of data
    - Black edges indicate an action with no data transfer

- ○ The single red edge indicates a dependency that is only reachable from a directly-connected blue edge (e.g. the red edge can only be traveled in the case that the load loop operation is performed)

Circuit Design

The design of the internal circuits can be seen below. The MPU will be able to power all of our components, in parallel, via GPIO pins. This includes stomp switches, rotary encoders, buttons, LCDs, and lastly pedals. For what has been completed, descriptions will be attached to the circuit diagrams below, respectively.

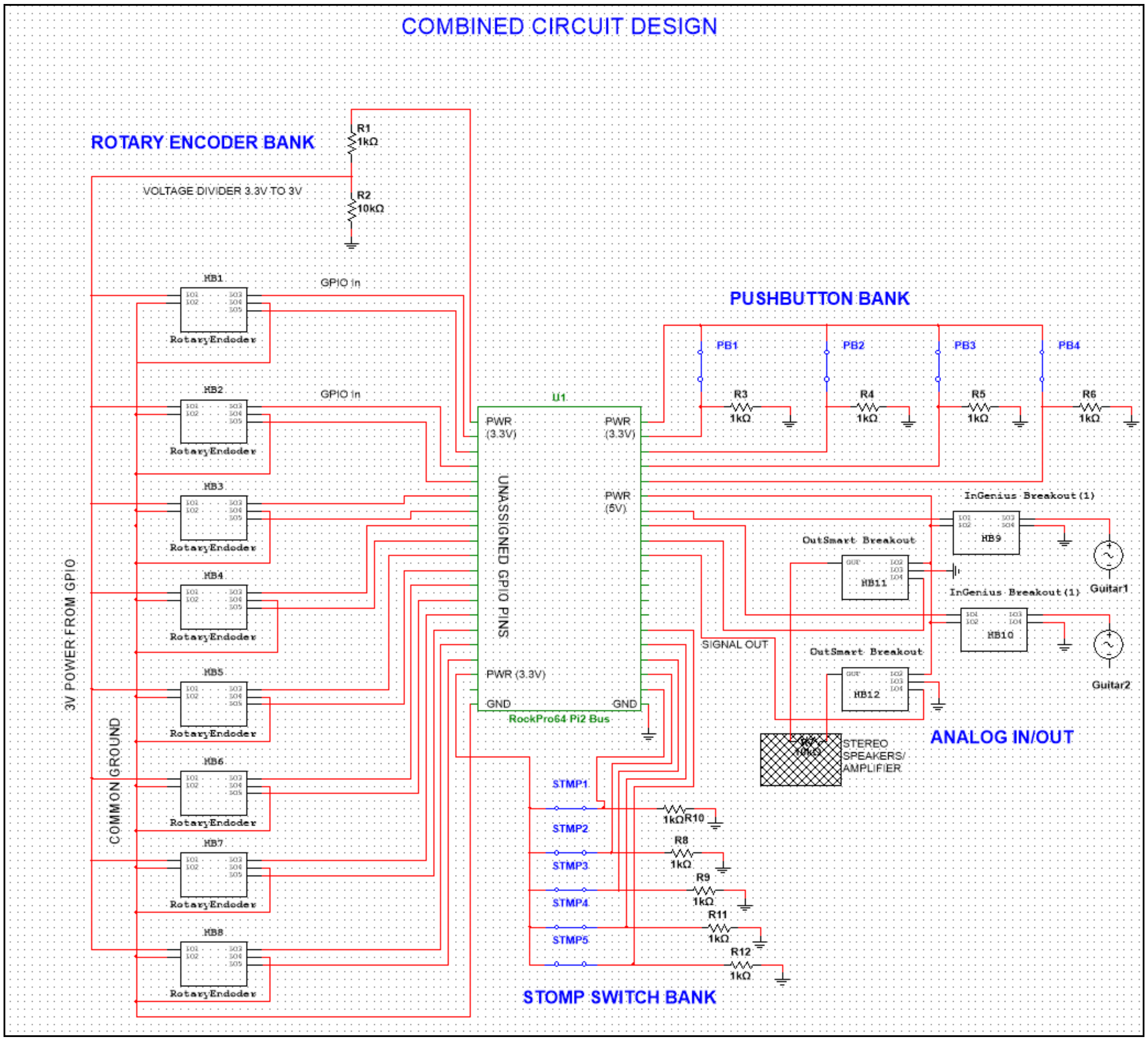


Figure 8: Block Diagram of Full Circuit

The left half of the diagram is for the rotary encoders. Each encoder will use two GPIO pins, a supply voltage of 3V, and a ground connection. Between the GPIO 3.3V pin is a voltage divider, calculated to drop the voltage to 3V. This makes the input voltage handleable. The quadrature code function of rotary encoders is currently being experimented with for proper knob operation.

The MPU seen here is a rough estimate of how we will use 40 GPIO pins for our components. The following diagrams are breakdowns of the peripheral circuits seen above. All assigned GPIO pins in the diagrams are subject to change.

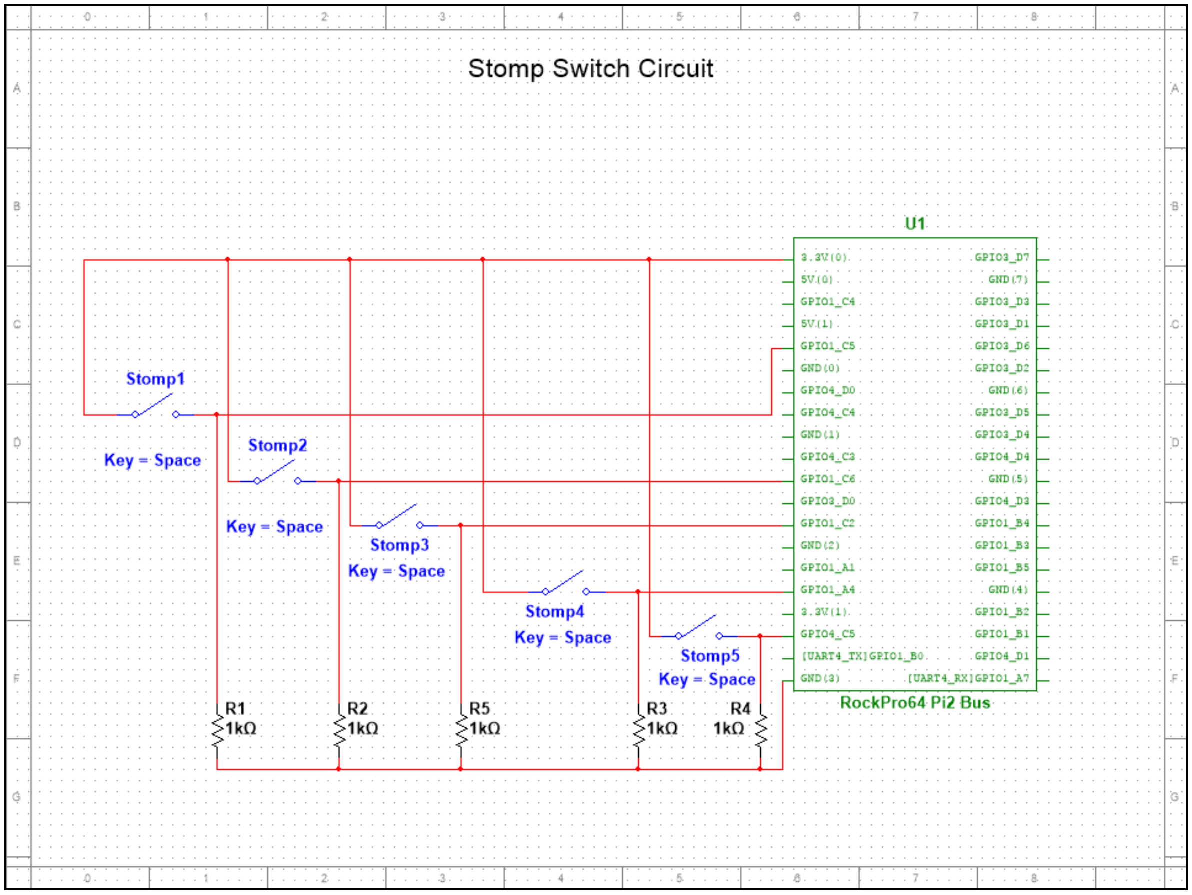


Figure 9: Stomp Switch Circuit

The stomp circuit is very simple. A readable voltage (3.3V for the RockPro64) will be fed through each switch. The 1kΩ resistors function as pull-down resistors, forcing the GPIO to read low when there is no voltage. The signal will be handled by code. Certain combinations of active stomp switches will dictate how effects are used or not used.

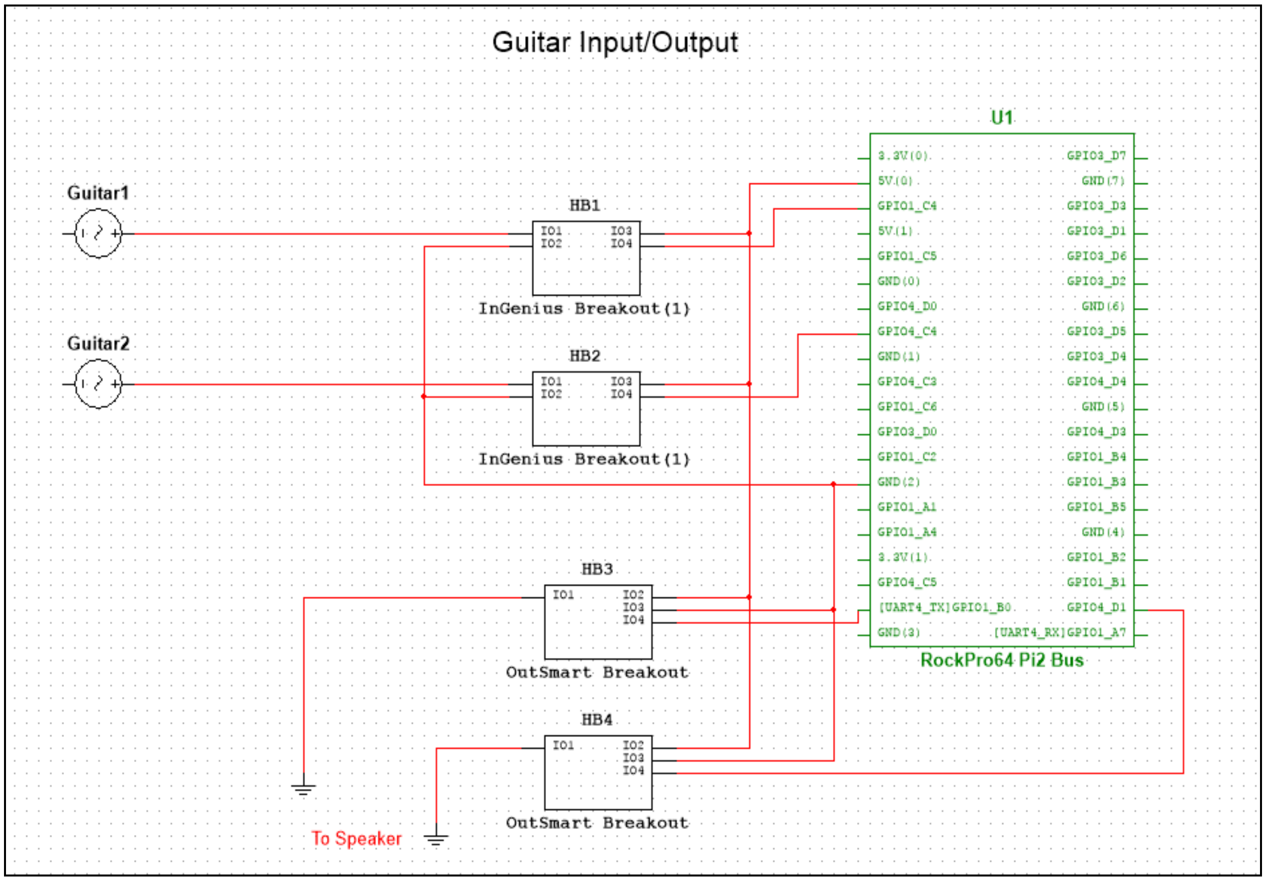


Figure 10: Input/Output Circuits for 0.25" TRS Plug

The analog signal input and output circuits make use of the InGenius and OutSmart breakout boards. These boards were chosen from SparkFun because of the CMRR rating circuits contained within the board. Essentially, the guitar input will be cleaned up by providing 5V to the boards.

The InGenius has a +/- voltage source option for unbalanced signals. As of right now it is assumed that the guitar signal will be balanced, so only +5V will be used to power the circuit. Along with this, research concludes that there is no need for peripheral circuits for input/output. The signals are ADC/DAC ready.

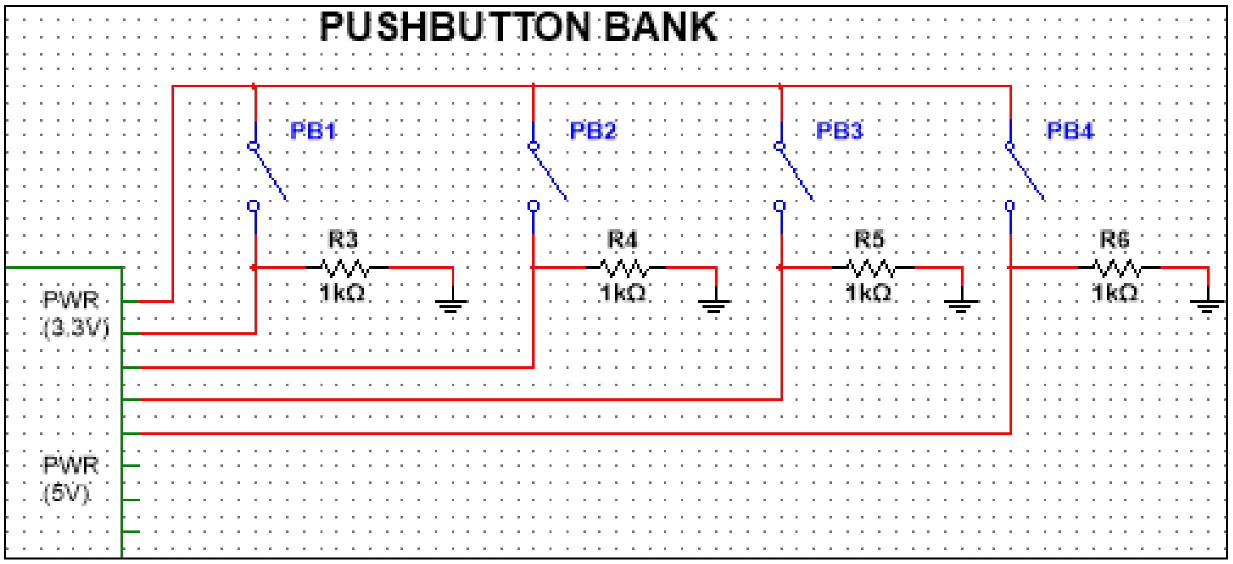


Figure 11: Pushbutton Circuit Diagram

The push button bank will primarily be used for navigating the right-hand 7” LCD panel. This panel displays the recorded layers in existence, so the buttons can be used to select specific layers.

The buttons will work with a 3.3V supply pin from the GPIO, and similar to the stomp switches, a pulldown resistor will need to be required. More research on these buttons will determine if other circuits will need to be included with the buttons.

Circuit Design Notes

There are a handful of remaining components that will need to be studied before implementing. These include: MIDI in/out, XLR in, expression pedal, and secondary pedals. Upon implementing these components it may be useful to find a GPIO extension, or a more efficient way to allocate space for GPIO wiring. There are approximately 6 pins left on the Pi2 bus.

3.3: User Interface

Physical Interface

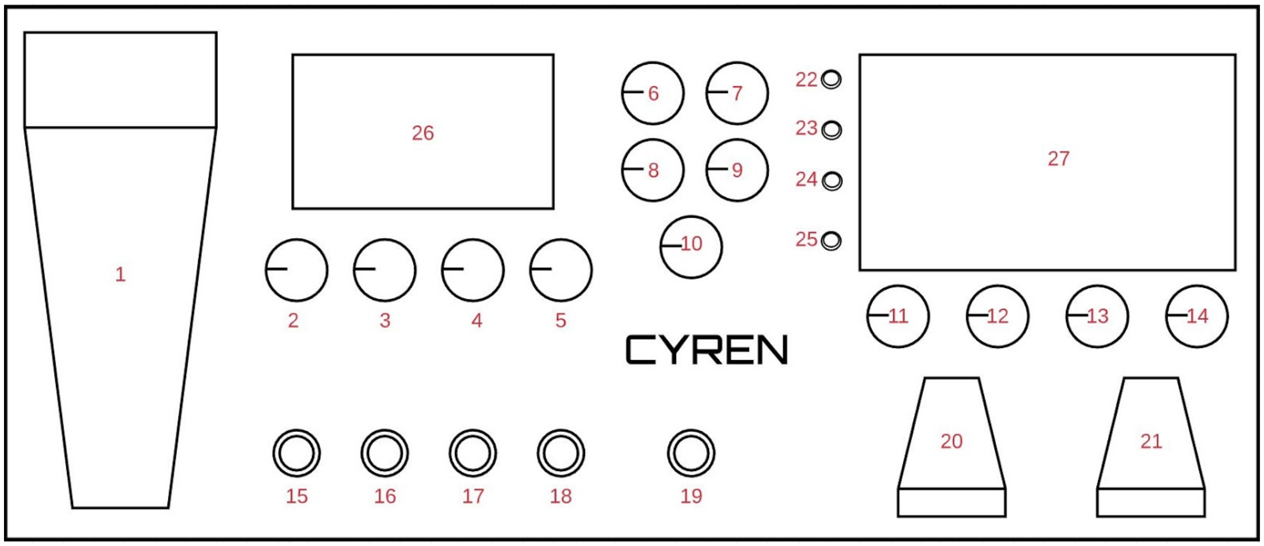


Figure 12: Design Diagram

Physical Interface Function Table

#	Name	Motion / States	Function
1	Expression Pedal	Rocking motion	On certain sound effects, this will adjust the effect depending on its tilt.
2	FX Control Knob 1	Non-stop rotation	Changes the level of an aspect of the current effect.
3	FX Control Knob 2	Non-stop rotation	Changes the level of an aspect of the current effect.
4	FX Control Knob 3	Non-stop rotation	Changes the level of an aspect of the current effect.
5	FX Control Knob 4	Non-stop rotation	Changes the level of an aspect of the current effect.
6	Aux Volume Knob	Non-stop rotation	Changes the volume for the aux input.
7	MIDI Volume Knob	Non-stop rotation	Changes the volume for the MIDI input.

8	Mic Volume Knob	Non-stop rotation	Changes the volume for the mic input.
9	IN Volume Knob	Non-stop rotation	Changes the volume for the IN input.
10	Master Volume Knob	Non-stop rotation	Changes the volume for the entire device.
11	Layer 1 Level Knob	Non-stop rotation	Adjusts the volume of layer 1 in the mix.
12	Layer 2 Level Knob	Non-stop rotation	Adjusts the volume of layer 2 in the mix.
13	Layer 3 Level Knob	Non-stop rotation	Adjusts the volume of layer 3 in the mix.
14	Layer 4 Level Knob	Non-stop rotation	Adjusts the volume of layer 4 in the mix.
15	FX Stomp Switch 1	Click Toggle On/Off	Activates the effect saved to effect slot 1.
16	FX Stomp Switch 2	Click Toggle On/Off	Activates the effect saved to effect slot 2.
17	FX Stomp Switch 3	Click Toggle On/Off	Activates the effect saved to effect slot 3.
18	FX Stomp Switch 4	Click Toggle On/Off	Activates the effect saved to effect slot 4.
19	Alt. Selection Stomp Switch	Click Activate	Tap once to switch between FX stomp switches selecting effects and selecting layers. Double tap to select all layers.
20	Loop Start Switch	Click Activate	Triggers the recording of a new loop.
21	Loop Stop Switch	Click Activate	Triggers the end of a new loop recording.
22	Layer 1 Indicator LED	On/Off	Indicates that layer 1 is currently selected.
23	Layer 2 Indicator LED	On/Off	Indicates that layer 2 is currently selected.
24	Layer 3 Indicator	On/Off	Indicates that layer 3 is currently

	LED		selected.
25	Layer 4 Indicator LED	On/Off	Indicates that layer 4 is currently selected.
26	FX Display	Dynamic	Displays the current effect's information. Also displays the menu when appropriate.
27	Layers Display	Dynamic	Displays the four layers currently recorded.

Table 3: Physical Interface Specifications

Left Display User Interface

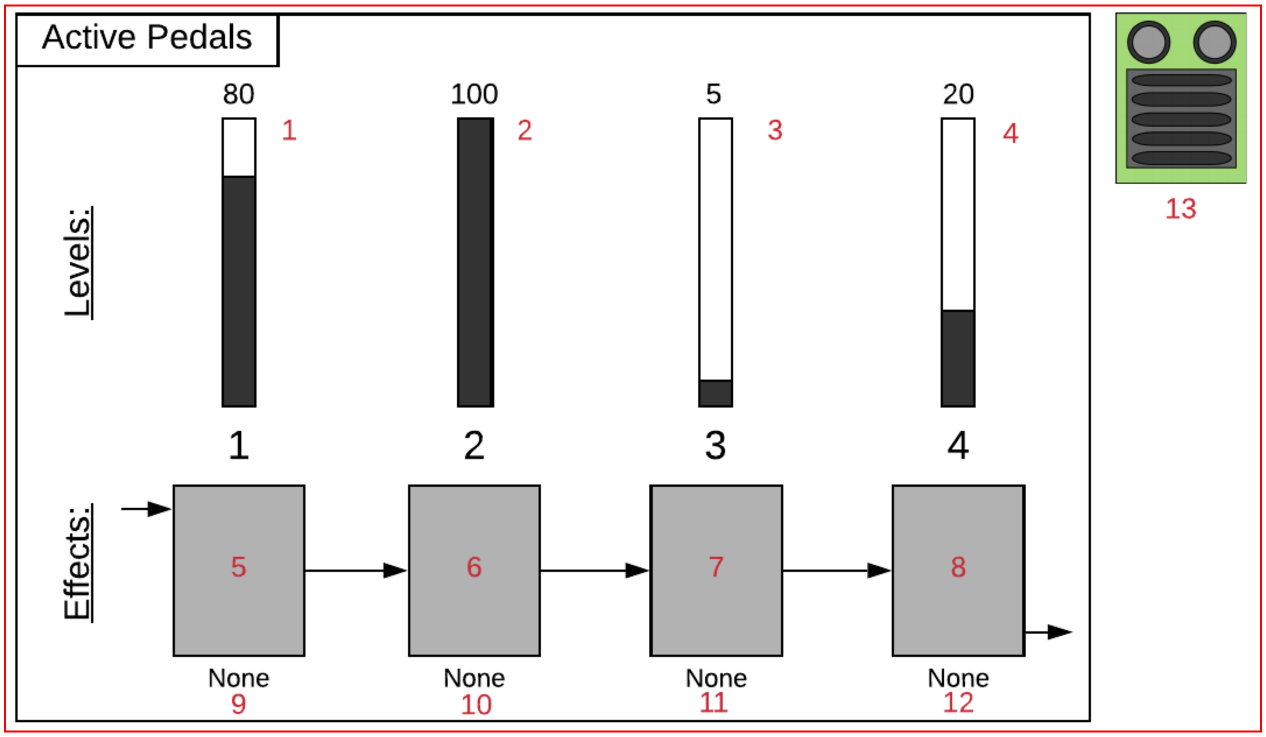


Figure 13: Left Display Design Diagram

Note: *Elements 5, 6, 7, and 8 have an associated rotary encoder (hardware) that can be pushed in.*

Left Display Interface Function Table

#	Name	Description	Rotary Encoder Interaction
1	Effect 1 Level	Amount that effect in slot 1 affects the sound output	Rotating associated knob increases or decreases this value and the UI reflects the changes
2	Effect 2 Level	Amount that effect in slot 2 affects the sound output	Rotating associated knob increases or decreases this value and the UI reflects the changes
3	Effect 3 Level	Amount that effect in slot 3 affects the sound output	Rotating associated knob increases or decreases this value and the UI reflects the changes
4	Effect 4 Level	Amount that effect in slot 4 affects the sound output	Rotating associated knob increases or decreases this value and the UI reflects the changes
5	Effect Slot 1	Effects placed here will be first in the sequence	Clicking on the associated controller will open the Effect Slot Menu [detailed below]
6	Effect Slot 2	Effects placed here will be second in the sequence	Clicking on the associated controller will open the Effect Slot Menu [detailed below]
7	Effect Slot 3	Effects placed here will be third in the sequence	Clicking on the associated controller will open the Effect Slot Menu [detailed below]
8	Effect Slot 4	Effects placed here will be fourth in the sequence	Clicking on the associated controller will open the Effect Slot Menu [detailed below]
9	Effect 1 Name	The name of the effect assigned to Effect Slot 1	None
10	Effect 2 Name	The name of the effect assigned to Effect Slot 2	None
11	Effect 3 Name	The name of the effect assigned to Effect Slot 3	None
12	Effect 4 Name	The name of the effect assigned to Effect Slot 4	None

13	Pedal Icon	Icon representing an effect	None

Table 4: Left Display Interface Specifications

Effect Slot Menu

This menu is opened whenever a rotary encoder associated with one of the four effect slots is clicked in. There are two versions of the menu; One for if the associated effect slot has a assigned effect, and another for if the associated effect slot has not been assigned an effect (i.e. the slot is “empty”). The options of the two menu versions are detailed below.

Menu Option	**Interaction**
Move	When this option is selected, the user will be able to move the associated effect left/right in the sequence by turning the rotary encoder.
Edit	Selecting this option brings the user to a new UI page where they are able to modify more values of the associated effect.
Save Preset	Selecting this option will prompt the user for a name to save the current associated affect with in memory.
Remove	This option will remove the associated effect from the sequence.
Cancel	This option closes the Effect Slot Menu.

Table 5: With Assigned Effect

Menu Option	**Interaction**
Add New	This option will add a new effect from a list of preset effect templates that is selected by the user.
Load Preset	Selecting this option allows the user to load one of the effect presets that they have saved to the system
Cancel	This option closes the Effect Slot Menu.

Table 6: Without Assigned Effect

3.4: Implementation Issues and Challenges

Digital / Analog Audio Conversion

We want our device to take audio input from a variety of devices, primarily instruments. These can be anything from guitars to synthesizers. The issue for our project is that some of these output a digital audio signal, while others output an analog audio signal. Our team is facing the challenge of being able to handle both types. We are doing this using an analog-to-digital converter (ADC) and a digital-to-analog converter (DAC).

Manipulation of Sound Data

After we have the audio data from the input, we want to be able to manipulate it. This manipulation will be an effect that modifies the sound wave to produce a unique audio profile. The members of our team do not have strong backgrounds in audio signals or sound waves prior to this project. As such, our team faces the challenge of figuring out how to perform the sound manipulation we desire. This spans from choosing the right programming language to selecting proper algorithms, as well as research into the physics of sounds.

Displaying Audio Information Intuitively

With our technological knowledge and backgrounds, we have a better toolset to understand the data that is presented by our device. Our team is keeping in mind, however, that the primary user for our device - musicians - may not have a technical background. The challenge this presents is displaying all necessary information in a way that is intuitive for our end-user to understand. As we work on the project, we plan to have many user-interface (UI) options to acquire feedback on, By the end of development, we hope to achieve the goal of an informative and intuitive UI design.

Minimizing Audio Latency Between Input/Output (I/O)

Our device is to be used primarily in a live setting - such as a concert or other music performance. In these types of settings, timing is everything. As such, we have the issue of latency between I/O and the device's processing hardware. Latency is essentially the delay between the request for the transfer of data and the actual transferring of the data. For example: a high latency would be like flipping a light switch, then the light switching on a couple of seconds later. A low latency would make the time between flipping the switch and the light coming on near instantaneous. Our team's challenge is reducing the amount to latency in our device, both hardware and software, as much as possible.

Having Enough Processing Power for Audio Manipulation

A major hardware-related challenge that we are working on is the processing power needed to manipulate and process the sound data. The biggest contributor to this challenge is the desire for our final product to be of a more compact size than carrying around a small computer.

To do this, we are utilizing a Single-Board Computer (SBC) as our main controller. While the Raspberry Pi is the most recognizable SCB, it does not fit our needs; instead, we are using the ROCKPro64. This Single-Board Computer has 4 GB of LPDDR4 RAM versus the Pi's 1GB of LPDDR2 - meaning more RAM at a significantly faster speed. Furthermore, the ROCKPro64 has a hexa-core processor, which provides more processing power than the Raspberry Pi's quad-core processor.

If we do find that more processing power is needed, the ROCKPro64 includes a PCIe x4 slot. With this slow, we are capable of using a dedicated sound card for our audio processing. This additional hardware is optimized for working with sound data and frees up additional processing power from the central processing unit (CPU) that can be used for other functionality, such as connectivity with the mobile app. The issue may still arise if further processing power is needed. If it does, we will look into alternative plans of action.

Managing Two Displays

Since our design calls for two displays to be working simultaneously, we have the challenge of managing the UI and functionality of both. Our devices has one display that is used for displaying menus, is controllable using hardware knobs on the device, and activates back-end code to modify the system. For this functionality, we need a display port that can handle input, in addition to output. The single-board computer (SBC) we have chosen, the ROCKPro64, includes a MiPi-DSI port. This port is a high-speed interface that connects the display to the processor, enabling a two-way connection. For our second display, we do not require control. This display will simply convert its output to HDMI and attach to the RockPro64's HDMI port.

Build Weight Versus Structural Integrity

Although our device will rest on the ground during use, it is not wholly stationary - i.e. the user needs to be able to carry it around. Therefore, we want our device to be light and easy to transport. On the other hand, being similar to guitar pedals, our device's switches will be mainly pressed quickly with a foot - hence the name "stomp boxes". This presents the issue of having the device be light without sacrificing the structural integrity and being easily damaged during normal use. To find a solution, we are experimenting with different build materials for the final product. For building prototypes, we are planning to 3D print an enclosure.

3.5: Design Analysis

Strengths

Versatility: Our device has the strength of being highly versatile, both in hardware and software. Below are highlights of this versatility.

Software

- Loops
  - Our device, the Cyren, has several software advantages, including the ability to work with loops. The device can record, play, and modify multiple loops from multiple inputs. Each of these instruments could be mapped to a different instrument, or be loops from the same instrument. To increase the versatility further, the loops can be played in tandem to produce a full song with a single device. The diverse musical applications are numerous with this feature.
- Effects
  - Alongside looping capabilities, our device also enables the user to modify their sounds using effects. These effects can be applied directly to the audio as it's being recorded. The effects can also be added to tracks that have already been recorded in a loop.
  - The effects are adjustable via physical knobs that are dynamically mapped to different aspects of the effect, including: gain, volume, speed, etc. Once an effect has been modified to the user's liking, they are able to save the effect as a profile that can be saved-to and loaded-from memory.

Hardware

- Interfacing With Devices
  - As stated above, our device can take input from multiple devices at once. Furthering that idea, we want to be able to support a diverse set of devices. To do this, we are allowing the device to connect with many different connection types. To name a few, there are: XLR, 3.5mm, aux, etc. This versatility in hardware is a great strength for the Cyren, allowing our users a greater toolset for their creative endeavors.

Visualization: In addition to versatility, our device's strengths come from the ability to visualize sound data in a meaningful way. While our backgrounds are in technical fields, the same may not hold true for our users. This fact means we must present the data in a way that they, mainly musicians, can understand. We do this primarily with the following aspects.

Visualizing Sound With Waves

- When coming up with how to display our data to the user, we looked at existing music manipulation software, also known as a digital audio workspace (DAW). These DAWs almost always displayed their data via waveforms. Our team implemented our visuals on this same concept. Using a visualization that is standard across other tools of the same nature, we reduce the learning curve of using the Cyren.

- The wavelengths of the recorded loops will be visible to users at all times and will be created in real-time when recording a loop. This will enable the user to make better, more informed decisions during a performance. For example, if they see that the wavelength is too large, they can know to play softer to obtain a better sound.

Standardized Physical UI

- In addition to the software, we also want the hardware to be somewhat familiar with our intended user, the musician. To achieve this goal, we researched other devices such as guitar effect pedals, keyboards, synthesisers, MIDI controllers, and more. From that research, we selected the most common hardware interfaces and used them for the Cyren.
- The most common aspect we found on almost all of these devices were knobs that controlled levels of some aspect. Our device also utilizes knobs in the same fashion - to control levels of components, such as volume. The next most common hardware component across these devices was the tactile foot switch. These switches are found on nearly every guitar effect pedal on the market. These pedals are also commonly called “stomp boxes”, as the guitarist will stomp on the switch mid-song to quickly activate an effect. In using these two components on our device, we have increased familiarity with new users who have previous musical experience.

Weaknesses

Large Amount of Knobs: One of the weaknesses of the current design of the Cyren is the copious amount of knobs. We currently have the following knobs:

- 4x knobs for the dynamically allocated effect controls
- 4x knobs for controlling the four layers of recorded loops
- Aux volume knob
- MIDI volume knob
- Mic volume knob
- IN volume knob
- Master volume knob

With a total of thirteen knobs, we fear the user will overwhelmed when first getting acquainted with our tool. We will look into alternatives for the next iteration of our design.

Placement of Pedals: Another weakness the current design of the Cyren has is the place of the pedals. We have three pedals: one for starting a loop recording, one for ending a loop recording, and an expression pedal for use with certain sound effects. Since these pedals are actuated using the users foot, they need to be oriented completely horizontal. This conflicts with the orientation of our screens and knobs, which are at an angle to provide the user with easier visibility. As a result, the current placements of the pedals are as separate units. This diminishes the portability and all-in-one aspect of the Cyren.

Accessibility to Main Control Unit: With the current design of the Cyren, it has a weakness with the ease of which the main control unit, the ROCKPro64, can be accessed. Since we are actively developing and modifying the connections to the control unit, we want to be able to access the controller's interface simple. However, the current design has the ROCKPro64 nested comfortably inside of the Cyren's enclosure. This makes it difficult to access during development, as well as making maintenance on the final product a frustrating task.

Wire Slack: There is a weakness with the current design regarding the inner wiring of all the components. As this is an early design and components are not yet permanent, we have not put the time into wire routing. This means that the bulk of the wiring is loose inside of the enclosure - causing issues like wires getting disconnected or shorted. Furthermore, this wire issue can become a fire hazard in the right conditions. Once we have finalized the component decisions, we will be able to design particular routing measures for the wiring.

Rough Visual Aesthetic: This weakness is simple, but a weakness nonetheless. Since this is design is more a proof-of-concept than a product to be sent to market, we have focused more on the functionality of the Cyren, versus its visual appeal. We expect this aspect of the device to approve with future iterations. The biggest change will come at the end of the development cycle, when we have the device functioning fully to design specifications.

Observations

Close Proximity of Controls May Be Frustrating: Our team wants the Cyren to be fairly compact and mobile. Combined with the desire for the system to be versatile, many of our controls sit close together on our hardware interface. Our users, being musicians, will need to adjust the controls while in the heat of a performance. The close controls in this situation could result in a knob being turned unintentionally, or a switch being triggered on accident. This is an aspect we will be paying attention to when testing this design.

We do not know the lifespan of individual components: Since many of our hardware components do not have documentation on extended usage and we do not have a large amount time for this sort of testing, our team can not know the individual lifespan of each individual component. If any of these components fail, then the entire system will not function to full capacity. As such, we cannot accurately predict the lifespan of the Cyren as a whole. With more time, we will be able to do extensive testing in this area - ideally before the Cyren goes to market.

3.6 Process Details

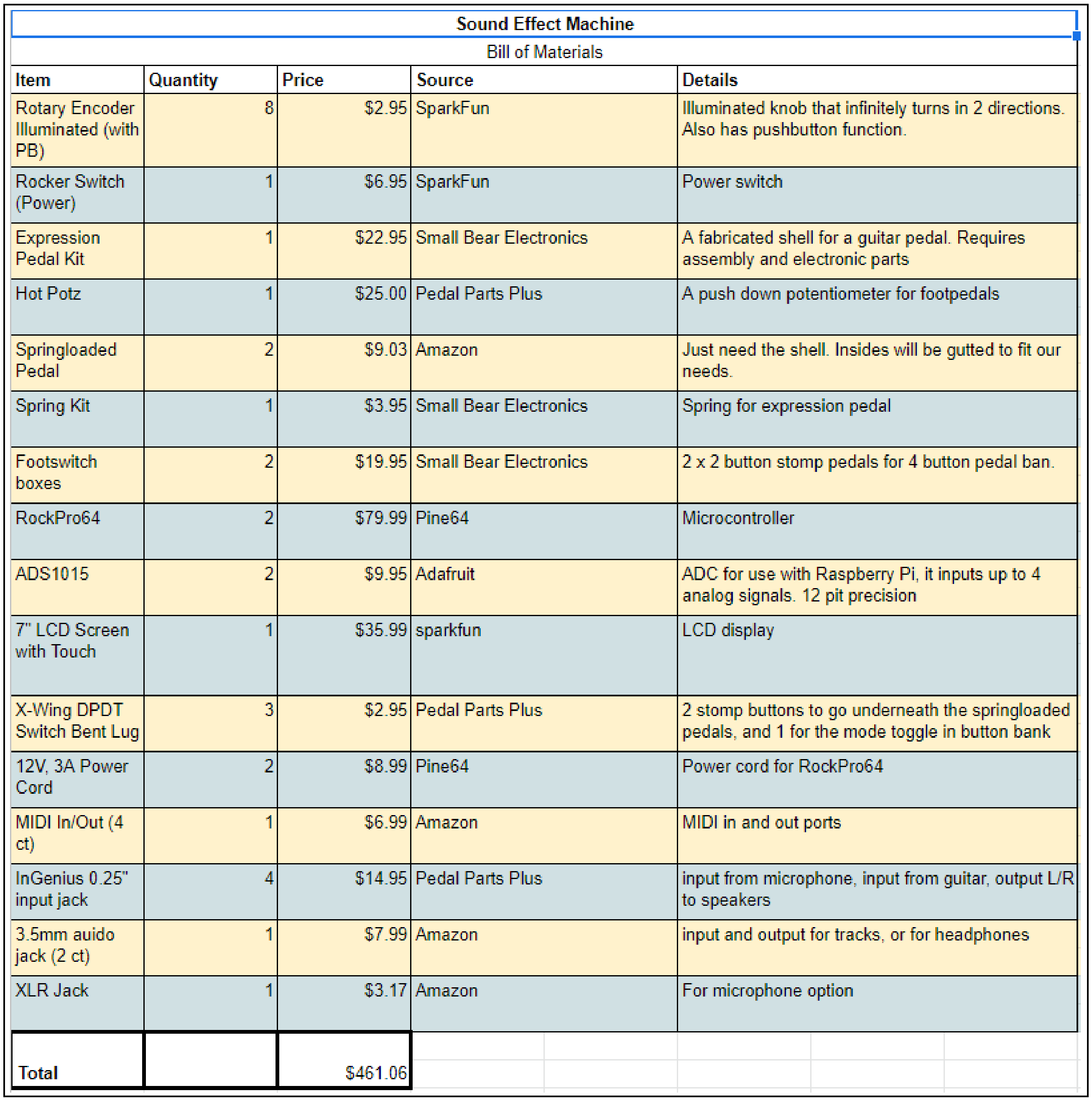
As we are still currently designing the majority of the project, our current progress is in the planning stage. Like mentioned in the technical approach, we are in the midst of performing research and proofs of concepts for the various different technologies we could be using. Being that there are so many different ways to perform the same thing, we want to make sure we use choose the technologies that suit us best. We managed to separate the project and their deliverables into these sections: input and output modules for microprocessor, the microprocessor, the enclosure, system level api to interact with input and output modules, core code base to run our software, and the user interface (visuals).

We have came to a few conclusions on the different technologies will be using and why. We knew we needed a microprocessor that could manage many inputs and outputs requiring a decent processor, a respectable amount of RAM, and as many input and output pins as possible. With that we decided on the Rock Pro as it meets our requirements and outperforms other microprocessors for what we need. When using analog inputs with a digital editing software and then outputting them through analog, we performed research on what analog to digital converter (ADC) and digital to analog converter (DAC) to use. We have not concluded on this yet, as it is one of the most important parts of our design. To utilize the ADC and DAC buffer best, along with all the various I/O buttons/screens on the device, we decided on designing our software in C. By using C we can easily interact with various hardware components compared to other languages. Being that we decided on using C for the core of the project, we decided to use GTK as it integrates with C seamlessly and provides a beautiful user interface. GTK will generate on-click listeners similar to Javascript that will point to C methods we created in the core. Lastly, we know we need a durable enclosure so we will be using CAD to design the most compact and durable enclosure we can. We plan to use metal for the enclosure to ensure its lifespan.

4.0: Estimated Resources

4.1: Hardware

This analysis is clearly a rough estimate. Much refinement is needed to cut costs and optimize our device. The assumptions for this project's costs were that we would use prefabricated materials, and use a basic microcontroller. Further component selection is needed in order to cut costs.



Sound Effect Machine

Bill of Materials

Item	Quantity	Price	Source	Details
Rotary Encoder Illuminated (with PB)	8	$2.95	SparkFun	Illuminated knob that infinitely turns in 2 directions. Also has pushbutton function.
Rocker Switch (Power)	1	$6.95	SparkFun	Power switch
Expression Pedal Kit	1	$22.95	Small Bear Electronics	A fabricated shell for a guitar pedal. Requires assembly and electronic parts
Hot Potz	1	$25.00	Pedal Parts Plus	A push down potentiometer for footpedals
Springloaded Pedal	2	$9.03	Amazon	Just need the shell. Insides will be gutted to fit our needs.
Spring Kit	1	$3.95	Small Bear Electronics	Spring for expression pedal
Footswitch boxes	2	$19.95	Small Bear Electronics	2 x 2 button stomp pedals for 4 button pedal ban.
RockPro64	2	$79.99	Pine64	Microcontroller
ADS1015	2	$9.95	Adafruit	ADC for use with Raspberry Pi, it inputs up to 4 analog signals. 12 pit precision
7" LCD Screen with Touch	1	$35.99	sparkfun	LCD display
X-Wing DPDT Switch Bent Lug	3	$2.95	Pedal Parts Plus	2 stomp buttons to go underneath the springloaded pedals, and 1 for the mode toggle in button bank
12V, 3A Power Cord	2	$8.99	Pine64	Power cord for RockPro64
MIDI In/Out (4 ct)	1	$6.99	Amazon	MIDI in and out ports
InGenius 0.25" input jack	4	$14.95	Pedal Parts Plus	input from microphone, input from guitar, output L/R to speakers
3.5mm auido jack (2 ct)	1	$7.99	Amazon	input and output for tracks, or for headphones
XLR Jack	1	$3.17	Amazon	For microphone option
Total		$461.06		

Table 7: Hardware Resources

4.2: Software

- GTK - free and open-source cross-platform widget toolkit for creating graphical user interfaces. GTK is licensed under the terms of the GNU Lesser General Public License (allowing free and proprietary software use).
- Tiny Test - Is a unit testing framework designed for testing code written in the C programming language. Tiny test is a very simple and straightforward testing framework that would not take much time to understand and create tests.
    - License: MIT License

- Or -

- CUnit - A unit testing framework designed for testing code written in the C programming language. CUnit is a slightly more complex testing framework that may need to be used in the case that mocking will be necessary for system testing (which appears unlikely).
    - License: GNU Free Documentation License
- FLAC - An audio coding format for lossless compression of digital audio, and is also the name of the free software project producing the FLAC tools, the reference software package that includes a codec implementation.

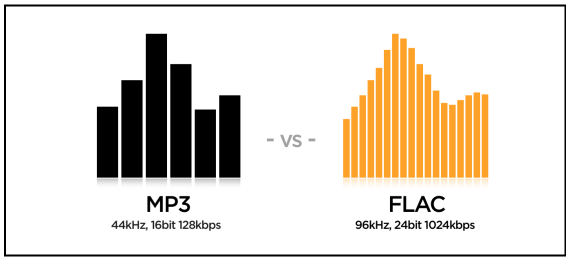


Figure 14: MP3 vs FLAC

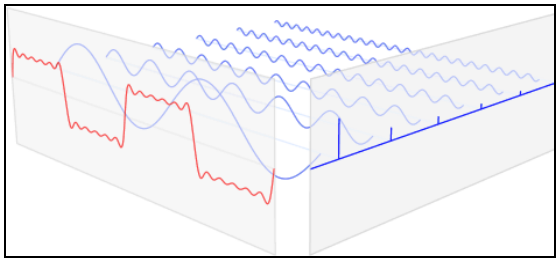


Figure 15: FFT Visualization

- Language: C
- License: GNU GPL; Libraries: BSD (BSD licenses are a family of permissive free software licenses, imposing minimal restrictions on the use and distribution of covered software.)
- FFT - C subroutine library used for computing the discrete Fourier transform (DFT) in one or more dimensions of both real and complex data. This is referring to the transformation of a sound (or sine) wave. This library is important because it will serve as our digital signal processor (DSP).
- FFTW (Fast Fourier Transform) is suitable for many types of applications because its flexibility. Below is an example of the principle behind the Fourier Transformation as applied to sound waves. Sound waves at different frequencies (blue sine waves) can be combined to form a final complex wave structure (red sine wave).
    - License: FFTW is a free software.

5.0: Project Timeline

For our project we needed to create a timeline to help us better be prepared and provide us a guide to follow. The better this timeline is, the more likely it will be easier to accomplish our goal. When creating this project timeline, we had to really break the project down and categorize what different parts there were in the project. Three of the biggest sections we found are the hardware, the user interface, and the connections between the two (API). As a group, we managed to discuss the type of process we would like to have moving forward. Being that there is a lot of research to be done, we wanted to use the Proof of Concept (POC) approach. By having members perform extensive research on certain topics, we can cover more ground. When someone finishes research, they are to be comfortable enough to share a synopsis of the research to the rest of the group while making a clear and decisive suggestion on whether the technology or idea researched would be ideal for our project moving forward, thus allowing us to not waste time during the research phase. We also provided time for much testing and refining at the end of the project's timeline to ensure a quality project in the end.

Below are listed four parts of the same graph. In the first section (Table 8a) you will observe each task being assigned an ID and what the task encompasses. The final column "predecessors" describes which task is needed to be completed before we can start on the current task. By using this part of the table, we are able to produce the actual timeline that is displayed in the remaining three parts of Table 8 (Table 8b, Table 8c, and Table 8d). In these tables you will observe each tasks' corresponding ID, if you move to the right you will find a gray denoted block that resembles the time expected to complete the task. Each cell of this chart resembles a week or seven days. If you look above, every four cells resembles a month that is clearly denoted as the header. This is to help envision the actual dates in which these tasks are expected to be started and completed. Be sure to check if a task runs between the different portions of the tables. For example, task 13 starts on the last week of April in Table 8b but then continues into the first two weeks of May in Table 8c.

ID	Task Name	Predecessors
1	Start	
2	Decide on controller	1
3	POC on other hardware	2
4	POC on conversion method for analog to digital	2
5	POC on main language used for project	2
6	Buy hardware and connect to system (rough draft)	3
7	Implement conversion methods for Analog to Digital	6
8	Design code base library to be used for project's software	5
9	POC on power options for the unit	6
10	POC on file types that we will store in database	7
11	Design database based on the API	10
12	Create connections to database from software using API	5,10
13	Install power management	9
14	Draft first enclosure for the device	9
15	Draft UI design	8
16	Implement drafted UI	15
17	Draft tests for API connections	12
18	Poll people on our enclosure draft	14
19	Poll people on our UI	16
20	Refine enclosure upon research completion	18
21	Refine UI based upon research gathered	19
22	Draft tests for UI	21
23	Perform final tests on the newly constructed research results	22
24	Refine prototype and create final product	23

Table 8a: Project Timeline

ID	January				February				March				April			
	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
1																
2	▨	▨	▨													
3				▨	▨	▨	▨									
4				▨	▨	▨	▨	▨								
5				▨	▨	▨										
6								▨	▨	▨	▨	▨				
7													▨	▨	▨	▨
8							▨	▨	▨	▨	▨					
9													▨	▨	▨	
10																
11																
12																
13																▨
14																▨
15												▨	▨	▨	▨	
16																▨
17																
18																
19																
20																
21																
22																
23																
24																

Table 8b: Project Timeline

	May				June				July				August			
ID	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
1																
2																
3																
4																
5																
6																
7	■															
8																
9																
10		■	■	■												
11					■	■	■									
12					■	■	■									
13	■	■														
14	■	■														
15																
16	■	■	■	■	■											
17								■	■	■						
18			■	■	■											
19						■	■	■	■							
20						■	■	■	■							
21										■	■	■	■	■		
22															■	■
23																
24																

Table 8c: Project Timeline

	September				October				November				December			
ID	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
1																
2																
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
16																
17																
18																
19																
20																
21																
22	▨															
23		▨	▨	▨												
24					▨	▨	▨	▨	▨							

Table 8d: Project Timeline

6.0: Conclusion

Cyren will be used by musicians all over to create unique sounds with our sound effect system combined with multiple loopers that can be layered over one another. We as a group envision Cyren being used by a multitude of users, from beginners to experts, we want to provide a product that will find use for almost any guitarist looking to add looping or sound effects to their music. Our design is well constructed and follows IEEE guidelines. The project itself has been iterated upon countless times to reach the optimal solution and as a group we are quite pleased with the end result and are excited to start working on development. We hope to provide with this document the level of planning necessary to fill in any gaps in understanding and make Cyren a reality.