

EEL 4914 - Senior Design 1 Project - Group C

Analog and Digital Effects Processing Technology (ADEPT)



Department of Electrical Engineering and Computer Science

University of Central Florida

Dr. Lei Wei

Group C

Diego Conterno - CpE

Tyler Michaud - EE

Alejandro Porcar - CpE

Dylan Walter - EE

Table of Contents

- 1.0 Executive Summary
- 2.0 Project Description
 - 2.1 Project Motivation
 - 2.2 Goals and Objectives
 - 2.3 Requirements Specifications
 - 2.4 House of Quality
 - 2.5 Block Diagram
- 3.0 Research Related to Project Definition and Part Selection
 - 3.1 Similar Products and Relevant Technologies
 - 3.2 Project Research and Part Selection
 - 3.2.1 Bypass Switch
 - 3.2.2 Analog Input and Output Buffers
 - 3.2.2.1 Input Buffer
 - 3.2.2.2 Output Buffer
 - 3.2.2.3 OPA2134 Operational Amplifiers
 - 3.2.2.4 2N2222A Transistors
 - 3.2.2.5 JFET Buffers
 - 3.2.2.6 Op Amp vs Transistor Buffers
 - 3.2.2.7 Passive Component Selection and Explanation
 - 3.2.3 Microcontroller
 - 3.2.3.1 Broadcom BCM2711, Quad-Core Cortex-A72
 - 3.2.3.2 FV-1
 - 3.2.3.3 MSP43x Series
 - 3.2.3.4 Cortex M4
 - 3.2.3.5 Programming Interface
 - 3.2.3.6 Memory Mapping
 - 3.2.3.7 Hardware Specifications
 - 3.2.3.8 I/O
 - 3.2.3.9 External Oscillator
 - 3.2.3.10 Heartbeat Program
 - 3.2.3.11 Liquid Crystal Display (LCD)
 - 3.2.3.12 Flash Memory
 - 3.2.3.13 User Interface
 - 3.2.4 Analog-to-Digital Conversion
 - 3.2.4.1 ADC (Analog-to-Digital Converter)
 - 3.2.4.2 Choosing an ADC
 - 3.2.4.2.1 ADS1675IPAG
 - 3.2.4.2.2 TLV321ADC3100

- 3.2.4.2.3 PCM186x
- 3.2.4.3 DAC (Digital-to-Analog Converter)
- 3.2.4.4 Choosing a DAC
 - 3.2.4.4.1 PCM1789-Q1
 - 3.2.4.4.2 PCM1742
 - 3.2.4.4.3 TLV320DAC3101
- 3.2.4.5 CODEC
- 3.2.4.6 Choosing a CODEC
 - 3.2.4.6.1 MAX98050
 - 3.2.4.6.2 PCM3060
 - 3.2.4.6.3 TLV320AIC23
- 3.2.4.7 Power
- 3.2.4.8 I/O
- 3.2.4.9 Analog Peripherals
- 3.2.4.10 Communication
- 3.2.5 Tone Section
 - 3.2.5.1 Bluesbreaker Tone Control
 - 3.2.5.2 Big Muff Tone Control
 - 3.2.5.3 Tube Screamer Tone Control
 - 3.2.5.4 Tone Control Selection
- 3.2.6 Power (9VDC)
 - 3.2.6.1 Power Efficiency
 - 3.2.6.2 Power Sequencing
 - 3.2.6.3 Grounding Methods
 - 3.2.6.3.1 Common Ground
 - 3.2.6.3.2 Earth Ground
 - 3.2.6.3.3 Chassis Ground
 - 3.2.6.3.4 Analog Ground and Signal Ground
 - 3.2.6.4 Circuit Isolation
 - 3.2.6.5 Droop Voltage
 - 3.2.6.6 Impedance Matching
 - 3.2.6.7 Ripple Voltage
 - 3.2.6.8 Linear Regulators
 - 3.2.6.9 Switching Regulators
 - 3.2.6.10 DC-DC Converters (Buck)
 - 3.2.6.11 DC-DC Converters (Boost)
 - 3.2.6.12 5V and 3.3V Digital Logic Power Supplies
 - 3.2.6.13 Transistor-Transistor Logic (TTL)
 - 3.2.6.14 Design Implementation
 - 3.2.6.15 Device Selection
 - 3.2.6.15.1 AMS1117 Buck Converter
 - 3.2.6.15.2 LT3045 Linear Regulator

3.2.6.15.3 Hurdles and Complications

4.0 Related Standards and Realistic Design Constraints

4.1 Standards

- 4.1.1 IEEE
- 4.1.2 IEC/UL 62368
- 4.1.3 UL60065
- 4.1.4 Joint Test Action Group (JTAG)
- 4.1.5 80 Plus Certification
- 4.1.6 Safety Requirements for Class 2 Power Units
- 4.1.7 CE - FCC - RoHS Certification
- 4.1.8 I2C
- 4.1.9 SPI
- 4.1.10 I2S

4.2 Constraints

- 4.2.1 Economic Constraints
- 4.2.2 Time Constraints
- 4.2.3 Environmental Constraints
- 4.2.4 Social Constraints
- 4.2.5 Political Constraints
- 4.2.6 Ethical Constraints
- 4.2.7 Health and Safety Constraints
- 4.2.8 Manufacturability Constraints
- 4.2.9 Sustainability Constraints

5.0 Project Hardware and Software Design Details

5.1 Initial Design Architectures and Related Diagrams

- 5.1.1 Input Buffer Testing and Part Selection
- 5.1.2 Output Buffer Testing and Part Selection
- 5.1.3 MCU Design Specifications and Part Selection
 - 5.1.3.1 Prototype
 - 5.1.3.2 High Speed External Oscillator (HSE)
 - 5.1.3.3 GPIO Configuration
- 5.1.4 CODEC Design Specifications and Part Selection
- 5.1.5 Tone Control Testing and Part Selection
- 5.1.6 Power Distribution Testing

5.2 Breadboard Test and Schematics

- 5.2.1 Input Buffer Breadboard Test
- 5.2.2 Output Buffer Breadboard Test
- 5.2.3 MCU Breadboard Test
- 5.2.4 CODEC Breadboard Test
- 5.2.5 Tone Control Breadboard Test
- 5.2.6 Power Distribution Breadboard Test

6.0 Project Prototype Construction and Coding

- 6.1 Integrated Schematics
- 6.2 PCB Vendor and Assembly
 - 6.2.1 Dimensions
 - 6.2.2 Layers
 - 6.2.2.1 Grounding Layer
 - 6.2.3 Thickness
 - 6.2.4 Track Width and Spacing
 - 6.2.5 Material Type and Lead-Free Options
 - 6.2.6 Mask and Silkscreen
 - 6.2.7 Vendor Selection
- 6.3 Final Coding Plan
 - 6.3.1 High Level Code Structure
 - 6.3.2 Effect Parameters
 - 6.3.3 Digital Effects
 - 6.3.3.1 Bitcrusher
 - 6.3.3.2 Chorus/Vibrato
 - 6.3.3.3 Compressor
 - 6.3.3.4 Delay
 - 6.3.3.5 Distortion
 - 6.3.3.6 Envelope Filter/Autowah
 - 6.3.3.7 Flanger
 - 6.3.3.8 Looper
 - 6.3.3.9 Phaser
 - 6.3.3.10 Pitch Shifter/Harmonizer
 - 6.3.3.11 Reverb
 - 6.3.3.12 Tremolo
- 7.0 Project Prototype Testing Plan
 - 7.1 Hardware Test Environment
 - 7.2 Hardware Specific Testing
 - 7.3 Software Test Environment
 - 7.3.1 STM32CubeMX
 - 7.3.2 STM32CubeIDE
 - 7.3.3 Version Control
 - 7.4 Software Specific Testing
- 8.0 Administrative Content
 - 8.1 Milestone Discussion
 - 8.2 Budget and Finance Discussion
- Appendices
 - Appendix A: References
 - Appendix B: Copyright Permissions
 - Appendix C: Datasheets

Figure Index

- Figure 1: House of Quality
- Figure 2: Block Diagram
- Figure 3: True bypass, standard bypass, and buffer bypass.
- Figure 4: Input Buffer (BJT - Emitter Follower)
- Figure 5: Output Buffer (BJT - Emitter Follower)
- Figure 6: Op Amp Buffer
- Figure 7: BJT Amplifier Specifications
- Figure 8: JFET Buffer
- Figure 9: Potentiometer Taper Chart
- Figure 10: STM32F446RC Memory Mapping Block diagram
- Figure 11: MCU Power Scheme
- Figure 12: STM32CubeMX Pin Programming Setup
- Figure 13: Digital Signal Resolution
- Figure 14: Aliasing Effect
- Figure 15: Interpolation of Discrete-Time Signal
- Figure 16: 4-Bit R-2R Resistor Ladder Network
- Figure 17: Block Diagram of 1-Bit $\Sigma\Delta$ DAC
- Figure 18: Using Variable Resistor as Control Input
- Figure 19: I2S controller pairing
- Figure 20: Musical Instrument Frequency Range Chart
- Figure 21: Bluesbreaker Tone Control Frequency Response
- Figure 22: Bluesbreaker Tone Control Schematic
- Figure 23: Big Muff Tone Control Frequency Response
- Figure 24: Big Muff Tone Control Schematic
- Figure 25: Tubescreamer Tone Control Schematic
- Figure 26: Tubescreamer Tone Control Schematic
- Figure 27: Power Supply Stages
- Figure 28: Common Ground Symbol

Figure 29: Earth Ground Symbol
Figure 30: Chassis Ground Symbol
Figure 31: Analog and Digital (Signal) Ground Symbol
Figure 32: Galvanic Isolation
Figure 33: Transient-Suppression Capacitors
Figure 34: Linear Voltage Regulator
Figure 35: Switching Regulator
Figure 36: Buck Converter
Figure 37: Boost Converter
Figure 38: Power Distribution Diagram
Figure 39: LTSpice Schematic, Analog Input/Output with MCU Input
Figure 40: AMS1117 3.3V DC/DC Buck Converter
Figure 41: 3.3V DC/DC Buck Converter Design
Figure 42: 3.3V DC/DC Buck Converter output simulation
Figure 43: 3.3Vcc and 5Vdd Linear regulator Diagram
Figure 44: Overall Initial Circuit Design Prototype
Figure 45: Input Buffer Small Signal AC Analysis
Figure 46: Input Buffer Transient Response ($V_{in} = 500\text{mV}$, 440Hz)
Figure 47: Input Buffer Frequency Response (Cutoff Frequency = 20Hz)
Figure 48: Output Buffer Circuit Simulation
Figure 49: Output Buffer Frequency Response
Figure 50: Output Buffer AC measurement with $V_{in} = 300\text{ mV}$
Figure 51: Output Buffer Small Signal Output Impedance
Figure 52: Prototype Diagram
Figure 53: Gmcrit Equation
Figure 54: Stray Capacitance Equation
Figure 55: Excel Sheet Formulation
Figure 56: External Crystal Oscillator Network
Figure 57: Minimum and Maximum Cutoff Frequency Calculation
Figure 58: Tone Control Circuit Simulation (Cutoff Frequency = 500Hz)
Figure 59: DC Power Supply for Testing (9V, 500mA)
Figure 60: Input Buffer Breadboard Test
Figure 61: Input Buffer Transient Response ($V_{in} = 500\text{mV}$, 440Hz)
Figure 62: Output Buffer Breadboard Test
Figure 63: Output Buffer Transient Response ($V_{in} = 500\text{mV}$, 440Hz)
Figure 64: MCU Testing PCB
Figure 65: CODEC Testing PCB
Figure 66: Tone Control Breadboard Test
Figure 67: 3.3V Buck Converter Breadboard Test
Figure 68: 5V Linear Regulator Breadboard Test
Figure 69: Final ADEPT Schematic (Eagle)
Figure 70: Recommended PCB Layering Scheme for STM32 MCU

Figure 71: Ideal Ground Layer Design Concept
Figure 72: Visualization of Distortion Effect on Sinusoidal Signal
Figure 73: Visualization of Flanging vs. Phasing
Figure 74: Diagram of Direct and Reflected Sound Waves
Figure 75: Visualization of Reverb Signal Structure
Figure 76: Semester Milestones Gantt Chart

Table Index

Table 1: Requirements Specifications
Table 2: Block Diagram Responsibility Chart
Table 3: Buffer Hardware Comparison
Table 4: MCU Hardware Comparison
Table 5: ADC Hardware Comparison
Table 6: DAC Hardware Comparison
Table 7: CODEC Hardware Comparison
Table 8: MCU/ADC Pin Connection
Table 9: MCU/DAC Pin Connection
Table 10: Budget and Parts List

1.0 Executive Summary

Since the dawn of modern music, musicians have altered the sound of their instruments. As early as the 1940's, recording engineers and musicians would experiment in their recording studios, using reel-to-reel magnetic tape recording machines outside of their intended function in order to achieve various "studio effects". By running two tape machines in tandem, with each machine playing the same piece of music simultaneously, various time modulation effects could be achieved, such as echoes (delay) and frequency sweeps (flange). In other cases, microphones were placed in large, empty rooms in order to achieve a larger sense of space in the recordings (reverb). Many guitar players at the time would turn up their amplifiers as loud as possible in order to saturate the vacuum tubes within their preamp section and coax harmonically-rich distortion and overdrive tones out of them.

This desire to stand out and create unique tones spawned an era of creative technological innovations, bridging the gap between technology and music - between science and art. Engineers began to experiment more, and soon came up with stand-alone effects units that could achieve tremolo and echo effects (such as the DeArmond Trem-Trol and the Maestro Echoplex, respectively). However, these stand-alone units were large, bulky, and inconvenient. They were also very expensive and required high voltages in order to operate.

There was a need for a new, more portable technology that wouldn't affect the meager income of a musician so heavily. Up until that point in time, only vacuum tubes were

available for audio amplification. The invention of the electronic transistor in 1947 by Bell Labs brought with it the portability and low cost that previous audio circuits did not have. This allowed for audio effect circuits to be put into “stompbox” enclosures that could be conveniently activated by stepping on a footswitch.

Since then, music technology companies like Boss, MXR, Dunlop, Ibanez, Electro-Harmonix, Digitech, and countless others have created incredible analog and digital effects pedals that have inspired musicians to create music that has been heard on millions of records. These industry giants have also inspired independent engineers and musicians to create their own “boutique” pedals. Companies like Earthquaker Devices, JHS, Walrus Audio, Keeley Electronics, and Rainger FX have brought a personal and unique touch to the world of audio effects, discovering new sounds that many musicians and music lovers have never heard the likes of before.

As more developments of this incredible technology become available to us, engineers and musicians alike seek to utilize this new technology in a unique and interesting musical context. The vast online community of DIY tinkerers and engineers has normalized the building of guitar pedals as not only a career, but a passion. Our Senior Design project is no different - after spending several years learning about electronics and coding, we want to use our knowledge and skills to pursue a project in music electronics: building a guitar effects pedal of our very own, hoping to innovate and create new sonic territories in the process.

2.0 Project Description

The following sections describe our inspiration for the ADEPT project, as well as what we achieved in pursuing it. We also outlined some of the specific features that we integrated in our design, and the responsibilities of each team member in the overall research and design process.

2.1 Project Motivation

As engineering students, as well as musicians, we found that a senior design project based around audio engineering, music technology, and sound design was very rewarding and informative. It was the perfect opportunity to apply the engineering fundamentals we have learned into something we are passionate about. Closing the gap between technology and music, we aimed to provide new alternatives for musicians who want to stand out and shape their sound in a unique way. We provided a digital multi-effects solution that has a creative impact on musicians with an interest in electrical and computer engineering.

Therefore, we proposed the idea of designing a musical instrument effects processing unit. This simple-to-use effects pedal is able to take the analog input from a guitar or other musical instrument (via a 1/4" instrument cable), convert that analog signal into a digital signal that can be manipulated within a processing unit, and then be converted back into an analog signal that can be heard by human ears.

The digital signal processing (DSP) was done by a microcontroller or microprocessor. This processor has a minimum requirement of a 16-bit resolution, and a sample rate of 44.1kHz. This guarantees a high-fidelity stereo sound output. The DSP chip can be either integrated into the main processing unit or it can be external. In this way we were not limited on the type of processing unit we used for our effects. We have a variety of modulation and time-based effects (delay, reverb, chorus, phaser, and tremolo), as well as some effects that implemented a manipulation of gain or bit depth (distortion/overdrive, bitcrusher, etc).

Since we needed to manipulate analog signals digitally, an ADC/DAC (Analog-to-Digital/Digital-to-Analog Converter) was implemented into the design. Since most guitar pedals that are commercially available run off of 9V – 12V (DC), the power requirements were very simple to integrate. Due to the nature of this project, and its integration between software and hardware elements, the workload was very balanced between the EE and CpE students on the team.

2.2 Goals & Objectives

Portable: The ADEPT prototype is able to fit in the palm of your hand, and weigh no more than 3lbs. This will make it tremendously portable for the gigging musician, who will simply attach it to his/her pedalboard or toss it in a backpack.

Affordable: Due to the immense amount of products with various prices and capabilities to choose from, we must be considerate in finding balance between price and quality. We want our product to cost less than \$250. In this way, most of our product's funding can go towards our MCU and our power supply. Components such as resistors, capacitors and inductors are reasonably priced. Overall, we want the most bang for our buck.

Easy to Use: We want our product to have obvious means of operation. In other words, we want any random user without any prior knowledge about the product or the fundamentals behind the operation of the product to operate it with little to no difficulty. How did we implement this? We can facilitate operation of our product through proper labeling, intuitive interface, and a user's guide. These three aspects will satisfy the ease of use criteria.

Low Power Consumption: In order to satisfy this objective, we must specify the total input voltage and current based off of our power supply. Our power supply will provide 9V and 500mA. Therefore, the maximum power distributed to the system is 4.5 Watts.

Optimal Noise Reduction: We must attain this objective by first specifying a signal to noise ratio (SNR) that would reflect clean audio signals passing through our system. We have specified a SNR to be -14dB. We first implemented this to make sure to use high quality components. Poorly made components can have unwanted inductance or capacitance when manufactured improperly. Moreover, we will want to minimize the signal path as much as possible as wire can have inductance which can increase common mode noise. Next, we will want to make sure to use proper grounding techniques to provide a clean neutral path for any excess noise going through our system to make sure that the noise does not want to go to unwanted parts of our circuit. Finally, we will want to have proper analog to digital and digital to analog conversion techniques to maintain our signal.

Maintaining Reliability: We want our guitar pedal to work the same way every time it is powered on. We want to make sure to have the proper protection protocols to prevent damage to the circuit. These protocols consist of reverse current protection transient response compensation and power on and power off sequencing. We want the customer to be able to operate our for a long time. We do not want our customer to only be able to use this device a few times. For example, not having proper power sequencing or transient response protection during powering on or powering off the device, which could cause the device to break on the 10th time of use. In other words, we want our guitar pedal to operate in a consistent way without failing.

2.3 Requirements Specifications

Requirements specifications are the functional and nonfunctional characteristics that our team develops before implementation. This approach streamlines the details of the product, the details in which the product will operate under, along with measurable attributes that were implemented into the design.

The reason for requirements specifications is to streamline the design process and transform the concept of the product itself into a tangible thing. This can include measurable values that reflect the performance of the product or illuminate physical properties of the product. This facilitates the functionality of the product by illuminating execution steps needed for the ideas making up the device. Requirements and specifications make measurable expectations come to life and allow our team to verify the attributes of the product efficiently.

The following requirements specifications include several important features that we liked to implement in the ADEPT pedal prototype. These features outline the integral

performance and sound quality deliverables of our project, both aesthetic and technical. Some of the aesthetic design requirements we have implemented include looper functionality, usb programmability, an LCD user interface, and dual footswitch functionality.

Technical design requirements implemented in our design include A/D Conversion, Total Harmonic Distortion, minimal added noise, high input impedance, low output impedance, standard ¼” signal and 9V power cable connections, and an affordable cost.

1. Analog-to-Digital Conversion
2. Total Harmonic Distortion (THD)
3. Minimal Added Noise
4. Input Impedance
5. Output Impedance
6. Looper Functionality
7. Standard ¼” Instrument Cable Compatibility
8. Power
9. USB Programmability
10. LCD Menu and Selection Screen
11. Low Cost
12. Dual Footswitch Functionality

<u>#</u>	<u>Specifications</u>	<u>Description</u>
1	Analog-to-Digital Conversion	The CODEC should meet a minimum requirement of a 16-bit resolution and 44.1 kHz, meeting CD industry standards.
2	Total Harmonic Distortion	A clean output signal (no effects added) should have a THD rating of less than 1% to the input signal in the audible range (20 Hz – 20 kHz)
3	Minimal Added Noise	The noise present within our circuit is no more than -14dB, leading to a high-quality
4	High Input Impedance	750kΩ
5	Low Output Impedance	475Ω

6	Looper Functionality	The guitar should have the capacity of recording and playing back at least 30 sec – 1 min of a sample recorded in the looper mode. The sampling rate being 44.1 kHz and the resolution 16-bits.
7	Standard ¼” Instrument Cable Compatibility	The device should have a ¼” cable input/output compatibility, as it is a standard for electric guitars and amplifiers.
8	Power	The device should be able to be powered by an external 9 VDC power supply rated at 500 mA. The power supply should be compatible with the standard 5.5 x 2.1 mm barrel plugs (center negative polarity)
9	USB Programmability	For debugging purposes, we will include a USB port. The user can then re-program the board and update the firmware that way.
10	LCD Menu and Selection Screen	Our pedal will utilize an LCD screen that the user can interface with, in order to select the desired effects on the fly.
11	Low Cost	Unlike current digital guitar pedals in the market, which starting prices range from \$200 - \$300, our device should have a total cost of less than \$150.
12	Dual Footswitch Functionality	Among the footswitches available (2), the tap tempo will have a dual functionality depending on the mode selected. The tap tempo will adjust the rate during the effect mode. In the looper mode, the switch will act as a sample start and stop.

Table 1: Requirements Specifications

2.4 House of Quality

Our house of quality was built to generate our concepts of our product into engineering characteristics. These characteristics need to be quantifiable in some way - one engineering characteristic will have a measurable effect on the other, and vice versa. For example, sound quality will have a strong correlation with cost, in the sense that the processing of the MCU. During analog to digital conversion, as the number of bits increases, the more accurate your signal is going to be when digitally processing the signal. When having high amounts of bit processing capabilities the cost of those products are higher than less bit processing capabilities. We quantified our engineering characteristics into six numerical values. Power efficiency, sound quality, weight, cost,

size, number of effects. We are able to judge how much those quantified characteristics will rise or fall based on a customer's requirements on the vertical axis. We feel that these are realistic and achievable goals and are satisfactory with the quality that we want in our product.

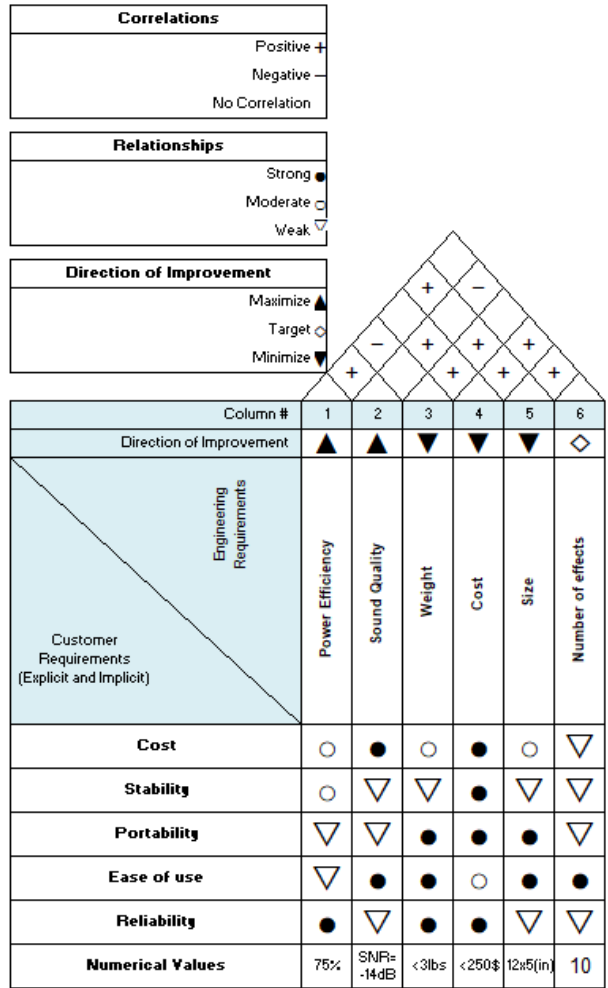


Figure 1: House of Quality

2.5 Block Diagram

In order to complete the ADEPT prototype on time, and in the most efficient way possible, we will need to allocate various project responsibilities to each member of our team. This will not only make our workflow much more streamlined, but will also allow us to hold each other accountable for project deadlines pertaining to the various parts of our research, design, and construction. In order to allocate these team roles, as well as get a better overall idea of what our project design entails, we have assembled a block

diagram that will take care of both of these requirements. We have also included a block status table that will allow us to keep track of the current state of each part of our design.

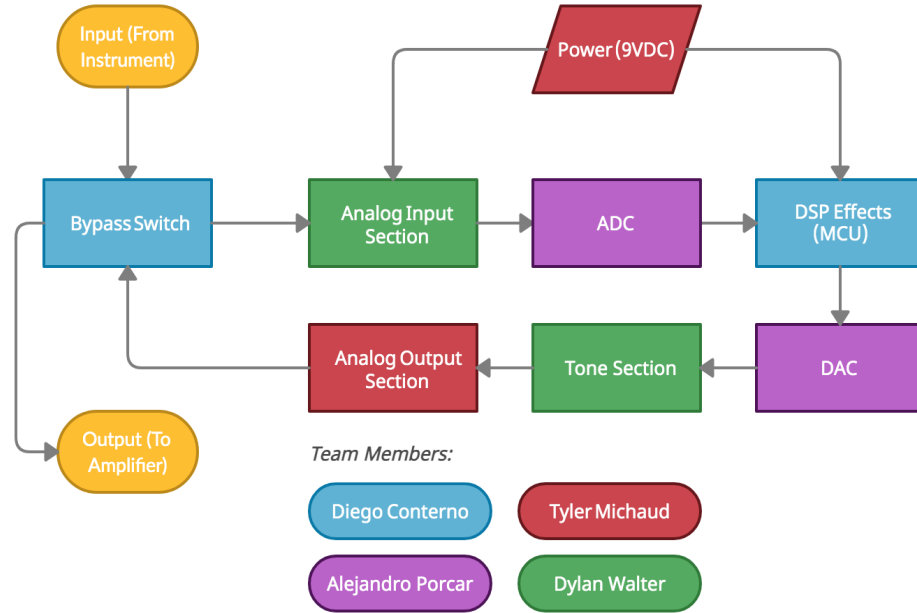


Figure 2: Block Diagram

Block Title	Block Status	Responsible Member
Bypass Switch	Complete	Diego
Analog Input Section	Complete	Dylan
ADC	Complete	Alejandro
DSP Effects (MCU)	Complete	Diego
DAC	Complete	Alejandro
Tone Section	Complete	Dylan
Analog Output Section	Complete	Tyler
Power (9VDC)	Complete	Tyler

Table 2: Block Diagram Responsibility Chart

3.0 Research Related to Project Definition and Part Selection

In order to successfully build the ADEPT prototype, extensive research must be done on each of its components. This includes both the hardware components (PCB and circuit design) as well as the software elements (MCU, effects coding). In the sections below, we

will provide some necessary research to define the design requirements of our project, and subsequently choose specific parts based upon this research.

3.1 Similar Products and Relevant Technologies

While the intention of the ADEPT project is to break new ground sonically, the technological fundamentals of what this project is based upon have been established for years by large music companies. These companies have released industry-standard effects that musicians have used for years. However, there are some eccentric musicians and hobbyists who find joy in tearing apart such exquisitely-made electronic devices, with the desire to understand how they work and how to replicate them. While at first glance it may seem sacrilegious to dismantle a guitar pedal (and likely void its warranty), many times this can yield remarkable results. When this reverse-engineering is done well enough, a “clone” is produced.

A guitar pedal “clone” is what it sounds like: a DIY, homemade copy of a popular guitar pedal that is normally manufactured in bulk by large music tech companies. However, a clone is no mere act of plagiarism - oftentimes these clones can become the foundation for new and exciting musical innovations. What starts as an exact replica of a popular pedal design can often rapidly evolve into a completely new and independent product. By manipulating certain parameters and components within pre-established circuits, many DIY pedal enthusiasts (some with no engineering background whatsoever) have found themselves turning their passion into a career.

In a similar way, we utilized the vast amount of resources available from previous musical and technological innovations in order to inform us of the best and most reliable parts and procedures to realize our pedal design to the fullest.

3.2 Project Research and Part Selection

In the following sections, we will provide detailed research into several advanced technologies. This research will inform our decisions of which hardware components we implemented into the design of the ADEPT prototype. We will make selections on the type of bypass switch we will use to turn the effect on and off, as well as decide upon the best hardware configurations for input and output buffers, the ADC/DAC, the microcontroller, the tone control configuration, as well as the power distribution and power regulation. By the end of this section, we will have decided upon all the hardware that we implemented in the final design of the ADEPT prototype.

3.2.1 Bypass Switch

For the bypass switching system, we have a couple of options to choose from. Most effects pedals use either a mechanical (passive) 3PDT (3 Pole, Double Throw) switch, or an electronic (active) SPST (Single Pole, Single Throw) soft switching relay system. The 3PDT switch is the easiest system to implement, but because of its mechanical switching system, is not very reliable (limited to roughly 30,000 switching cycles). The relay bypass system is rated to several million cycles, which makes it far more reliable, but is a bit more difficult to implement, as there is a coding element. Many audiophiles and tone purists may also complain that the active relay bypass system may affect the tone of the input signal, and does not faithfully reproduce the tone of the instrument because of all the secondary circuitry required. In addition to the passive and active switching options shown above, there are also switching options that utilize FETs and bistable flip-flops as well. For our purposes, we will try to simplify our design as much as possible, so these solutions will likely not be considered.

Looking at the different designs for true bypass switches in the market, we determined which one would best fit our demands. Let's start by looking at the main ways a pedal bypasses the effects section.

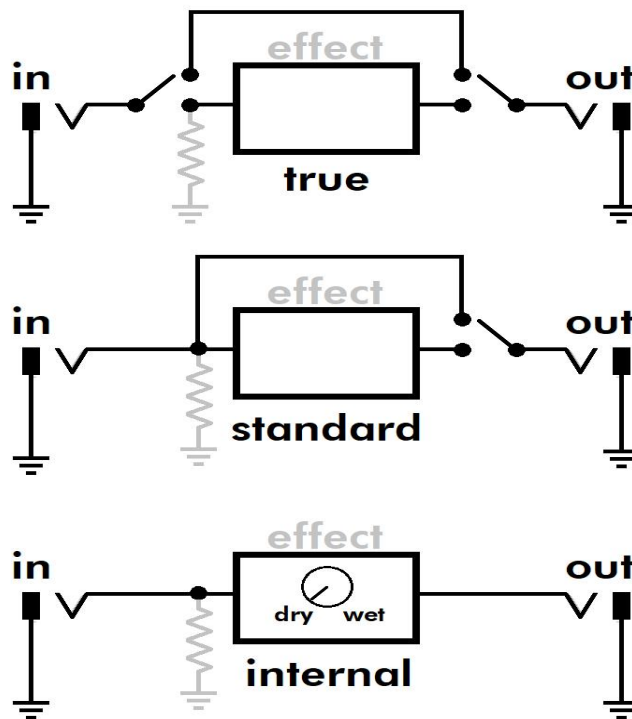


Figure 3: True Bypass, Standard Bypass, and Buffered Bypass

In the picture depicted above, we can see the three main approaches most commonly used in the industry for bypassing the effects block. The first diagram depicts a true bypass, the

gray resistor represents the input impedance of the effects block, which in our case was the DSP block (CODEC and MCU). The true bypass totally neglects the input impedance of the effects block and thus the “load” applied by the low input impedance does not reduce the level of the guitar signal coming into the device. Thus, a true bypass connects the input jack directly to the output jack.

The standard bypass switch is somewhat similar to the true bypass with the exception that the input impedance is not neglected and thus it affects the output signal. This can generate a high amount of disturbance in the signal depending on the guitar pickups (the microphones of the guitar) impedance and the length of the cable used to connect to the pedal, because cable capacitance can also affect the overall distortion.

Finally, the buffer scheme for bypassing the effects block is a bit more complicated in the sense that the bypassing mechanism is handled by the internal circuitry inside the effects block. In essence, the effects block recognizes the state applied by the user, and thus applies little to no effects to the signal. However, the loading applied by the input impedance cannot be avoided. Though the loading effect of the effects block might seem like a non-desirable phenomena, sometimes it can help the signal of the guitar sound better than a true bypass switch. This is due to the guitar pickups utilized and the input impedance of the guitar amp utilized.

Furthermore, there are other ways of implementing a bypass system for a guitar pedal, though they imply a more complex implementation. An example of this is the relay bypass scheme, in which the bypass mechanism is realized using a soft SPST switch, a relay, and the MCU. Although this approach is complex in its design, the benefits of having a more durable switching mechanism and the use of a relay are a very desirable design implementation. However, the cons of implementing a scheme such as the relay bypass also implies the addition of extra components to the PCB, including the relay itself and other peripherals.

One of the most common mechanical foot switches used in effects pedals is the 3PDT. This switching solution is a good choice for a simple implementation of what we are trying to achieve. Fundamentally, as the name implies, we are trying to bypass the DSP block in our device, and thus not apply any effects or processing to the clean instrument signal in any way. The following question arises then, why would we want to do this? The answer is simple: guitar players sometimes don't need effects, and just want a clean guitar sound. A simpler and more compact solution was to have a soft SPST switch that connects only to the MCU and functions as a button, which sets a flag to apply or not apply effects to our processed signal. However, processing is still required by the MCU and the CODEC, and the signal might not be considered “clean”, as there could be an involuntary addition of distortion to the signal. In a similar manner, the soft SPST switch will raise a flag in the MCU. However, this flag will not start or stop the effects - instead,

its purpose is to signal the MCU that the switch has changed its position and requires effects. Simultaneously, the soft SPST switch will trigger the power line and power the LED, signaling the user that effects are being applied. Furthermore, this implementation will require that the connection from the MCU to the soft SPST switch was set to pulldown, so that when the switch is not pressed it will read low and once pressed it will read high.

We have decided that the implementation of a true bypass switch for our design was best, since it does not involve a complex implementation of hardware and software to realize it. Although we want to build a reliable system, we believe that a mechanical switch like the 3DPT was more than sufficient for the purpose of testing and proving the concept we would like to deliver for our presentation.

3.2.2 Analog Input & Output Buffers

3.2.2.1 Input Buffer

A buffer is an active electronic circuit that can provide a change in electrical impedance, or resistance. For example, if a circuit design requires a low impedance output, but has a high impedance input, we can use a buffer to transform the overall impedance of the circuit from input to output. In our case, that is exactly what we want to do for the input stage of our guitar pedal circuit.

The reason for this necessary change in impedance in our circuit is due to the effect that impedance has on a guitar's signal. If a musician is running his/her guitar through a long signal path of effects pedals, or even just a long instrument cable (both of which increase in resistance the longer they are), the tone of the instrument will degrade significantly, and the resilience and clarity of the frequency spectrum will become compromised. In other words, the long lengths of wire that lie between the instrument and the amplifier/speaker are literally "impeding" the signal from flowing strongly enough, which results in an objectively low quality guitar sound.

Therefore, it was necessary to implement a unity gain buffer at both the input and the output of our guitar pedal circuit. Unity gain simply means that our signal does not increase or decrease in amplitude - only the impedance is adjusted. This was important to be implemented at the circuit's input - guitars have a very high impedance output from the pickups, which we want to "boost up" by reducing the impedance to prepare the signal to enter the rest of the circuitry.

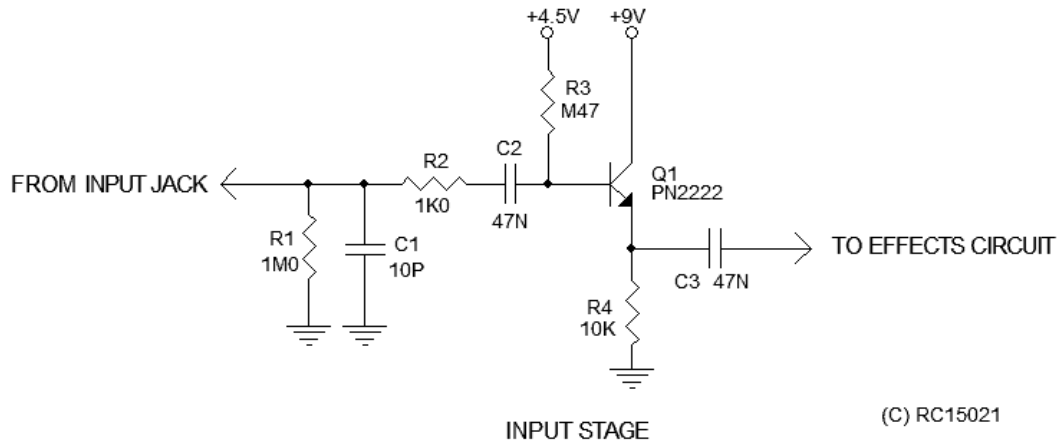


Figure 4: Input Buffer (BJT - Emitter Follower)

3.2.2.2 Output Buffer

As mentioned above, we are striving to lower the impedance going into our circuit at the input. The same applies to the output - the output section of our circuit will also require the utilization of a unity gain buffer that will have a low output impedance. This is intended to keep the signal strong in the signal chain that lies between the instrument and the amplifier/speaker. This signal chain may include other pedals or music devices.

Since the instrument signal has weakened throughout the previous parts of the circuit, it needs to be “boosted up” again before being sent out to another effects pedal or to an amplifier/speaker. As with the input stage, there are multiple resistors and capacitors being used in tandem with the transistor in order to achieve similar desired utilizations. Generally in this stage, a volume potentiometer is also implemented to control the overall loudness level of the circuit.

In order to implement a buffer amplifier to decrease the impedance at both our input and output, we must use some sort of semiconductor device. We considered implementing this by using FETs (Field Effect Transistors), BJTs (Bipolar Junction Transistors) or operational amplifiers.

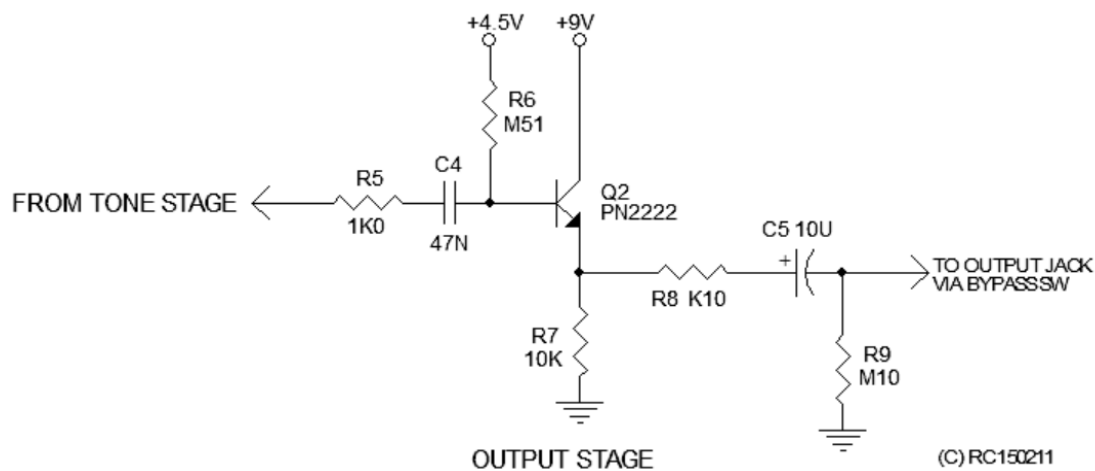


Figure 5: Output Buffer (BJT - Emitter Follower)

3.2.2.3 OPA2134 Operational Amplifiers

Operational amplifiers are widely used to increase the output gain in sound systems. In order to simplify things. Operational amplifiers have various FETs, resistors, and capacitors in them to produce the gain that we desire. We must take into account the type of op-amp we need to maintain the desired sound quality in our system. When choosing the right op amp, we must consider the following: Input impedance, output impedance, noise, slew rate, maximum supply voltage, and bandwidth.

Input impedance: Op amps have a very high input impedance between the negative and positive terminals of the input. Ideally it is treated as an open circuit however it is simply such a high impedance that a very minute current passes through the positive and negative input terminals.

Output impedance: Op amps are designed to have a very low output impedance. This is because we want a very small amount of current at the load going to ground. This minimizes our loss at the output terminal.

Noise: Op amps have components inside them that cause parasitic noise. They occur at the output and back to the input. Equivalent Input Noise Voltage is the most common type of noise found in an op amp.

Slew rate: The slew rate of an op amp is the rate of change in the output voltage caused by a step-change in the input. The higher the value of slew rate, the faster the output can change and the more easily it can reproduce high-frequency signals.

Maximum supply voltage: We will need an op amp that can handle or peak voltages at our input. Moreover, We needed to supply the necessary power to the differential inputs at our op amp to implement this. If our input approaches the magnitude of DV power to our op amp, clipping may occur. This causes unwanted distortion in our sound output. So we must consider this in our design.

Bandwidth: The bandwidth of an op amp is the allowable frequency range of the signal at input. Ideal op amps allow all frequencies to pass through them. This is not the case in the real world. We will need an op amp that has a bandwidth that falls within the operating frequencies at the analog output of our guitar pedal.

We will need to couple op amps to resistors and capacitors in order to smooth out our signal, particularly if any edges occur on our sine wave signal after digital to analog conversion. The capacitors play the role of filtering out the small DC components that make up the edges in our sine wave. The resistors have a direct relationship to the magnitude of our gain in order to maintain signal quality. Smoothing is a technique used after digital to analog conversion. Regarding the edges of the alternating wave after conversion, we can use capacitors to filter out those DC components to reduce distortion and maintain the integrity of any wave form passing through our circuit. This is essential for our project or else will run into low sound quality at the output of our speaker. We must also consider noise at our output.

Noise will occur along our signal and that noise will consist of various frequencies rippling along any wave form passing through our system. Therefore, we will need to put more capacitors with different capacitance to filter out noise at different frequencies along our system.

A good op amp for our project was OPA2134 by Texas Instruments. This op amp is ideal because it has high slew rate, bandwidth, and very low noise. Here is a basic example of a differential operational amplifier with a DC offset that maintains and smooths our signal. Note that the capacitor at the output is key for ideal sound quality:

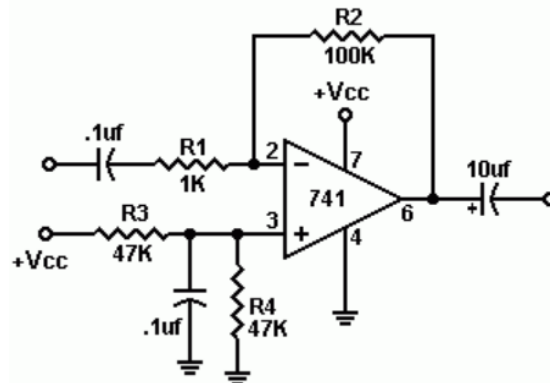


Figure 6: Op Amp Buffer

3.2.2.4 2N2222A Transistors

In many effects pedals, the most common buffer circuits implement an NPN transistor in the Common Collector (Emitter Follower) configuration. This type of BJT (Bipolar Junction Transistor) amplifier configuration is characterized by a voltage gain roughly equal to 1. This means that the gain/volume of the circuit is not amplified - the function of this configuration is mainly used in order to lower the high impedance (High-Z) input coming from the instrument (or from other music gear/effects pedals) down to a low impedance (Low-Z) that will feed into the next section of the circuit (ADC/effects). This is exactly what we want, as this will keep our signal strong and preserve all the frequencies and overtones that are so important to a guitar's unique tonal characteristics.

Based on these requirements needed for the input buffer stage (as well as for the output stage), we will consider the use of the 2N2222A NPN Transistor in our pedal circuit. This transistor has the capability of functioning as necessary for the Emitter Follower configuration (high input resistance, low output resistance), as well as providing adequate current gain and being able to withstand the levels of current that is flowing through our guitar pedal circuit.

Our team also considered the use of a similar transistor, the 2N3904. This transistor has an almost identical construction when compared to the 2N2222A, even having the same maximum collector-emitter voltage $V_{ce0} = 40V$. However, the main difference between these two components is that the 2N2222A has a higher maximum collector current $I_c = 1000mA$ (1A), with the 2N3904 having a max value of $I_c = 200mA$. Since the ADEPT pedal is only working with small currents between roughly 10mA-500mA, either transistor would likely suffice in our design. For the purposes of our component comparison, however, we will only consider the 2N2222A, as the higher collector current will provide our design with fewer limitations than the lower collector current of the 2N3904.

Configuration	Voltage gain	Current gain	Input resistance	Output resistance
Common emitter	$A_v > 1$	$A_i > 1$	Moderate	Moderate to high
Emitter follower	$A_v \cong 1$	$A_i > 1$	High	Low
Common base	$A_v > 1$	$A_i \cong 1$	Low	Moderate to high

Figure 7: BJT Amplifier Specifications

3.2.2.5 JFET Buffers

JFET buffers are actually really popular in maintaining signals produced by electronic pickups in instruments. They are low cost and have decent input impedance. For input and output audio buffers, we use JFETs and are used in common drain configuration or source follower configuration. Where the Gate pin of the transistor is the input, the source is the output and the Drain acts as the common. The downside to JFETs is that they will have to be re-tuned if we swap out one model for another. Op amps with the same pin configuration will not have this problem. Here is a standard JFET buffer configuration:

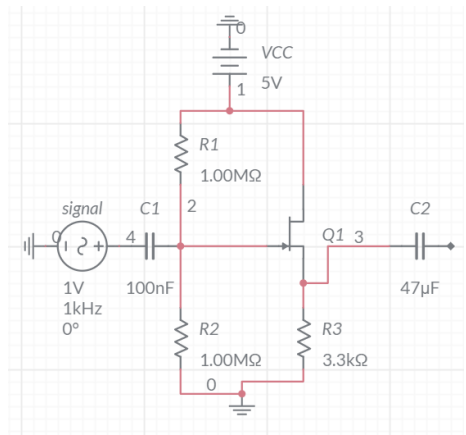


Figure 8: JFET Buffer

3.2.2.6 Op Amp vs Transistor Buffers

Our main goal for both input and output stages is to maintain the signal either coming into or out of our system as best as possible. We must find a balance between power consumption, performance, cost and simplicity of design.

Op amps have a very high input impedance. A high input impedance is beneficial to reducing noise in our signal coming from the guitar pickups. When a signal is traveling from the pickup through the cable, unwanted capacitance and inductance causes unwanted filtering of our signal which we are trying to maintain. Having a high input impedance helps negate these issues by limiting current fluctuations which leads to the reduction of voltage fluctuations in which the noise/filtering occurs.

Op amps have this advantage over transistor buffers. We can try to emulate a nominal input impedance by manually maximizing our input impedance of a transistor buffer. However, we will never come close to what is found on the market with any op amp. Simply because op amps are optimized for maximum performance by using more complex DC coupling methods and more complicated internal design.

Two advantages that transistor buffers have over op amp buffers are cost and power consumption. As you can see in the table, the BJT is 6 times cheaper than the op amps alone. By choosing the op amp, we are paying more to maximize performance. Op amps also need DC biasing which will typically range from +/- 15V or +/- 12V. Again, there is no free lunch.

	<u>2N2222A Transistor</u>	<u>OPA2134 Op Amp</u>
<u>Cost</u>	\$0.85	\$4.95
<u>Output Current</u>	600mA	40mA
<u>Output Impedance</u>	Null	0.6Ω
<u>Gain</u>	300	10E12
<u>Noise</u>	4dB	-7dB

Table 3: Buffer Hardware Comparison

3.2.2.7 Passive Component Selection and Explanation

There are several different types of passive elements (resistors and capacitors) that was utilized in our circuit. These components have many varied uses and applications in audio circuits, which we will discuss below:

The capacitors in an effects pedal circuit can be used for multiple different functions, depending on where they are placed in the circuit. Power filtering capacitors are used to filter out any noise that comes from the 9V power supply. Coupling capacitors allow AC signals to flow through them, but not DC; this is integrated into many effects pedal designs in order to prevent DC current from combining with the signal coming from the instrument, and thereby preventing more noise from being generated. Finally, capacitors can be used in RC (resistor and capacitor) circuits in order to shape the tone of a circuit and filter out unwanted frequencies. More information about this RC circuit integration was covered below in the Tone Section.

While variable capacitors are available for use in the design of our circuits, they will most likely not be integrated due to the fact that potentiometers (variable resistors) are a much simpler solution if real-time adjustment is required at any point to adjust the analog/digital parameters of the effects pedal.

The resistors in an effects pedal circuit can also have different functions depending upon where they are placed and what other components they are used alongside. At the most fundamental level, resistors can be used in a voltage divider circuit, which can be tremendously useful in power applications. More about voltage division and power management was covered in the Power (9VDC) Section below.

Since resistors limit the current through a circuit, they are most often used to adjust the level into or out of components by decreasing the amplitude of the voltage. A resistor can be placed at the beginning of an audio circuit to change the overall gain of the circuit (such as when used alongside a transistor or operational amplifier); they can be placed at the end of a circuit to adjust the overall volume (loudness) of the output. Resistors can also be used to absorb excess voltage present in a circuit; this excess voltage is most often held within capacitors that have been charged, and may cause static/popping noises to be heard when the pedal is engaged. To prevent this, high value “pulldown resistors” are connected to ground along the signal path and are used to drain/bleed this charge out of the circuit, thereby removing the potential for noise.

Oftentimes in effects pedals, variable resistors (potentiometers) are used instead of static resistor values in order to adjust these analog parameters such as gain and volume on the fly. The volume control of a circuit is essentially a variable voltage divider, with a potentiometer connected to the input, output, and ground of a circuit in order to limit the overall current through the circuit and reduce the volume level.

Since we were integrating the use of potentiometers in many parts of the ADEPT prototype, it is important that we discuss the various different types available, and some relevant information about their construction. Potentiometers are most often categorized by the type of “taper” that they have, which refers to the relationship between the position of the potentiometer’s rotation and its resistance. These types of tapering generally fall

into the two categories: “audio” and “linear”. In the United States, the naming convention for potentiometer tapers is as follows: A-type potentiometers have a logarithmic sweep, B-type potentiometers have a linear sweep, and the less common C-type potentiometers have a reverse logarithmic sweep. These different types of potentiometers have various applications in different circuits, as we will discuss below.

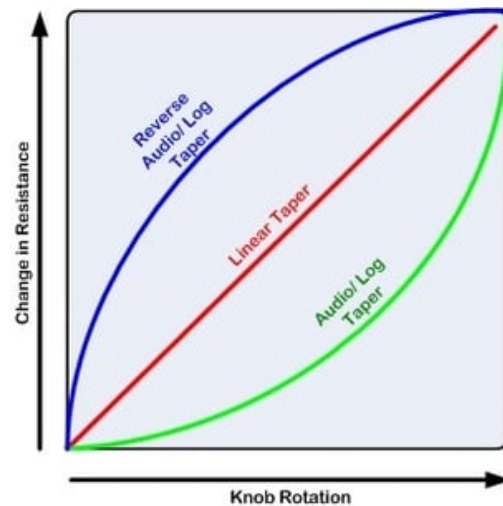


Figure 9: Potentiometer Taper Chart

Since human hearing is logarithmic, it is most common for the amplitude of audio signals to be manipulated in this way. As you rotate a standard A-type logarithmic potentiometer, the amount of resistance will increase very slowly at first, and then increase drastically as it nears the end of its rotation. In other words, the resistance of a logarithmic potentiometer increases the most in the latter 50% of its overall rotation. Volume/level controls (as discussed above) are one application of the use of a logarithmic potentiometer in a circuit. Another use of logarithmic potentiometers is in tone control or equalization circuits. The auditory behavior of this type of taper is very natural to our ears, as opposed to linear potentiometers, which are generally used in different types of applications.

Linear potentiometers, as their name implies, have a perfectly linear sweep in resistance. This is ideal for such applications as light dimmer circuits, in that you can accurately adjust the brightness of the light to be 50% when the knob is set to halfway, for example. Linear potentiometers are also used in many digital applications where the proportional relationship between the resistance and the knob position are crucial. Such applications include interfacing with a digital device or MCU via an Analog-to-Digital Converter (ADC). This type of application was implemented in the ADEPT prototype in order to adjust various digitally coded effect parameters by simply adjusting an analog input

voltage with a linear-taper potentiometer. This implementation is discussed in more detail in the Analog-to-Digital Conversion Section below.

3.2.3 Microcontroller

The microcontroller unit (MCU) is the brain behind our project. The main objective of the MCU unit is to control and operate the other devices in the system. The MCU is in charge of modifying the digital signal received from the ADC/CODEC, as well as storing it. Furthermore, the microcontroller will then relay the output digital signal to the DAC/CODEC so it can then be converted into an analog signal that can be further manipulated to make it audible.

As a fundamental element in the design of our guitar pedal, we have decided to use an MCU to process and modify the signal in a digital domain instead of an analog domain. Most guitar pedals in the market are done using analog elements, the reason we deviate from this status-quo lies behind the fact that we want to be able to have an all-in-one guitar pedal, one in which we can apply different kinds of effects stored in memory, whilst still allowing the possibility to continue adding more effects to the list, as long as there's sufficient memory available.

That being said, the more important aspects we look for in an MCU are high clock-speed, for processing large amount of instructions in real-time, SPI/I2C communication, in order to send and receive data between the MCU and the CODEC/ADC/DAC, large flash memory, for storing effects and the firmware (though we can also use an external flash memory for extra space), a floating point unit (FPU), for computing complex algorithms required for the effects, and the capacity to connect to an external clock oscillator in order to keep in sync with the other devices (CODEC/ADC/DAC). Using these points as consideration for picking the correct MCU, we have listed and shortly described a couple of the options we considered.

3.2.3.1 Broadcom BCM2711, Quad-Core Cortex-A72

The Broadcom BCM2711 is the processor utilized in the most recent version of the Raspberry Pi series. This is by far the most power processor we have in our list of potential MCU's, running a clock speed of 1.5GHz the 64-bit ARM architecture is able to handle pretty much anything we throw at it, however, the documentation on handling DSP is very scarce and having to deal with an operating system seems like more of a hassle rather than adding headroom for future expansions.

In addition, when trying to find the cost for purchasing a single chip there were no major retailers that would sell it, only adding more problems to something that should be

straightforward, this might be due to the fact that the processor was probably designed for the Raspberry Pi board rather than a standalone IC.

3.2.3.2 FV-1

The FV-1, although not the most powerful chip in the list, was designed for guitar pedals. The architecture and ISA are optimized for DSP. However, we found that working the FV-1 would bring certain limitations to what we envisioned for our project, in the sense of adding extra features such as having an LCD, a more complex user interface, among other features. Therefore, we decided to look at other chips that could allow us more versatility and headroom. Additionally, the FV-1 programming is not as straightforward as we thought. The language we would write the programs in was in Assembly, with the use of their specific instructions. This would, in part, be a hassle in the process of coding and debugging new effects. Finally, the FV-1 has 64 registers, out of which 32 are available for the user to store audio samples, thus the memory is very limited.

3.2.3.3 MSP43x Series

Among all the microcontrollers we looked at, the MSP430 was one of the first ones we considered due to the fact that we were all already very familiar with the MSP43x family. Thus, having a good background with the microcontroller didn't seem so daunting. Texas Instruments has a wide variety of microcontrollers and also has forums for students to ask any question regarding their projects.

Some of the microcontrollers we have previously worked with such as the MSP430FG4618 and the MSP430G2 LaunchPad were strong candidates. We have seen similar projects such as DIY synthesizers done with this line of microcontrollers, which was encouraging to know regarding the capacity of these microcontrollers to be implemented in real-time digital signal processing. Additionally, Texas Instruments supplies a library for the MSP430 DSP which can be downloaded from their website for free.

These microcontrollers usually come at a low cost, depending on which microcontroller we are talking about the value ranges from \$1.10 (MSP430G2230) to the more expensive type like MSP430F1471IPM which goes for \$11.60 in digikey.

3.2.3.4 Cortex M4

We have looked at a variety of different boards such as the Teensy 3.2, which are very popular among audio projects and the STM32 series that use a 32-bit Cortex M4 processor. Although they vary in different aspects, the overall performance of this processor is outstanding, and it has a lot of backing in the online community regarding

DIY projects of all kinds, not only for audio purposes. It is also very cost friendly, coming at about \$20 dollars for the Teensy 3.2 board itself, the processor on its own (MK20DX256VLH7) costs about \$11.46.

<u>MCU</u>	<u>Clock speed</u>	<u>Memory</u>	<u>RAM</u>	<u>GPIO Count</u>	<u>Cost</u>
FV-1	48 kHz	768 B	4096 B	3	\$17.75
MSP430G2553	25 MHz	256 KB	512 B	16	\$2.38
Teensy 3.2	72 MHz	256 KB	64 KB	40	\$11.46
STM32F446	180 MHz	256 KB	128+4 KB	50	\$7.45

Table 4: MCU Hardware Comparison

Note: When we refer to the Teensy 3.2 we are referring to its processor (MCU) the MK20DX256VLH7.

The Teensy 3.2 and MSP430 proved to be tough contenders regarding the addition of the extra features, both had a fast enough clock speed to do real time processing of audio signals, both had plenty of I/O's to allow us to control different elements, and more importantly, both had a very simple programming methodology. Having a dedicated IDE that can interpret a more familiar language to us, such as C and C++ is very beneficial and paves the way for developing effects and other program applications we considered to implement in the project. The Teensy 3.2 has a floating-point unit, unlike the MSP430, which instead uses a Fixed Point Math Library. Although this last point does not affect the overall decision of picking the right MCU for our application, we thought we would go with the STM32F446 since it covered all the bases we looked for and offered a lot more features we could take advantage of in later developments, on top of the fact that there's a lot of information available regarding DSP and other similar applications.

The STM32F446 line provided a great performance, running a maximum clock speed of 180 MHz, as well as providing a decent amount of memory for storage. Depending on the version of STM32F446, the flash memory will change, however we have decided to utilize the STM32F446 RC, which has a flash memory capacity of 256 kB and an SRAM of 128 kB. Other outstanding features that were useful in our project are the Dual mode QuadSPI interface, the general-purpose DMA, the external memory controller with up to 16-bit data bus, and the debug mode. The STM32 line can be easily programmed using the Joint Test Action Group (JTAG) interface in conjunction with the free IDE (STM32CubeIDE) provided by STmicroelectronics. Furthermore, the STM32F446 RC

package that we decided to use is the LQFP 64, which is more than sufficient for our project.

3.2.3.5 Programming Interface

As previously mentioned, programming on the STM32F446 was done using the STM32CubeIDE. Most of the code that was implemented is done in C in accordance with the required libraries and interrupts for the STM32F446.

In terms of connectivity, the STM32F446 utilizes the ST-LINK/V2 (debugger/programmer) to connect the USB and the JTAG. The STM32F446 has JTAG and SWD (debuggers) interfaces. Both have become an industry standard. JTAG became an IEEE standard (IEEE Std. 1149.1-1990) since 1990, Additionally “On JTAG devices with SWD capability, the TMS and TCK are used as SWDIO and SWCLK signals, providing for dual-mode programmers” [1].

The standard JTAG requires a 20-pin connection. We can opt for using an ST-LINK/V2 that connects through a flat ribbon connector to the 20 pins required, however, a note in page 13, section 4.2 of the user manual [2] suggests using an adapter to reduce the number of pins to 10 [3]. This could be done to reduce the space used in the PCB. Furthermore, external pull-up resistors might be required, however the processor contains embedded pull-up and pull-down resistors, as mentioned in section 5.2.2 [4]. Figure 14 [4] shows the required connections to elaborate this scheme.

Additionally, we can also include USB connectivity in the PCB. This can enable debugging without the use of an external debugger, such as the ST-LINK/V2. We would like to include both of these options in our test PCB board, in the case the USB does not work we can still continue working with the external debugger. The USB 2.0 connectivity in the STM32F446RC has two USB connections, one of which can provide a high speed interface solution.

Furthermore, the debugger or usb port connects directly to the computer, in which the user can program the chip by using the STM32CubeIDE software. The IDE STM32CubeMX can also be used to debug and program the GPIO pins of the device, consequently, when programming the USB interface we can select for what purpose we would like to use the USB connection for; this can range from audio device class to regular communication device class (Virtual Port COM).

3.2.3.6 Memory Mapping

In order to program the MCU we will need to know the specific blocks that are reserved for the required connections, timers and memory. This is an important part in the process

of embedded system programming, since we will need the appropriate addresses for when we are coding the interrupts and functions for our firmware. The following image is a compact representation of the memory mapping in the STM32F446RC. This image can also be found in section 5 [7].

Figure 15. Memory map

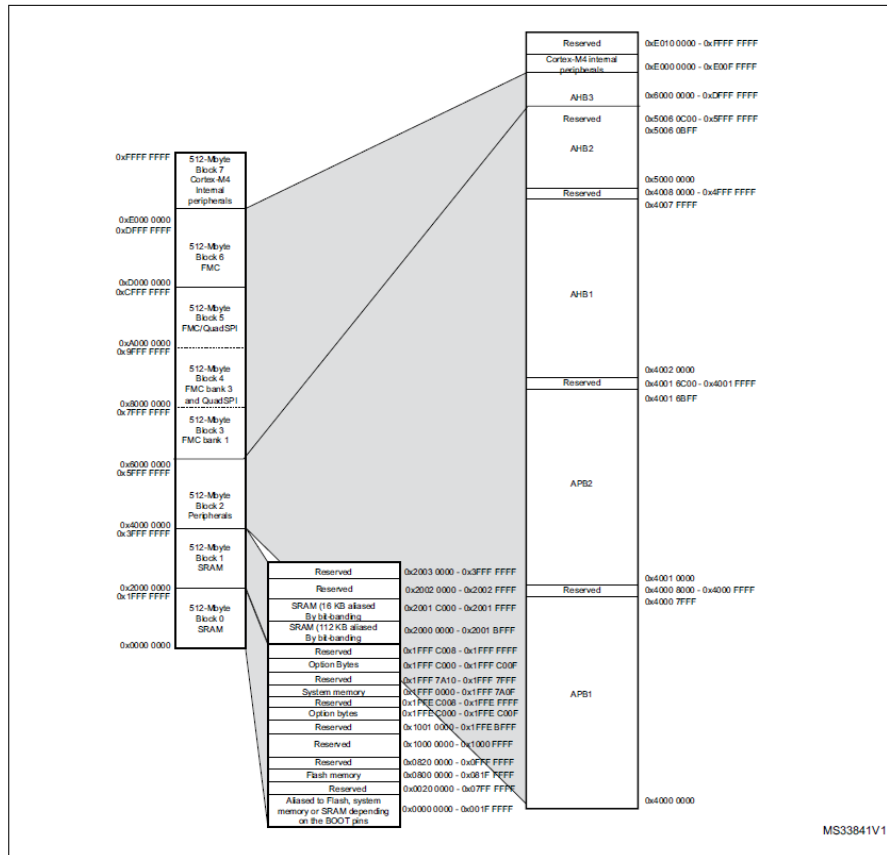


Figure 10: STM32F446RC Memory Mapping Block Diagram

3.2.3.7 Hardware Specifications

The operating voltages should be in the range of 1.7 – 3.6 V. The V_{DD} pins must be connected to an external decoupling capacitor, the suggested type of capacitor as mentioned in section 2.2 [4] is a 100 nF Ceramic capacitor. If no external power supply is used, then it also indicates that we connect the V_{BAT} pin to the V_{DD} pin with a 100 nF external ceramic decoupling capacitor. Since we will not be using the embedded ADC

component then the V_{DDA} is not required to be biased. Our design will have an external CODEC with its own power line. In addition to the ceramic coupling capacitors, the PCB design will follow the guidelines mentioned in section 8 [4] to achieve a high-quality sound. Furthermore, the document also mentions that we should include a single ceramic capacitor (10 μ F) connected in parallel for the power supply.

Based on the current characteristics, table 14 [7], the total sum of all currents (all V_{DD} power lines) should amount to a maximum of 240 mA, ideally each individual V_{DD} power pin should have a maximum of 100 mA. The peripherals we used will have a direct power line from the power supply rather than the MCU.

The figure below displays the power lines supplied by the external power source and the power lines supplied by the MCU. In essence, MCU will only provide power to the LCD component.

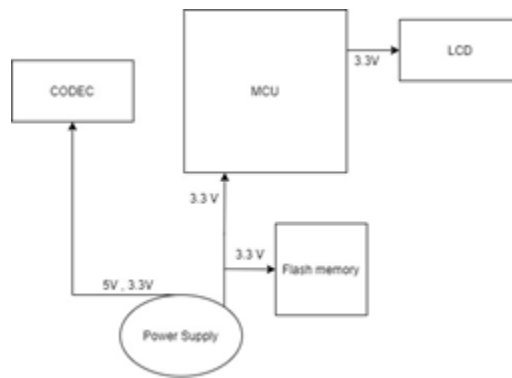


Figure 11: MCU Power Scheme

Because we used the NRST pin for the JTAG, it is important to remember that a system reset was triggered once a low-level signal is detected from the NRST pin. Fortunately, since we are using the embedded resistors, no action is required on our part, since this is done automatically.

We should also notice that there's an option to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the power control register (PWR_CR) as mentioned in section 3.2.3 [4]. This was a good indicator for when we are implementing a system check routine in the firmware. If there's an abnormality in the power supply, we can utilize this to trigger a system emergency shutdown.

As previously mentioned, we have selected the package LQFP64 of the STM32F446RC. There are two main reasons we selected this package: 1) The amount of interfaces required

and 2) Since we are working with high speed interfaces, the signal integrity and noise emission are crucial factors that will affect the quality of the output signal, therefore we want to select a small package that will provide a better signal integrity. This is mentioned in section 4.1 [4].

Since we won't be using all the IO pins, clocks, and counters, we will set all unused IO pins to be "frozen" or disabled. This can be done through the firmware. By doing this we are increasing the EMC performance.

3.2.3.8 I/O

The system will require a couple of peripherals to achieve the following objectives: A user interface that allows the user to select parameters easily on the fly, a visual interface to display the current state and effects being used, a thermal sensor to detect any overheating of the system, and an external reference clock to keep both the CODEC and the MCU in-sync. These different peripherals will require specific schemes for connecting properly to the MCU and avoid any unwanted noise or degrade in the quality of the signal, especially for components such as the CODEC and the external oscillator. The communication protocol used for these peripherals will vary depending on each one. Each one of these peripherals were described in length in their respective sections.

The image below depicts a rough approximation of the pins reserved for all the IO components required from the MCU. These pins are subjected to change depending on whether or not the peripheral picked was changed after further consideration or testing. The STM32CubeMX allows the programmer to utilize a simple GUI to program the pins and it automatically generates the required code for the setup of those pins. It is a very powerful tool that will allow us to easily set up the IO pins required and organize the board in such a way.

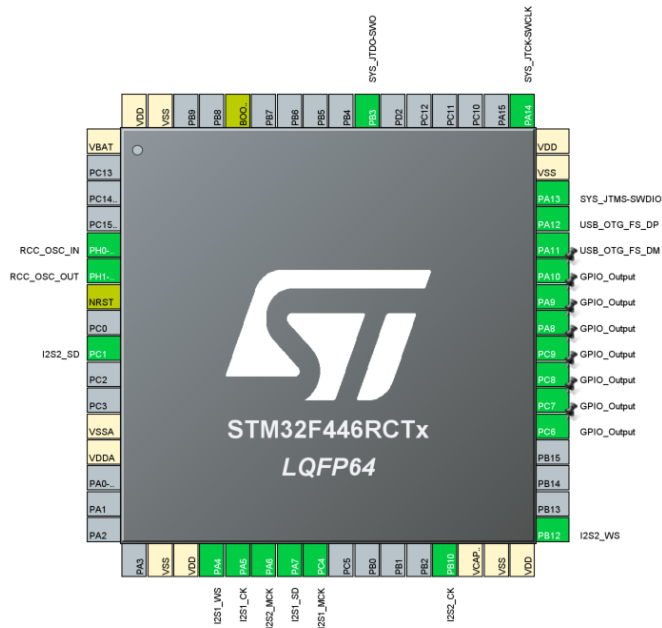


Figure 12: STM32CubeMX Pin Programming Setup

3.2.3.9 External Oscillator

The system required an external clock reference to prevent any phase shift if the data transmitted between the CODEC and the MCU. This will provide a stable and reliable clock rate between the two components. Following the guidelines mentioned in section 6.1 [4]. We used the high-speed external clock signal (HSE) crystal/ceramic resonator. Figure 16 [4] shows the correct way in which the external oscillator should be connected.

Following the recommendations mentioned in [5] for the compatible high-speed resonators, we should find a oscillator that meets a frequency range between 4 – 26 MHz a g_{min} of 5 mA/V and $G_{m_crit_max}$ of 1 mA/V. Whether the resonator is ceramic, or quartz-crystal does not matter as long as it meets these requirements.

As far as the frequency of the crystal itself is concerned, the ideal choice was one that makes it easiest to get the target SYSCLK frequency. Commonly, we thought that having a high crystal clock meant we were able to get the desired requirements which are having a sampling rate of 44.1 kHz, however, the STM32F446RC came with a pair of PLLs.

The CODEC clock frequency required to get 44.1 kHz, was at least 88.2 kHz, based on Nyquist-Shannon theorem, the PCM3060 had a digital input clock (SCKI1 & SCKI2) of 2.048 - 36.864 MHz and a sampling frequency range of 16 - 96/192 kHz. Now that we

had the CODEC clock frequency, we then looked at what frequencies the PLL can generate; looking at the datasheet, the PLLSAI characteristics list: input clock, min: 0.95 and max: 2.10 MHz, and the PLLSAI multiplier output clock, max: 216 MHz. Keeping in mind the division factor M to have the specified PLL input clock values. So, when we decided to pick a more common frequency for external crystals, such as 8 MHz, then our calculations would have been as follow:

Ext. Clock Freq. = 8 MHz, Division Factor = $M = 8$, PLLSAI Input Clock = 1 MHz, thus if we want the SYSCLK running at 180 MHz we would multiply out PLLSAI by 180, to achieve a total SYSCLK frequency of 180 MHz.

The reason behind choosing an 8 MHz external clock instead of a 25 MHz was very simple. First, 8 MHz external clocks were more common than 25 MHz, especially finding one that is compatible with the STM32 family. Second, If we knew our target sampling frequency is that of 44.1 kHz, a 25 MHz crystal will have a 0.02%, because the closest it can come to 44.1 is 43.989 kHz. If we increase the sampling frequency to 48 kHz the error would increase to -0.27 %, which is not good at all for audio applications.

Furthermore, based on suggestions given by STM engineers, we decided to change the connectivity scheme. Rather than using the crystal as both MCU's HSE and the CODECS SCKI, we will, instead, use the external oscillator for the MCU only, then through I2S mode of the SPI peripheral we can handle the generation of the 44.1 kHz, and additionally, it also has an output called MCK, which generated a clock that is 256 times the sampling frequency (44.1 kHz), the MCK could be connected directly to the SCKI pin of the CODEC.

Further inspection during the implementation of I2S for the MCU and the CODEC revealed that the MCU cannot act as a master for the CODEC, therefore we decided to use the CODEC as master for the I2S communication. Although 44.1 kHz is good enough for audio applications, the CODEC we have chosen can operate at a frequency of 48 kHz. The only difference in the connection scheme is that there will be no MCK lines required since the CODEC is provided with an external high speed clock.

3.2.3.10 Heartbeat Program

The heartbeat program was a simple way to detect whether the system is working correctly. It is a simple blink LED program. This was a good way to start the debugging process and know if the external clock was working, which was a good way to isolate whether the problem is the clock or not.

3.2.3.11 Liquid Crystal Display (LCD)

Part of our goal for this project was to have a reliable guitar pedal that allows the user to fully interact with all its features, whether it was the state or effects they have selected. To make the experience less daunting and complex, we desired the user to have a visual interface to interact with. The LCD displays the current settings selected whether it be by the footswitch or the user interface parameters. There's a lot of options that we were to select from, but having a high-end LCD was not necessary. If the display could show the select effect and state of the pedal it would be more than enough. This was achieved with an 8-bit character display, 4 lines by 20 characters, ideally powered by the same voltage level as the MCU (3.3V), or in the case we want to design a more minimal looking interface, we can always go with the common 16x2 LCD with a 5V or 3.3V power rail. The scheme for implementing the latter one is the same as that for implementing a 20x4, though the only difference is the available amount of characters and lines that can be displayed, so at the end is a matter of how much information we want to display for the user, and since we aren't looking to make a complex user interface we believe that the information displayed was very brief and straightforward. A common 16x2 LCD would be the HD44780.

However, if we wanted to implement a more elegant and aesthetically pleasing visual interface we would use a pixel addressable LCD, such as the LCD-TFT display controller (LTDC), which also enables having a more interactive interface in which the user can touch the screen to select the desired settings.

3.2.3.12 Flash Memory

Memory for our project was one of the most important points we touched on, other than DSP. Since we were loading a lot of different features on the firmware such as pedal state and effects, each one of these required a lot of storage. Additionally, one of our goals was to provide an all-in-one guitar pedal. One of the most praised features found in the world of guitar pedals was having a looper. Being able to store a sample on the fly is usually a must have when building a guitar pedal board.

In order to have a long enough sample storage we required an external flash memory, since the 256 kB embedded on the MCU wouldn't be enough to store long samples. The goal of our pedal is to record audio at 48 kHz. At this sample rate the data transmission between the MCU and the flash memory is 24 bits, therefore, a 1 second track * 48,000 samples/second * bits/sample = 1,152,000 bits (144 kB). 144 kB of memory is quite a lot for a 256 kB storage capacity, and 1 second wouldn't cut it for a live performance where the samples are usually between 30 seconds - 1 minute.

3.2.3.13 User Interface

The user interface was composed of a set of parameter knobs, a selector knob, and a pair of footswitches. In essence, this was the primary way in which the user will interact with the pedal. The set of parameter knobs allows the user to adjust the selected effect to its liking, for every effect will have designated adjustment parameters mapped to the 3 knobs. Most of the effects have a depth percentage, and a rate percentage that define the overall conversion of the signal once processed through that effect in particular.

The selector knob was used to travel through the different effects and states of the pedal, this can either be in the effect state or the loop state. The way the selector was encoded depended on how we planned the GUI to display on the LCD. Moreover, the selector provided values to the MCU. There are many ways in which we implemented and approached the selector, however our goal was to make it as easy and convenient for the user such that he/she doesn't spend too much time travelling through the menu.

Furthermore, our project wouldn't be called a guitar pedal unless we implemented footswitches, therefore we decided to build our program around 2 switches. One of the switches was a hard switch, which by default once pressed it changes its state and remains in it until pressed again, we can say that the switch acts a "latch" sort of speak. The second footswitch is a soft switch, which unlike the hard switch which "latches" this pedal only a triggers a momentary change of state and then it goes back to its initial default state, this was a convenient way for implementing functions such as the start and stop of a loop sequence, or using it a tap tempo in order to adjust an effect rate or to change the overall tempo of the recorded samples.

Additionally, the tone section was also included as part of the user interface. Although the tone section is not connected to the MCU, it was still part of the overall design of how we implemented the user interface.

3.2.4 Analog-to-Digital Conversion

3.2.4.1 ADC (Analog-to-Digital Converter)

In order to be able to take in the initial signal from our instrument to then be modified or altered to create the desired effects, we needed to convert that raw analog signal into a digital signal with discrete values that can be processed. For this, we needed an Analog-to-Digital converter implemented in our design.

An ADC works by taking samples of the input analog signal at several time intervals. Each of these samples was then assigned to a specific discrete value inside a range of voltages, and in this way generates a step stair wave that represents the original analog signal and can be interpreted by a computer. There are many different types of ADCs, and

each achieved their functionality with different implementations. We go over a few of these architectures and their advantages later on.

Since we are dealing with sound, quality is a top priority. Thus, we needed the original signal to be sampled and translated as accurately as possible. To achieve this, we chose an ADC by considering the following characteristics: Resolution, Accuracy, Sampling Rate and Noise.

The resolution of a converter is the number bits utilized to represent the signal for each conversion cycle. The greater the resolution, the more precise the measurement values. For example, an 8-bit ADC was able to express the signal using 256 different discrete values, while a 16-bit ADC had 65536 values at its disposal to represent the same signal. Related to resolution, accuracy refers to how close a converter's output comes to representing its maximum theoretical resolution. Below is an image that illustrates the effects on a signal as resolution increases.

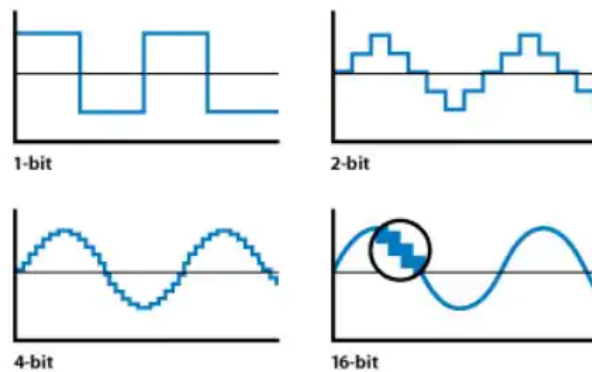


Figure 13: Digital Signal Resolution

(Example on how resolution affects the digital signal. (Source: Apple Inc – Soundtrack Pro 3: Audio Fundamentals))

The sampling rate is the frequency at which the converter samples the input signal, usually expressed in Hertz (Hz). If the sample rate is too low, not enough data points were recorded and therefore, the resulting digital signal is not representative of the original analog waveform. To avoid this, we needed to consider the Nyquist theorem concept, which suggests that for obtaining accurate data, our sampling frequency needed to be twice or larger than the highest frequency of our incoming signal. The image below is an example of how a low sampling rate can wrongly represent a signal.

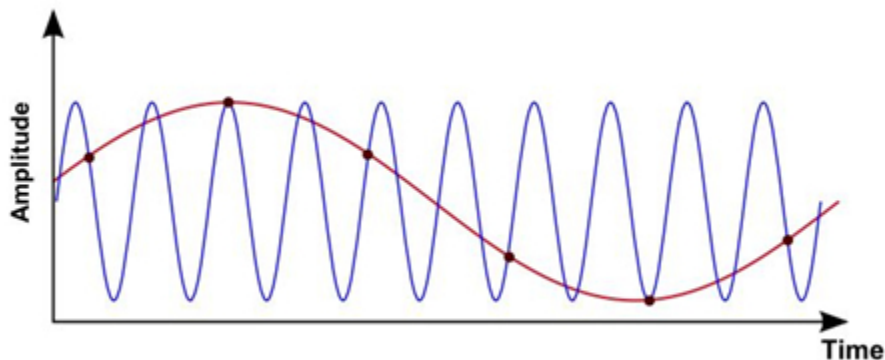


Figure 14: Aliasing Effect
(Example of Aliasing effect, blog.dataphysics.com)

Another two important specifications to classify converting devices have to do with performance. These are the SNR (Signal-to-Noise Ratio) and THD+N (Total Harmonic Distortion + N) measurements. The first is computed as the ratio of the signal power and noise power of the entire signal. This measurement is commonly expressed as a dB value, and a greater value, which indicates that the signal is more prominent than the noise, and a lower value tells us that the noise measured is more dominant, with a chance of the relevant signal getting lost in all this noise.

THD + N is the measurement given by the sum of the first 9 harmonic voltages of the signal plus the noise for these intervals, all divided by the RMS signal voltage. This value can be represented as a percentage or in dBs as a negative number. This value told us about the relationship between harmonic distortion (unwanted overtones) to the total signal in dB. If given as a value in dB, the more negative the value indicates less overall distortion which is desired.

When it comes to noise in a converter, the most predominant is that of quantization noise, which is caused by small differences between the analog input voltage being sampled and the specific bit-resolution of the ADC. Oversampling is one of the most effective techniques to deal with this noise, where the incoming signal is sampled at a frequency that is a large multiple of the Nyquist rate. This method also offers several benefits such as greatly reducing aliasing and creating a better SNR, which translates into a more desirable bit-resolution.

Given all the different comparison points and parameters discussed, we referenced these to talk about potential different devices. After making some comparisons, we made a decision on a single device and explained why it fitted our needs the best.

3.2.4.2 Choosing an ADC

There are many ways in which ADCs can be implemented, but after considering several different architectures, two that stand out for their characteristics aligning with our design specifications were the Delta-Sigma and the Successive-approximation (SAR) converters. Both architectures are able to effectively minimize quantization noise, but they differ in their resolution and sampling rate specs. The Delta-Sigma converters offer greater resolution capability, but slower sampling speed, while SAR converters can achieve much higher sampling speeds, in turn of a lower resolution. To make a final decision, we considered how fast of a sampling rate is sufficient for audio processing.

The typical sampling rate for audio processing is 44.1 kHz, and even for High fidelity audio, the rate used is about 96 kHz. Considering this, the Delta-Sigma architecture was able to achieve this speed with ease, as well as to offer that higher resolution. This specific design makes use of oversampling, in addition to decimation filtering and quantization noise shaping, to output a low-noise, high resolution bitstream that can describe an acoustic signal with great precision.

The final design we specified needs to be as compact as possible in order to meet our size requirement specifications. Because of this, we needed to examine the possibility of using two separate chips for ADC and DAC, versus using one chip that features capabilities for both conversions, called a CODEC. Today, these kinds of devices are very capable of producing quality sound with minimal noise conversions, while running on ultra-low power modes, making them extremely efficient and a very desirable choice. We explain the CODEC in its own section. Taking all of this into consideration, there were a couple of options for our ADC chip that served our needs:

3.2.4.2.1 ADS1675IPAG

This is a high-speed, high-precision, 24-Bit ADC by Texas Instruments that can operate at speeds of up to 4MSPS (Million Samples per Second) employing an advanced Delta-Sigma architecture. It uses serial control communication, which allows it to communicate with a wide range of microcontrollers, DSPs, and FPGAs. The chip operates from an analog input supply of 5V, and a digital supply of 3V, while dissipating 575mW of power.

3.2.4.2.2 TLV320ADC3100

This ADC from Texas Instruments is a 24-Bit Stereo audio chip that can achieve a sample rate of 96kHz. This chip also features on-board digital filtering such as Low-Latency IIR and Linear Phase FIR filters, as well as a programmable high-pass filter. This chip supports many different Audio Serial data communication protocols such as I2S, DSP, and TDM Modes. The TLV320x series operate on an analog voltage supply

of 3.6V, and a digital supply of 1.95V and can consume anywhere from 6mW to 17mW depending on the power mode selected.

3.2.4.2.3 PCM186x

This last chip is also from Texas Instruments, and it is a similar Delta-Sigma 24-Bit ADC chip, but now with a maximum sampling frequency of 192kHz. It has 4 different independent channels available to perform conversions, and is controlled by using either I2C or SPI. When it comes to audio serial protocols, it supports I2S and Left-Justified modes. This chip employs a single 3.3V Power-Supply operation, and has a power dissipation ranging from 85mW to 145mW.

<u>Name</u>	<u>Max Sample Frequency</u>	<u>Communication Protocol</u>	<u>Cost</u>
ADS1675IPAG	96kHz	Serial	\$3.93
TLV320ADC3100	96kHz	I2S, I2C	\$3.08
PCM186x	192kHz	I2S, LJ, SPI, I2C	\$39.14

Table 5: ADC Hardware Comparison

3.2.4.3 DAC (Digital-to-Analog Converter)

Once the digital signal was processed by our DSP chip and altered to create the desired audio effect, we needed to transform this signal back into analog to be able to clean it up and eventually hear the result. To perform this binary to analog conversion, we implemented a DAC (Digital-to-Analog Converter) chip into our design.

DACs operate by first taking the digital signal consisting of many discrete voltage values to create a stair step wave. This wave contains a series of small gaps in between each digital reading, which then the converter closes by using a method called interpolation. This process consists of looking at the two values of each neighboring digital reading on the stair step wave and calculating the values in between them. By performing this operation for all values in the digital signal, the DAC is able to produce a smooth waveform that is now an analog signal.

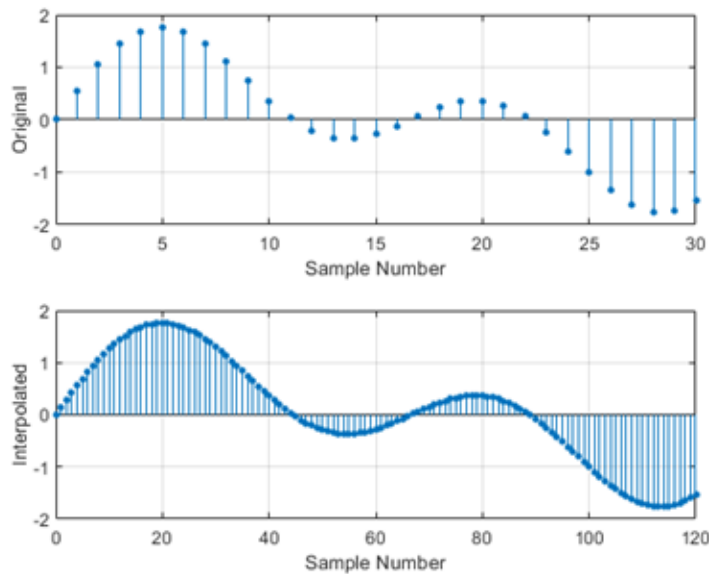


Figure 15: Interpolation of Discrete-Time Signal

(Interpolation of discrete-time signal, Mathworks.com)

When it comes to audio applications, there were two main DAC architectures that are mainly employed: R/2R and Delta-Sigma. The first kind is based on a resistor matrix which uses base resistor values of R and 2R arranged in a ladder configuration that converts the values of input samples from code to proportional voltage values. These resulting voltage values are then passed through an analog filter that essentially smooths out and connects these values, transforming the stair-step signal into an analog signal.

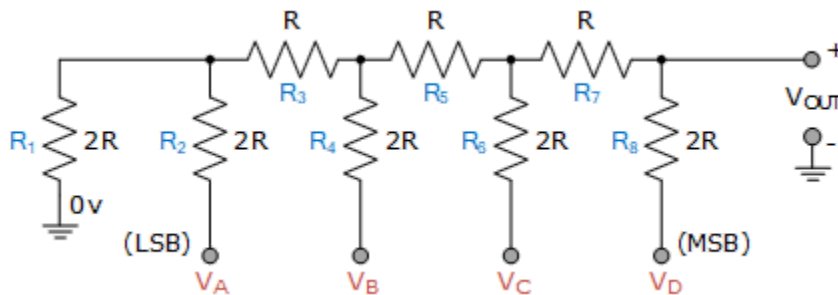


Figure 16: 4-Bit R-2R Resistor Ladder Network

(4-bit R-2R Resistor Ladder Network, electronics-tutorials.ws)

The DAC Delta-Sigma architecture works in a similar way than in an ADC, as described earlier in its respective section. However, for a DAC, the converter is mostly digital, and it performs all the operations described for the ADC, but now in reverse. The first stage

of this architecture is an interpolation filter which receives data at a low rate, and outputs digitally filtered data with a much higher rate. The second stage is a Delta-Sigma modulator that acts as a low filter for the signal, but as a high filter for the quantization noise, and converts the signal into a high-speed bit stream. The third stage is responsible for converting the incoming bit stream to reference voltage values, which are then passed through a simple LPF filter to obtain the desired analog signal in the last stage.

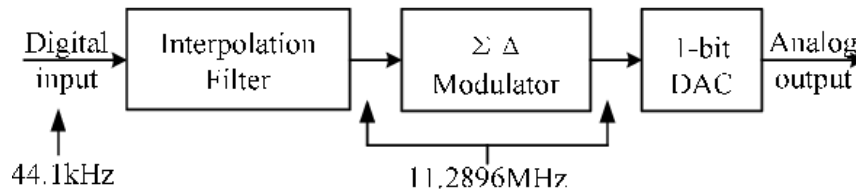


Figure 17: Block Diagram of 1-Bit $\Sigma\Delta$ DAC
(Block Diagram of 1-Bit $\Sigma\Delta$ DAC, Semanticscholar.org)

When comparing the two mentioned architectures, we found advantages and disadvantages for each. The R/2R has a simple design advantage which at first sight, indicated that there was less risk for unwanted noise, due to the fact that it had fewer, simpler components. Nevertheless, each of the matrix resistors in this design had a value deviation, which translated into non-linearity issues when converting bit values into voltages that degraded sound quality.

Now, for the Delta-sigma architecture, one of the key advantages was that the modulator gets rid of all non-linearity issues, and can deliver linear conversion from input to output. This type of DAC was also very simple in design, since it relied on mostly digital components to perform the conversion, which also gets rid of any potential physical issues like overheating. An issue that did not arise, was that of overload due to the linear nature of this design. More modern implementations have different ways of overcoming this issue, a common one being noise-shaping.

Looking at all the pros and cons of each architecture, we concluded that the R/2R had a simpler, more analog focused design, but came with some setbacks that could reduce precision and therefore sound quality. The Delta-sigma architecture possessed some great advantages when it came to noise reduction, as well as taking away many of the physical problems that come with mainly analog designs.

3.2.4.4 Choosing a DAC

To choose an appropriate DAC we considered the same parameters we did for choosing our ADC. Therefore, choosing a DAC with sigma-delta architecture seemed to be the

right approach. This architecture offered many benefits for audio applications that delivered great quality of sound without having to deal with too many noise issues.

Some of the DAC chips that we considered were the PCM1742 and the PCM1789. These were both high-performance single-chip devices with up to 24-Bit and 32-bit resolutions, Delta-Sigma modulators, and can achieve a sampling rate up to 192-Khz. There was also the TLV320DA chip, offering Stereo 32 Bit resolution at 192KhZ sampling rate. Our goal when it came to resolution is 24 Bits, anything above that seemed to be unnecessary and it was said that it did not make a difference in the sound quality of the output analog signal. Next, we presented a few of the DAC options we had under consideration:

3.2.4.4.1 PCM1789-Q1

This chip is a 24-Bit Stereo Sigma-Delta DAC, with a sampling rate up to 192kHz. It contains an Analog Low-Pass filter as well as a 8x Oversampling Digital Filter to greatly reduce any noise in the output signal. For audio interfacing, this chip is capable of I2s, L/R-Justified, or DSP. For Mode control, it can do SPI, I2C or serial communication. To power this component 5V is required for Analog, and 3.3V for Digital, with a power dissipation in the range of 160mW.

3.2.4.4.2 PCM1742

This Texas Instruments DAC has 24-bit Stereo data and can achieve sampling frequencies in the range from 5kHz to 200kHz. Embedded, there is a 4x/8x oversampling Digital Filter to help with noise reduction. This device is capable of I2S and Left-Justified audio formats, as for mode control, it has only serial communication. As most devices of this kind, it operates with a 5V Analog power supply, and a 3.3V Digital supply, with a power dissipation of around 85mW -100mW.

3.2.4.4.3 TLV320DAC3101

This is a low-power DAC that can do up to 32-bit length data, at a sampling rate of 192kHz. It includes user-programmable Biquad and FIR digital filters to achieve high quality output. It also contains a programmable PLL (Phase Locked Loop), for flexible clock generation. This chip supports L/R-Justified, DSP, and TDM audio interfaces and I2C for mode control. To power this device, it requires 3.3V for Analog and Digital, and a power dissipation of around 50mW.

<u>Name</u>	<u>Resolution</u>	<u>Max Sample Frequency</u>	<u>Communication Protocol</u>	<u>Cost</u>
PCM1789-Q1	Stereo 16, 20, 24, 32 Bit	192kHz	I2S, SPI, I2C	\$4.81
PCM1742	Stereo 24 Bit	200kHz	I2S, Serial	\$3.08
TLV320DAC3101	Stereo 32 Bit	192kHz	I2S, I2C	\$3.27

Table 6: DAC Hardware Comparison

3.2.4.5 CODEC

The third type of component we implemented into our signal processing section was a CODEC, which stands for “CODER/DECODER”. This is a single chip device that is capable of performing both A/D and D/A conversions. This actually brought several benefits such as overall lower power consumption, and decreased the physical footprint of this system to leave more room available in the board for other components. Since we wanted to make our final prototype more portable, a CODEC helped us get closer to that goal.

A CODEC chip works simply by having both internal ADC and DAC chips that can usually work either synchronized or in an asynchronous manner. For this, when looking for a suitable CODEC, we examined the same parameters as in earlier sections: Resolution, Sampling Rate, Architecture. From previous research we learned the benefits of employing Delta-Sigma converters for audio applications such as ours, so we will stick to that. We also looked for A/D and D/A of at least 24 Bit resolution and a sufficient sampling rate.

3.2.4.6 Choosing a CODEC

After considering many different alternatives that met the requirements we needed for our D/A conversions, we had a few good options to choose from.

3.2.4.6.1 MAX98050

The first option was a Low-Power CODEC from Maxim Integrated which featured both ADC and DAC with stereo 32-Bit resolution and a maximum sampling frequency of up to 384kHz. It is capable of generating low noise output signals and avoiding playback disruption by employing on-board programmable Low-Latency Digital filters. For audio

interfacing, it can do I2S, Left-Justified, or TDM communication. For mode control, it uses I2C. When it comes to power supply, it requires 3.6V for Analog and 1.8V for Digital. This chip is mainly used for portable or wearable devices such as headsets, therefore it has very minimal power consumption.

3.2.4.6.2 PCM3060

This is another high-performance CODEC device from Texas Instruments which has as main characteristics a resolution of 24 bits for both ADC and DAC, and a sampling rate that can be set up to 92Khz for the ADC and 192kHz for the DAC. This stereo audio chip uses a delta-Sigma architecture, and is capable of I2S, L/R Justified audio interfacing with 16/24bits. For mode control, we have the choice between SPI or I2C serial control interfaces. This chip also featured anti-aliasing and decimation filters for its ADC, as well as a low-pass and oversampling filters for the DAC. When it came to power, this device required a typical 5V for analog and 3.3V for digital voltage lines and has a power dissipation of around 82mw. Lastly, it has typical SNR values of 99dB and 105dB, as well as -93dB and -94dB for ADC and DAC respectively.

3.2.4.6.3 TLV320AIC23

This device by Texas Instruments is a CODEC capable of data transfers up to 32-Bit in length, and with maximum sampling rates of 96kHz. The ADC features a third-order multibit architecture with a 90dBA SNR, and DAC uses a second-order multibit architecture with up to a 100dBA SNR. This is a Low-Power device that is able to output high-quality audio playback while consuming less than 23mW. For audio interfacing, it can use I2S and Left-Justified formats. For mode control, it is able to do 2-Wire SPI as well as TI McBSP Serial port.

<u>Name</u>	<u>Resolution</u>	<u>Max Sample Frequency</u>	<u>Communication Protocol</u>	<u>Cost</u>
MAX98050	Stereo 16, 24, 32 Bit	192kHz	I2S, I2C, SPI	\$2.98
PCM3060	Stereo 24 Bit	96kHz	I2S, I2C, SPI	\$5.69
TLV320AIC23	Stereo 16, 20, 24, 32 Bit	96kHz	I2S, SPI, Serial	\$10.15

Table 7: CODEC Hardware Comparison

For all the reasons listed throughout the last few sections, we considered that our best choice was to use a CODEC audio chip, to be able to save some physical footprint, make

connections simpler, and build a power efficient guitar pedal. The CODEC that we selected is the PCM3060. With 24-Bit resolution, enough sampling speed of 96kHz, both SPI and I2C for control as well as the I2S audio interface, combined with an affordable price point, made it the most fit component to adjust into and benefit our design.

3.2.4.7 Power

The digital and analog power supply lines of the PCM3060 were bypassed to the according ground pins with 0.1uF and 10uF capacitors as close to the pins as possible, to increase performance of both the ADC and DAC. Even though there were two power lines, these were generated from one source of 5V for both a Vcc of 5V and Vdd of 3.3V. This was to avoid any potential issues generated from incorrect supply sequencing.

3.2.4.8 I/O

For the CODEC chip to connect and talk with the microcontroller, it uses I2S, a communication audio interface that is based on a 3-wire connection. A BCLK for serial clock, a LRCLK for Word Select, and a DIN for data in, or DOUT data out. This chip has the following pins: BCLK1, LRCK1, DOUT for the ADC, and BCLK2, LRCK2, DIN for the DAC.

The input of this device was an analog signal, which was first processed by the on board ADC. The analog input gets read in by using the 2 pins VinR and VinL. This data was used to generate a digital step wave that was sent to the MCU to modify using the DOUT pin. After modifying the digital wave, such was sent to the DAC using the DIN pin, where it was converted back to an Analog signal. The resulting signal is finally outputted by using the VoutL and VoutR independent output channels.

3.2.4.9 Analog Peripherals

The CODEC was responsible for interfacing with the various analog and digital peripherals present within our design. One of the many design requirements of the ADEPT is being able to provide a way to control various effect parameters with the simple turn of a knob. In order to do this, we interfaced with the MCU via the CODEC's ADC input.

This design aspect was executed by implementing the use of a potentiometer that was placed in between a reference voltage source of 3.3V and the analog input of our CODEC. This provided a way for an analog parameter (voltage) to control a digitally coded parameter via the Analog-to-Digital Conversion provided by the CODEC.

After some experimentation during development, we found out that it would be a better option to leave the CODEC dedicated only for the processing of the audio signals. Hence, to implement the parameters dials as intended, we used instead the 12-bit ADCs that are part of our MCU in order to read the 0V to 3.3V range outputted by the potentiometers.

Before being replaced by MIDI (Musical Instrument Digital Interface), Control Voltage (CV) was an analog standard that was very commonly used in vintage synthesizer circuits in order to externally control various auditory and effect parameters onboard the synth, such as filter cutoff, LFOs, and arpeggiator tempos. We were using a similar application here to interface with the MCU's digital parameters.

It was important that the potentiometer we used for this application has a linear taper. A linear taper potentiometer provided a consistent and smooth transition for the reference voltage values going from 0V to 3.3V as the potentiometer is turned. If a logarithmic potentiometer was used, the parameter in question would be increased/decreased in a nonlinear fashion that would feel unnatural to the user. For example, if implementing a distortion effect, we would want the gain to increase in a linear fashion, where the halfway point of the knob would offer 50% gain. A logarithmic potentiometer, in contrast, would very slowly increase the gain for the first 50% of the knob's turn, and then very quickly and unnaturally jump up to max gain after the halfway point of the knob's turn. This undesirable behavior proved the need for the implementation of a linear potentiometer for our particular application.

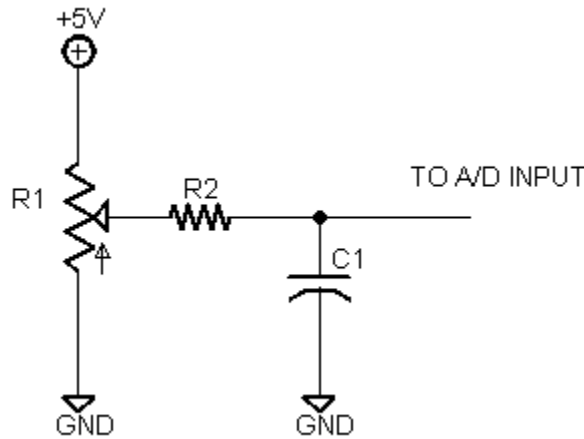


Figure 18: Using Variable Resistor as Control Input

3.2.4.10 Communication

We utilized I2S communication protocol in order to transfer the data from and for the MCU and the CODEC. I2S is a standard communication protocol that is used to communicate PCM audio data, this communication standard is composed of a 3-line bus for serial and synchronous data transmission. The data is transmitted through the SD line, synchronized by the rising or falling edge of the SCK, and falling edge of SCK for the receiver. The WS (word select) line is used to determine if the word being sent is the right or left one, this is needed since the data represents stereo digital sound, so each sample contains two words. The scheme implemented for this is transmitting each word over half a sampling period, therefore the sampling rate is doubled and these two words are transmitted per period.

The PCM3060 has two system clock inputs, one for the ADC and the other for the DAC. Since we are using an external clock as our master clock, the two clock inputs were connected to an external 12.288 MHz external oscillator. Similarly, the MCU will also act as a slave to the external clock. Therefore, there were 2xI2S channels required from the MCU. The other two channels: WS (LRCK1 and LRCK2) and the SD (DIN and DOUT) were connected directly between the MCU and the CODEC.

Similar to the diagram depicted in section 1.1, figure 2 [6], we had a similar connection using the external clock as the controller for the 2xI2S channels.

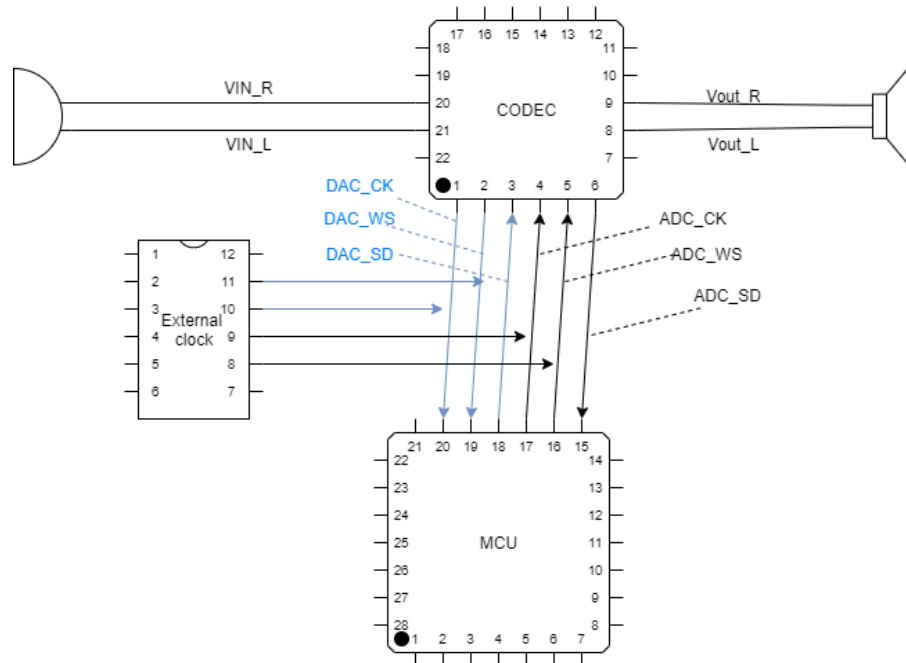


Figure 19: I2S Controller Pairing

Note: *The pins are not the corresponding ones to the actual target connections.*

In order to set up the CODEC correctly we first have to configure it using the I2C connection and the proper instructions required to configure the CODEC's operation mode and sampling frequency desired. In our case, we ended up establishing a sampling frequency of 48kHz (per instructions given on the datasheet and based on the external clock supplied to the CODEC, which is 12.288 MHz). Additionally, the communication protocol we utilized for the CODEC is I2S. This being a half-duplex line based on the pins provided for the SPI line. Once we have enabled I2S, then those SPI lines are disabled for the SPI protocol. Furthermore, the I2S connection is configured in the MCU, and we set the MCU to be a slave for this communication.

3.2.5 Tone Section

The next section of our circuit that comes directly after the digital effects processing stage of our effects pedal is the tone section. This section was responsible for shaping the overall tone of the effect. A tone control on any piece of audio gear is generally defined as a simplified method of equalization that can directly affect the amplitude of low, middle, and high frequencies in the human hearing range.

You can often find active multi-band graphic equalizers in many high-end stereo systems. This type of equalizer can amplify (or attenuate) very specific frequencies (in Hz) along the audio spectrum. This differs from a passive tone control in that a passive tone control does not control individual frequency bands, but rather rolls off the frequencies above or below a certain cutoff frequency. The difference between active and passive electronics is that passive circuits do not require a power source. We observed both types in this section.

In its simplest form, a passive tone control is created by using an RC (resistor and capacitor) filter circuit. These filters can be either low-pass or high-pass. When a low-pass circuit is implemented, high frequencies are cut off and the tone will become warmer or darker as the potentiometer (variable resistor) value is increased. When a high-pass circuit is implemented, low frequencies are cut off and the tone will become thinner or brighter as the potentiometer (variable resistor) value is increased. The cutoff frequency of these filters in Hz were calculated by using the equation $f = \left(\frac{1}{2\pi RC}\right)$.

Since the four octave frequency range of a standard six-string electric guitar falls anywhere between roughly 80Hz - 1200Hz, we wanted to implement a tone control circuit that was able to tame the unwanted frequencies within this range. With the typical frequency of the open "A" string on a guitar being 440Hz, we used this as a guide for our frequency manipulation in our circuit. Other instruments that were compatible with the ADEPT pedal, such as piano, have a wider frequency range (roughly 30Hz - 4000Hz), but the fundamentals of our tone control remained the same.

Various tone controls have been used in guitar pedal circuits throughout the years, including those used in the Marshall “Bluesbreaker”, the Electro-Harmonix “Big Muff”, and the Ibanez “Tube Screamer”. All of these tone controls do essentially the same thing, but in slightly different ways, with incrementally different frequency responses. In order to make the best choice as to which type of tone control configuration is the best suited for the ADEPT prototype, we compared these 3 examples below to outline their pros and cons, and their most common applications tonally.

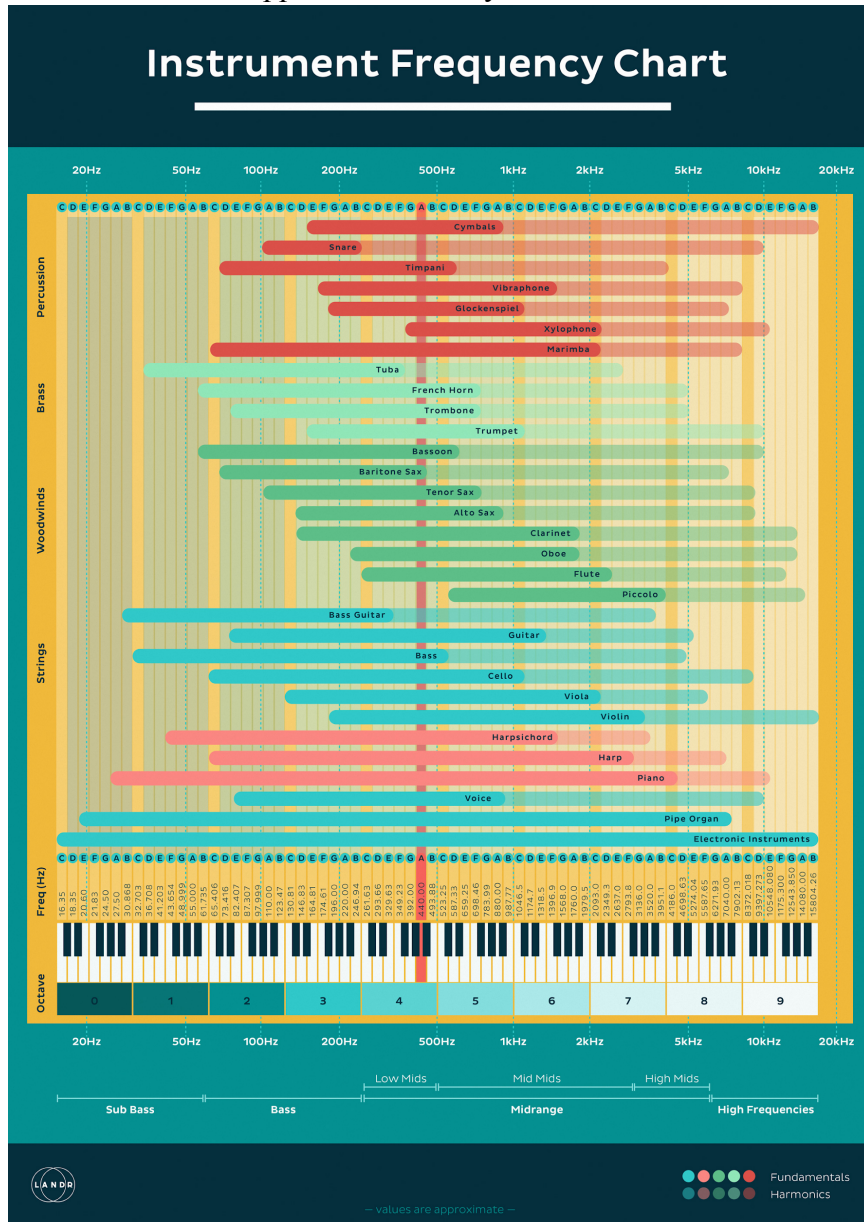


Figure 20: Musical Instrument Frequency Range Chart

3.2.5.1 Bluesbreaker Tone Control

The “Bluesbreaker” style of tone control very simply uses a single passive RC low pass filter circuit to roll off the high frequencies anywhere above 400-500Hz. This option was certainly the easiest to implement, and uses the fewest amount of components in order to achieve the desired effect. This simple passive low pass filter circuit is also the same circuit that is implemented in the onboard pickup tone control of an electric guitar. Some other guitar effects pedals that use this type of tone control configuration include the ProCo “RAT”, the AnalogMan “King of Tone”, and the JHS “Morning Glory”.

As mentioned above, this was a very simple circuit to implement, and uses very few passive circuit components. This made for a very inexpensive tone control implementation, which was desirable to our team in order to afford the costs of the more expensive digital components, such as the MCU, ADC/DAC, and LCD screen. However, since the ADEPT pedal was intended to be a very diverse multi-effect that covers many different sonic characteristics, we considered the desire to have a more complex tone circuit that can boost or cut specific frequencies. So we analyzed another type of tone circuit that has these features in the following section.

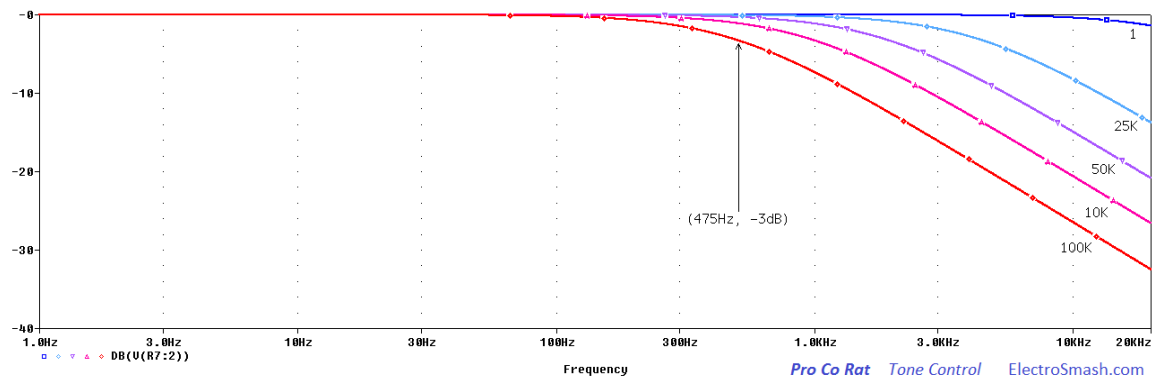
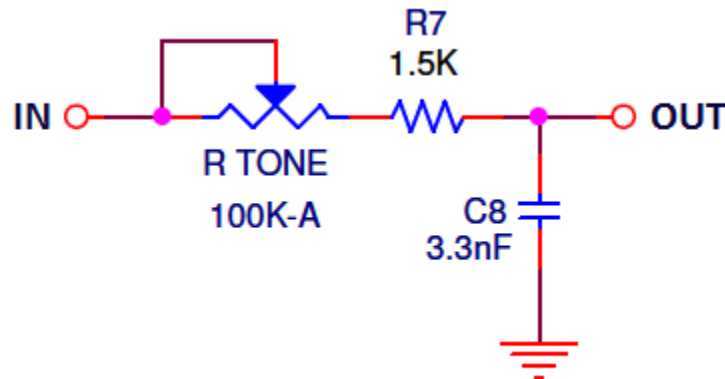


Figure 21: Bluesbreaker Tone Control Frequency Response



Pro Co Rat Tone Control ElectroSmash.com

Figure 22: Bluesbreaker Tone Control Schematic

3.2.5.2 Big Muff Tone Control

The “Big Muff” style implements both a passive RC low pass filter and a passive RC high pass filter simultaneously. These two filters are blended between one another using a single potentiometer. Because of this unique design, and depending upon the specific component values used in each of the filters, this type of tone control has a very characteristic feature of either a “mid-boost” (mid frequencies increased) or “mid-scoop” (mid frequencies decreased). This can be seen by analyzing the frequency response of the circuit.

Mid frequencies are characterized by any frequencies that lie between 300Hz - 5000Hz. Most of the musical instruments in a rock/pop band lie within this frequency range, including the electric guitar. However, too much mid frequency presence can contribute to an overall mix that is “muddy” or lacking in sonic detail, making it difficult for the human ear to point out individual instruments in a recorded audio mix.

Since the frequency range of the electric guitar lies within this “midrange”, many guitarists will attempt to add clarity to the band mix by cutting these mid frequencies from their signal, making room for the frequencies of other instruments to stand out during live and recorded performances. This is the case with the Big Muff’s tone control.

While this tone control is very unique to the Electro-Harmonix Big Muff Pi, many “clones” of this pedal have been made by other pedal companies. Some of these clones include the Earthquaker Devices “Hoof”, the Way Huge “Swollen Pickle”, and the JHS “Muffuletta”.

This tone circuit is only slightly more complicated than that of the standard passive RC low pass filter shown in the previous section. Because of the limited number of parts, the affordability remains the same. There is also the added feature of the mid-frequency

cut/boost, as explained above. However, there was one more type of tone control that we observed, in order to have enough options to choose from when it came to deciding on our tone control implementation for the ADEPT prototype.

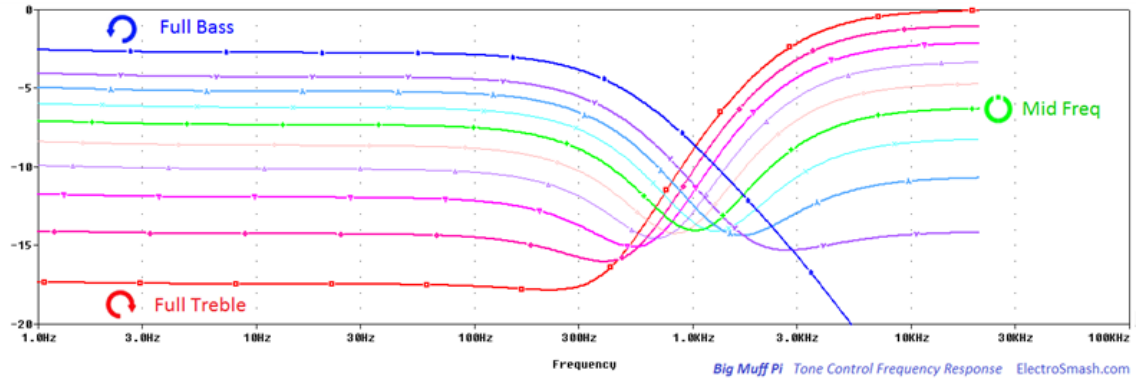


Figure 23: Big Muff Tone Control Frequency Response

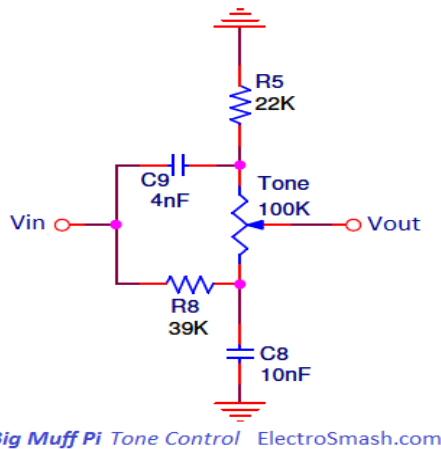


Figure 24: Big Muff Tone Control Schematic

3.2.5.3 Tube Screamer Tone Control

The “Tube Screamer” tone control was by far the most complex design that we have observed thus far. It is created by implementing an operational amplifier to create a unique configuration that combines both a passive low pass filter (which rolls off the harsh high frequencies above 2000Hz) and an active tone circuit (which boosts the treble frequencies enough in order to smooth out the response from the low pass filter). This makes for a very elegant tone control that has been emulated in many other guitar pedal circuits.

Because of the larger amount of parts required, however, this tone control configuration was the most expensive of the ones we have observed.

As was the case with the Big Muff Pi, the basic model of the Ibanez Tube Screamer overdrive pedal has been revised and replicated by many other companies, in the form of boutique “clone” pedals. Some of these clones include the JHS “Bonsai”, and the Earthquaker Devices “Plumes”.

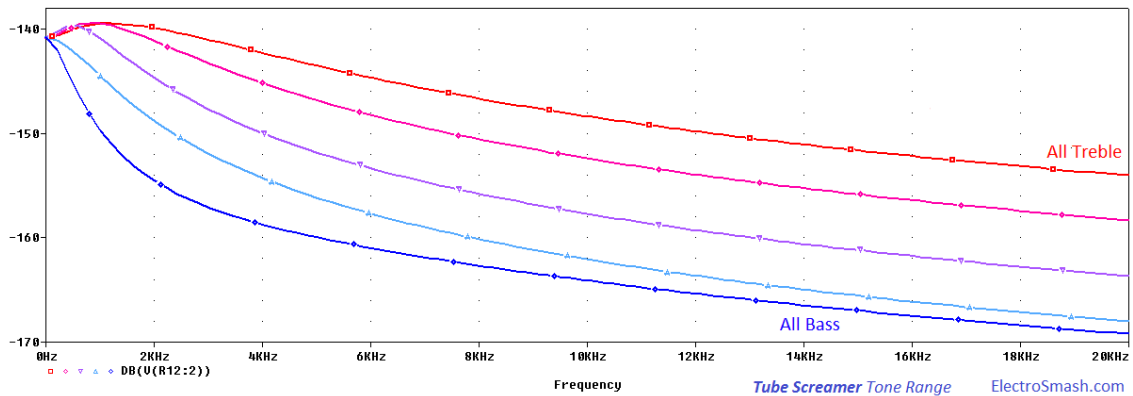


Figure 25: Tube Screamer Tone Control Frequency Response

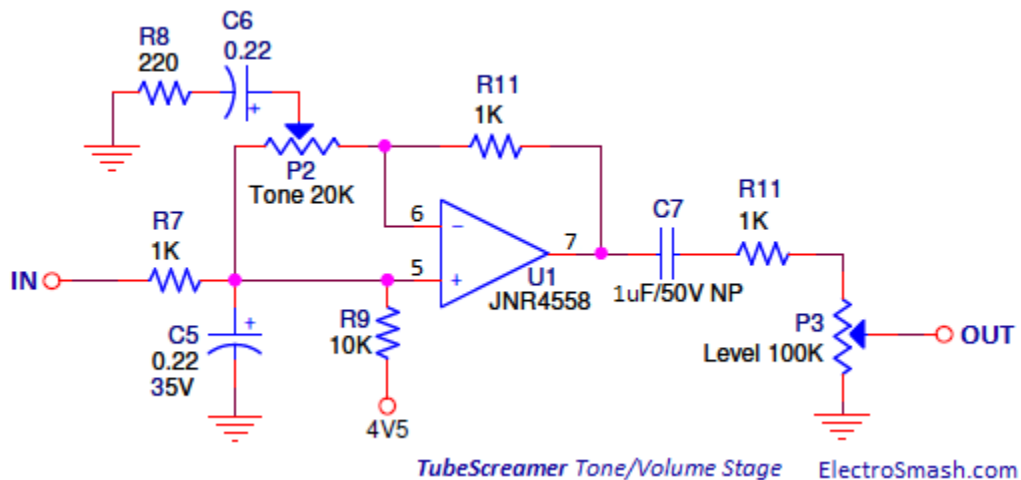


Figure 26: Tube Screamer Tone Control Schematic

3.2.5.4 Tone Control Selection

There were a lot of factors taken into consideration with our tone control, including cost, ease of implementation, and frequency response. Our team felt that the tone control configuration most suited for the ADEPT was the Bluesbreaker-style tone control. We

have chosen this tone control because it is the configuration with the lowest amount of parts, the easiest implementation, and the most simple frequency response.

We felt that it was better not to over-complicate our tone control design, since we were spending a lot of time on other parts of the circuit such as the input and output buffers, the CODEC and MCU implementation, and the overall power distribution.

3.2.6 Power (9VDC)

We used a power supply that converts AC voltage to DC voltage with approximately 3 different possible DC outputs. The desired voltages are 9,12, and 24 VDC at 2.5A Output. We were using a supply that has isolation features along with capacitive coupling to handle any transient responses going into the system. Why did we use these features in our system? Isolating any circuit protects it from taking in voltages too high for the circuit to handle. Typically, to isolate any circuit, we will use a transformer tied with a relay to protect our circuit. In this case. Our power supply was plugged into a standard power outlet that you see in your home. In the United States, most homes use electrical power in the form of 120 volt, 60 hertz, single phase, alternating current. Most standard electrical outlets don't have any protection against power surges or spikes. The voltage from the outlet could then spike to over 170 volts. If the voltage regulator in the AC to DC converter of the power supply has a tolerance under the supposed power spike, it could destroy our regulator and in turn, the power supply. Figure 27 below is a standard diagram of the stages within our power supply.

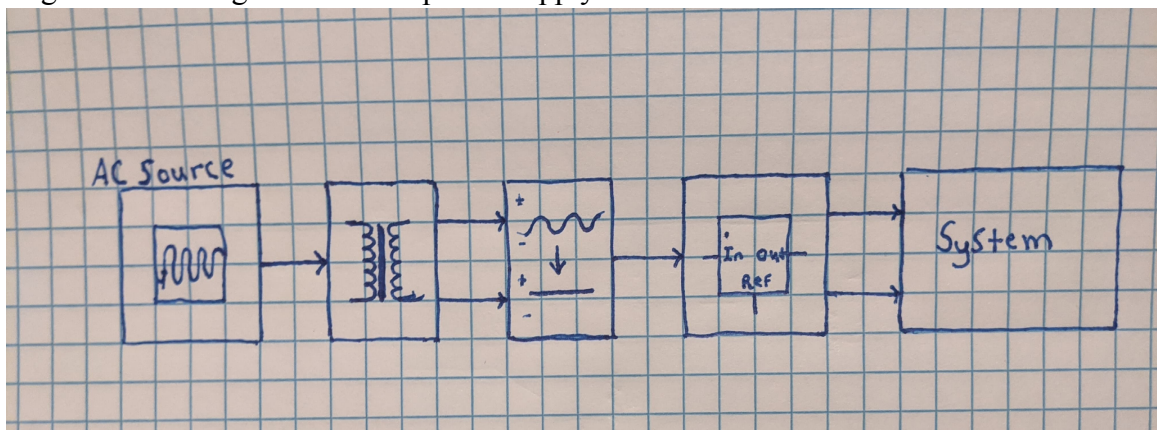


Figure 27: Power Supply Stages

3.2.6.1 Power Efficiency

Power efficiency was a major consideration when implementing our design. What is power efficiency? Power efficiency is the ratio of output power to input power.

$\frac{P_{out}}{P_{in}} \times 100 = \text{Power Efficiency}$ where the output power is the product of the output voltage and output current. Input power is the product of the Input voltage and input current. As the power efficiency ratio decreases, the power lost in the system increases. Resulting in heating of the system. Sometimes increasing the power efficiency of the system as a whole comes with trade offs such as noise generated in a system. We will go over this when considering voltage regulators below.

In order to maximize power efficiency, we kept our output voltage and output current high and our input voltage and current low. How may we manage power efficiency? We must first consider what aspects we are attempting to regulate or simply keep constant when supplying power throughout our guitar pedal. For our case, our main focus is maintaining and regulating voltage. Therefore, the amount of current is subject to change based on our design. How may we manage output current? Since our goal was to maintain a steady and accurate output voltage, we selected the components that affect current at the output. The obvious component values to manipulate are resistors. Simple Ohm's law exercises this premise.

Intuitively this was an obvious choice since resistors dissipate heat and moreover, power. Increasing the resistance at any output stage will create power loss. For example, having a high output resistance will decrease the current at the output. We saw that this decreases output power and inturn will decrease the power efficiency ratio. How could we manage input current? The opposite relationship between resistors and current applies to input current management. However, with our power supply, the internal impedance was a constant. This means that the output power of our power supply(our input power to our guitar pedal) was present.

Why didn't we take capacitors and inductors into account? When it comes to powering our system, we were supplying DC only. One may argue that we must take into account equivalent series resistance or capacitors and inductors. However, their values were found to often be very small and were negligible when analyzing power efficiency of the DC aspect in our guitar pedal. Methods of voltage regulation had a very significant effect on power efficiency. We went through those considerations in the sections below.

3.2.6.2 Power Sequencing

A key factor considered in developing a stable design was power sequencing. The concept of self-explanatory and its own term. We wanted to make sure that each stage in our system is powered on at the right time concurrent with one another. If we send a sudden impulse of power through three different devices at once we can expect unpredictable behavior based on return current and unknown current paths throughout our system. To avoid these problems, we considered powering on one stage at a time.

First thought of power on our digital to analog and analog to digital converters. After a short time delay the MCU will power on. Our input and output buffers were tied directly to our 9 volt power source so you will not need to worry about DC coupling or buffers either before or after the power sequencing is finished. We wanted to use a two millisecond time delay in between each stage powering on. How could we have implemented this? There were various ways to implement power sequencing; some can be very complicated. Considering that we had a relatively simple system regarding power we did not need to use complicated power sequencing techniques such as utilizing an fpga or any other programmable chip for that matter. A good option was finding a power sequencing chip that has a minimal number of inputs and is designed with a fixed time delay in between outputs. A good option was the lm3880 three rail power sequencer with fixed time delay. Where two of the pins were tied to the CODEC and one pin was tied to the MCU. Each of the corresponding voltage regulators for both analog and digital power will have its according voltage regulator at the 9 volt input before the stage of the power sequencer. In this way, we would have been able to avoid damaging the chip from high voltage damage.

3.2.6.3 Grounding Methods

Having a stable ground is essential to having a properly operating circuit. By stable ground, I mean having minimum noise along a common voltage potential and making sure that each different ground of each different stage of our system will have a minimal voltage difference between each stage at each ground. There are many different types of ground and each stage was referenced to its own ground internally. We tried to have as much common ground as possible within our system. However we may run into situations where we cannot share a common ground between stages. We needed to implement various tests to confirm that the grounding is optimal in our system. Here, we went over how grounding works and the nature of various grounding methods. Furthermore, we analyzed these methods to see which fits our needs for designing our guitar pedal.

What is Grounding?

First of all, any charge flowing through any circuit needs a common path to discharge through. Grounding does exactly this by creating a common voltage potential of 0 volts to channel the charge flowing through the circuit into that null potential. Any circuit with charge flowing through it will act unpredictably if that system is not grounded. This can cause violent damage to all components across the circuit. All positive charge will flow towards the negative direction of the circuit. Therefore, any charge flowing through a circuit will need a common node to travel to. This completes the overall flow of charge through the system and makes it operate properly. Furthermore grounding is a safe way to distribute charge among the system. If you touch a system that is not grounded the positive or negative potential will always want to find a pathway to complete the circuit.

If the person touching it is a better pathway to any common voltage difference then charge will flow through that person and can injure that person. Overall, grounding stabilizes voltage levels, protects against electrical overloads, and minimizes resistance to the common node.

Grounding Methods

There are five types of ground commonly used in powering electronics. There is earth ground, chassis ground, common ground, analog ground, and signal ground.

3.2.6.3.1 Common Ground

Voltage is a relative characteristic when analyzing any circuitry. A voltage value will only make sense if you have another voltage value to compare it with. If you were to just measure a voltage value without any reference you the measurement device would read an utterly meaningless value on its display. You absolutely need something as reference to measure against, this reference is ground. Balanced signals are often made into an equidistant proportion to ground. For example, in a wire the voltage goes to +3V on the other wire it will go to -3V. However, ground is a neutral path of 0V so when measuring across components it will not be +3V to -3V rather than the actual voltage difference across any specific component *with reference* to ground. Common ground is often used interchangeably and can be confusing when doing such. The common ground symbol is an “overall” concept applied when designing circuits and is mostly relevant when designing simulation in an “idea” environment. Common ground symbol is denoted as the figure below.



Figure 28: Common Ground Symbol

3.2.6.3.2 Earth Ground

Earth ground is exactly as it sounds. It's a ground physically (and electrically) connected to earth via a conductive material such as copper, aluminum, or an aluminum alloy. An actual earth ground needs to meet the National Electrical Code. This code applies to all the United States, National Electrical Code (NEC) is the set national standard deemed safe electrical for all electrical systems, to protect people and from electrical hazards. By NEC standards, earth ground consists of a conductive pipe, or rod, physically driven into the earth to a minimum depth of 8 feet. The earth provides an electrically neutral body, this provides a neutral common path for charge to flow through. Earthing is a

complicated concept in the sense that it is “assumed” to be a neutral path for electrons to flow into since the earth can consist of various materials with different conductance. For example, the bottom prong of a standard home electrical outlet is tied to earth ground. Obviously, We will not be using any earthing in our circuit itself but our power source was tied to regular ground. Therefore, it is a necessary aspect to consider in our design. Earth ground symbol is denoted as the figure below.



Figure 29: Earth Ground Symbol

3.2.6.3.3 Chassis Ground

A really interesting method of grounding is chassis ground. Oftentimes electrical systems are enclosed in a metallic box. This box is off then conductive and can offer a path of neutrality for charge to flow into. The reason why we put electrical systems in metallic enclosures is to work something like a faraday cage. A faraday cage is an enclosure that blocks all electromagnetic waves from entering the system. This Faraday cage concept is also a convenient way to provide a neutral path. Constraints regarding space and mechanical logistics require a convenient grounding method. This convenient method is known as chassis ground. For example some radios can have a metallic enclosure in which chassis ground is utilized to provide a common node within that enclosure. We have not disregarded this possible method of grounding as we may have to enclose our guitar pedal and a metallic enclosure. Given issues of space within that enclosure chassis ground was considered as a convenient method to provide a neutral path for our circuits to operate.



Figure 30: Chassis Ground Symbol

3.2.6.3.4 Analog Ground and Signal Ground

Analog ground is often referred to as the grounding for the analog power components of any processing system. In contrast to signal ground or digital ground it is typically less noisy and has less stringent constraints regarding variation of input voltage. Some

systems separate analog ground from digital ground or signal ground. However it can be wise to tie analog ground and signal ground together. This eliminates any voltage differential between different grounds to have a single neutral path. The reason why digital ground is typically more noisy than analog ground is because the sudden change in a digital signal can cause abrupt spikes that turn the transient response from going high to low or low to high digital value. If digital ground is tied to analog ground, the noisy aspects of digital ground can leak over into our analog input and deteriorate the power stability in our MCU. We considered either separating the more noisy digital ground and less noisy analog ground or tying them both together. In our case we decided not to separate digital ground from analog ground because the frequencies of our data communication between the CODEC and STM32 weren't high enough to cause an issue.



Figure 31: Analog and Digital (Signal) Ground Symbol

3.2.6.4 Circuit Isolation

We considered isolating our circuit in two ways, through using an isolation transformer or an Isolation relay. Isolation Transformers are made up of two coils which are coupled magnetically with an iron core. These coils transfer energy to each other through a magnetic field. There are two types of Isolation Relays, Electromagnetic relays and solid state relays. Electromagnetic relays use electromagnetic and mechanically movable parts that complete the circuit when a voltage is set at the input. Solid state relays use the same except they use an optically controlled diode. Both of these methods for isolation use a technique called galvanic isolation. Galvanic simply refers to the current produced through a chemical action.

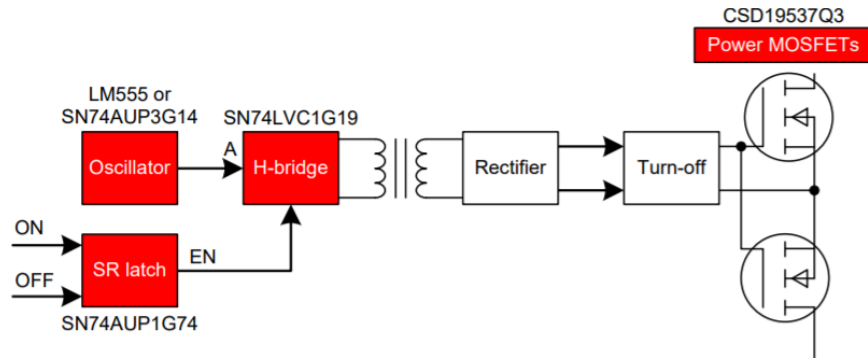


Figure 1. TIDA-00751 Block Diagram

Figure 32: Galvanic Isolation

Even with DC voltage, there is a transient response in a system every time it is powered on or off. This is why a lot of board schematics couple capacitors at the outputs to ground. Any transient response has frequency, even if it is for a brief moment in time. Unwanted transient responses can damage our system. To better protect our system from unwanted transient responses, we must couple our outputs with capacitors. Different capacitor values can handle different transient responses. Therefore, it was wise to couple multiple capacitors with different values from output to ground as they will short any transients to ground. This method is called transient suppression.

In order to further isolate the sensitive components of our guitar pedal circuit from any power mishaps, we considered implementing a 1N4001 rectification diode. This diode was placed in our circuit to protect it in the case that a power supply with an incorrect/inverted polarity is used (since we were using a center-negative power supply, this diode would protect against the accidental use of center-positive power supplies).

As far as our DC power management is concerned within the circuit, we can utilize a simple voltage divider in order to split the original incoming 9V DC signal to have access to half the voltage (4.5V DC). This can be done by using the same resistor value for both resistors R1 and R2 and applying the formula $V_o = V_i (R_2/R_1+R_2)$. The reason this half voltage is necessary in our circuit is because we were using it to bias the transistor buffers in the input and output stages.

3.2.6.5 Droop Voltage

A problem we considered running into when supplying two ports for USB power is called droop. Which is the power surge or drop when a configuration in our hub, such as

plugging in or removing a new device. Droop in a loaded port produces on its V-Bus line when a device is connected or removed from anything tied together in a circuit.

How to protect from “droop”? Simply use decoupling capacitors. This enables us to provide sufficient power to an DSP or MCU to maintain voltage stability. If the voltage increases unprecedentedly, decoupling capacitors will absorb the excess energy trying to flow through any device, hence maintaining voltage stability. This concept is very similar to transient suppressive coupling capacitors. You can see a basic configuration of these concepts below.

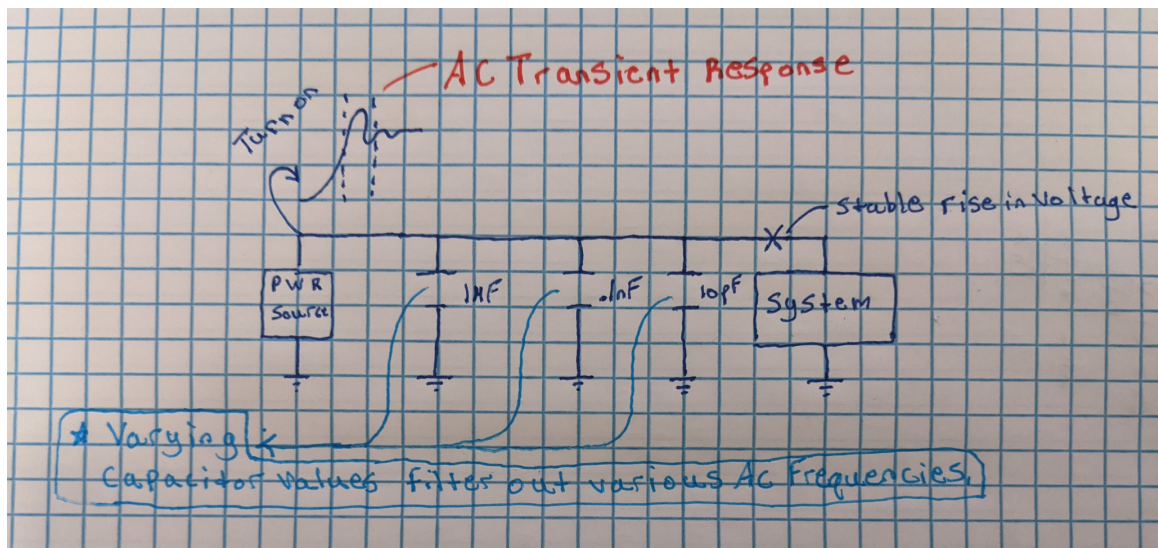


Figure 33: Transient-Suppression Capacitors

3.2.6.6 Impedance Matching

One of those most important considerations of USB communications is impedance matching. When transmitting information on USB data lines. Bits can be lost due to noise. This noise can be produced externally or internally. In this case we will focus on internal noise. When sending any signal through a medium, small electromagnetic fields can intertwine with each other. If constructive interference occurs, the integrity of the data in our signal is at risk of being lost.

How to implement impedance matching to optimally reduce noise on our data path? First let's consider what impedance matching is. Every medium that electricity passes through has some sort of impedance even if it is very small. Obviously, different levels of impedance have different frequency responses. Theoretically, when the impedance of data lines are exactly the same, there is absolutely no noise produced in our system. For

example, if the data lines (D+ and D-) of our usb have equal amounts of impedance, internal noise is eliminated. Hence, impedance matching.

In the real world they can never be truly equal. However, we can get really close. Typically when you see wires twisted together, they are some sort of lines for data transmission. Twisting wire is a method of impedance matching. When twisting wires together you create a voltage differential that induces deconstructive interference rather than constructive interference. Essentially the signals resonating from the wires are 180 degrees out of phase, making the potential difference caused by the noise to be zero. Furthermore, one wire being longer or thicker than the other causes an impedance mismatch and in turn a noisy signal. This was heavily considered if we wanted to implement USB communication. Especially when soldering data lines from one port to another.

3.2.6.7 Ripple Voltage

Ripple voltage is the amount of AC voltage along a DC voltage output. This will always happen during any AC to DC conversion since no methods exist where you can completely eliminate ripple voltage. However, it can be minimized and managed when designing our power elements in our system. Ripple voltage typically is caused by varying output voltage during rectification. It typically looks like a pulsating voltage rising from zero to a maximum and back to minimum. The ripple voltage can be very problematic for sensitive components. Current will rise and fall as a result of ripple voltage and may cause overheating and damage components over time. Furthermore, Rippled voltage can cause errors in digital circuits such as the MCU and CODEC. They can also be considered a noise source in this regard. Since we wanted to maximize performance with a clean audio signal we must take ripple voltage and its consequences into account. We were able to manage the negative effect of ripple voltage by choosing the appropriate elements and regulators when powering our guitar pedal.

Implementation of voltage regulation:

3.2.6.8 Linear Regulators

Voltage regulators come in many models with different tolerances and capabilities. The advantages of linear regulators are that they have a simple design, low noise due to the absence of switching, few external parts, and low cost. The disadvantages are that they only have step down capabilities, poor power efficiency, and due to their poor power efficiency are subject to generate undesirable amounts of heat.

The components of a standard switching regulator consist of a transistor, operational amplifier, capacitor, and resistor. Linear regulators use a closed feedback loop to bias a

pass element to maintain a constant voltage across its output terminals. A linear regulator works by taking the difference between the input and output voltages, this difference dissipates power and generates heat. As the difference between the input and output voltage increases, the power dissipation and heat generated increases.

In most cases, a linear regulator wastes more power stepping down the voltage than it delivers to the output. The transistor inside the regulator is connected between the input and output terminals, the resistors used as voltage dividers at the inverting and non-inverting inputs of the op amp need high voltage and current to maintain a constant voltage at the output. Therefore, high input-to-output voltage differential combined with high load current results in large amounts of power dissipation. Despite these drawbacks, our team will consider the use of 78L05 linear voltage regulators for the power distribution. However, other options were discussed, and the best one was ultimately chosen as the appropriate hardware solution.

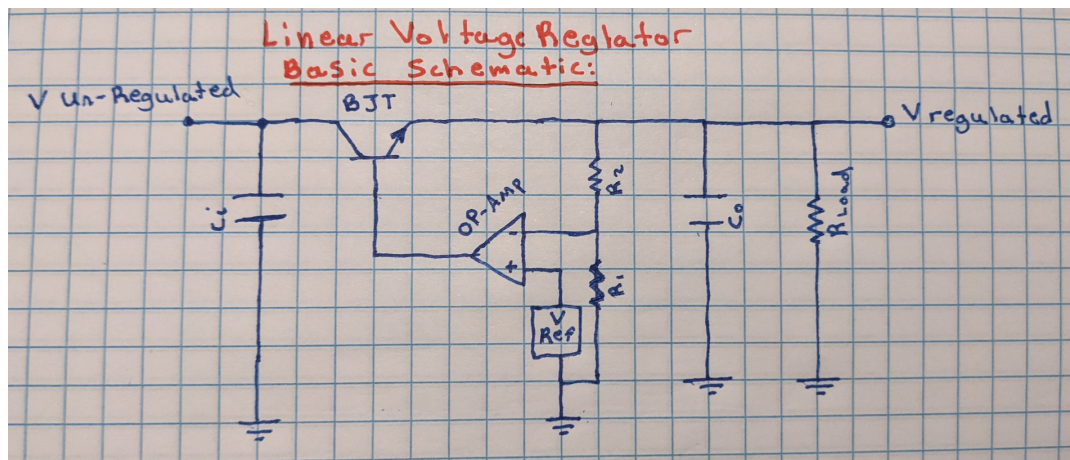


Figure 34: Linear Voltage Regulator

3.2.6.9 Switching Regulators

The advantages to switching regulators are that they have step up, step down and negative voltage capabilities. Furthermore, they are much more power efficient than linear regulators and, as a result, produce less heat. Some switching regulators can be up to 98% power efficiency while linear regulators operate at around 70% power efficiency. The downside is that they require more external parts, are more expensive, and produce high amounts of noise.

How do switching regulators work? A switching regulator works by taking small bits of voltage by rapidly switching on and off from the input voltage source to the output. The

internal switch regulates the rate at which energy (voltage) is transferred to the output. Hence “switching regulator” was deemed the name of this device.

Why are switching regulators more power efficient? The energy losses involved in switching of energy around were relatively small. Since their efficiency is less dependent on input voltage, they can power useful loads from higher voltage sources.

Why do switching regulators create noise? Any sudden change in the input of energy will always produce some sort of transient response. For example, when the switching regulator switches on to flow energy, the sudden impulse will shoot up and oscillate for a short period of time before coming steady. The quantity of these independent transient responses per switch is directly proportional to the switching frequency.

How may we eliminate noise at the output of a switching regulator circuit? First if we are able to measure the number of oscillations per period of time, we can find the frequency of the transient response of the switching regulator. We can design a low pass filter with a cutoff frequency less than the transient response induced by switching. Furthermore, an op amp can be provided in this filter with a gain of one to eliminate any attenuation at the output.

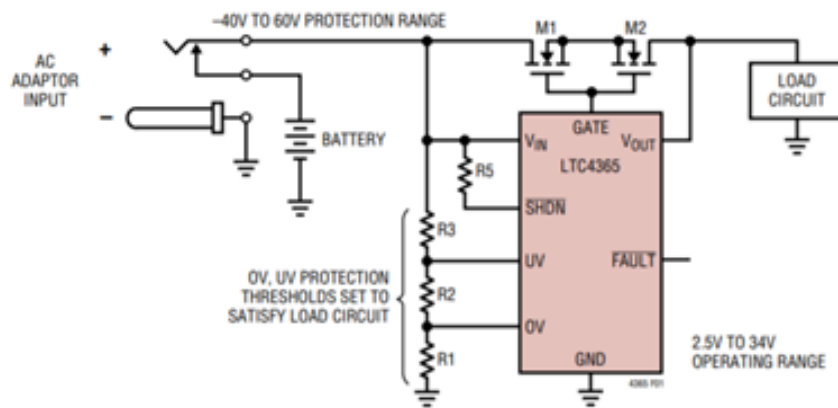


Figure 35: Switching Regulator

3.2.6.10 DC-DC Converters (Buck)

DC-DC converters are also another application for voltage regulation. The advantages of converters are design simplicity, cost, along with boost and buck capabilities. They are also more power efficient than linear regulators. Buck converters step down voltage from the input to output while boost converters step up voltage from input to output. Buck converters consist of a diode, FET, inductor, and capacitor. Buck converters also have potentiometers in them that vary the voltage level at the output. They are typically

referred to as “turn pots”. A switch at the input turns on to let current flow to the output capacitor. The voltage across the capacitor cannot change instantly, and the inductor initially limits the current to the output. Therefore, the voltage across the capacitor during the switching cycle is less than the voltage of the input voltage.

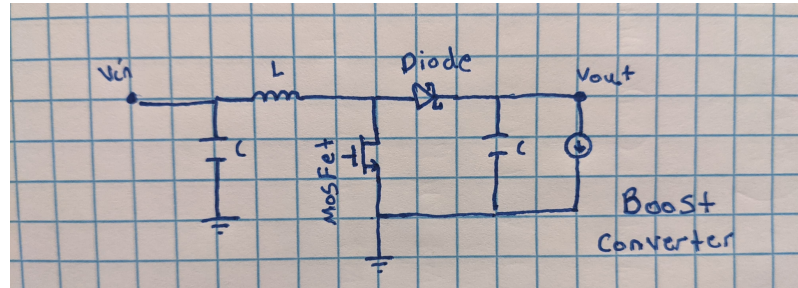


Figure 36: Buck Converter

3.2.6.11 DC-DC Converters (Boost):

Boost converters have all of the same elements in the circuit except the placement of the inductor and the FET are oriented differently. At the initial turn on of the circuit, the inductor’s current at the input does not change instantaneously. This allows for smooth current flow. The FET diverts the current from the inductor. During this time the output capacitor stays charged since it can’t discharge back due to the bias of the diode. When the switch turns back off, the FET turns off and the capacitor takes the extra energy stored in the inductor from turn on. This provides extra charge inside of the capacitor. Hence, increasing voltage across the output.

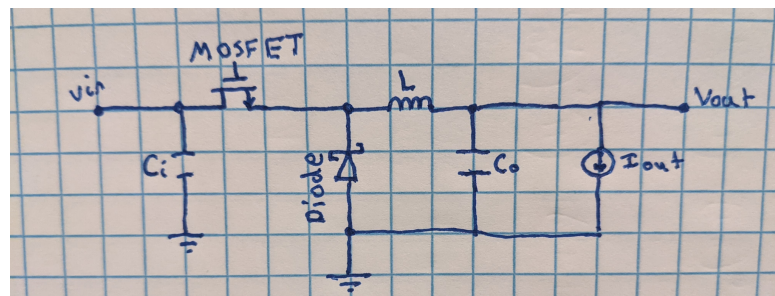


Figure 37: Boost Converter

Overall DC-DC converters can either increase output voltage (boost) or decrease output voltage (buck) relative to the input voltage.

Constraints of providing power:

3.2.6.12 5V and 3.3V Digital Logic Power Supplies

Another aspect we implemented in our design was powering our power to ground for our USB communications. There is no doubt we will need our system to be USB compatible. Since we are using 9V-12V in our power supply, voltage regulators needed to provide stable USB power of 5 volts and a maximum of 1.5A. Voltage regulators are devices that maintain the voltage of a power source within acceptable limits. We will need a regulator to supply the 5V to at most two ports. We must tie two regulators or a single regulator (given its capabilities) to our main power supply to output the proper 5 volts per port. Two of the most commonly used logic level voltages are 5V and 3.3V. 3.3 Digital logic DC power is a typical voltage used for low-power digital devices. 5 Digital logic DC power falls under the standard of “Transistor Transistor Logic” (TTL) voltage used by digital devices. Both 5V and 3.3V power supplies for any microcontroller have very stringent voltage regulation requirements.

Having an unnecessarily high voltage can damage components and having an unnecessarily low voltage can cause erratic behavior in a system. 5V logic requirements lay within 5.2V and 4.75V while 3.3 logic requirements lay within 2.7 volts and 3.6 volts. When implementing our design we wanted the components used to regulate the voltage at the power input to uphold these restrictions. It came down to a trade off between power efficiency, cost, noise immunity, and accuracy. This was why we needed to be conscientious when deciding between the types of voltage regulating systems such as linear regulators, boost/buck converters, and switching regulators.

3.2.6.13 Transistor-Transistor Logic (TTL)

We also needed to consider Transistor-transistor logic (TTL), a digital logic implementation in which current pulses manipulate the behavior of bipolar transistors. TTL ICs usually have four-digit numbers beginning with 74 or 54. A TTL device uses transistors with more than one emitter therefore having more than one output. The characteristics of TTL is having high switching speed, and decent immunity to noise. Most circuits using TTL have a disadvantage of drawing high current. However, there are low current TTL devices on the market, but the tradeoff is the reduction in operating speed and that they tend to be more expensive. Its principal drawback is the fact that circuits TTL logic gates come in integrated circuits (IC's).

3.2.6.14 Design Implementation

We used a 9V 500mA AC to DC power source going into the analog input. Furthermore, we were tying that 9 volts to a buck down 3.3V voltage regulator where the output of the regulator goes into the VDD pin of the STM32F446 MCU. Since we digitally implemented our effects, we required less power as the Digital implementation

components require less power to do proper processing to produce those effects. Our MCU will require 12mA at the input to the internal load of the MCU. We anticipated requiring approximately 700mA for the analog and digital sections of our guitar pedal. Therefore, our power supply current rating was sufficient to power our device.

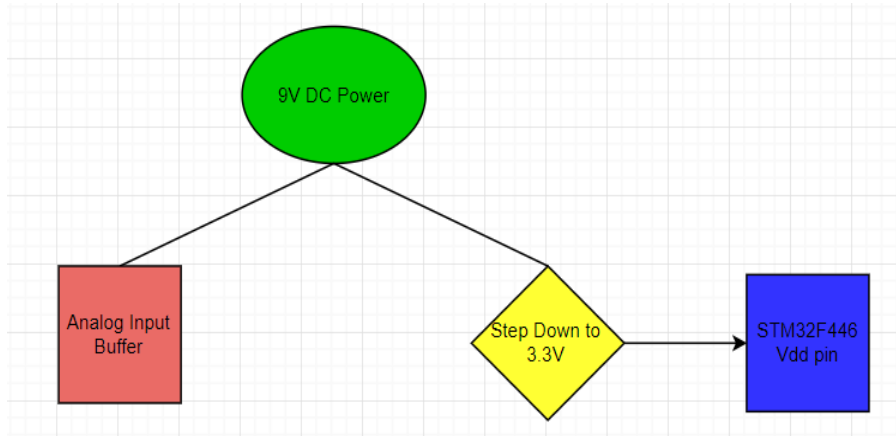


Figure 38: Power Distribution Diagram

Below is a spice schematic that has the input buffer, output buffer and the regulated voltage going into the STM32F446 MCU along with the internal resistance found in the MCU to achieve the right amount of bit processing capabilities needed to implement the digital effects. We put in hypothetical AC signals to the input and output buffer to emulate if the behavior at the output was working properly (which it does). Furthermore, we measured various current across the components of each buffer to make sure that it will not be drawing excessive current that the first stage of our 9V power supply cannot handle. The current was found in the micro amps so we will have plenty of available current from the power supply to spare if we need to make any unexpected changes in the future.

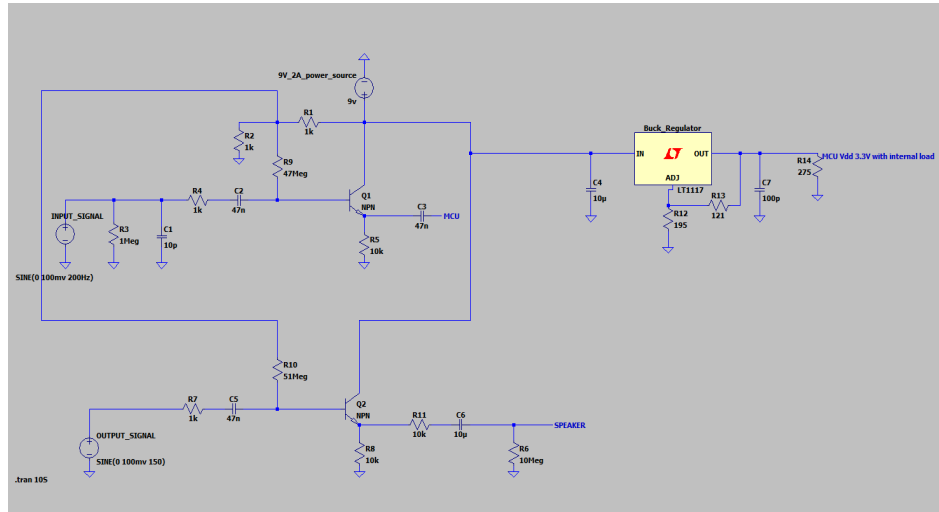


Figure 39: LTSpice Schematic, Analog Input/Output with MCU Input

3.2.6.15 Device Selection

After doing extensive research on how we were to implement the power distribution architecture for the ADEPT pedal prototype, we then selected the specific components that were purchased and integrated into our overall design.

3.2.6.15.1 AMS1117 Buck Converter

For this design we have chosen a DC/DC bipolar linear Buck converter. We have analysed the benefits between accuracy, noise, efficiency, and cost of each type of voltage regulator listed in section 3.2.6. The buck converter was ideal for powering the STM32F446 MCU since any linear regulator is very accurate and creates low amounts of noise. We must sacrifice power efficiency to do this. Compared to switching regulators, We were sacrificing about 12% of our power efficiency as the average power efficiency of a switching regulator is ~97% while a DC/DC linear buck converter is 85%. We used a AMS1117-3.3V DC 4.75V-12V to 3.3V Buck Converter Voltage Linear Regulator Step Down Power Supply Module with a maximum current output of 800mA. This output of 3.3V was tied to our Vdd pin of the STM32F446 MCU. In case this does not work. We could've used the LT1117 adjustable step down converter. When using an adjustable converter We use the equation $V_{out} = 1.25(1 + \frac{R2}{R1})$ to find the proper resistor values to yield a 3.3V output. R1 is recommended to be 125ohm or lower. Therefore, we will set R1 to 121ohms using the equation we adjust R2 to be 195 ohms. We had to adjust the R2 value slightly to generate a more accurate output in LTspice. We set the capacitor values couple to the input and output to ground in order to eliminate ripple voltage caused by any induced transient response occurring in the circuit. Occurrence from the transient

response can be caused by noise in the power supply, hard start or hard stop (unplugging and plugging), or simply power cycling the device.

When running the spice simulation we measured an output of 3.299V and 12ma going into the Vdd pin of the STM32F446. This measurement demonstrates our estimated resistor values are correct. We also measured the currents across the adjustment resistors to estimate how much current was needed to be supplied across the circuit. The current across these resistors were measured to be about 10mA. We also measured the currents found in the input and output buffer. The maximum amount of current measured across the components was less than 10 mA. It is safe to say that the analog input/Output and the voltage regulation designs will not need a lot of current. This left plenty of current from the 9V 2A power supply to be available for the MCU.

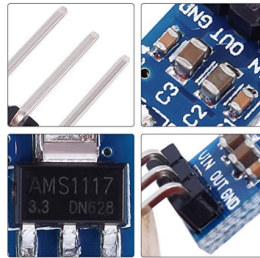


Figure 40: AMS1117 3.3V DC/DC Buck Converter

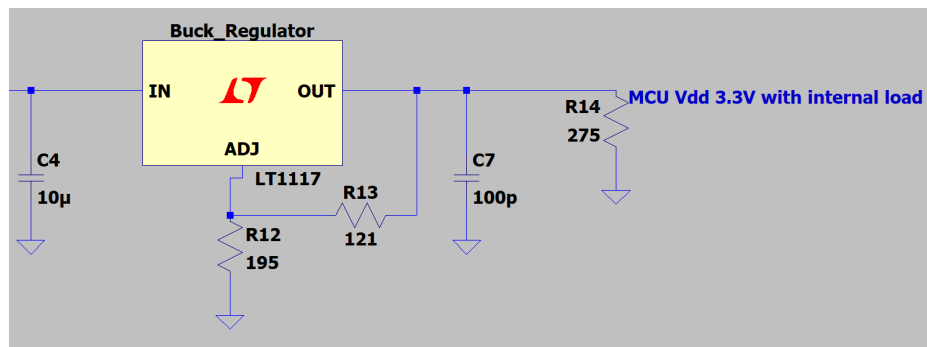


Figure 41: 3.3V DC/DC Buck Converter Design

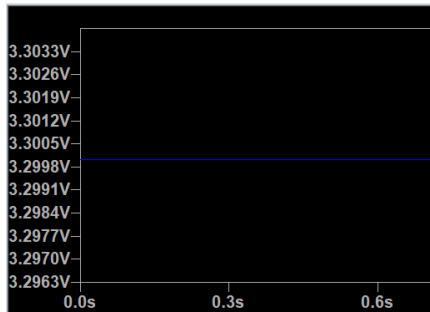


Figure 42: 3.3V DC/DC Buck Converter Output Simulation

3.2.6.15.2 LT3045 Linear Regulator

This is a 20V, 500mA, Ultralow Noise, Ultrahigh PSRR Linear Regulator. Ultrahigh PSRR means power supply rejection ratio. We considered using this regulator to power the PCM3060 CODEC. We were supplying 5V analog power and 3.3V digital logic power. Recall that 3.3 logic requirements lay within 2.7 volts and 3.6 volts. This regulator is accurate enough to sustain those margins with an output error margin of $0.5 \mu\text{V}/\text{V}$. Therefore we had plenty of precision needed to implement an accurate digital 3.3V margin along with an accurate 5V margin.

When implementing digital supply voltage, low noise occurrence is essential in the sense that the 3.3V is tied to the data lines. The presence of noise in the data lines can cause poor data transmission. Ultra low noise regulators will increase the performance of our system in the sense that we are minimizing error when processing our audio signals. Hence, we will process and produce a clean signal, improving the sound of our effects. Again, like all linear regulators, they aren't as power efficient as other regulators, such as switching regulators. In our design, we are aiming to attain maximum performance. Therefore power efficiency is much less important than minimizing noise in our system.

This voltage regulator is designed to operate as a precision to have a reference current followed by a high performance voltage buffer. Furthermore, we can reduce noise even more, increase output current, and spread heat on the PCB, we can couple the device in parallel. This device supplies a typical 260mV dropout voltage with 500mA current. The typical quiescent current is 2.2mA and drops to a significantly less value than $1\mu\text{A}$ during shutdown. The LT3045 maintains unity gain operation. This provides constant output noise, bandwidth PSRR, and load regulation, while operating independently of the output voltage. It also has a wide output voltage range (0V to 15V). Furthermore, we can program the current limit, fast start capabilities and a power good feedback loop into the system to exemplify operability and stability in the system as whole. The fast start feature uses a capacitance value to control the ramp rate at turn on. Fast start features can cause small ripple along the ramping voltage at turn on. The fast start feature was disabled since

we want a steady climb without any ripple caused by the internal capacitance used to increase the slew rate to steady state. It also was not necessary for our application. The LT3045 is stable with a minimum $10\mu\text{F}$ ceramic output capacitor. The voltage regulator also has Built-in protection. This includes reverse-current protection, internal current limit with foldback, reverse-battery protection, and thermal limit with hysteresis. Where the change in output voltage at ambient temperature before and after the device is cycled over the operating temperature range. The LT3045 is available in thermally enhanced 12-Lead MSOP and 10-Lead $3\text{mm} \times 3\text{mm}$ DFN packages.

We could also decrease the output voltage noise by adding a capacitor across the SET pin resistor. Adding the capacitor will bypass the SET pin resistor's thermal noise and the current noise of the voltage reference pin. By doing this we set the output noise equal to the error amplifier noise. We could also decrease the start up time by using a bypass capacitor.

We were considering using the first regulator (5V_{out}) to power the device as a whole. The output of the second regulator was to be applied to the 3.3V data lines. Both regulators were tied to our 9V power supply. Both regulators were mounted on our PCB. Here is a diagram of our LT3045 voltage regulator:

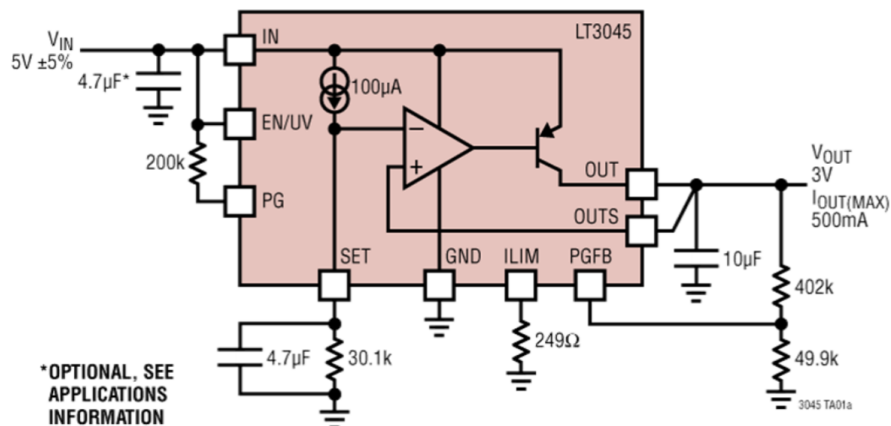


Figure 43: 3.3V_{cc} and 5V_{dd} Linear Regulator Diagram

3.2.6.15.3 Hurdles and Complications

Linear regulators cause excessive amounts of heat due to their lack of power efficiency. Once internal temperature gets over 125 degrees celsius (257 degrees fahrenheit), the nature of operation can become unstable and can deteriorate the shelf life of the product. To combat this we bought 5 units for testing. In the case that the internal temperature rises high enough to damage the buck converter due to improper insulation techniques (if we even need any), we were able to replace and troubleshoot the problem without any

real time setbacks. Furthermore, we can use insulating heat putty as a heat sink. This will dissipate heat generated internally of the chip to the putty. Since it works as an insulator we wouldn't have to deal with unexpected shorts across the components. It also worked as a good way to mount our buck converter anywhere if we needed to move it.

Another important constraint to consider was that resistance of materials can change over temperature. The resistance of most conductor materials varies with temperature. The temperature coefficient of resistance (TCR) is a constant that represents the change in resistance per degree Celsius. This is expressed as ppm/°C (parts per million per degree centigrade). Note that this constant is specified over a specific range of temperatures change over a specific temperature range that typically go from -55C to 145C. All of the resistor values used on the market are specified under ambient temperature. The formula can be denoted as $R = R_{ref}[1 + \alpha(T - T_{ref})]$

Where:

R = resistance at material temperature

Rref = resistance at Tref

α = temperature coefficient of resistance

T = material temperature in °C

Tref = reference temperature at which the temperature coefficient is specified

For example if you had a TCR of 30ppm/°C. That means its resistance can change up to 0.000030 ohms (30.1,000,000) per ohm per degree change in temperature.

4.0 Related Standards and Realistic Design Constraints

The following section discusses the various standards and constraints that affect how the ADEPT prototype was designed. We took into account various limitations that we worked around in order to realize our final design.

4.1 Standards

The ADEPT project must adhere to a number of standards in order to comply with laws and government regulations. This allowed us to ensure the legality of our design implementations, as well ensure that we make proper design choices pertaining to the various integral communication standards that we needed to utilize.

4.1.1 IEEE

IEEE standards are a primary consideration when developing our design. IEEE stands for “Institute of Electrical and Electronics Engineers”. IEEE is an organization that reviews various technologies in computer and electrical engineering to streamline characteristics in implementing designs that are an open source database. The goal of IEEE standards was to improve the quality of designs for the overall benefit of humanity.

4.1.2 IEC/UL 62368

The IEC 62368 international standard aims to provide and facilitate a list of requirements any ICT and AV product should meet in order to be certified and approved in terms of safety. It is a wide-known standard that manufacturers around the globe rely on to ensure that their products are safe for the public and operation. The IEC/UL 62368 came into effect as of December of 2020, previous standards such as the IEC/UL 60065 haven’t not been replaced but rather embedded in the IEC 62368. The new updated version has an overarching description of the requirements that must be met as well as a risk factor for those specific requirements, thus allowing the manufacturers more room to decide on the design for those specific requirements that have a high enough risk factor. Having your product be certified will facilitate trade export in the US and the EU.

4.1.3 UL60065, 8th Edition, September 30, 2015

UL Standard for Safety Audio, Video and Similar Electronic Apparatus - Safety Requirements. “This international Safety Standard applies to electronic apparatus designed to be fed from the MAINS... and intended for reception, generation, recording or reproduction of audio, video and associated signals.” [8].

4.1.4 Joint Test Action Group (JTAG)

“The Joint Test Action Group formed in 1985 to develop a method of verifying designs and testing printed circuit boards after manufacture. In 1990 the Institute of Electrical and Electronics Engineers codified the results of the effort in IEEE Standard 1149.1-1990” [1].

4.1.5 80 Plus Certification

1. By EPRI (Electric Power Research) and Ecos Consulting
2. Designed to streamline power products to 80% efficiency and above. Rating from white to platinum.
3. Requires three different load levels
4. Standardized Power Factors per rating from white to platinum.

4.1.6 Safety Requirements for Class 2 Power Units

1. UL 1310 per CUI inc
2. Maximum potential of 42 VAC peak / 60 VDC for exposed wires and terminals.
3. Protection from “backfeed” voltage.
4. Cord and plug-connected units with a 15 or 20A plug for 120/240 VAC mains supply.

4.1.7 CE - FCC - RoHS Certification

1. CE mark or CE marking is a European conformity marking displaying that the product is compliant with all relevant European requirements.
2. FCC - “Federal Communications Commission.” This means that the product is approved by US safety standards.
3. RoHS - “Restriction of Hazardous Substances.” The device was manufactured without certain hazardous chemicals.

4.1.8 I2C

1. Developed by NXP Semiconductors(used to be Phillips Semiconductors) in 1982 and became a de facto standard.
2. I2C is a bidirectional 2-wire bus for facilitating IC control.

3. A serial data line(SDA) and a serial clock line (SCL) are required to specify I2C description and standard.
4. Each device must be software accessible where each device has a unique address and there must be a slave and master relationship at all times where the master can operate both as receiver and transmitter.
5. Must include collision detection and arbitration to prevent corrupted data.
6. Data transfer features must have 100 kbit/s Standard-mode, 400 kbit/s Fast-mode, 1 Mbit/s Fast-mode Plus, and/or 3.4 Mbit/s High-speed mode.

4.1.9 SPI

1. Developed by Motorola in 1979 and became a de facto standard.
2. communicate in full duplex mode using a master-slave architecture with a single master.
3. SPI consists of one master and multi slave communication.
4. Master is the Transmitter and the slave is the receiver.
5. Capable of maintaining communication simultaneously between multiple slaves.

4.1.10 I2S

1. Serial bus interface used for communicating digital audio devices.
2. Transfer PCM audio data between devices.
3. Separates clock and serial data signals.
4. Simpler receivers than asynchronous communication systems that need to recover the clock from the data stream.
5. Unrelated to I2S despite the similar name.
6. Standard was introduced in 1986 by Philips Semiconductor (now NXP semiconductors), further revised on June 5, 1996.

4.2 Constraints

The following section will outline some of the design and administrative constraints pertaining to our project. These constraints are essentially the challenges that we were facing during the design and implementation process of our project, and provided some context to us in order to be prepared for any hurdles that came our way.

4.2.1 Economic Constraints

Although the market for guitar pedals is known for having a high bar for prices, most of the manufacturers that produce guitar pedals are met with a high demand but a low supply, therefore the prices increase. Although there are not that many digital guitar pedals in the market, most of the options available come at a very high price. Thus, the

number of DIY options have increased in the past years. Another factor that prompts the increase of people doing their own guitar pedals at home is the availability of power IC's. Ever since boards such as the Arduino or the Raspberry Pi started to come out the DIY communities have also increased in size.

Our project did not steer away from the motives found behind why a lot of guitar pedal users decided to go the DIY route. Our project uses relatively inexpensive components such as the MCU which, although is a powerful one, only cost about \$8. Since the overall cost of our project is projected to be under \$200, the economic constraints that our team faces were quite minimal. However, we were very wrong in that assumption. Having to respin out PCB and get expedited shipping added a huge cost.

4.2.2 Time Constraints

Irrefutably, the most important requirement of this Senior Design Project is that it must be completed by the end of next semester's Senior Design 2 course. This deadline could not be reasoned with, and our team always managed our time wisely, such that we were able to deliver a functional and impressive product as our end result.

Due to the COVID-19 pandemic, time has become even more of an issue for students wishing to pursue such ambitious projects as those seen in Senior Design. Shipping times during the pandemic have increased significantly, making it much more difficult to receive parts on time.

Even more notably, the unprecedented silicon shortage in 2021 is a major concern for our project. This shortage has come about as a result of the pandemic and other factors related to it, including a higher demand for electronic devices during quarantine and limited manufacturing at tech companies, who continue to comply with risk reduction tactics such as having employees work from home.

Since the ADEPT prototype used several semiconductor devices to function properly, including the ADC/DAC, MCU, and several other integrated circuit chips, we had difficulty obtaining some of our parts on time (or at all), which could delay the prototyping process and put our team under time pressure.

4.2.3 Environmental Constraints

The environmental constraints of this project are an important factor, and must be addressed with great detail. With the decline of the CD, and as more individuals consume music digitally, the environmental footprint of the music industry has significantly reduced over the years. We wish to continue this trend of being as environmentally conscious as possible when designing our product.

We must also consider the environmental impacts of the manufacturing processes for the various physical components of our design. This included the PCB (printed circuit board), enclosure, and any electronic circuit components that were used for our project.

Traditionally, PCBs have been made with toxic materials (lead, mercury, and cadmium, among others) that have a high potential risk level for landfill contamination and harm to production workers. With the publication of the Restriction of Hazardous Substances Act (RoHS) in 2011, the use of certain harmful materials was either limited or completely banned in the manufacturing of electronic products. Because of this, many PCB manufacturing companies have become RoHS-compliant in recent years, and the PCB production process has significantly reduced the amount of harmful materials that are used in these products.

There have also been countless efforts to implement the recycling of electronic products like PCBs. By recycling used and defective PCBs, we can recover various metals and alloys (including copper, tin, and gold) that can be used in the creation of other electronic products, thus significantly reducing the amount of natural resources that must be mined in order to manufacture these products.

In addition to the environmental impacts of PCB manufacturing, we must also observe the impact that semiconductor technology has on the environment. It's clear that without semiconductors and other electronic devices, PCBs would be a lot less useful. However, the process of manufacturing these integral devices is not without flaw. The semiconductor industry is responsible for a small percentage of the United States' greenhouse gas emissions, which primarily comes from the use of fluorinated gases used for advanced production processes. However, this industry is working hard to maintain a small carbon footprint. The Semiconductor Industry Association (SIA) actively works to reduce these industrial emissions, monitoring them and reporting them to the Environmental Protection Agency (EPA) on a regular basis. Because of this industry's self-accountability, they have successfully reduced their collective emissions by 50% since 1999, and currently only create about 0.2% of all industrial emissions.

Companies like Hammond that manufacture metal electronics enclosures have also made strides in becoming environmentally conscious in recent years. These improvements include recycling millions of pounds of scrap materials from their manufacturing processes, as well as going completely paperless for all company documentation. They have also implemented a closed-loop system that prevents any water-based chemicals used in their powder coating/painting process from leaking to clean water supplies, as well as filtering this waste water so that it can be reused.

4.2.4 Social Constraints

We hold, as musicians and engineers, accessibility as one of our main pillars. We believe everyone should be able to have access to devices such as this project in order to expand and further enjoy their passion for music. Allowing others to modify and build this project at home is one of our visions for this project, hence open source was another pillar of ours. In that regard we would like to support a community of engineers and musicians to make the device as user friendly and accessible as possible.

This is a project for everyone who is willing to spend the time and money to do at home. We hold in our hearts that a great community of people who share the same passion for music technology will further envision and utilize our project as a stepping stone for their personal development.

4.2.5 Political Constraints

After careful consideration, our team came to the conclusion that there is no political constraint affiliated with our project. Since this was purely a musical and technological endeavour, we wished to keep it that way, and not involve any sort of unnecessary outside influence from media or news outlets.

4.2.6 Ethical Constraints

We believed our project would have a greater impact if we made it available for everyone, instead of patenting it. We believed open source was the driving force in the betterment of a project that was designed to suit the individual needs of the musician. Instead of defining the features and constraints of what the product can and cannot do, we desired to leave it open to interpretation depending on what the user requires. We wanted to see how the community would welcome it and modify it to their needs. In this way we would get more reliable feedback and we would also have a better understanding on what the most popular features are.

4.2.7 Health and Safety Constraints

While few and far between, there have been fatalities and injuries related to musical equipment in the past. Vocalists have touched improperly grounded microphones on stage and suffered severe electric shocks. In addition, many musicians and road crew members have been injured by attempting to lift heavy amplifiers and audio equipment by themselves, thus resulting in lower back injuries.

Since the ADEPT prototype runs on a power input of 9VDC, with a current of no more than 500mA, the risk of any injury due to electric shock is extremely unlikely. Our design utilized an isolated grounding system as well, which further reduced the likelihood of any

inadvertent or latent discharge from the chassis or any connected cables. The ADEPT pedal is also extremely lightweight, under 3lbs. This small footprint and portable nature completely negates any potential for injury.

In order to bridge the gaps between the components of our circuit and bond them to our PCB, we used a low-melting point metal alloy known as solder. This alloy can contain toxic elements, most notably lead. Misuse of lead solder (including the inhalation of smoke from the soldering process, and improper handling of solder waste) can potentially have negative impacts on the health of the individuals that come into contact with it. With proper ventilation and handling of the solder, however, these negative impacts can be reduced or completely mitigated. In addition, our team opted to use low-lead or lead-free solder when assembling our prototype.

4.2.8 Manufacturability Constraints

Thankfully we didn't really find too many of these constraints limiting our project. When it comes to sourcing our components and materials, all of them were available at very reliable online stores, so this was not an issue at the current project scope. This could have changed if the project was taken to a bigger scale, manufacturing many guitar pedals would require us to maybe purchase equipment to build the PCBs necessary. For our final prototype, we were able to get our PCBs done by a third party company, thus, it was not too much trouble.

4.2.9 Sustainability Constraints

Audio devices tend to last for a long time if properly taken care of. Our objective is to build a device that can withstand the test of time. We stand behind the fact that all the components were mounted on a single board, and assuming we have followed all the guidelines and suggestions mentioned in the development of the PCB, we could achieve not only a high quality of sound but also a pretty durable device.

In terms of reliability, we wanted to make our design reliable enough to the point where the user can check for temperature. We also want to handle extreme situations which might prompt a system failure if no measures are taken beforehand, therefore we considered implementing an emergency shut-down triggered by sensors for overheating and for power abnormalities, we also like considered to have a system check routine once the device is power on, to assure that the device can operate correctly.

We could not guarantee how many years the device will last in terms of lifespan, however the average analog guitar pedal can last between 10-30 years. A digital guitar pedal could

last between 5-15 years. Of course the assumption here is that the user will take care of and won't expose it to any hazard or dangerous environment. At the end of the day, the components we used can be very sensitive to temperature and humidity.

Temperature was a huge factor that was taken into consideration when talking about performance and lifespan, for example, the MCU has a thermal resistance of 46 degree celsius, whilst the CODEC SNR and dynamic range are also affected when the temperature increases past 25 degree celsius.

If we considered a device like this and its performance in the current market, depending on the marketing behind it, we considered that it could possibly have a good standing and make a name for itself like many old analog guitar pedals have. However, since nowadays a lot of people can recreate old guitar pedal sounds and reverse engineer them it is hard to say that it would survive as a product itself.

5.0 Project Hardware and Software Design Details

In this section, we outlined the design process for the ADEPT prototype. This includes both the hardware design aspects (including the simulation and testing of our schematics, as well as PCB layout and tracing) and the software design/programming aspects.

To demonstrate how we designed the hardware for this project, we provided several simulations of the various parts of our overall circuit design, using both MultiSim Live and LTSpice softwares in order to do so. In addition, we included mathematical calculations that justified our selection of component values within our circuit.

We also provided any software-related design aspects that were necessary to realize the digital specifications of the ADEPT prototype, including the theory and equations behind all of our specific effects coding algorithms.

5.1 Initial Design Architectures and Related Diagrams

The first step in our design process was to generate a prototype diagram of our overall circuit layout based on our block diagram. This circuit schematic layout includes all the analog components that were used in our design, including the input and output buffers, the power supply, and the tone control. This schematic also includes the digital components of our circuit, including the ADC/DAC/CODEC and MCU chips, written in block diagram format for simplifications purposes.

This preliminary schematic did not have any assigned component values, and only used basic unlabeled symbols for each component. This is because we chose component values based on our Multisim Live and LTSpice test simulations, where we can easily swap out component values and decide what values will best realize the design functionality. Component values were provided later on, as did their respective justifications with the function of the circuit.

After drawing up a basic prototype schematic, the next step in our design process was to break the larger overall circuit down into its individual parts for testing. In this way, we were able to isolate any problem areas and focus on them without being distracted by the larger picture. As mentioned above, tests were done using circuit simulation software such as Multisim Live and LTSpice, and numerous calculations were made in order to realize the proper functionality of our design, pertaining to the requirements specifications we previously defined.

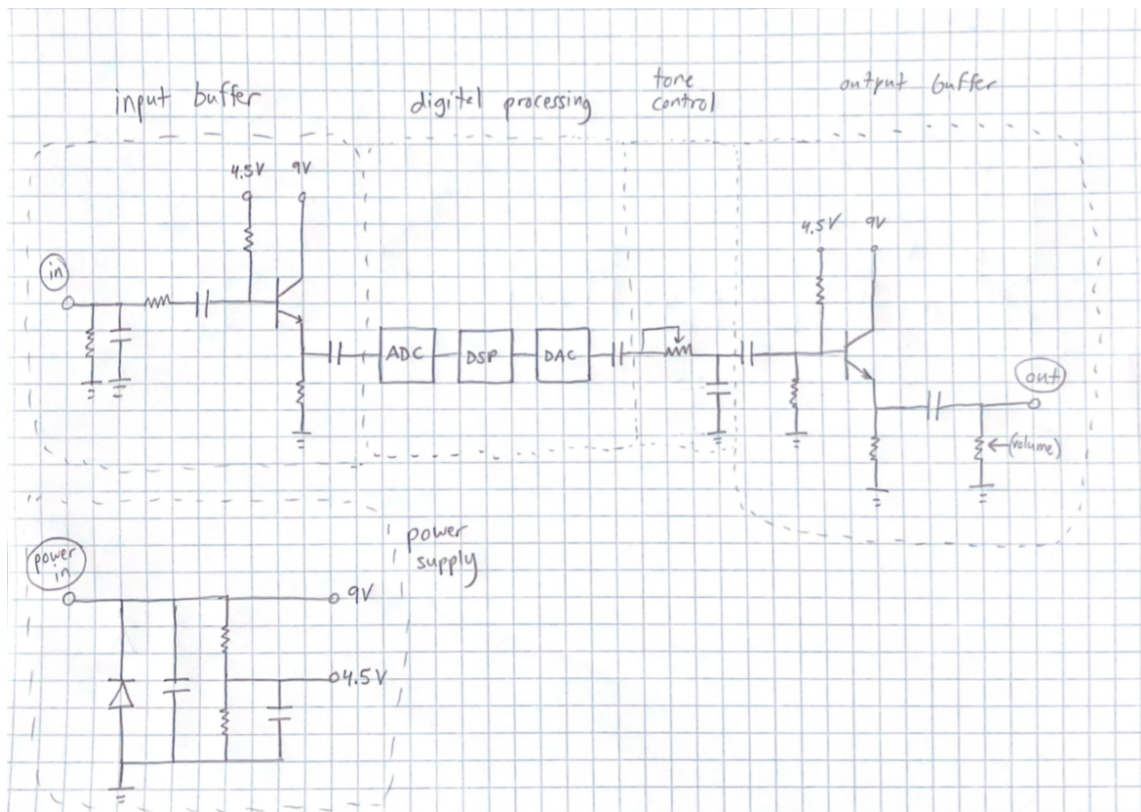


Figure 44: Overall Initial Circuit Design Prototype

5.1.1 Input Buffer Testing and Part Selection

We first analyzed the input buffer stage of our schematic by performing a small-signal AC analysis of the circuit, as well as simulating it in the Multisim Live schematic simulator software. By doing so, we were able to quickly select various resistor and capacitor values and adjust bias voltages in order to accurately realize the Common Collector/Emitter Follower transistor buffer configuration. As discussed before, this configuration is implemented in order to take the high impedance input from the guitar pickups and lower it down to a lower impedance to prepare it to enter into the rest of the circuit. This preserves signal integrity and prevents high frequency signal loss throughout the circuit.

When it comes to audio circuits, impedance is directly related to sound quality. The longer the signal path (distance between the instrument and the amplifier), the higher the impedance. The higher the impedance of the circuit, the lower the tonal quality of the signal will become. High frequencies will get removed from the signal, resulting in muddiness and a lack of clarity. This is why many guitar players who use long instrument cables (30 feet or more) experience a darker tone than those who use shorter cable lengths. The same can be said for those who have a large amount of pedals and effects in between their guitar and amplifier. The increase in length of the signal chain, as well as the lack of buffering along the signal path, will add to an overall decrease in tonal quality.

In order to mitigate these sound quality issues, we must ensure that the input and output impedances of our buffers are properly calculated such that there is no signal loss or tonal compromise. The input impedance of our guitar pedal at the input buffer should be anywhere between $470\text{k}\Omega$ to $1\text{M}\Omega$. Any less than this range would yield a muddy or muffled sound, and any more would have the opposing effect of a bright and thin sound that is harsh to the ears. This impedance range that comes directly from the guitar's pickups is very high, and will need to be brought down to a lower impedance level to prepare to enter into the CODEC. Such is the reason why we have chosen to implement the Emitter Follower BJT buffer amplifier configuration.

We wanted to make sure the impedance level going into the CODEC is as small as possible. This is because large impedances can amplify the effects of small capacitances, such as the stray capacitances that arise from various aspects of PCB design. These capacitances can arise from variances in trace width and spacing, the amount of layers in the PCB, and several other factors that can lead to propagation delays and noise in our digital components. More information about these phenomena is covered in Section 6.2 (PCB Vendor and Assembly).

The way we analyzed the Emitter Follower circuit was doing a small signal AC analysis of it. The small signal AC analysis of an Emitter Follower amplifier shows that any DC sources in the circuit act as a ground. In addition, all capacitors in the circuit behave as a short. In this way, the 9V DC voltage at the collector (V_c) behaved as a reference node equal to 0V (ground).

The maximum value of the 2N2222A transistor's DC current gain (β) is 300. We used this value to assist in calculating the input and output impedances of our buffer. Since we know that we want our input impedance to be high (between $470\text{k}\Omega$ to $1\text{M}\Omega$), we used this range as a reference when plugging in resistor values during our small signal AC analysis. The value of r_π was negligible in our analysis, since it was in parallel with resistor R2 in our circuit, and had an incredibly small value relative to the value of R2.

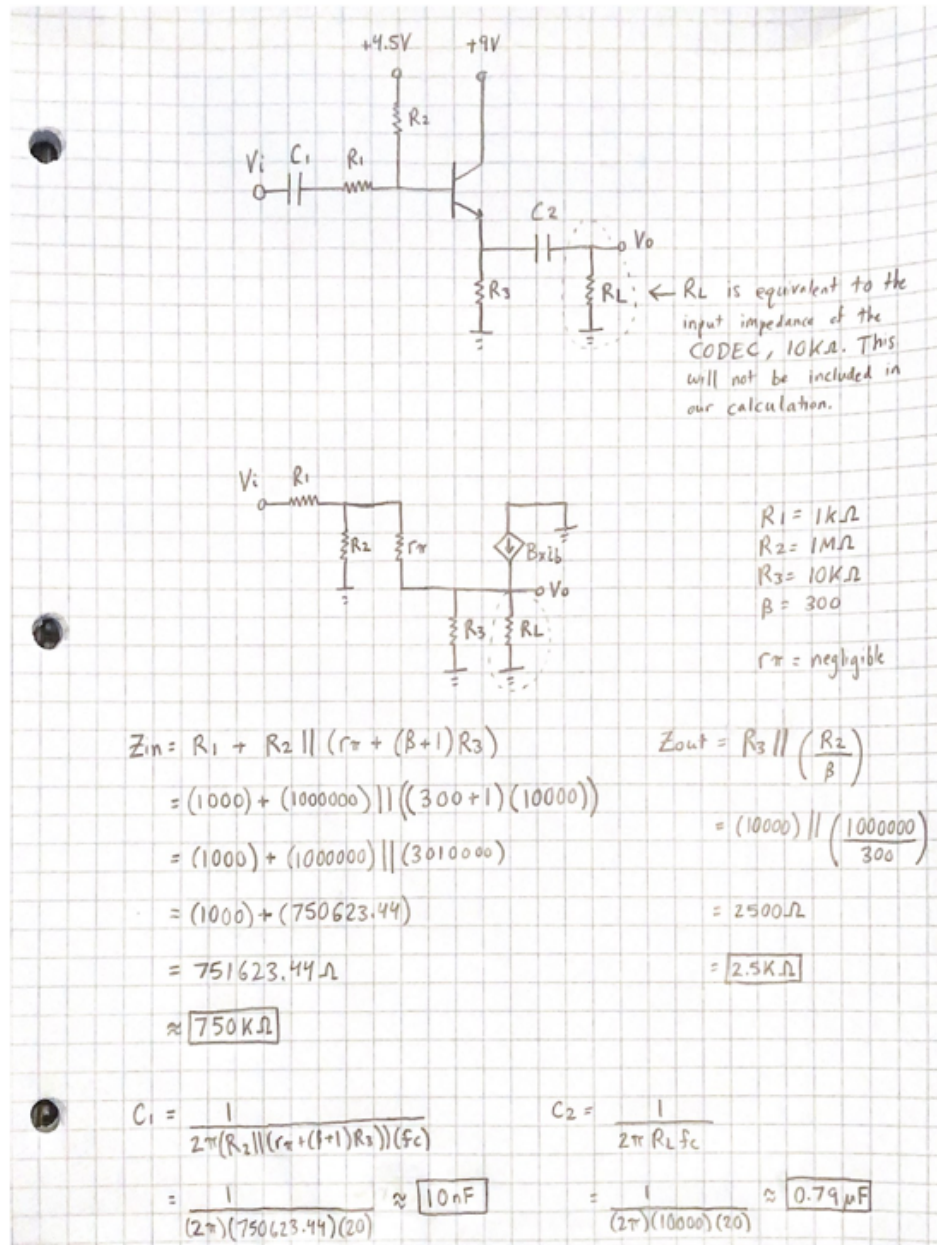


Figure 45: Input Buffer Small Signal AC Analysis

The input impedance of the CODEC is $10\text{k}\Omega$, and we calculated the values of the resistors and capacitors in our buffer such that we can accommodate this smaller impedance. A good rule of thumb was that the load impedance (in our case, the input impedance of the CODEC) should be much larger than the source impedance (output impedance of our buffer). This concept was known as “impedance bridging”, and was especially useful to implement in audio circuits, since it increases signal level, reduces distortion, and optimizes noise levels. In many cases, a common ratio between the source impedance Z_s and the load impedance Z_L (known as the damping factor) was 10. This means that the load impedance should be 10 times the source impedance that precedes it.

As can be seen in the calculations above, R_1 was equal to $1\text{k}\Omega$, R_2 was equal to $1\text{M}\Omega$, and R_3 was equal to $10\text{k}\Omega$. Using these values, along with the value of the current gain $\beta = 300$, we were able to find that the input impedance of our buffer was equal to $750\text{k}\Omega$, and the output impedance of our buffer was equal to $2.5\text{k}\Omega$. Since our input buffer had an output impedance of $2.5\text{k}\Omega$, our damping factor was 4 when compared to the CODECs load impedance of $10\text{k}\Omega$, but this was still acceptable and produced a desired result.

The base of the transistor must be biased to 4.5V in order for it to turn on and enter into the forward active region. This bias voltage was achieved by utilizing a simple voltage divider circuit that halved the voltage at the collector terminal (9V) and send it to the base of the transistor.

The coupling capacitors C_1 and C_2 in our buffer are placed such that the AC signal coming from the guitar and the DC signal coming from the 4.5V bias voltage do not mix. These capacitors, along with the resistors in the circuit, create high pass filters that are used to remove low frequencies that could cause aliasing or noise later on. The value of these capacitors were chosen such that the cutoff frequency of the high-pass filter was outside of the audio range. Therefore, any capacitor value chosen that yielded a cut off frequency of 20 Hz or less was acceptable. In our calculation, we found that a value of 10nF for C_1 worked the best to yield the cutoff frequency that we desired.

Since the input impedance of the CODEC was $10\text{k}\Omega$, we must use this load value to calculate the value for the other coupling capacitor C_2 at the output of our buffer. Similarly to the first capacitor we calculated for, the cutoff frequency of the high pass filter created by resistor R_L and capacitor C_2 should be outside of the audio range, 20 Hz or less. Therefore, the value of this coupling capacitor was about $0.8\mu\text{F}$.

In this configuration, the output voltage at the emitter was never more than the input voltage. This means that the voltage gain of the emitter follower transistor configuration was equal to 1. This was ideal for our project design, as we did not want to amplify our

signal at the buffer stage, but instead only reduce the impedance coming from the input of our guitar signal. The voltage at the emitter, V_e , was 0.7V less than the voltage at the base, V_b . This voltage drop becomes evident when observing the output waveform from the buffer - although not significant, there was a very slight voltage drop at the output of the buffer. This behavior was expected.

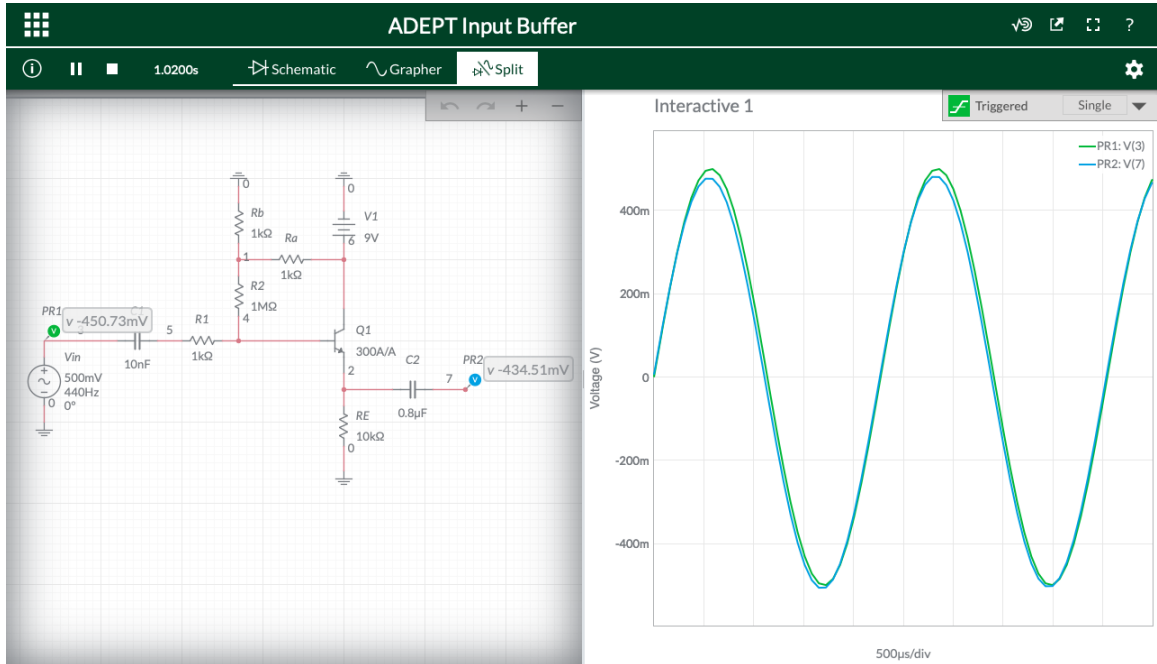


Figure 46: Input Buffer Transient Response ($V_{in} = 500\text{mV}$, 440Hz)

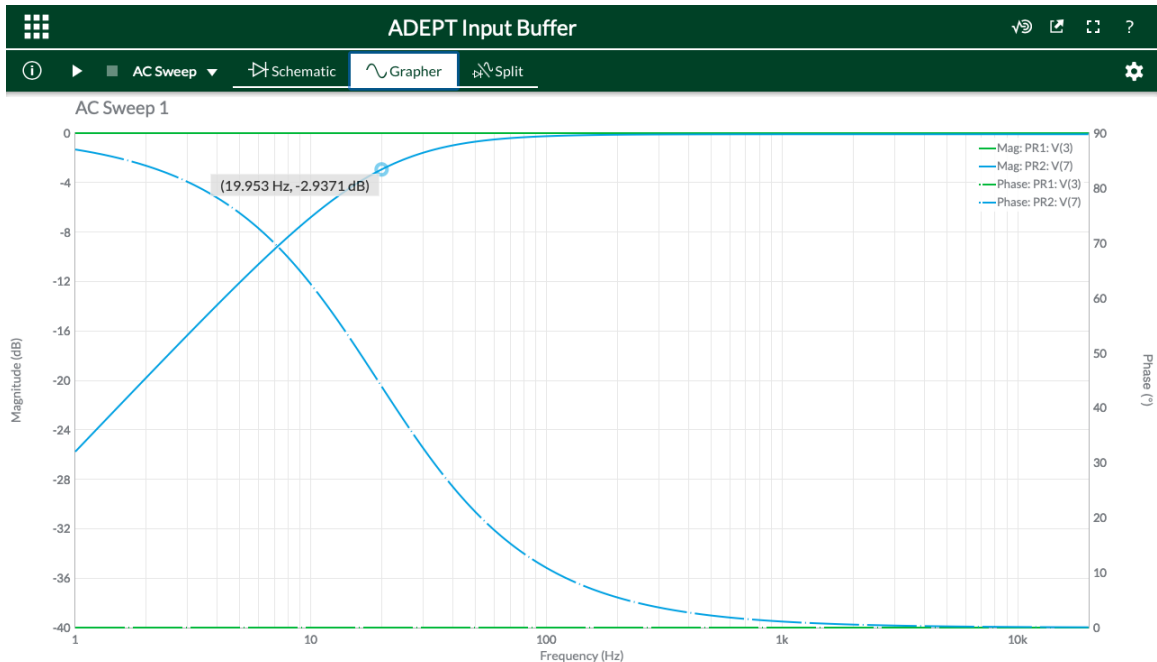


Figure 47: Input Buffer Frequency Response (Cutoff Frequency = 20Hz)

5.1.2 Output Buffer Testing and Part Selection

This section is very similar to the development of our input buffer. We implemented a detailed analysis of the output buffer stage of our schematic by simulating the behavior of our circuit in Ltspice circuit simulator. This helped us to hone in specific values of components, such as resistors and capacitors, needed to implement our design. We were also able to understand the behavior of the output buffer. We will construct our output buffer in the emitter follower configuration. The purpose of the emitter follower was to increase or maintain voltage gain to our speaker. This accelerates the process of estimating ideal resistor and capacitor values in our circuit. The goal was to send a clean signal with a low output impedance to the guitar amplifier.

First off, we will need to take impedance matching into account to ensure that the impedance was the same or nearly equivalent with respect to our output buffer and guitar speaker. If the impedance mismatch was high between the speaker and the output buffer, the signal can deteriorate and cause unwanted distortion when using our guitar pedal. The input to the speaker will have a high input impedance. Therefore, the output of our buffer needed to have a low output impedance. However, if the difference between the two impedance values is too high, the signal travelling from the output buffer to the guitar amplifier will undergo attenuation. We needed to carefully select the output impedance

value to avoid attenuation. The impedance of the cable will also add on to the series output impedance of the buffer. The goal was to decrease this impedance to avoid attenuation along the signal path. We cannot control the impedance of the cable. Therefore we must compensate for this when designing our output buffer.

The input impedance of a standard guitar amplifier was $1\text{ M}\Omega$ with an output impedance of 1Ω to 3Ω . So the same concept applies. Have high input impedance and a low output impedance to avoid attenuation. This was why we wanted a low output impedance of our output buffer.

First we analyzed all components in the emitter follower output buffer and understood the reason for each individual function. The base was biased to $+4.5$ volts through R5 to turn the transistor into forward active condition. Resistor R1 and Capacitor C1 function as a high pass filter that removes low frequencies from the signal path. C1 acted as a coupling capacitor to isolate DC components from the common emitter amplifier. We used a voltage divider bias of 4.5 Volts through R5 to the base of the BJT. We optimized our R5 value to as far in the forward active region as possible to maximize gain at the output. This did not affect the DC biasing of the circuit. R2 puts the transistor into common emitter condition by R2 being tied from the emitter to ground. C2 was used as a coupling capacitor. This isolated any DC current from causing a DC offset in the guitar amplifier. It also contributed to the frequency response of the High Pass effect of the circuit. Below are a schematic, AC Sweep, and AC input/output measurements as a function of time.

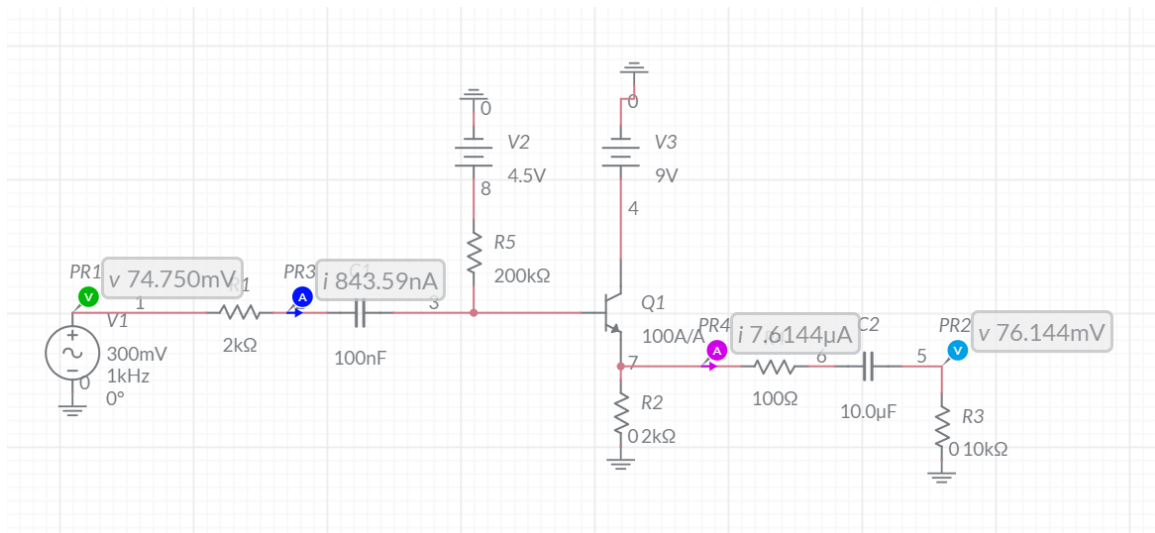


Figure 48: Output Buffer Circuit Simulation

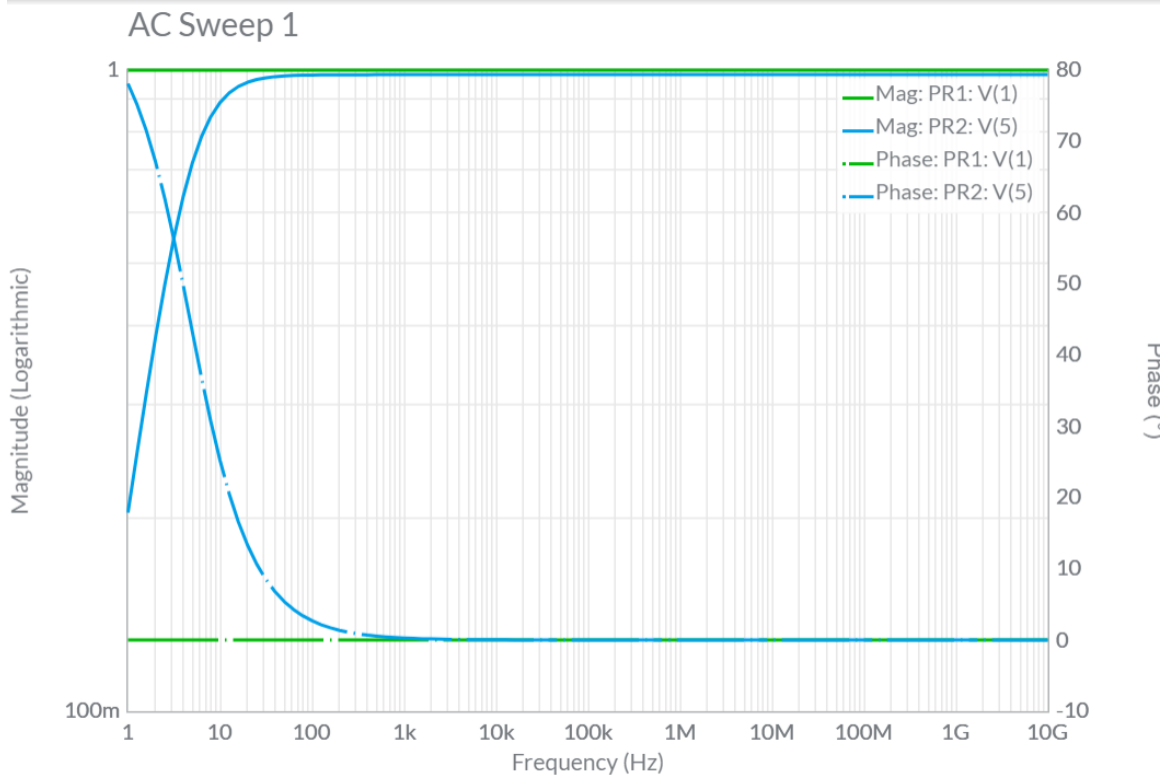


Figure 49: Output Buffer Frequency Response

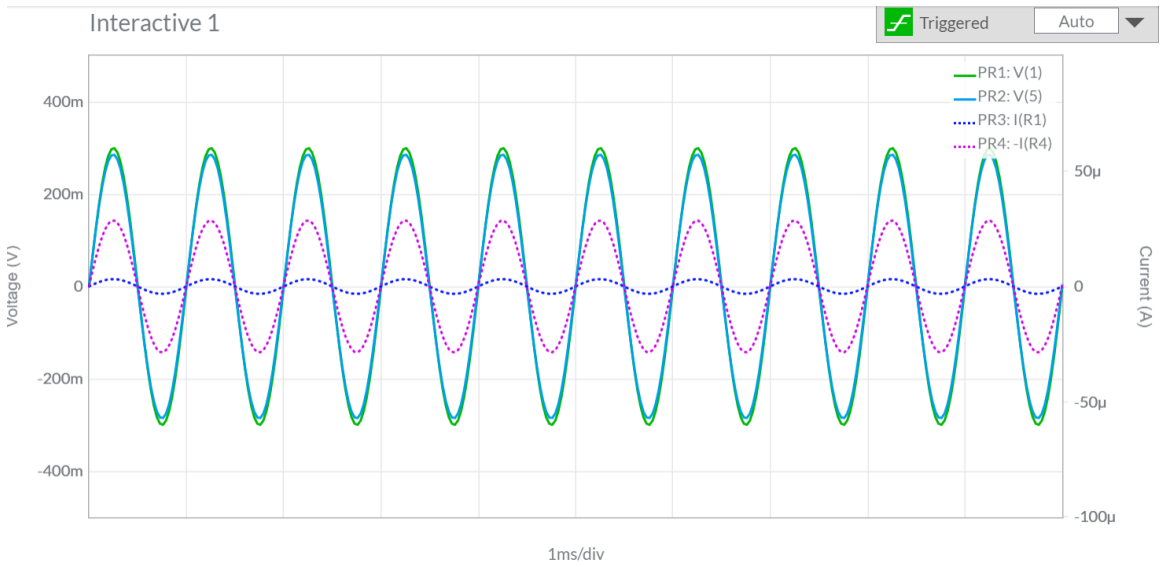


Figure 50: Output Buffer AC Measurement ($V_{in} = 300mV$)

When selecting values for our values for the analog output buffer, we wanted to make our output impedance as small as possible. We aimed to make R2, R3, R4, and R5 as small as possible while maintaining functionality of the buffer. We performed a small signal analysis to find the ideal resistor values and check those values in the simulation to make sure that the buffer would still operate properly. We created a hybrid-pi model to represent the small signal circuit of the output buffer. Since we are finding the equivalent impedance from the small signal circuit we eliminated the current source from the model. This left the current source as an open circuit. Also take note that r_{π} was much smaller than R5. Hence we eliminated r_{π} from the calculations. Below is a handwritten calculation to find the output impedance of the analog output buffer.

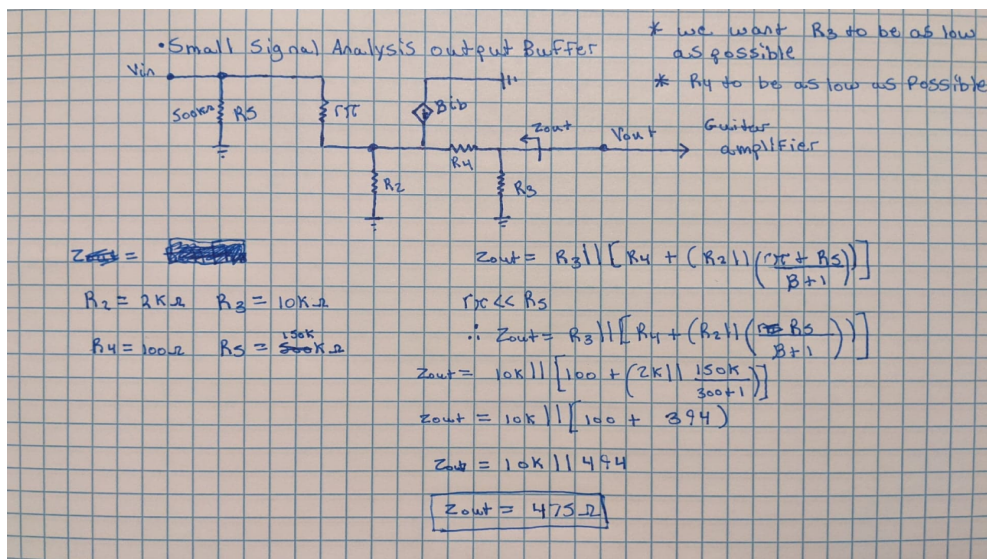


Figure 51: Output Buffer Small Signal Output Impedance

5.1.3 MCU Design Specifications and Part Selection

5.1.3.1 Prototype

The envisioned idea for our product's design was to encompass our idea of having a pedal that was simplistic yet provides enough inputs for the user, to facilitate menu traveling and selection of options. We also wanted a design that can be easily adjustable in the case we face a roadblock during the development of the firmware, an example of a potential problem we could face would be having to redesign the way in which the user travels the menu by the use of a knob, if we were faced with the decision of needing the user to confirm the selection then we would need the addition of a button in order to capture that expression. We believed that we could implement both of these ideas by using a push-button rotary encoder, this allows the user to perform the menu travel action and also the option selection action using the same knob.

A push-button rotary encoder works in a similar manner as a regular rotary encoder. The shaft rotates either clockwise or counterclockwise, and of course there's no limit to the rotation, thus one can fully rotate the shaft endlessly. Every 22 degrees the shaft will output a two bit binary code from its output pins, resulting in four unique values in a full rotation, once the user has repeated a full rotation the code will repeat itself. The order in which the output values are sent represents the direction of the rotation. Suppose a clockwise rotation gives the following list of values 0, 1, 3, 2, and when rotated counterclockwise we get 0, 2, 3, 1. By polling the state of the encoder and saving the last value obtained, we can then determine the direction of the rotation. We can use this to realize the action for traversing the menu, moving the selector up or down depending on the direction of rotation.

Furthermore, we have also incorporated 3 knobs for parameter adjustment in case we want to further the expression of each effect, this is unrelated to the tone section which controls the volume and dry/wet parameter of the overall mix. The parameter knobs were subject to change depending on how much we are able to accomplish in terms of code and implementation of the effect, however it was always a good idea to leave "room for error" in case we have the option to include such features.

Additionally, another feature we would like to implement/"make room for", was the tap-tempo/looper softswitch. The concept of using a tap-tempo feature will mainly come into play when we are implementing time-based effects, such as delay or tremolo. By using a tap-tempo adjusted the number of repetitions of the effect in a specific time period. Another state or function for the softswitch was as a start/stop recording button for the looper state.



Figure 52: Prototype Diagram

5.1.3.2 High Speed External Oscillator (HSE):

In order to correctly choose the proper high speed crystal resonator, one must first understand and have in mind the desired frequency for the oscillator. In our case, the desired frequency for our oscillator was 8 MHz, as described and realized in the external oscillator section of the MCU.

In this section we will explain the general approach for selecting a compatible oscillator for the STM32F446. The compatibility based on the table 39 [7] and the guidelines provided in [5]. As previously mentioned, knowing the frequency of the oscillator was a pre-requirement for starting out selecting a correct oscillator, however, a good tip for future revisions was to also know the stray capacitance, which was the total capacitance of both the crystal's leads and the pcb traces on the board. Before we dive into the explanation of how we approach the lack of having this latter requirement, let's first look

at the initial specifications we looked for in an external oscillator.

For the purpose of this example we used the ECX-3SX SMD Crystal [8], by looking at the datasheet we can see the following specifications that we should look for:

ESR = 100, Co = 7 pF (max), CL = 20 pF (typ).

Following the advice in [7] after table 39, CL1 & CL2 should be in the range 5 - 25 pF and both are usually the same size, then we can assume proper values for CL1 and CL2. Thus, we could then proceed to start doing the calculation steps in section 3.4 of [5]

$$g_{m\text{crit}} = 4 \times \text{ESR} \times (2\pi F)^2 \times (C_0 + C_L)^2$$

Figure 53: Gmrit Equation

The result for the Gmrit for the ECX-3SX equal to 7.3676×10^{-4} A/V, this result is acceptable based on table 39, since it is less than 1 mA/V. Moving forward we can calculate the Gain margin assuming that the Gm = 25 mA/V, where Gain margin = Gm / Gmrit. The Gain margin obtained is 33.93, which is sufficient to start the oscillation and meets the condition Gmargi > 5.

Recalling the point we left at the beginning of this section, let's calculate the stray capacitance based on the CL1 and CL2 values we assumed using the following formula provided in section 3.3 of [5].

$$C_L = \frac{C_{L1} \times C_{L2}}{C_{L1} + C_{L2}} + C_s$$

Figure 54: Stray Capacitance Equation

Thus, the stray capacitance calculated came to 15 pF, with the assumption that both CL1 and CL2 are equal to 10 pF. Of course, once we had a solid value for the stray capacitance of our board we then proceeded to change the values of CL1 and CL2 accordingly in order to satisfy the requirements on table 39.

The following is a screenshot of an excel sheet used to process the calculations for different crystals:

1	STM8S								
2	STM8A								
3	STM32Fxxx								
4	Pierce Oscillator								
5									
6	Gain Margin is the key parameter that determines whether the oscillator will startup, we are looking for a minimum of 5								
7	Gain_margin	=	$\frac{G_m}{G_{merit}}$						
8									
9									
10	Assume:								
11	Assume:								
12	Assume:								
13	Assume:								
14	Gmerit	=	$4 \times ESR \times (2\pi F)^2 \times (C_o + C_L)^2$						
15	Example:								
16									
17	Gmerit	=	$4 \times 80 \times (2 \times \pi \times 8(10)^6)^2 \times (7(10)^{-12} + 10(10)^{-12})^2 = 0.23mAV$						
18									
19	Enter values	F (Hz)	8000000	Co (Farads)	7E-12	CL (Farads)	1E-11	ESR (Ohms)	80
20	Gmerit	=	0.233661305	mAV					
21									
22	Assume:								
23	Gain_margin	=	$\frac{G_m}{G_{merit}}$	=	25	=	106.9925		
24									
25									
26									
27	General Case:								
28	Enter your values	F (Hz)	24000000	Co (Farads)	7E-12	CL (Farads)	8E-12	ESR (Ohms)	2
29	Gmerit	=	0.040931154	mAV					
30	Gain_margin	=	$\frac{G_m}{G_{merit}}$	=	5	=	122.15634		
31									
32									
33	Your Case:								
34	Enter your values	F (Hz)	32000	Co (Farads)	8E-13	CL (Farads)	7E-12	ESR (Ohms)	65000
35	Gmerit	=	0.000639472	mAV					
36	Gain_margin	=	$\frac{G_m}{G_{merit}}$	=	0.00056	=	0.8757225		
37									
38									

Figure 55: Excel Sheet Formulation

Finally, we then calculated R_{Ext} as shown in section 3.5.3 of [5]:

An initial estimation of R_{Ext} is obtained by considering the voltage divider formed by R_{Ext} and C_{L2} . Thus, the value of R_{Ext} is equal to the reactance of C_{L2} .

Therefore $R_{Ext} = 1 / (2 \pi F C_{L2})$, and so, with an oscillation frequency of 8 MHz and $C_{L2} = 15$ pF, we have $R_{Ext} = 1326 \Omega$.

So, the value obtained for R_{Ext} in this example comes to 1989.437 ohms. Additionally, as mentioned in the notes in section 3.5.3, it was recommended that R_{Ext} was added to the ESR and the G_{merit} be recalculated in order to check if $G_m \gg G_{merit}$. If this remains true, then we know that the oscillator startup will stay within acceptable parameters. When we calculate once again the G_{merit} in the equation with the addition of R_{Ext} to ESR, then the value obtained was 0.015394 A/V, about 15 mA/v, if we commute once again the gain margin using this value for G_{merit} , then we obtain a result of about 1.6. In conclusion, once we added R_{Ext} to the calculation, we determined that the oscillator we picked as an example will not meet the conditions, since 1.6 was not greater than 5 and the newly computed G_{merit} value was not less than 1 mA/V. The following figure depicts the typical application of an 8 MHz crystal attached to the STM32F446 chip.

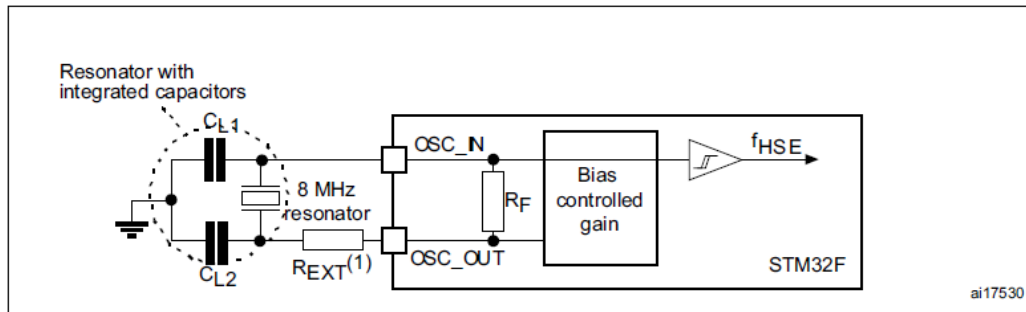


Figure 56: External Crystal Oscillator Network

5.1.3.3 GPIO Configuration:

For most of the connections established with the MCU and any external component, we decided to use the STM32cubeMX IDE pinout configuration feature, which prompts the user a simple UI with the corresponding MCU package they are working with. This was a simple way to start selecting the necessary pins that were needed in the project.

HSE: In order to enable the high speed external crystal pins we selected System Core > RCC. Then, we continued to select from the drop-down menu Crystal/Ceramic resonator, this enabled the corresponding pins on the Pinout view layout.

USB/JTAG: Similar to the HSE, we selected System Core > SYS. Then, we selected from the Debug drop-down menu Trace Asynchronous Sw, this is the USB initialization for the SWD debugger, if we were to use the JTAG, we would then select the JTAG 5-pin option from the drop-down menu. For testing purposes we liked to select the pins assigned to both of these options since we implemented an external ST-LINK/V2 interface.

CODEC: To set up the correct communication with the CODEC, knowing that the MCU was the one driving the clock, the following shows the necessary target pins using the I2S mode of the SPI peripherals.

MCU	Pin #	ADC	Pin #
I2S1_SD	PA7	DOUT	3
I2S1_WS	PA4	LRCK1	4
I2S1_MCK	PC4	SCKI1	6
I2S1_CK	PA5	BCK1	5

Table 8: MCU/ADC Pin connection

MCU	Pin #	DAC	Pin #
I2S2_SD	PC1	DIN	12
I2S2_WS	PB12	LRCK2	11
I2S2_MCK	PA6	SCKI2	9
I2S2_CK	PB10	BCK2	10

Table 9: MCU/DAC Pin connection

Additionally, we wanted to select in the Parameter Settings in the IDE the following: Data and Frame Format: 24-bits data on 32 Bits Frame and Selected Audio Frequency: 48 kHz.

LCD: For the implementation of the LCD we first have to look at the connection scheme we will use. Popular approaches for connecting an LCD to the STM32's was by using I2C with the help of an external GPIO extension and programming libraries such as HAL. This was a neat way to minimize the use of GPIO pins on the MCU, but in return you are also adding an extra external component on the PCB as well as adding a heavy library to the code. Instead, we have decided to use a more direct approach, since we have a lot more pins than what we need on the MCU. We used a direct connection between the GPIO pins in the MCU and the pins of the LCD.

There's different approaches to how we can connect the LCD to the MCU, we could use 11 pins (8-bit data bus) or we could instead use 7 pins (4-bit data bus). We are thinking of using the latter one since we didn't want to utilize that many pins in our MCU in the case we wanted to add more components to it in the future.

The 7-pin approach requires 4 GPIO pins (data bus), and the usual RS (data/command), RW (reads the busy flag) and E (strobe) pins, although this approach isn't as efficient as the 11-pin approach, the difference in delay was relatively negligible.

Any GPIO pin could be selected as long as they are 5V tolerant, since we were using an LCD that was powered using 5V. Most of the pins in the STM32F446RC were 5V tolerant, this can be seen in section 4 of the STM32F446 datasheet. The I/O structure markings for 5V tolerant pins were denoted by "FT" and also "FTF", however, the latter one usually denoted pins that have I2C FM+ options, which was something we used in the development. Part of selecting the correct pins for the LCD depended on the usability and options the pins have, if a pin has alternate functions that were necessary for other

component's connection then we wanted to leave those alone, this constraint implies that the availability of having 7 pins of the same port and placed consecutively might not be realizable, though did not affect the overall programming of the pins, it did affect the convenience of having to program one port instead of multiple independent pins. Additionally, we also ensured that the power and control pins we assigned in the MCU had push-pull capability, and in the case that we would power LCD using the MCU we must ensure that the VDD pin we are using for power can supply 5V.

External Flash Memory: In order to provide the MCU with enough memory space for long recording we set up an external flash memory, such as the W25Q128JVYIQ, which is a NOR flash memory that operates on a single 2.7 to 3.6 V power supply. It has 16 MB of memory and was organized in 65,536 programmable pages of 256-bytes each. The chip also communicated through SPI connection with up to 66MB/S transfer speeds. Even though the chip is capable of using both Dual and Quad SPI, we decided that using standard SPI would be more beneficial since we required less pins from the MCU in order to interface with the memory while offering sufficient transfer speeds.

More specifically For the GPIO connections between MCU and memory, we used one the dedicated SPI peripherals in order to connect to the Chip Select Pin, Data Out Pin, Data In Pin, and Clock Pin of the memory. We supplied 3.3V of power through the VCC Pin and a respective ground through the GND Pin. Additionally, we tied the pins Write Protection and Hold to a constant high since they are active low pins and we did not use those features from this memory.

5.1.4 CODEC Design Specifications and Part Selection

Inside of the PCM 3060 CODEC datasheet, we found a section regarding typical connection arrangements that we used as a guide. An input of 3.3V and a ground line are needed to power the device. I2C was used for mode control, and I2S for audio data communication.

Many of the pin connections made from the CODEC were to the MCU. To start, SCL and SDA I2C lines will go to the corresponding pins in the MCU. For I2S, we must have pins LRCK1, BCK1, and DOUT lines going to our audio receiver and LRCK2, BCK1, and DIN to our audio transmitter. SCLK1 and SCLK2 lines were connected to the external oscillator, which synchronized the whole signal processing and conversion process in between elements.

For the analog input pins, they were required to use a 4.7uF capacitors for our VinL input line. It is important to remark that we are working with mono signals only, and therefore we only used the left channel pins for both input and output. We also included two 10uF electrolytic capacitors between Vcc and AGND1 lines for one, as well as Vdd and

DGND lines for the other to reduce any kind of noise from reflections. The analog output lines are four, VoutL+, VoutL-, VoutR+, and VoutR-, but as mentioned before, we only used the left channel lines, which means that we only connected VoutL+ and VoutL- to the post filtering section of the pedal. We discuss this post filtering section, also known as the tone control of our pedal, in the following section. Calculations were made in order to determine the cutoff frequency of the tone control, as well as any other design choices pertaining to the hardware integration.

5.1.5 Tone Control Testing and Part Selection

For the tone control of our circuit, we decided upon using the simple “Bluesbreaker” configuration, which has been utilized in many other guitar pedal designs, as discussed before. This simplified design is essentially a low pass RC filter, which consists of a capacitor, resistor, and potentiometer (variable resistor). This will allow us to quickly tweak the tone and attenuate the high frequencies of our output signal coming from the pedal. We have decided to integrate this tone control configuration into our pedal due to its simplicity and ease of use, as well as its low cost. The low cost of this configuration was due to the very few parts that it takes to integrate - only 3 components, as mentioned above.

We will use a $100\text{k}\Omega$ potentiometer in series with a $1.5\text{k}\Omega$ resistor. This resistor was connected to ground via a 3.3nF capacitor. This created a low pass filter that rolls off frequencies above roughly 500Hz .

It is common in guitar pedals to have the tone knob configured in such a way that turning the knob to the left makes the tone darker, and turning it to the right makes the tone brighter. We implemented this configuration such that tone control adjustment was more intuitive to the musician using the ADEPT pedal.

When the resistor was turned all the way to the right (0% resistance), the maximum cutoff frequency was very high, at 32kHz . In this way, a very large majority of the frequency spectrum was let through, resulting in a bright tone. When the resistor was turned all the way to the left (100% resistance), the minimum cutoff frequency will lower to 475Hz , resulting in a dark tone.

The use of the tone control is subjective to the musician using the ADEPT pedal. Depending on the particular effect, some of the effects that our pedal provides may benefit from a darker tone, which could result in a “lofi” or low fidelity sound which many musicians find appealing.

The equations shown below were used to calculate the cutoff frequency of the low pass filter that we implemented for use in the tone control of the ADEPT. As can be observed

from these equations, we also can calculate the minimum cutoff frequency and maximum cutoff frequency of the filter. These minimum and maximum values pertained to the values of the cutoff frequency when the tone control knob was turned all the way to the right (maximum cutoff frequency, bright tone) and when the tone control knob was turned all the way to the left (minimum cutoff frequency, dark tone).

$$f_c = \frac{1}{2\pi \cdot (R_{TONE} + R_7) \cdot C_8}$$

$$f_{cmin} = \frac{1}{2\pi \cdot (100K + 1.5K) \cdot 3.3n} = 475Hz$$

$$f_{cmax} = \frac{1}{2\pi \cdot (0 + 1.5K) \cdot 3.3n} = 32KHz$$

Figure 57: Minimum and Maximum Cutoff Frequency Calculation

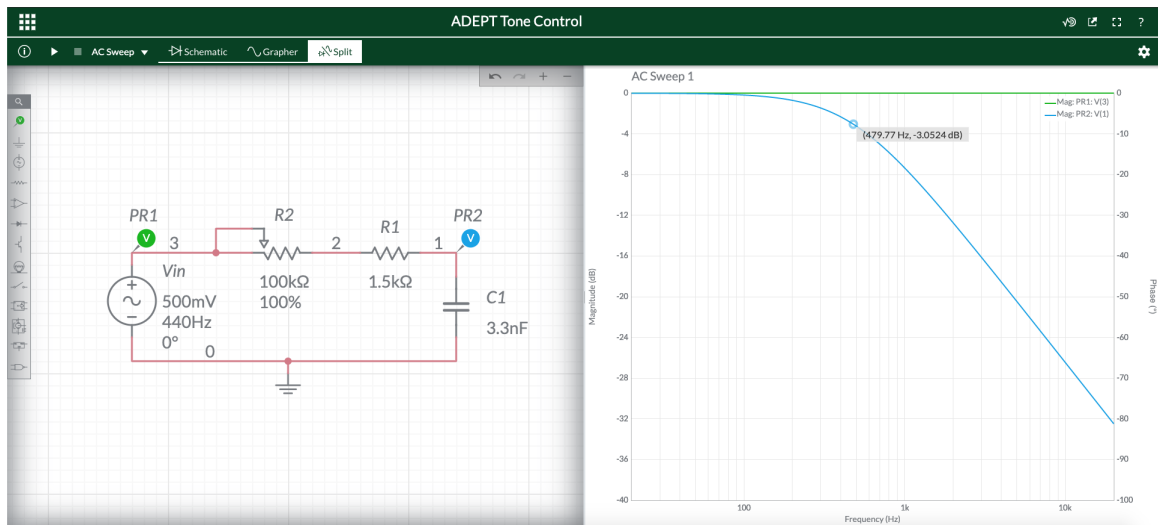


Figure 58: Tone Control Circuit Simulation (Cutoff Frequency = 500Hz)

5.1.6 Power Distribution Testing

Objective

These tests were to verify that each power component was functioning properly. We were doing a piecewise approach to isolate the individual power devices to eliminate unnecessary variables. We needed to measure the data needed to confirm that the devices were operating properly and compared them to results found in simulation.

Temperature Test

Equipment needed: Digital Multimeter, Power Source

Summary:

This test was to confirm that the temperature of the voltage regulators were within temperature margins specified in each datasheet. We used a J-Type thermocouple with a digital multimeter. If the temperature of each voltage regulator was within specified margins found in the datasheet, the passed and we moved to the next step.

Output Test

Equipment needed: Digital Multimeter, Power Source

Summary:

This was a simple test to verify that the voltage regulators are distributing the desired output with an input of 9V. We will use a digital multimeter to measure the input of 9V at the input pin of the regulator and we will measure the output and compare it to the values found in the simulation.

Voltage Regulator Load Current Test

Equipment needed: Digital Multimeter, Power Source, Load Box

Summary:

The reason for this test was to emulate scenarios in which other devices may unexpectedly draw more current than usual. We confirmed the dropout voltage by testing various loads at the output of each voltage regulator. To do this, we set up a load box in parallel to the output resistor that was equivalent to the input resistance of the device in which we are supplying voltage. We used our original 9V 500mA supply to make the test more realistic because as we increased the load current at the output of the regulator, it would require more current draw at the input of the power source.

We wanted to confirm that the maximum load current would not drop out under 1A since the devices we are using will not draw any more current than 1A. We slowly increased the load and analyzed the drop in output voltage. Once the output voltage was less than

0.8V of the specified output regulator voltage, we will consider the voltage to have “dropped out”. Furthermore, we would not run the load for more than 5 seconds. This was because any high current draw would either be caused by some transient response in the system. Since transient response was very brief in time, there was no need to stress the components out in this nature for more than 5 seconds.

Power Supply Load Current Test

Equipment needed: Digital Multimeter, Power Source, Load Box

Summary:

We applied various load currents directly across the output of our power source; this was to confirm that the power source was realistically able to supply 500mA without any issues. We applied a load current with the load box and incremented the load current to 485mA. We did not put a 500mA load as the performance would reach its peak just under 500mA. We did not want to damage our power source in the process of this test.

Grounding Continuity Test

Equipment needed: Digital Multimeter, Power Source. Oscilloscope

Summary:

This test was to verify that there was minimal potential difference between different grounds. First we powered on all devices and used a digital multimeter and measured across each ground in reference to one another. We then verified that the voltage difference between each ground was less than 1mV. Furthermore, we powered off our device and measured the resistance between grounds to confirm continuity. If there was a significant voltage difference between one ground to another, we would have needed to go back and analyze our grounding methods by exposing the points of discontinuity and eliminating any elements causing impedance between grounds.

Grounding Noise Stability Test

Equipment needed: Power Source, Oscilloscope

Summary:

This test was to verify that all grounds among all devices in which we were distributing power were absent of noise. We powered on our device and set the oscilloscope to AC coupling. We then measured across each ground in reference to one another. Next, we measured the magnitude of AC voltage for any noise signals significant enough in magnitude to effect the operation of our devices. If the noise was higher than 39mV, the system failed the test and we needed to analyze and verify where the source of noise was.

Power Supply Isolation Test

Equipment needed: Digital Multimeter

Summary:

This test was to verify that the power source was isolated with some sort of transformer. We could not simply open our power source without special tools to physically see what was inside the plastic enclosure. To verify that the system was isolated, we used a digital multimeter to measure the resistance between the input and output nodes. If the circuit was isolated, we will measure a very high resistance typically in the mega ohms value. If the resistance on the multimeter measures less than $2M\Omega$ then we know that the circuit was not isolated.

5.2 Breadboard Test and Schematics

The following section includes the various physical breadboard tests and relevant test results that we found when implementing our physical components for the first time. Breadboarding was an important step in the design process, as it allowed us to quickly and easily swap out components and implement real-time testing processes in order to prove that our various circuit configurations will function properly when mounted on our PCB.

We used a standard 0-35V, 0-3A power supply to supplement our 9V, 500mA plug-in adapter for the testing of our voltage regulators and input/output buffers. We used the Digilent Analog discovery 2 board to implement a function generator and oscilloscope. We also confirmed the functionality that the oscilloscope feature was operating properly by measuring directly at the output of the function generator pin. Since our power source was not calibrated recently. We confirmed the accuracy of the power source output by measuring the voltage with a digital multimeter. We made sure to measure at the opposite ends of the wires to confirm that there would be minimal voltage drop at the inputs to our devices. We set the function generator to a 500mV, 440Hz input to emulate the guitar pickup's output. Pickup output signals are typically 300mV and 20Hz to 300Hz. We made our function generator values slightly higher to be conservative in by proving that the circuits were stable and not attenuate at high frequencies. Below is a picture of the power supply that was currently supplying power to our bread boards. As you can see, the device was drawing current to confirm that we did indeed implement the test without putting it together and simply taking pictures.



Figure 59: DC Power Supply for Testing (9V, 500mA)

5.2.1 Input Buffer Breadboard Test

Our first breadboard test involved the construction of a simple unity gain Emitter Follower transistor buffer using a 2N2222A transistor. We also used six 1/4W standard sized resistors of various values, as well as two film capacitors and one ceramic capacitor. In order to test the functionality of this breadboard prototype, we attached a 9V, 500mA power supply to the positive and negative rails of our breadboard and tested various points with a multimeter. With this setup, we were able to successfully recreate the results from our MultiSim and LTSpice tests, yielding a high input impedance and low output impedance, as well as a voltage gain roughly equal to or less than 1.

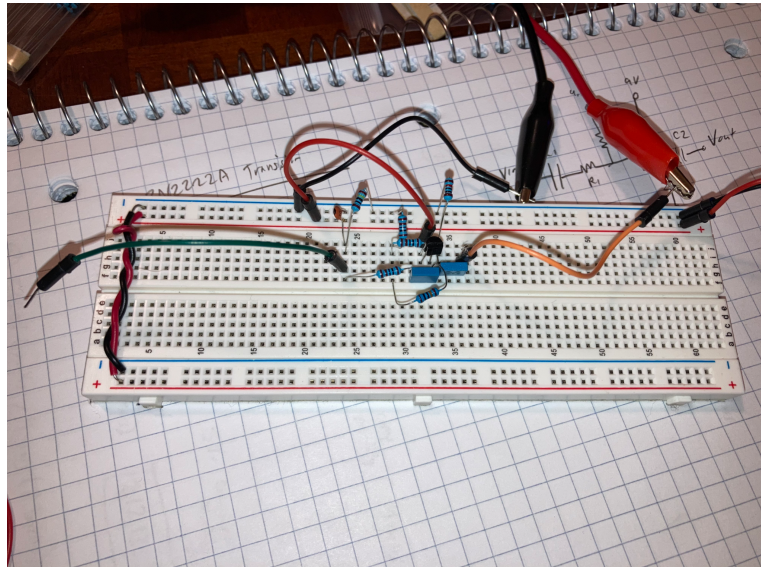


Figure 60: Input Buffer Breadboard Test



Figure 61: Input Buffer Transient Response ($V_{in} = 500\text{mV}$, 440Hz)

5.2.2 Output Buffer Breadboard Test

The output buffer was similar to the input buffer in the sense that it was in the emitter follower configuration. We chose the same type of transistor, resistor values, and capacitor values of the simulation and compared the behavior of our circuit to simulated results with respect to the input and output. We saw a slight phase shift between the input and output on the oscilloscope. Also having the power supply exactly at 9V caused a slight DC offset. We were able to negate this offset by setting a 1k resistor between the

9V power supply and the collector node of the BJT. Furthermore, we confirmed that the capacitor at the base of the BJT was the reason for the slight phase shift. However, this fortunately will not affect the quality of the audio signal. The signal did not attenuate under relatively higher or abnormal voltages or frequencies. We have confirmed that the output buffer design was sufficient enough for properly implementing our design and keeping a good quality signal.

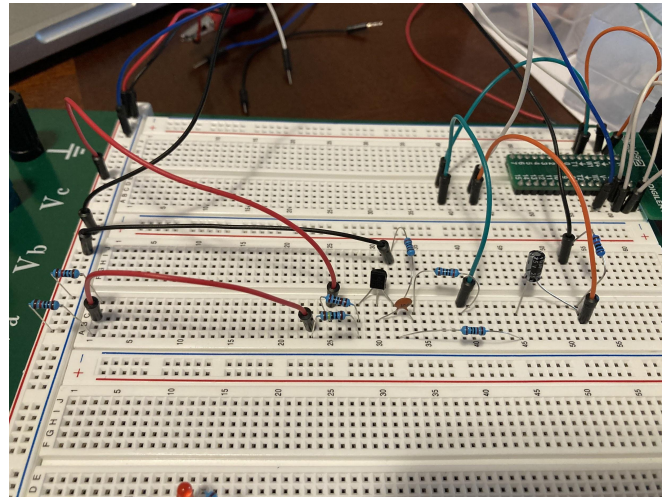


Figure 62: Output Buffer Breadboard Test



Figure 63: Output Buffer Transient Response ($V_{in} = 500\text{mV}$, 440Hz)

5.2.3 MCU Breadboard Test

The following PCB prototype was drawn up in the Eagle software with the intent of having a functional way to test our STM32 MCU. Since the MCU was surface-mounted (SMT), it was necessary for us to create a way for us to interface with it. The design

below allowed us to do so via a breadboard, since all the pinouts of the STM chip was to be routed to long vertical pins that could be inserted into a breadboard.

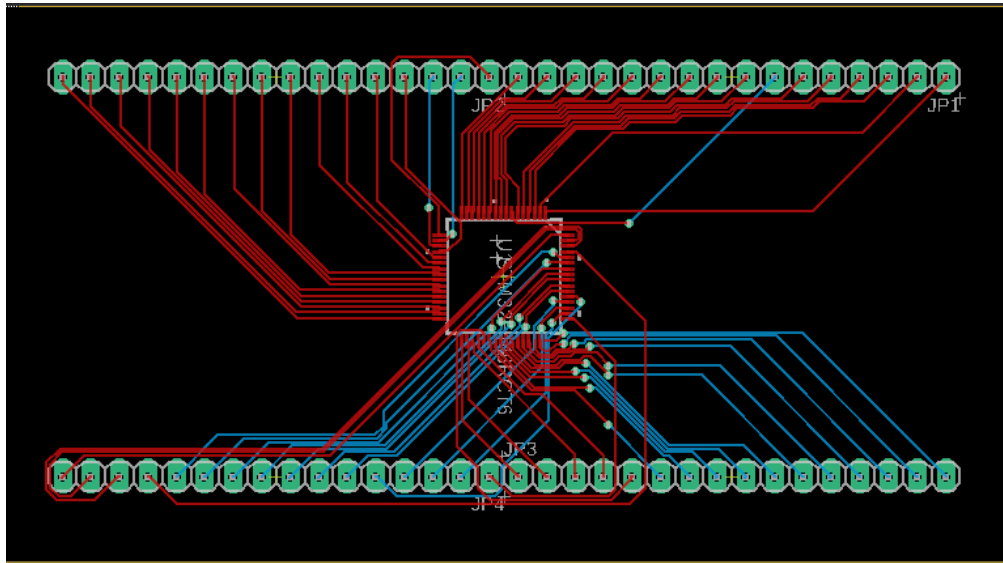


Figure 64: MCU Testing PCB

In addition to the development of a breadboard testbench for the MCU, we also include a prototyped evaluation board in which the main components (CODEC, external oscillator and flash memory) were already routed. The reason for developing an evaluation board where we already include all these components was due to the fact that, even though they can be tested individually, the importance of having them working together was crucial to the success and foundation for future testing.

The reasoning behind designing an evaluation board (PCB) instead of using a breadboard was mainly to test all the main components once incorporated onto a PCB, since the factors of random noise addition and routing can greatly affect the outcome of the tests performed. Therefore, we should elaborate a series of testing phases before finalizing the design of the first version of the prototype. The reason we would want to set up different testing stages was so that we could confidently move onto the next testing points that required us to have the main components working in conjunction.

Phase 1:

During the initial stages of phase 1, we test the stability of power provided to the main components that we were testing (MCU, CODEC, external oscillator, flash memory). Once we had confidence that each component was receiving a stable power supply, then we verified that the USB/JTAG modules were working, since both these were indispensable in the process of debugging and programming of the MCU.

The goal of phase 1 was to determine the following:

1) Power & Programming interface: As mentioned, we first needed to check that all the components were receiving a stable power and that the connections to the PC were also working. In these ways we were able to guarantee that there's no problem with the power or programming of the board.

2) MCU-External Oscillator: The first step in phase one was two make sure that the external oscillator was working correctly with the MCU, in order to test this, we were able to write a simple blinking LED program that will essentially turn on and off the LED in terms of the frequency the oscillator was providing the MCU. This was obviously a very high frequency for us to detect a change in state of the LED, therefore we had to divide the frequency provided by the correct multiple. Another way of testing that the connection with the external oscillator was working was by using the debugger interface in the STM32CubeMX IDE. We were also expecting to use this IDE to set up the correct pins for the target connections. Least to say, we were also testing different parameters and set ups provided by the STM32CubeMX IDE.

3) MCU-CODEC: Once we guaranteed that the MCU has a proper clock (provided by the external oscillator), then we started testing the connections between the MCU and the CODEC. Ultimately, what we were looking for in this testing was to see that the data was flowing between the CODEC and the MCU and that the packages are not being lost. A successful testing will show that the CODEC was able to process the incoming analog signal, sending it to the MCU, receiving the digital stream back from the MCU, and finally the analog output from the CODEC was the one expected. Furthermore, we would also be testing the correct parameters used in the set up such as the desired frequency (44.1 kHz) and the resolution (24-bits). The methods for testing this were done using a function generator that matches the average frequency of a guitar, and an oscilloscope for analyzing the input and output signals. Additionally, we also liked to extend the testing further by calculating the THD and SNR levels in the system.

4) MCU-Flash Memory: Another side of this testing phase was to guarantee that the connection to the external flash memory was taking place without any problems. We would like to elaborate a small programming function to record and store data into the external flash memory. We would also like to test the resolution quality for the recovered data from the external flash memory, our expectation were the resolution was at least that of 16-bits.

Phase 2:

Once we have confidence that the fundamental elements of the DSP block are working correctly, we can then proceed to testing peripherals.

1) Bypass switch: Although the bypass switch was not part of the DSP block it was still important to know that the MCU was receiving and sending the processed signal, thus we needed to check that these connections were working correctly.

2) LCD: The testing required for the LCD was done in order to make sure that the GPIO pins selected are working correctly and that the 7-bit scheme used for the LCD worked. This testing was done by starting the development of the user interface that was displayed on the LCD, or by simply displaying a test program. All the segments of the LCD were tested.

3) User interface: The scheme realized for the user to navigate the menu should also be tested, this can either be done by utilizing the LCD. We expected to have a barebones version of the menu working by the time we test the LCD and the menu selector.

Phase 3:

Finally, in the last phase of the breadboarding testing cycle, we got the final adjustments and necessary fixes to ensure that the prototype version wouldn't require any PCB fixes. Ultimately, we wanted the first iteration of our PCB design, our evaluation board, to expedite the process of debugging and testing before we started using the prototype to develop the firmware. We thought this method would grant us more time and less stress when it came to reconfiguring a PCB board or diagnosing an issue with a particular element.

5.2.4 CODEC Breadboard Test

When it comes to testing the CODEC, the surface-mount package type of the device was not ideal for testing on a breadboard. Also, to function properly, the chip needed input from a clock, as well as control lines all coming from an MCU. For these reasons, we were testing the functionality of the CODEC once it was mounted on the PCB board, similarly to the MCU.

We implemented a simple breadboard for testing, which we planned to utilize alongside the breadboarded prototypes of the MCU, as well as the power supplies and input/output buffers and the power supplies. In this way, we were able to test the functionality of all our components before ordering a PCB containing our finalized circuit design.

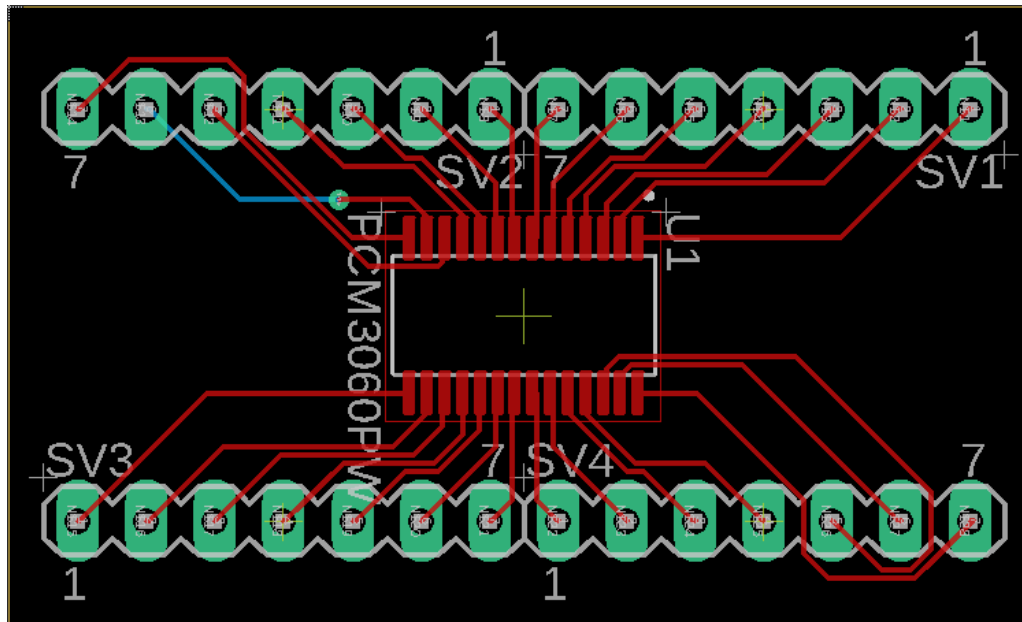


Figure 65: CODEC Testing PCB

We were testing mostly to find the right settings for the CODEC in which we get the best quality conversion from it. Some of these settings include the clock frequencies for both ADC and DAC and the sampling frequency. To determine the quality of the output signal, we measured the SNR and jitter with different chip settings, to find the combination that gave us the most favorable values.

In order to test the CODEC we needed a couple of inputs. We needed a power source of 3.3V and one of 5V, a ground line, a sinusoidal clock source, MODE control lines following I2C, I2S communication lines, an analog input source, and lastly a connection to send the resulting signal output to. Most of these mentioned lines were provided by the MCU, which made testing a lot more efficient once the components were mounted and connected on the PCB.

The test itself consisted of choosing a clock rate and data rate, as well as other values coming from the MCU to set up the CODEC with. We then sent an input analog signal to the chip to transform. The signal was converted into digital format, then once again converted to analog for the output. We then compared the input signal against the output signal. By examining the difference between the two, it was possible to come up with a Signal-to-Noise ratio value, which determined the overall quality of the CODEC's conversions.

The procedure of this test was then to convert an input signal many times by using slightly different settings each iteration, and then proceeded to calculate the output

signal's SNR value. After many trials, we were able to tell which settings improved or hurt the CODEC's performance, which was a vital piece of this guitar pedal.

5.2.5 Tone Control Breadboard Test

The next breadboard test we conducted was for the tone control configuration. In the image below, we configured both the "Bluesbreaker" schematic (left) and the "Big Muff" schematic (right). Both tone controls utilized a 100k Ω logarithmic potentiometer in order to adjust the value of the cutoff frequencies. We conducted multiple tests with both of these configurations.

The first test we conducted was using a multimeter to measure the resistance of the potentiometer as we turned the knob from its minimum value to its maximum value. As expected, at 0% knob rotation, the potentiometer had no resistance. At 50% knob rotation, the potentiometer had about 25% of its total resistance value, or 25k Ω . This odd 25:50 ratio was due to the behavior of logarithmic potentiometer's frequency taper, which we had discussed previously in Section 3.2.2 (Analog Input and Output Buffers). At 100% knob rotation, the potentiometer measured 100k Ω , or 100% of its resistance value. By using the aforementioned equation for cutoff frequency calculation, we were able to successfully recreate the frequency response from our previous Multisim Live test results in Section 5.1 (Initial Design Architectures and Related Diagrams).

The other test that was performed was a simple auditory test. We adjusted the value of the potentiometers and listened to the characteristics of the frequency sweep, and how the tone sounded when rolled all the way off when rolled all the way on. These characteristics proved to us that using a logarithmic potentiometer was the right choice, as the gradual change in tone felt very natural to the ear. Based on the results from this test, and from the Multisim Live circuit simulations, our team ultimately decided to implement the "Bluesbreaker" style of tone control, because of its simplicity and ease of integration.

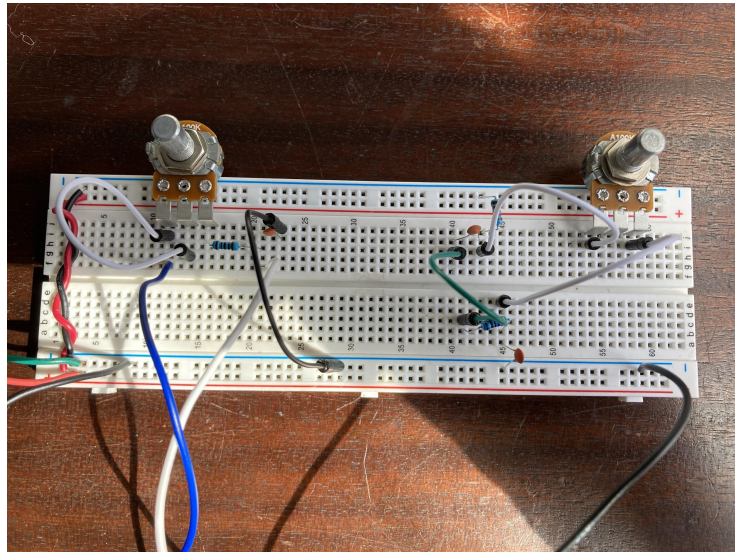


Figure 66: Tone Control Breadboard Test

5.2.6 Power Distribution Breadboard Test

In this test, we confirmed the functionality of our AMS1117 DC/DC buck converter (Figure 67) and a 78L05 linear voltage regulator (Figure 68) by using a digital multimeter and the power supply stated above. We also let the buck converter and regulator run for over a period of 4 hours to confirm that the converter did not get too hot. We did not need to use a thermocouple in this case since the converter was ambient to the touch. Since the functionality of this converter becomes unstable at 120 degrees celsius, there was no need to take any further more precise measurements.

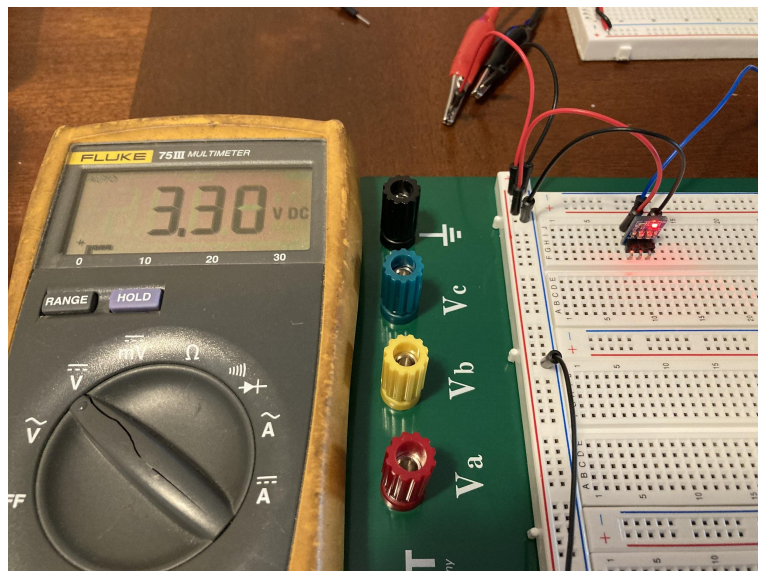


Figure 67: 3.3V Buck Converter Breadboard Test

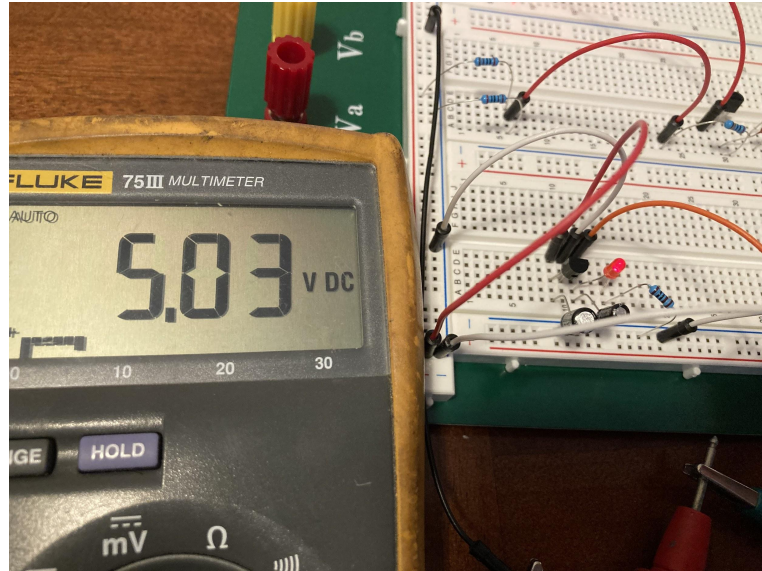


Figure 68: 5V Linear Regulator Breadboard Test

6.0 Project Prototype Construction and Coding

After all the necessary preliminary testing has been completed on the analog and digital hardware elements of our project, our team had to then be assembled as a working prototype during Senior Design 2 in the Fall semester. Despite all the challenging work that has been done thus far in researching components and understanding exactly what we want to achieve with our design, this assembly stage most certainly had its own set of unique challenges that we learned to overcome. The following sections of our paper describes our plan for the assembly of our physical prototype, as well as the coding elements required that allowed our pedal to deliver the high quality effects processing that we require in our design.

6.1 Integrated Schematics

Seen in the image below was our final design schematic. We have decided to utilize the Eagle software in order to assemble our overall circuit schematic and layout our PCB design. In order to do this, we had to download various Eagle libraries that gave us access to the specific parts that we used in our design, as well as their footprints. These specific part footprints were important in order to make the proper connections between the various parts of our circuit. They were also important to implement such that our PCB would have proper pin spacing for all the SMT components (MCU, CODEC, Buck Converter).

Whether or not we accommodated these specifications was up to how complicated our PCB design ended up being, as well as what companies offer the best deals and the widest variety of options. Some of the companies listed below also offered assembly services, which meant that we could potentially have all of our circuit components professionally soldered onto our PCB for an additional fee. We considered whether or not this option works for us, and will factor that into our decision as well.

6.2.1 Dimensions

Based on our design, we knew that our PCB would have to be at least 100mm by 75mm (roughly 4in by 3in). This size would fit perfectly inside of our anticipated enclosure, the Hammond 1590BB. However, this PCB size specification was subject to change, depending on how many components ended up in our final design schematic. In addition, if we ended up using a larger or smaller enclosure size, we may have had to consider altering the dimensions of our PCB.

6.2.2 Layers

The amount of layers in a PCB refers to how many layers of copper are present in the board. These copper layers are the foundation for the traces/connections that are laid out between the many components of a circuit. PCBs are available in configurations of anywhere from 2 layers to 14 layers (or more), depending on the PCB vendor used. PCBs with 2 layers are most common, and can accommodate a large variety of simple circuit layouts. The top layer of this configuration is often described as the “signal layer”, and is home to the main signal path of the circuit. The bottom layer is most often used as the “ground layer”, and is where all the components’ ground/reference node connections are routed to.

More layers are required when implementing a more complex PCB design with a lot of components, connections, and varying power distributions/voltage levels. However, some issues can arise as you increase the amount of layers in a PCB. These issues include impedance and propagation delays, and can lead to latency and noise in a circuit. Propagation delay occurs due to the intrinsic capacitance of a PCB’s layout, which can occur between the traces of a PCB as well as between layers. When increasing the amount of layers in a PCB, this capacitance can also increase and lead to higher amounts of propagation delay. We took this into account when deciding how many layers we wanted to implement into our PCB design.

Our MCU, the STM32F446RCT7, has a recommended configuration of at least a 4 layer PCB stackup. This 4 layer configuration was particularly useful for applications using several digital components, as the 4 layers could be utilized for signal path, grounding, as

well as various reference voltages. Between the MCU and the CODEC being used in our design, we will require a 3.3V voltage reference and a 5V voltage reference. If we were to dedicate these reference voltages to separate planes/layers of our PCB, we could greatly simplify our PCB design and reduce the overall impedance of the PCB's connections, which was ideal. This also reduced the number of vias we would have to implement. Vias are simply holes in the PCB that allow traces to pass underneath one another or attach to different layers within the PCB. Shown below is an image of a commonly recommended scheme for the PCB layers utilized alongside an STM32 MCU chip.

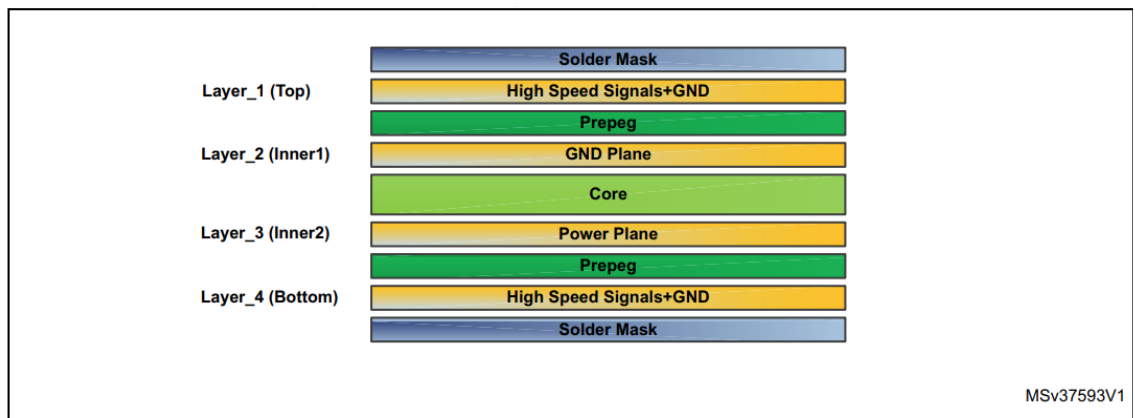


Figure 70: Recommended PCB Layering Scheme for STM32 MCU

6.2.2.1 Grounding Layer

Proper grounding methods were essential in designing a PCB layout. Without a stable ground, it was nearly impossible to pass clean signals from one device to another. It was essential to avoid ground current loops. Ground current loops occur when two points in a circuit that are intended to have the same ground reference have a potential difference. Hence, causing a ground loop and therefore noise passing through our system which can interfere with the signal quality. We also minimized the return path distance to reduce impedance along the path to the ground common.

We used one of our layers as a single ground plane to connect all of our grounds and we will utilize vias to shorten the return path to our grounds. Another consideration when implementing a good grounding design was taking into account that we will have both power ground and signal ground for our devices mounted on the PCB. Analog power grounding was typically a lot easier to manage than digital signal grounding because the transitioning of the state of the signal can cause current spikes and could lead to noise. The best way to have a proper return path for both power and digital signals was to use only one ground plane and partition the PCB into digital and analog routing sections. By

partitioning, that meant cutting through the ground plane between grounds as much as possible without splitting the grounds.

Analog signals were routed solely in the analog section of the ground plane of the board and digital signals were routed solely in the digital ground plane of the board. By sectioning off the different grounds properly, the digital ground signals generated by the currents will remain in the digital section of the board so they won't interfere with the analog ground. This explanation was difficult to conceptualize in physical form. We constructed a hand drawn example diagram of how we wanted to partition ground planes on a single layer within our PCB as an example of how we went to design our ground layer:

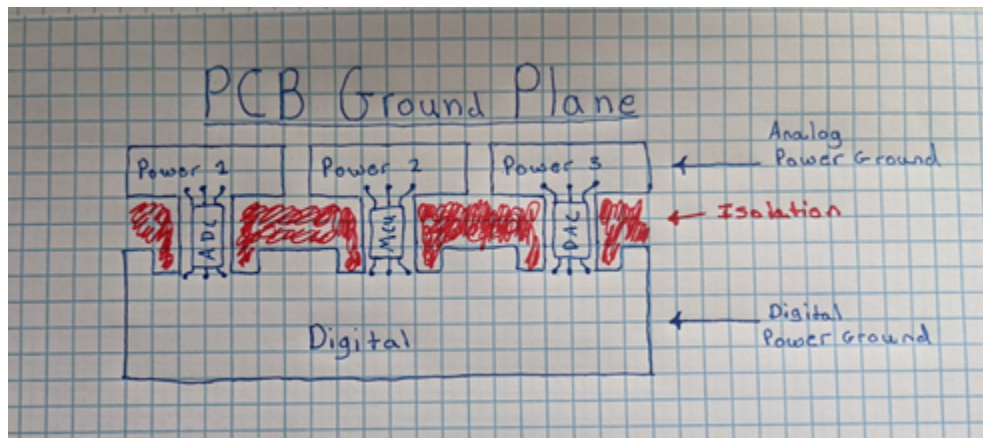


Figure 71: Ideal Ground Layer Design Concept

Having a long return path to ground could generate undesired noise. Since noise from either digital or analog sources was alternating and the wires within the PCB have their own inductance, the greater the length of the wire causes more inductance. Therefore, a low-inductance return path improves noise performance and reduces board radiation. We implemented this by minimizing trace lengths, via lengths, and the overall distance of the system to the return path. When tracing our PCB, we needed to be very considerate of this aspect as we aim to reduce noise as much as possible.

6.2.3 Thickness

The thickness of our PCB was directly dependent upon how many layers we chose to implement. The thickness of the various layers can also contribute to changes in impedance and capacitance of our board connections. This was due to the material that was sandwiched between the copper layers being dielectric. Dielectric materials are the same ones as those used in capacitors to create space between their conductive parallel plates. Therefore, the thickness of the dielectric layers, and of the board overall, directly

correlates to these undesirable impedance and capacitance issues, and was considered when choosing both the number of layers and the overall board thickness.

6.2.4 Track Width and Spacing

When it comes to PCB traces, the thicker the track width, the better. Obviously, space is also a large factor when deciding the thickness of the PCB traces, as well as the allowable space in between them. We hoped to implement a track width that was large enough to maintain an ideally low impedance and reduce capacitive propagation delays, but small enough to make the best use of space on the surface area of our PCB.

6.2.5 Material Type and Lead-Free Options

In accordance with our environmental constraints, we have chosen to be as environmentally conscious as possible when selecting our circuit components, enclosure, and PCB. Several PCB manufacturing companies offer the option to use lead-free materials in the fabrication of their PCBs, and this was a large deciding factor for our team when choosing which company we were ordering our PCB from. By ordering lead-free PCBs, and using lead-free solder, our ADEPT PCB prototype was virtually lead-free and RoHS compliant. In addition, any PCB prototypes that do not make the final product will ultimately be recycled.

6.2.6 Mask and Silkscreen

While mostly an aesthetic facet of our PCB design, we hope to choose a unique colorset for our PCB solder mask and silkscreen printing. Many PCB vendors have a wide variety of colors and materials available to choose from, and these were considered when searching for the right PCB vendor to buy our PCB from. On the more technical side, the solder mask can have some undesired effects on our PCB, depending on the trace width that we choose. The solder mask can decrease the impedance on thinner traces, while not having much of an effect on thicker traces. If there are options to decrease the thickness of our solder mask, we will do so as to not run into these technical issues later on during PCB assembly.

6.2.7 Vendor Selection

The following section will outline the pros and cons of 5 different PCB vendors that we have researched. We also made the final choice as to which one best suited the needs of our project's PCB design requirements.

The first PCB vendor we considered was ALLPCB. The cost per PCB was roughly \$15, which was a bit expensive, but this company offers a single free prototype PCB, which

would certainly cut down on the cost requirements for our project. They also offer PCB assembly services, up to 16 layer boards, and the option of using lead-free materials as well. Overall, the services offered by ALLPCB satisfy many of our PCB requirements, and they will definitely be considered for our final selection.

BasicPCB was the next company considered. We found this vendor to be the worst option that we considered, for a number of reasons. BasicPCB was exactly what it sounds like: it only offers very basic options, at a limit of only 2 layers. In addition to the lack of options, this vendor's prices were also extremely expensive, costing nearly \$100 for just 5 PCBs. This vendor also required the use of the Gerber file type, as well as the inclusion of a tool list, which could complicate our PCB prototyping process and require an extra learning curve.

JLCPCB was a very good option that we considered, for several reasons. Firstly, it was certainly the cheapest option out of all the ones we researched, with an unbelievable price of under \$5, excluding shipping. JLCPCB also offered up to 6 layers per board, as well as lead-free material options and assembly services, similar to those offered by ALLPCB. The only downside to this option was that it only accepts Gerber files for the upload and submission of the PCB design to the site. It became more apparent as we did more research that Gerber files are important in PCB design and fabrication processes.

The next PCB vendor considered was OSH Park. Very popular among hobbyists and independent engineers, OSH Park is an affordable and simple-to-use service that offered several simplified preset options for PCB fabrication. These preset options, while a bit limited, offer standard specifications for trace width and spacing, board thickness, etc. Both 2-layer and 4-layer options were available, and all their products used lead-free materials. The best aspect of this service was that only an EagleCAD or KiCAD file was required to submit a design to the OSH Park website. This greatly simplified our PCB design process, requiring minimal effort to process specialized files within the CAD software of our choice. Another unique aspect of OSH Park's PCBs was the solder mask color: a rich purple that was not commonly seen on many PCBs in production today. We felt that this unique color scheme would be an attractive quality for the ADEPT prototype.

The final option for PCB fabrication services that we considered was from the company PCBWay. This vendor offers affordable products, with a wide variety of options for various specification setups, all of which utilize lead-free materials. Urgent fabrication and fast shipping options are also available, which was a big selling point for our team. PCBWay, much like ALLPCB and JLCPCB, also offer assembly services. PCBWay also offers the most board layers out of any of the options we have discussed above, with a whopping amount of 14 individual copper layers.

After careful consideration of all of the options above, our team decided to go with OSH Park for our PCB fabrication. While only having simplified options available, OSH Park satisfies nearly all of our PCB design requirements, including offering affordable. 4-layer, lead-free PCBs with standard specifications that cater perfectly to our needs. In addition, the simplified ordering process using only an EagleCAD file will make our prototyping process much faster and easier, and their unique purple solder mask design offers an aesthetically pleasing visual aspect that we believe will stand out when compared to other PCB designs.

6.3 Final Coding Plan

6.3.1 High Level Code Structure

In the development process for a large codebase we wanted to first come up with a rough draft of the structure of our code, this was a basic outline of the tasks and functions that we implemented, this gave us an idea of the overarching components in the code.

Since we were working in embedded programming we first understood how the program worked. The device should be in a constant state of checking the incoming signal and then applying the effect processing to it. This involves a great deal of dynamic memory allocation, since we were reading and writing to and from the internal memory space and also the external memory. The device also had a start up protocol, which set and initialized any variable or pins that were required. Furthermore, the looper functionality of our pedal would likely shift from the current processing to the looper mode, so we would have to come up with a mode selection in order to change the current state of the program.

As in most systems, the main function was the driver function of the firmware. We had to run an infinite while loop in it to constantly check for any changes whether in the effects or the mode. Additionally we also had the firmware compartmentalized, this meant that we divided the mode and effects into different functions, as well as start up routines and check for events in the menu selection. The overall structure of the code was easily readable and also faster to debug. This was also a good rule of thumb in the process of developing since we can always make sure each module was working before it was integrated into the final version.

Carrying on from the last point, version control was a crucial part of debugging and developing our codebase, we utilized tools such as GitHub to carry different versions of our code and also shared the progress immediately after we were done pushing a new update. This facilitated the processes of sharing and debugging code as well as keeping a neat timeline with snapshots of our code.

Tentative Outline:

1. Initialize peripherals
2. Clear memory
3. Pulse check (Heartbeat LED)
4. Enable audio
5. Initialize mode to default
6. Run infinite loop

1. Initialize peripherals: An important part of starting the device was the set up function, in which all peripherals were initialized, meaning, setting the correct clock frequency, configuration, and parameters. These peripherals include but were not limited to: External clock, LCD, LEDs, USB/JTAG, encoder, buttons, internal clock config, SPI communications.

2. Clear memory: The memory stored previously was of no use and will only bring problems, therefore the device will clear all the memory

3. Pulse check: The device starts the blinking LED program that then lets the user know the device was properly working, if the LED does not blink then this will alert the user that the device's external clock is not working.

4. Enable input audio: The device will initialize the communications with the CODEC and will start capturing the incoming signal and also release the output processed signal.

5. Initialize mode to default: The device default mode should be set to a clean effect signal with no processing applied.

6. Infinite loop: Will check for any events, such as: changing effects (moving up or down in the menu), selecting effects, changing modes.

6.3.2 Effect Parameters

All of our audio effect implementations follow a similar control scheme: 3 adjustable parameters per effect (with the exception of the looper). We decided to implement this control scheme in order to simplify our digital design and coding requirements. This simplified control scheme was also very intuitive to the user of the device, removing the need for any complicated instruction guide on how to operate the ADEPT.

6.3.3 Digital Effects

We have decided that for this project to implement most of our guitar effects digitally, which consisted of doing all the necessary distortions and modifications to the source guitar signal inside of the DSP chip with code. This decision was made since doing everything digitally will save us physical space in our device, and will also give us a simplified implementation that will ultimately save time and cut down on material cost.

We implemented all of the effects in the STM32F446, along with peripherals such as an additional external memory, which were necessary for effects such as delay and looping. Fortunately, to help us do this, the STM32 includes a very useful software interface to set all the output pins, and all settings correctly. There were also many useful DSP and filtering libraries that have been created specifically for the STM32 that will come very handy.

Our team was also planning on creating an interface in which the user can interact with the microcontroller in order to select different effects, as well as setting specific values relevant to each effect. This was done by a series of buttons and knobs on the device, including a few that could be stepped on, since the user or guitar player is usually standing and has their hands busy playing the instrument.

In the following sections, we discuss the different effects (listed below) that we attempted to implement for our project. Each effect section includes an explanation of the effect, some background, as well as how we planned to implement it in code. We also discussed how all of these effects were controlled, by using the 3 parameter knobs that we implemented in our design.

1. Bitcrusher: decreases sample rate and bit depth
2. Chorus/Vibrato: time-based doubling effect that sinusoidally alters the pitch of the signal using delay lines
3. Compressor: alters the dynamics of the signal such that quiet sounds are increased and loud sounds are suppressed
4. Distortion: increases gain to the point of saturation
5. Delay: repeats signal impulse a certain number of times [feedback] at a specified rate [delay time]
6. Filter/Autowah: utilizes an envelope filter in tandem with a cutoff/resonance filter that will sweep through the filter frequencies upon the input of an impulse from the instrument
7. Flanger: similar to chorus, but with shorter delay lines
8. Looper: allows the user to record, play, overdub, stop/pause, erase musical loops
9. Phaser: sweeps through frequency spectrum at a specified rate and depth
10. Pitch Shifter/Harmonizer: alters the original pitch of the signal, with the ability to mix with dry signal for harmony
11. Reverb: similar to delay but more spatial/atmospheric, and with more delay lines

12. Tremolo: rapid increase and decrease in volume at a specified rate and depth

6.3.3.1 Bitcrusher

A bitcrusher is a type of distortion effect that is achieved by reducing the sample rate or resolution of an audio signal. This drastically reduces the fidelity of the audio signal, producing a “lofi” sound that can simulate the sound of an old radio, or a broken electronic device. By digitally reducing the bandwidth of the original guitar signal, we introduce quantization noise, which is what produces the unique scratchy and bell-like tones that a bitcrusher can yield.

As we have discussed in earlier sections, slowing down the sample rate or reducing the resolution of an audio sample results in quantization noise due to conversion irregularities. Slowing down the sample rate, or down-sampling, intentionally reduces the audio quality to produce the desired effect. Reducing resolution also has the same audio quality reducing effect, since we transition from a higher bit to a lower bit resolution, much of the detail in the signal gets lost, and waveforms become noisy and buzzy, producing the butcrushing effect.

To reproduce such an effect using code we needed to find a way to produce this quantization noise in our signal. We planned on achieving this by having two separate processing functions, one specialized in reducing sample rate, and one for reducing bit resolution. Each one of these functions is directly affected by the user selected parameters of Sample Rate, Bit resolution, and Mix/Blend. The first controls how much we want the rate to slow down. Resolution controls how many bits we want to reduce the resolution by. Mix/Blend is the same as discussed previously.

After modifying the input signal by going through either or both filters, the resulting signal was sent to the DAC to convert back to analog. The final soundwave had the characteristic “lofi” effect added to it.

6.3.3.2 Chorus / Vibrato

Chorus is a rich doubling effect that has been utilized as a standard effect in studios and on pedalboards for decades. This particular effect manipulates the input signal to make it sound as if there are many different versions of it being played simultaneously, coming from many different directions towards the listener. Such was the case when listening to a choir singing (hence the name chorus) or listening to many string instruments playing in unison in an orchestra. In these live performance cases, the slight variances in pitch among all the individual performers are not perceived as being out of tune, but rather add to the thickness of the overall tone of the performance. Such was the idea behind the implementation of the chorus effect.

As a studio effect, chorus is achieved by simply “overdubbing”, or recording multiple takes of the exact same instrument or vocal part. The subtle inconsistencies in these recorded performances, whether it be slight differences in timing or in tuning, can produce a very full-sounding effect that is perceived as a washy and shimmering effect.

Acoustically, this effect can be produced by physically having many instruments playing (or people singing) at similar pitches. Several acoustic instruments also have inherent chorus-like qualities built into them, most notably the 12-string guitar. 12-string guitars utilize 6 “courses”, or groupings of strings, in pairs of 2. These courses can be tuned such that the notes of the strings within one course are either one octave apart, or in unison. This produces a lush, brilliant sound. Other acoustic instruments that can create this lush effect include honky-tonk pianos, mandolins, accordions, and many more.

Vibrato is an effect that uniformly changes the pitch of an audio signal sinusoidally. This is an effect that can be acoustically achieved by a vocalist through variations of the vocal cords, or by a stringed instrument player by vibrating or moving the string on their instrument back and forth, uniformly varying the pitch. Rotary speakers (more commonly known as Leslie speakers) also create the vibrato effect by means of the Doppler effect. This physics concept is defined as a perceived change in the frequency of a wave based on the listener moving closer to or further away from the source of a sound. This is what is occurring when listening to a Leslie speaker that is rapidly rotating around, projecting sound waves around the room it is placed in.

The electronic alternative to produce this effect is done by using signal processing software, where the original sound is mixed with one or more pitch-modulated and slightly delayed copies of itself. Based on what we know about the chorus effect, we can conclude that the vibrato effect is essentially just a chorus without the dry signal mixed in. The way that this was implemented in our digital design is as follows: the dry instrument signal was able to be blended with the wet vibrato signal via the Mix/Blend parameter knob. Placing the knob anywhere in between it’s two extremes (dry signal or fully vibrato signal) will result in a lush chorus sound. Through this method, we are not only able to achieve the desired chorus effect, but also a vibrato effect if the dry signal is removed completely.

For this project, we produced this effect in an electronic fashion. Using the STM32 digital signal processing chip, we processed the input with a program written to simulate the chorus effect. This program took the input signal and created multiple copies of it. Each copy was modified slightly in pitch according to the parameters of Rate, Depth, and Mix/Blend to then finally get mixed with the original copy of the signal. The digital signal was converted back to analog by the CODEC to be filtered and eventually heard through the amplifier or speaker.

6.3.3.3 Compressor

While not the most conspicuous or obvious effect, compression is a dynamic audio effect that is incredibly important to many musicians and recording engineers. It has been applied to not only guitars, but also basses, keyboards, drums, vocals, and virtually every other recorded instrument for decades.

Compression is a transparent audio effect that reduces, or compresses, the overall dynamic range of an audio signal. This reduction in dynamics essentially means that loud sounds are attenuated and quiet sounds are amplified.

We will incorporate three parameters for this particular effect, which the user will have control of using the physical knobs. The first variable is Attack/Sustain, and it controls the soundwave similar to how treble does it, affecting the definition of each note. The next setting is Release, which controls the sustain period of each note played. Lastly, we have Mix/Blend, which as discussed before, is used to control how much of the effect modified signal we want to hear.

To translate this effect into code, the idea was to create a function that was able to level out the soundwaves volume. To achieve this, we set a desired amplitude or dynamic range value for the compressor to aim for first. We will also consider the three parameters discussed above in real-time, to modify the sound accordingly. The input signal was sent to the function, where if any peaks or troughs seem too far off from the target volume, they were altered to match the specific amplitude. The volume uniform signal was lastly sent to the DAC for conversion back to analog.

6.3.3.4 Delay

This effect consists of recording a segment of sound played in the guitar, to then play it back with a delay that can be adjusted by the user. This creates an echo illusion, similar to the reverberation, but if we were to compare them, we discussed how reverb resembles sounds played in a big room, while delay is more similar to when we yell down a deep well, and we hear a reproduction of our voice come back after some little time.

Usually, guitar pedals dedicated to this kind of effect are very versatile, depending on what settings you select, you would be able to generate anything from long extended playback times that create cascading walls of sound (For atmosphere music), to quick playback times which produce an instant, snappy, “slapback” effect. Many iconic guitarists and bands have experimented with delay through the years, finding unique ways to create new sounds and give different feelings to their music. A few of these

pioneers are Eddie Van Halen, Brian May from Queen, David Gilmour from Pink Floyd, and The Edge from U2.

When it comes to the settings included in modern delay pedals, we found three adjustable parameters. The effect level, feedback, and delay time. The first controls how much of the processed delay signal was output from the pedal. Highly processed signals are usually referred to as “wet”, while unprocessed signals are considered “dry”. The feedback setting allows the user to control how many times the delayed sound was reproduced - turning it up to maximum would allow cascades of delays to come out of the pedal indefinitely, resulting in self-oscillation. Lastly, the delay time setting simply controls how long the pedal waits before replaying the sound recorded by the player.

To implement this functionality in code, the original sound was recorded, to then be outputted with a specific delay determined by the player. Hence, we had a function that takes audio snippets and assigns delays accordingly. It was important to point out that it all needed to be done in real-time, thus, after a delay was assigned, each delayed track was sent to the DAC at the corresponding time. All the settings for the delay function and other values were set depending on the user's input of the three discussed parameters.

6.3.3.5 Distortion

In a power amplifier, when the volume or signal gain is turned up so high that the signal amplitude becomes greater than what the amplifier was built to reproduce, the top of the signal waveform cannot be reached and gets chopped off, resulting in a hard clipped waveform. In sound, this translates into a distortion effect that can be heard by the listener.

The distortion effect gives the audio signal a “dirty” and compressed sound. Distortion pedals usually achieve this by hard clipping the signal, to produce a great amount of gain, usually a tonal characteristic of high gain amplifiers. In the 1970s, guitar players had access to tube amplifiers, a kind of amplifier that when pushed to its limit, could produce a specific sound that we now recognize as distortion. This allowed musicians to experiment with new sounds in their music, which eventually led to a revolution in rock music.

In order to reproduce this effect, the original signal must be hard clipped at the top and bottom of the wave, as shown in Figure 72 below. To achieve this digitally we implemented a program where the idea was basically to set up some threshold values for both the positive and negative spectrum, where if a signal value surpasses any of these, that specific value point would be replaced by the value of the threshold. In this way, we could simulate the hard clipping that occurs when a guitar player turns up their amp past their working volume limit. The rules by which the signal was modified is determined by

the parameters set by the user through the physical knobs. These parameters are Drive/Gain, Tone, and Mix/Blend. Gain refers to how much volume over the limit we want to simulate, which results in overdrive or distortion. Tone controls the treble, or the definition and clarity of the overdriven signal. Mix/Blend controls how much of the distorted signal we want to hear at the output, relative to the original signal.

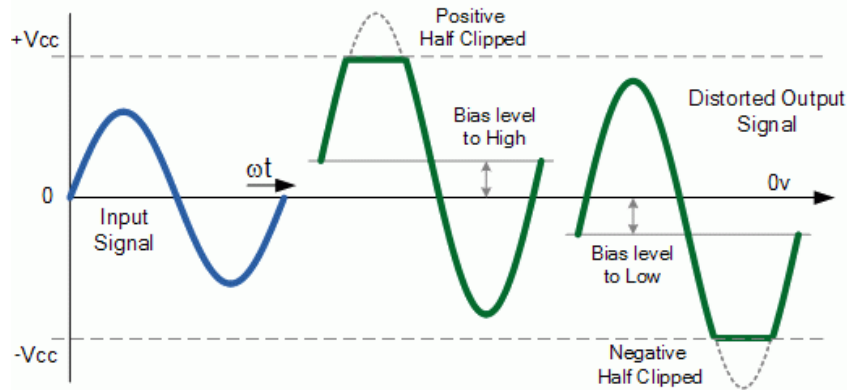


Figure 72: Visualization of Distortion Effect on Sinusoidal Signal (electronics-tutorials.ws)

6.3.3.6 Envelope Filter/Autowah

Often heard in disco and funk music from the 1970's, envelope filters are a dynamic guitar effect that have a distinctive “quacking” sound. This unique sound was created by manipulating the cutoff frequency of a filter using an envelope detector circuit.

The envelope detector circuit was used to represent the volume of the input impulse signal (for example, the pluck of a guitar string). This volume level is represented as a voltage value which, in turn, controls the sweep of a filter's cutoff frequency. The filter in question can either have a low pass, high pass, or band pass response, and generally a high Q point. This resulted in a very resonant filter that produced a dramatic effect.

For this particular effect the control options for the user were cutoff, Resonance, and Sensitivity. Cutoff and Resonance were characteristics of the filter, while sensitivity controls the amplitude of the signal, so it works as a gain control. The parameters from the user were used to set up new filter objects in code. We essentially simulated an envelope detector circuit using code.

6.3.3.7 Flanger

One of the earliest studio effects to be created was flanging. One early method of achieving this sweeping effect was by utilizing two separate reel-to-reel tape machines.

By simultaneously playing back the song recordings on the two separate reel-to-reel tape machines, and subsequently applying gentle pressure to the edge of one of the tape reels, or “flanges”, the playback of one of the recordings is very slightly slowed down, resulting in what we know as flanging. This slowing down and speeding up of one playback reel relative to the other essentially creates a variable notch filter, or comb filter.

Flanging can also be reproduced digitally, upon understanding the fundamental ideas behind the effect. Very similar to the chorus effect, flangers are created by simply mixing two identical signals together, one of them being delayed by a very small and gradually changing period. This period is usually very minimal, commonly less than 20 milliseconds. The resulting signal has “peaks” and “notches” similar to the ones created for the phasing effect. Varying the time delay causes these to sweep up and down the frequency spectrum. Additionally, part of the output signal can be fed back into the input to produce a resonance that intensifies the effect. The sound produced is that of a “comb filter effect” similar to the swooshing heard from a phaser, but more subtle. In the image below, we can see the difference between phasing and flanging on the sound wave.

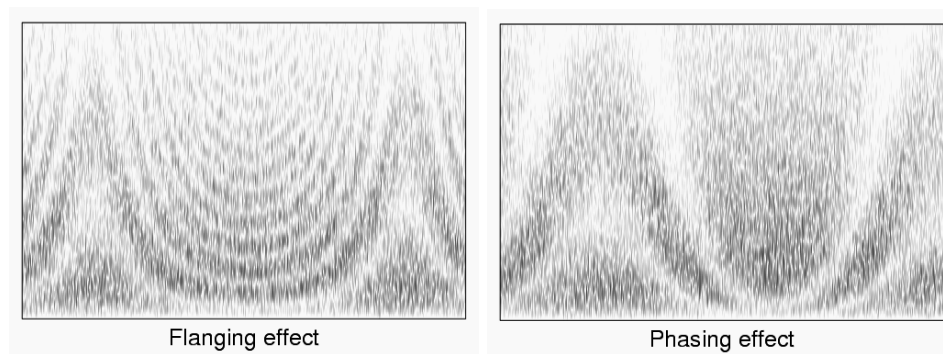


Figure 73: Visualization of Flanging vs. Phasing (wikiwand.com)

To reproduce the flanger effect in code, we must first create once again a filter object, that now this time around was responsible for speeding up or slowing down a given sound sample. This modification was done according to the parameters set by the user through the physical knobs. We will have the same parameters as for the flanger effect, with pretty much the same influence in sound. Once the filter objects are ready, we can send copies of the input signal to them to change the speed accordingly, to then mix them with the original signal before going to the DAC.

6.3.3.8 Looper

Looper pedals allow solo musicians to record a small sample melody, which can then be played on repeat, or in a loop. With this loop playing in the background, the musician can play on top of this recorded “track”, with the ability to overdub himself and stack these

recorded loops on top of one another. This can be utilized as a tool for songwriting, when there are no other musicians around to play along with. This was a very valuable effect to have for any guitar player, which was why we wanted to implement it in our design.

The first looper pedal was developed by Les Paul in 1953, and today, it stands as one of the favorite one-person-band tools of choice. It has been well established in music history, starting with “A Rainbow in Curved Air” as one of the first songs to ever employ this effect pedal, to then used by Radiohead’s Ed O’Brien, Mastodon’s Bill Kelliher, and many more. Today it is still very popular in many genres of music, used by artists like Ed Sheeran, The Black Keys, and Tash Sultana.

This effect can be produced by simply recording a copy of the incoming sound for the desired period of time determined by the player, to then be played back over and over again. To program this effect, we will use similar techniques as to what we have been using so far. We will first prompt the program with a button press to start recording the sample to loop, it will listen, and once the button is pressed again it will store a copy of the track in the external memory included in our STM32 microcontroller. The program will then be able to retrieve the track to play it over for as many iterations as the player desires.

6.3.3.9 Phaser

This particular effect is created by taking the signal, splitting it and passing it through several all-pass filters. These filters do not modify the signal, but instead only shift the phase of each of the split signals. All the different signals are summed up and then combined with the dry signal. Once all the signals are put together, this creates some cancellations in some parts of the output signal, but it also creates “peaks” and “notches” which result in very interesting sounds.

There are many settings that can be played with in a common phaser, which we tried to implement for the user to be able to control. For example, one very important setting is that of “stages”, which controls how many filters we are going to pass the signal through. The resulting sound can be very different depending on the settings used, but generally speaking, phasing creates a swooshing, synthesized sound.

In order to implement the phaser effect with code, we first created different filter objects, which are able to take an input and change its phase. The settings for the filter will change dynamically depending on the user’s input given through the knobs. After the filters are set up, the input is taken and many copies of it are made. Copies were sent to the different filters, to then be collected and summed all together to the original sound. We implemented the parameters of Rate/Speed, Depth, and Mix/Blend. Rate refers to how many times per time unit the phasing occurs. Depth has to do with how wide of a

frequency sweep the player wants. Mix/Blend as discussed before, is the mix between the original signal and modified signal to be outputted.

6.3.3.10 Pitch Shifter / Harmonizer

The pitch shifter effect, made popular from guitar pedals like the Digitech Whammy, causes each note that is played to be lowered or increased in pitch by a pre-designated musical interval before going to the amplifier. By combining this modified pitch-shifted signal with the original dry signal, a harmonizer effect was created.

We can see examples of pitch-shifting on TV and movies when a character's voice is modified with high-pitching in cartoons. If a character were to be some kind of monster, filmmakers can give it a deep, low voice using pitch-shifting. As it comes to music, one very popular example of this effect in use can be heard in the bass intro song "Seven Nation Army" by The White Stripes, where Jack White plays an electric guitar through a pitch shifting effects pedal set to an octave below.

There have been different ways in which this effect has been reproduced in past music, as well as for cartoons and movies. One method to modify pitch can be done by replaying a sound to a different speed than it was recorded. It could also be accomplished on reel-to-reel tape recorders by changing the motor on them. With vinyl records, simply placing a finger on the record as it plays will make the track slow down, decreasing the pitch, while spinning the record faster can speed it up, increasing the pitch.

In order to reproduce this effect in software, we had to implement a filter object which we can use to change the pitch of a given input. The idea is to pass copies of the input signal to multiple filter objects that will change the pitch by the preset interval. If the harmonizer is enabled, the modified copies will then get mixed with the original signal before getting sent to the CODEC's DAC. Each filter was generated corresponding to the pitch parameter set by the user, and the final signal output depended on the Mix/Blend value as well.

6.3.3.11 Reverb

This effect is used by musicians to create the illusion that the recording was done in a large room. This occurs because of the natural behavior of sound waves to bounce off when they hit a surface which does not absorb any of the sound. The sound waves in a big room may bounce off walls in different directions before hitting the receiver or target. There might be a direct sound coming straight from the source, but all those other sound waves bouncing around the room take some time to get to it, therefore creating some kind of echo effect. Below a graphic representation of how this occurs.

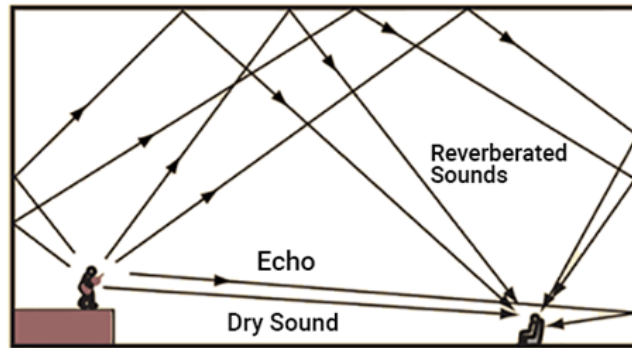


Figure 74: Diagram of Direct and Reflected Sound Waves (softdb.com)

Back in the 60s, musicians used to obtain this reverb effect by simply playing across a large room on opposite sides of the recording device; this method was called chamber reverberator. In this manner, the sound waves would be able to spread and bounce across the room, and some of them would be a bit delayed. The delayed sound waves along with the “dry sound” from the source created the desired effect.

Another method commonly used around the same time-frame, consisted of passing the sound waves across a metal plate, and setting the receiver or recorder on the other side. A similar method was even implemented into a commercial amplifier called the “Fender Twin Reverb Amp”, where a spring was used instead of a metal plate as a medium for the sound, and the tightness on the spring would determine the time and level of this resulting sound.

In today’s age, the most common method used to create reverberation is through digital processing. This method consists of using a system very similar to what we were trying to implement, which is to pass the sound signal through an ADC, then modify the signal using several delay lines whose input is taken at different sample rates to mimic the different sound waves coming from many directions. All the different signals will get summed up to create the final reverb signal, which was passed to a DAC to transform back into sound. In the image below, we can see how the source signal is represented, followed by the reverb body, which is all those other delayed signals that are added up to the original to create the desired effect.

As with most other effects, the user has a set of parameters available that it can mess with. These are decay length, Tone, and Mix/Blend. Decay length refers to how long the delay tail will extend (as in Figure 75 below). Tone, as we know, controls the treble, more or less how much clarity and definition the sound will have. Mix/Blend decides how “wet” or “dry” the sound will be, which means how much of the modified sound is going to actually play through the amplifier.

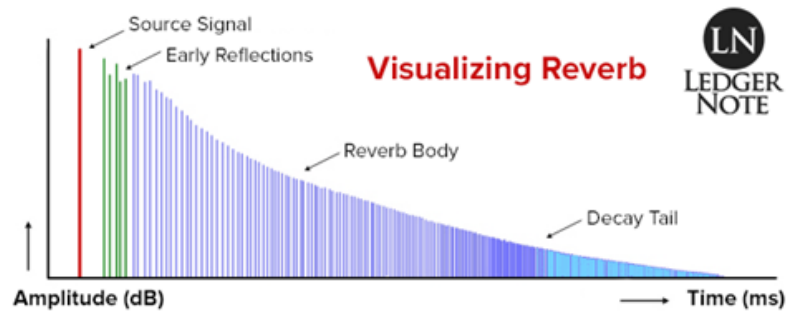


Figure 75: Visualization of Reverb Signal Structure (Ledgernote.com)

To implement the Reverb effect, we used a very similar method as to what we are doing for Delay. The original sound wave was recorded and stored, and will then be passed to some delay function, which will assign a different delay to each copy accordingly. The big difference in implementation against delay, is the timing and amount of delay lines that were used. For reverb, there are many more delay lines. Once again, there were setting controllers in the code that will respond to input from the user when turning the knobs, to modify the appropriate values and produce the desired effect.

6.3.3.12 Tremolo

As we discussed earlier in the paper, one of the very first stand-alone guitar effects was a tremolo effect, the DeArmond Trem-Trol. Often confused with vibrato, tremolo is an audio effect that is characterized by a uniform, repeating variation in the overall level or amplitude of the instrument signal.

In the early days of guitar effects technology, the tremolo effect was produced in various unique and unusual ways. The DeArmond Trem-Trol created the tremolo effect by sending the guitar signal into a canister filled with electrolytic fluid, and using a motor to oscillate and shake the canister, which altered the amplitude of the signal. Guitar amplifiers manufactured by well-known music companies Fender and Gibson implemented the tremolo effect by the means of something known as “bias wiggling”. Bias wiggling is the analog process in which the bias of a guitar amplifier’s preamp vacuum tube is modulated in a sinusoidal manner, creating the subsequent modulation in signal amplitude that we hear as tremolo. This sinusoidal modulation was later achieved via the use of optical circuits, which used light-dependent resistors (LDRs) to create the pulsating sinusoidal modulation required for the preamp circuit of the amplifier to create the tremolo effect.

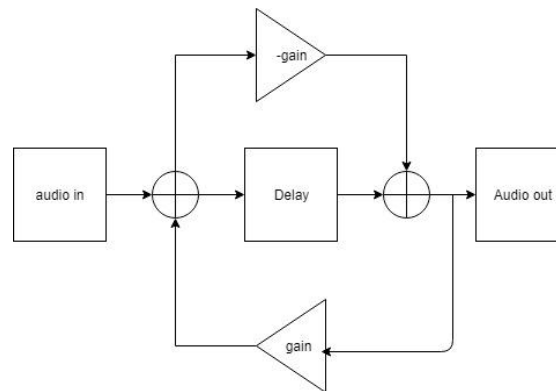
Nowadays, thanks to digital technology, we were able to try to implement this vintage guitar effect much easier. Thus we implemented the tremolo effect in code. To do this, we need to recognize that for this effect we must output almost the same signal at different

volume levels. Hence, we will create a function that will take the input sound wave, and reduce its amplitude, which acoustically will reduce the volume. Now, depending on three familiar parameters explained previously of Rate, Depth, and Mix/Blend the function will output accordingly. The idea is to vary the volume level by time, so all of the input signal went through the filter, but different parts of it alternate in volume.

The final amount of effects that we were able to implement amounted to a total of 6. Next, I will continue to explain how the effects were implemented and how the algorithms used to realize them.

Allpass filter/”Pipe”

The structure of the filter is as follows:



The allpass filter shown above is analogous to the all-pass filter from analog filters. As the name implies it allows pass of all frequencies, in other words the amplitude response of an all-pass filter equals to 1 for all the frequencies, whilst the phase response (determines delay vs. frequency) can be arbitrary. For our implementation we have changed the gain parameter to 0.7, this results in a sound like if you played a signal through a pipe, hence the name “pipe”.

As you can see, the allpass filter structure requires a feedback loop making it an IIR filter, it also has a feedforward loop.

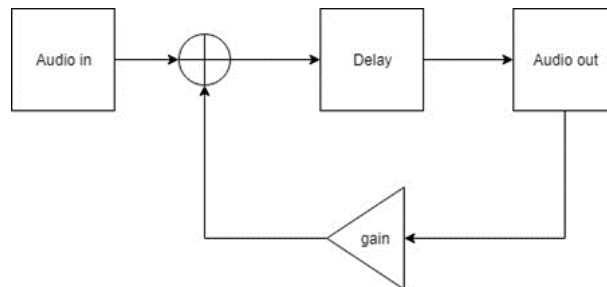

```

float Do_Allpass0(float inSample)
{
    float readback = apbuf0[ap0_p];
    readback += (-ap0_g) * inSample;
    float new = readback*ap0_g + inSample;
    apbuf0[ap0_p] = new;
    ap0_p++;
    if (ap0_p == ap0_lim) ap0_p=0;
    return readback;
}

```

The following code shows the implementation of the Allpass filter. The variable `-ap0_g` is the gain (feedforward loop) multiplied by the input audio sample. Similarly, we multiply the `ap0_g` (feedback loop) by the input sample audio, and then store the result into the allpass buffer. We also had to make a condition to reset the buffer since our memory usage is limited.

Comb filter (a.k.a. Delay)



The implementation we utilized for our delay simulates that of a feedback comb filter. The comb filter is an IIR (“recursive”) digital filter since there’s *feedback* from the delayed output to the input. The result is a sequence of “echoes” exponentially decaying.

$$y[n] = b_0x[n] - a_My[n - M].$$

Difference equation

For stability we want $|a_M| \leq 1$, otherwise the result would be an infinitely increasing series of echoes, where each one will be louder than the previous one.

```

float Do_Comb0(float inSample)
{
    float readback = cdbuf0[cf0_p];
    float new = readback*cf0_g + inSample;
    cdbuf0[cf0_p] = new;
    cf0_p++;
    if (cf0_p==cf0_lim) cf0_p = 0;
    return readback;
}

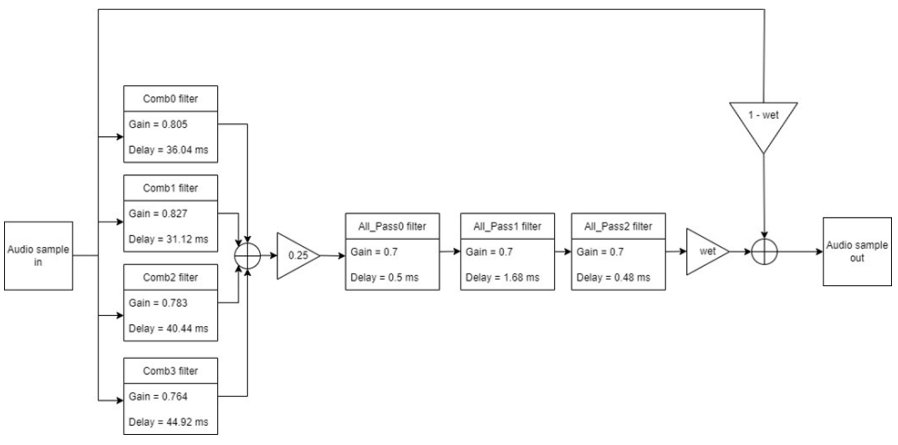
```

The implementation of the comb filter is very similar to the allpass filter previously explained. As you can see, we also made a condition for checking the limit allowed for the number of delays we want.

Reverb

The implementation used for the reverb filter follows Schroeder reverberation algorithm. Which is a popular reverb algorithm published by Schroeder for the Audio Engineering Society.

The way it is implemented is using 4 comb filters in parallel with 3 all pass filters. There's also a feedforward that allows us to use the parameter of dry/wet, allowing the user to determine how much of the clean signal will be added to the output.



As seen in the diagram, the reverb filter is a combination of the previously shown comb filter and the allpass filter. This configuration is also known as the Schroeder Allpass section. There's many different ways of implementing Schroeder's reverberation algorithm, the implementation we used has 4 comb filters in parallel that are then

normalized by a factor of .25, which then brings the overall energy obtained of the comb filters into a series of 3 allpass filters. There's also a feedforward that allows us to use the parameter of dry/wet, allowing the user to determine how much of the clean signal will be added to the output.

```
float Do_Reverb(float inSample)
{
    float newsample = (Do_Comb0(inSample) + Do_Comb1(inSample) + Do_Comb2(inSample) + Do_Comb3(inSample))/4.0f;
    newsample = Do_Allpass0(newsample);
    newsample = Do_Allpass1(newsample);
    newsample = Do_Allpass2(newsample);
    return newsample;
}
```

Distortion

In order to reproduce the effect of distortion, the original signal must be hard clipped at the top and bottom of the wave, as shown in the figure above. To achieve this digitally we implemented a program where we are able to amplify the incoming signal, as well as specify the thresholds of +Vcc and -Vcc such that the amplified signal will be distorted/clipped at the top and bottom of the signal's waveform.

The values by which the signal was modified are determined by the parameters set by the user through the parameter potentiometers. These parameters are Amplitude, Threshold, and Mix/Blend.

Compressor

The goal of a compressor is to gradually reduce the overall gain of an input signal if it is passed a certain threshold. The compressor has three main parameters: Attack, release, and hold. If the input signal jumps to a higher level above the threshold, then the attack phase is initiated; attack means that the audio input signal level is reduced step-by-step over time until the final gain reduction is complete. The attack parameter is set to a specific time in milliseconds. Once the signal level is below the threshold then the compressor recognizes this and waits for the hold time to set the signal back to the 0 dB gain (gain of 1). Over the time of the defined release time the audio signal is not manipulated anymore.

Pitch-Shifter

From a musical point of view, pitch shifting corresponds to the shift of a melody one or more semitones up or down. Now, from a signals point of view, pitch shifting consists of scaling the fundamental frequency and its harmonics by a specific factor.

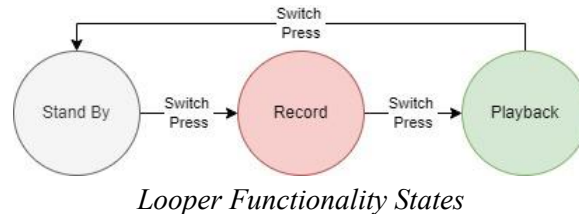
Additionally, we also were able to implement a looper functionality that I have included under this section.

Looper

The way this is implemented in the code is by utilizing a circular buffer that introduces every new

sample and sums a value (the Shift desired) to it, thus returning a scaled version of the sample.

To implement this feature, we used the external flash memory to store guitar input audio and our SPST foot-switch to prompt the program to cycle through three different states as shown in the diagram below.



During the recording state audio data gets stored in the flash memory, the next state retrieves the audio from memory to play it in a loop, and a final switch press places the pedal on stand-by state ready for the next recording. Although we were able to store and reproduce audio, it was a challenge to play it back at its original tempo due to the difficulty of synchronizing the read speed of the flash with the I2S transmit speeds. We decided to keep this feature in our design since the sped-up playback results in a very interesting effect.

7.0 Project Prototype Testing Plan

In any sort of development of a product, there is a sequence in which to efficiently design a product in a manner that will increase the overall quality, performance, and reliability. To simply make a design on a pcb layout and implement it in real life would lead to problems in future development, specifically, testing the product. Here are the steps in which to approach developing at testing a new design.

Step 1: Concepts and calculations:

First we implemented the theory around the design. Such as picking components, deciphering flow of functionality, and applying theory and concepts. Main considerations consist of the environment in which the system would operate, the frequency of use in which it will operate, and the ability to replace or change certain components in the future.

Step 2: Simulation:

After choosing component values and applying theoretical concepts to the design, we carry on that design to some sort of simulation software to analyze the behavior across all components. In this project we used a combination of Ltspice and Multisim Live. We essentially took the ideal concepts scenario and saw if our design would align with the theory applied to generating the flow of the system.

Step 3: Breadboard testing:

When the simulation of our design complied to our operation standards, we implemented the system in real life by making a rough design that was not compact but easy to measure and replace components. We also analysed the behavior of our design in a non ideal scenario. This was essential to make a properly functioning system before investing time and money into getting a PCB manufactured.

Step 4: *Software Testing:*

This was where our input and output analysis took place along with the structure of the logic on our software program. For example, analysing memory space and allocation, along with running simpler programs that emulate the desired operation of our product.

Step 5: *Order PCB and Test:*

For breadboard and software testing we made a piecewise approach to verifying the functionality of the design. Once a PCB was ordered and implemented, we started to combine all elements and observations found in previous testing. In this step, we tested the system as a whole to see if our concept actually works.

Step 6: *Create final product:*

Finally, once our testing was complete and everything functioned properly. We implemented housing and other miscellaneous characteristics to our final product.

Types of testing implemented:

Design Verification Testing:

The design verification process has two basic goals. First, was to ensure that the requirements at the lowest level of the design architecture, including derived requirements, are traceable to the functional architecture. Second, design verification must ensure that the design satisfies the validated requirements baseline. This was accomplished by developing a verification matrix that ties each element of the functional architecture and each element of the requirements baseline to a specific test or series of tests.

I/O Functionality Test:

Equipment needed: Function generator, Digital Multimeter, Oscilloscope, and Power supply.

Execution: This test was to verify functionality of the STM32F446 MCU. In this section we verified that the input and output pins of the MCU are functioning as desired. For inputs, we emulated a signal using the function generator and sent them into specified input pins and observed the behavior. For the outputs, we measured the DC outputs with

a digital multimeter and AC outputs with an oscilloscope. The reason why we were using a digital multimeter to measure DC values was for the fact that the multimeter can measure DC values with more accuracy. Thus, increasing the precision of our data collection during testing. The same concept applied to the reason that we are measuring AC signals with the oscilloscope. By doing this, we were able to isolate any issues with the MCU by eliminating extra variables added in testing.

Power Efficiency Test:

Equipment needed: Function generator, Digital Multimeter, Oscilloscope, and Power supply.

Execution: This test was to verify that we met our overall power efficiency specified in our design. We measured the input and output current along with the input and output voltage of our system. We will then take our measurements and calculate the ratio of input and output power and compare that ratio with our specified power requirement. We will use DC power analysis for DC variables and AC power analysis for AC signals using an oscilloscope.

Signal to Noise Ratio Test:

Execution: This test was to verify that our system complies with our specified maximum signal to noise ratio (SNR) value of -14dB. First we will measure all amplitudes of our AC signals passing through our device, then divide the sum of the amplitudes present in the spectrum in which our system operates. We will then measure the values occupied by noise. We will use the following formula to input our measurement to attain the SNR: $SNR = (average\ signal\ power) / (average\ noise\ power)$ and $SNR_{dB} = 10 \log_{10}(SNR)$

Interface and Operability Test:

Equipment needed: Development Environment, MCU

Execution: This test was to verify that our visual interface, in this case our LCD screen, was working properly. We used an IDE to implement test patterns on the LCD to verify that our communication and design was satisfactory. We will also toggled through our menu to observe the behavior of our system. This was so we could isolate the nature of our user interface and isolate abnormalities found in either our screen or our MCU.

Reliability Qualification Testing:

Our design was tested under certain conditions to determine if it complies with what we specify as our own requirements for the system. If our design was in compliance with these specifications then we could present our product to the “customer”.

Temperature Operability Test:

Equipment needed: Thermal chamber, Oscilloscope

Execution: This was a simple test to verify that our system will operate at a higher temperature. We were going to emulate an environment similar to a hot summer day outside. We wanted our product to work, for example, in a concert outside. To emulate this environment we considered using a thermal chamber to see how the system operates at 32 degrees celsius. First we hooked up our power source outside of the chamber and led the power line into the through hole to our system inside the chamber. We then ramped the chamber to 32C and let it soak for 1 hour. We would then measure the AC signals going across our system and make sure that it was operating normally.

Maximum Load Current Test:

Equipment needed: Digital multimeter, Load Box

Execution: In this test we verified that our system can handle various loads while maintaining operability. We measured the input voltage to the system and analysed the load value in which the voltage drops out. This way, if any stage pulled an unexpected amount of current, we were able to see how our system behaved.

7.1 Hardware Test Environment

For our hardware test environment, we needed to set up a controlled environment in which we can isolate problems to specific stages of our system. Once breadboarding testing was completed and we were sure that each component was working as expected, we then moved onto developing an evaluation board (PCB) where we began testing entire modules of our device, ranging from the input section to the output section. The goal was to be able to track down any bugs in specific modules in order to reduce the backtracking time required for debugging. These details are specified in Section 7.0.

7.2 Hardware Specific Testing

Once our prototype was complete, we needed to assure that it worked according to our requirements specifications. In order to do this, we will create hardware and software

testing environments that allow us to accurately measure the functionality of our prototype.

Before we could even be sure that our prototype would turn on and deliver high quality effects processing to our instrument signal, we ensured that a few basic functions were present in the prototype. The first basic function was that the pedal should be able to turn on when the power adapter was plugged into the DC jack. This functionality can be confirmed by observing the status LEDs on the pedal, as well as the LCD screen - if all these light up, we can confirm that the ADEPT is properly receiving power.

Another basic hardware function that we will have to test is the bypass switch. The pedal should be able to, at the very least, pass a clean signal from the input to the output when the footswitch was not engaged. This proved that the 3PDT switch used in our design was connected properly, and that engaging the footswitch should switch the signal path from a true bypass clean signal to the signal that was being sent through the circuit.

Once the ADEPT was up and running, we also performed a sound quality test. This was done by plugging in an instrument and listening to the various effects and their operations. This included adjusting all of the effect parameters and trying to get a wide variety of sounds out of the pedal, ranging anywhere from subtle to dramatic. Since we aim to deliver a product with a high fidelity audio quality, this was evident upon listening to the pedal through an amplifier.

We also wanted to assure that the digital hardware within the ADEPT was functioning properly. This included the MCU and CODEC, the heart of our pedal. Testing the sound quality will already prove to us that these digital chips are working, but we wanted to go the extra mile to ensure that they are functioning at their peak performance level. As previously mentioned, the evaluation board should be divided in modules in order to facilitate debugging. Each module will have specific requirements that will assure us the module was working as expected. Depending on the module the overall noise induced in the circuit was a crucial parameter we should keep an eye out for. Although we were not overly concerned with the total power consumption of our device, we did want to keep a check on the power delivered to every module, making sure that the stability was persistent despite the amount of load at any given time.

MCU:

In terms of hardware testing for the MCU we expected to have the correct input voltage and current required to power the MCU. These supply connections would also have the suggested coupling networks mentioned on the STM32F446RC datasheet, these connections have also been mentioned and explained in previous sections in these documents concerning the MCU and its peripherals.

7.3 Software Test Environment

7.3.1 STM32CubeMX

STM32CubeMX is a graphical interface for the configuration of STM32 microcontrollers and microprocessors. It operates under C coding with multiple types of processors. This environment allows you to choose from a multitude of different devices. We chose the specific MCU that we used in our design to test our equipment properly. Next we specified the peripherals of communication to establish forms of communication. Each peripheral corresponded to a pin. So we specified the type of peripheral and number of each type of peripheral to establish the pinout properly. In this case we used I2S and SPI peripheral in our STM32F446 MCU. Then a visual pinout tab was generated in the IDE that emulates the real pinout of our MCU. Furthermore, the environment allowed you to prevent collisions by sending error warnings of which data paths conflicted with each other when debugging. We were also able to configure power sequencing of the pin out. We made sure to confirm which components are to be powered on in concurrence with one another through debugging. After the pinout and peripheral configuration was complete, we transferred it to the IDE in the next section. The pin outs needed to be accurate or else we would have ran into major complications when debugging.

7.3.2 STM32CubeIDE

The STM32CubeIDE is a development platform that runs on the C and C++ language. This environment facilitates the peripheral configuration along with debugging capabilities. Once the appropriate pin outs developed in the STM32CubeMX environment are created, the project was imported into the STM32CubeIDE and created an environment to implement programs into the MCU in which we were able to compile and debug our code that we used to process the effects in our guitar pedal. We desired to make sure memory allocation was working in an efficient manner. So we utilized advanced debugging features that provide views of CPU core registers, memories, and peripheral registers.

7.3.3 Version Control

An important part of debugging and backtracking was having a clean and organized codebase. In the process of development we utilized GitHub as our main version control tool. This allowed us to have an organized repository with different branches for developing different modules of the code and a main branch which will act as the updated consolidated version of the code. GitHub also allows developers to go back to previous versions of the uploaded code and see a snapshot of it. Having a good version control can

greatly reduce the hassle of debugging, and it was also helpful in the case one of the group members lost all their code, since it was stored in the cloud. GitHub was easy to use and comprehensive for large projects such as this one, especially when a lot of people were working on the same codebase.

7.4 Specific Software Testing

In order to ensure a good quality guitar pedal, we made sure that every single effect was working exactly as intended, without outputting any extra noise or distortions other than the effect. We also needed to make sure that the input from all physical knobs was read and used accordingly, since the knobs will acquire different controls depending on what effect was currently selected. For this reason we created a series of tests that we could use to measure both the functionality and quality of each individual effect. Each separate effect was put to the test first by using an oscilloscope to examine the output sound waves, and secondly by plugging in a guitar and amplifier for sound tests.

During the first phase of testing, a function generator was used as input to the system. In this manner we can get a standard control signal to modify and test the effects on. The prototype's output will go to an oscilloscope, where the modified wave was displayed. At this point, we could examine the resulting shapes of each wave and compare them to each effects' typical waveforms. For instance, If we were testing on distortion, we expected to see a waveform that should be hard-clipped on the output. This method was also a great way to test each individual knob and their effect on the soundwave's shape, to again compare them to expected results. Once we tested all effects through their parameters range, we proceeded to correct any discrepancies that we observed in the test.

As it pertains to the second stage, it consisted of plugging in a guitar and amplifier to conduct a sound test. In a similar fashion to stage one, we tested each effect in its range of parameters, but this time just listening and paying close attention to the sound quality, noise, or any discrepancies that could be heard. For obvious reasons, looking for each effect to simply sound good to the ear, and could be used in a musical setting. Once again, if we found any issues in the sound for any of the effects, we debugged the code to find and resolve the issue.

8.0 Administrative Content

In this administrative section, we provided some of the administrative information pertaining to the ADEPT project. Provided below is an overall budget for the project, which will allow us to manage the cost of buying the parts necessary to build our prototype. In addition, we included a milestone table that provided specific dates of completed events/accomplishments of the project, spread across the two semesters.

8.1 Milestone Discussion

Project Report				
Divide and Conquer Document	Alejandro, Dylan, Tyler, Diego	100%	4/26/21	1/29/21
Table of Content	Alejandro, Dylan, Tyler, Diego	100%	1/29/21	4/16/21
60 page draft report	Alejandro, Dylan, Tyler, Diego	100%	1/29/21	4/2/21
100 page draft report	Alejandro, Dylan, Tyler, Diego	100%	1/29/21	4/16/21
100 page final report	Alejandro, Dylan, Tyler, Diego	100%	1/29/21	4/27/21
Research, Documentation & Design				
Block Diagram	Dylan	100%	1/25/21	1/29/21
Components and parts list	Alejandro, Dylan, Tyler, Diego	100%	1/27/21	4/1/21
Microcontroller/Microprocessor	Diego & Alejandro	100%	1/27/21	4/1/21
ADC/DAC/CODEC	Alejandro & Diego	100%	1/27/21	4/1/21
Network & connections schema	Diego & Dylan	100%	1/27/21	8/1/21
Effects	Diego & Alejandro	80%	1/27/21	8/1/21
Power supply	Tyler & Dylan	100%	1/27/21	8/1/21
PCB layout	Tyler & Dylan	90%	1/27/21	8/1/21
Testing & Verification				
Tone section breadboarding	Dylan	100%	3/1/21	8/1/21
MCU/CODEC External Clock	Diego & Alejandro	90%	5/1/21	8/1/21
MCU & CODEC Communication	Diego & Alejandro	90%	5/1/21	8/1/21
Power Supply	Tyler & Dylan	100%	5/1/21	8/1/21
Systems Check Routine	Diego & Alejandro	0%	5/1/21	8/1/21
Switches & User Interface	Diego & Alejandro	0%	5/1/21	8/1/21
DSP Effects	Diego & Alejandro	0%	5/1/21	8/1/21
PCB layout	Tyler & Dylan	0%	5/1/21	8/1/21
Important Meetings				
1st Meeting w/ Lei Wei	Alejandro, Dylan, Tyler, Diego	100%	2/2/21	2/2/21
Pre 60 page Draft Delivery	Alejandro, Dylan, Tyler, Diego	100%	4/1/21	4/1/21
Pre 100 page draft delivery	Alejandro, Dylan, Tyler, Diego	100%	4/26/21	4/27/21
100 page final report	Alejandro, Dylan, Tyler, Diego	100%	4/27/21	4/27/21

Figure 76: Semester Milestones Gantt Chart

8.2 Budget and Finance Discussion

Since the cost of these parts is relatively minimal, our team will not require a sponsorship for our project, and was self-funded. The total estimated cost of all the parts necessary in order to build the working ADEPT prototype was approximately \$150.

Parts	Cost
PCB	\$30-60
PCM3060 CODEC	\$5.69
Resistors (various values)	Already purchased
Capacitors (various values)	Already purchased
Diodes (1N4148 and 1N4001)	Already purchased
Transistors (2N2222A)	Already purchased
Linear and Logarithmic Potentiometers	\$0.55/ea
Encoder Rotary Switch (for preset selection)	\$2.00
STM32 MCU (for DSP)	\$7.45
Memory chip (for preset selection)	\$1.69
External Oscillator Crystal	\$1.00
LCD (16x2)	\$19.97
9V DC Power Jack (Center Negative)	\$0.65/ea
Metal Enclosure (Hammond 1590BB)	\$6.00
Knobs	\$0.65/ea
3PDT true bypass switch	\$2-10
Soft-click momentary switch	\$2-10
¼" mono input/output jacks	\$2.00/ea
Indicator LEDs	Already purchased
Power Supply	\$30.00
ST-LINK/V2 USB connector (for programming)	\$22.61

Table 10: Budget and Parts List

9.0 Results and Conclusion

After countless hours of development and troubleshooting, we were able to come up with a functional prototype that satisfied the requirements that we set at the beginning. The device was able to perform analog to digital conversion of the guitar signal, for it to be modified by the MCU through software and lastly to be outputted to a sound amplifier with minimal noise added to it. We implemented almost all of the effects that we tentatively selected at the beginning including reverb, delay, distortion, compressor pitch-shifter and a looper functionality, as well as other effects that we created through experimentation with the initial effects such as “pipe”. In addition, the prototype adhered to all of the power and electrical requirements while being lightweight and compact at the same time. Given all of this, we believe that we were able to successfully envision, design, and develop a working guitar effects pedal prototype that could be used by both musicians and engineers to get a better understanding of digital signal processing concepts.

Appendices

This final section of our paper includes all the references used for our research, the proof of copyright permissions for any third party images used to supplement our research, and any relevant datasheets pertaining to the hardware used in our design.

Appendix A: References

Dissecting a Distortion / Overdrive Guitar Pedal:
https://www.neatcircuits.com/doctor_pedal.htm

Digital Signal Processing using a Raspberry Pi:
<https://www.electrosmash.com/forum/pedal-pi/207-basics-of-audio-dsp-in-c-for-rapsberry-pi-zero>

DSP chip Alternatives:
<https://cdn-shop.adafruit.com/datasheets/vs1000.pdf>
<http://www.vlsi.fi/en/products/vs1103.html>
<https://ww1.microchip.com/downloads/en/DeviceDoc/70292G.pdf>
<https://www.ti.com/product/TMS320C6711D>

ProCo RAT Circuit Analysis:
<https://www.electrosmash.com/proco-rat>

Electro Harmonix Big Muff Pi Circuit Analysis:

<https://www.electrosmash.com/big-muff-pi-analysis>

Ibanez Tube Screamer Circuit Analysis:

<https://www.electrosmash.com/tube-screamer-analysis>

Passive 3PDT Bypass vs. Active Relay Bypass:

<https://www.coda-effects.com/2016/04/relay-bypass-conception-and-relay.html>

Capacitors in a Guitar Pedal Circuit:

<https://www.coda-effects.com/2020/01/all-you-need-to-know-about-capacitors.html>

Resistors in a Guitar Pedal Circuit:

<https://www.coda-effects.com/2015/08/all-you-need-to-know-about-resistors-in.html>

Frequency Range of a Guitar:

<http://recordingology.com/in-the-studio/guitars/#:~:text=The%20fundamental%20frequencies%20in%20the,at%20multiples%20of%20these%20frequencies>

Comparing Different Tone Controls for Guitar Pedals:

<https://www.youtube.com/watch?v=HZAUhjt75X4>

History of Effects Units:

https://en.wikipedia.org/wiki/Effects_unit

Power and voltage regulation:

<https://dronebotworkshop.com/powering-your-projects/>

Instrument Frequency Response Chart:

<https://blog.landr.com/eq-cheat-sheet/>

Silicon Chip Shortage:

<https://www.fastcompany.com/90607876/why-is-there-a-silicon-chip-shortage-three-factors-are-to-blame>

Environmental Sustainability in Music:

<https://medium.com/music-x-tech-x-future/the-urgent-need-for-a-sustainable-music-industry-the-tech-that-makes-it-possible-c5c15141db19>

Hammond Enclosure Manufacturing - Environmental Impacts:

<https://www.hamdfg.com/news/135-improving-efficiency-and-the-environment>

PCB Manufacturing and Environmental Compliance:

<https://www.mclpcb.com/environmental-impact-semiconductor/#:~:text=These%20components%20have%20the%20highest,substances%20than%20non%2Dcompliant%20products.&text=Therefore%2C%20PCBs%20don't%20have,they%20did%20in%20the%20past>

Power Supply Safety Standards:

<https://www.cui.com/catalog/resource/power-supply-safety-standards-agencies-and-marks>

80Plus Power supply Standards/Certification:

<https://www.clearex.com/80plus/>

Guitar Pedal Buffers:

<https://www.analogman.com/buffer.htm>

Audio (Logarithmic) vs. Linear Potentiometers:

<https://www.sixstringsupplies.co.uk/audio-or-linear-pots>

Using a Variable Resistor as a Control:

https://www.w9xt.com/page_microdesign_pt14_ad_pot_control.html

Difference Between 2- and 4-Layer PCBs:

<https://www.electronicdesign.com/industrial-automation/article/21805842/whats-the-difference-between-2-and-4layer-pcbs>

Impedance Bridging:

https://en.wikipedia.org/wiki/Impedance_bridging

ADC Properties:

<https://www.analogictips.com/adc-specs-resolution-accuracy-repeatability-throughput-faq>

Electronics Standards:

https://webstore.ansi.org/industry/electronics?_ga=2.82520725.248318190.1617342406-815733927.1617342406

Standards Website: www.nssn.org

UL60065 Standard:

https://global.ihs.com/doc_detail.cfm?document_name=UL%2060065&item_s_key=00413927

TI Digital Guitar Pedal

<https://www.ti.com/lit/ml/sprp499/sprp499.pdf>

FV1 Bit Crusher

<https://www.youtube.com/watch?v=VuZOQ0T5Y0w>

Arduino Mega Guitar Pedal

<https://hackaday.com/2018/05/08/stomping-microcontrollers-arduino-mega-guitar-effects-pedal/>

FV1 8FX Pedal

<https://www.youtube.com/watch?v=5byraXi2TQg>

Arduino Vocal Effects Box

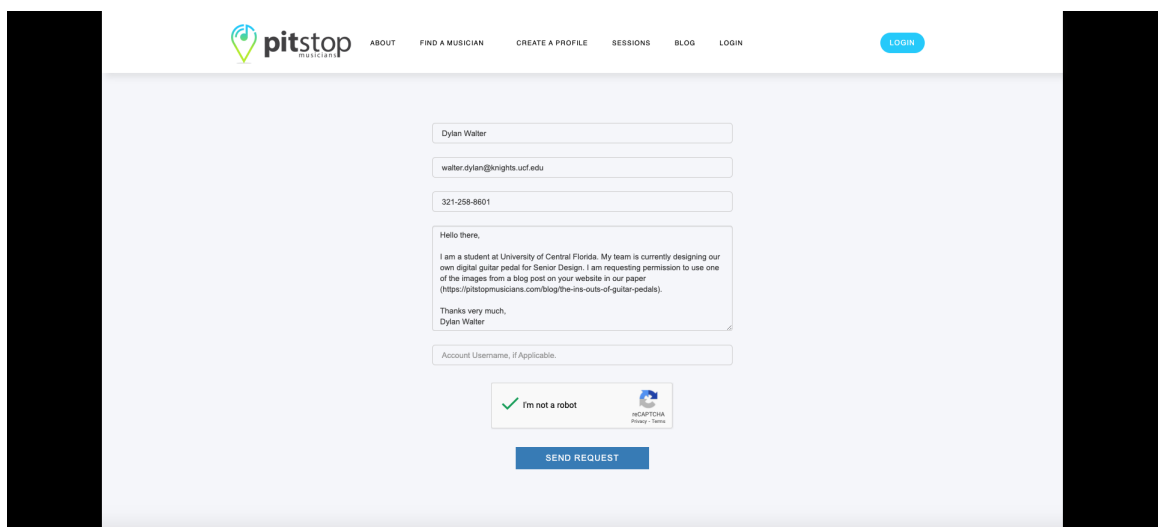
<https://www.instructables.com/Arduino-Vocal-Effects-Box/>

MCU & External Oscillator References:

1. En.wikipedia.org. 2021. *JTAG - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/JTAG> [Accessed 31 March 2021].
2. ST. *ST-LINK/V2 in-Circuit Debugger/Programmer for STM8 and STM32*. STMicroelectronics NV, 2018
www.st.com/content/ccc/resource/technical/document/user_manual/65/e0/44/72/9e/34/41/8d/DM00026748.pdf/files/DM00026748.pdf/jcr:content/translations/en.DM00026748.pdf.
3. *TC2050-ARM2010 ARM 20-Pin to TC2050 Adapter* .
www.tag-connect.com/wp-content/uploads/bsk-pdf-manager/2021/02/TC2050-ARM2010-2021.pdf.
4. Getting started with STM32F4xxxx MCU hardware development. (n.d.). Retrieved March 28, 2021, from
https://www.st.com/content/ccc/resource/technical/document/application_note/76/f9/c8/10/8a/33/4b/f0/DM00115714.pdf/files/DM00115714.pdf/jcr:content/translations/en.DM00115714.pdf
5. Oscillator design guide for STM8AF/AL/S, STM32 MCUs and MPUs. (2018). Retrieved March 28, 2021, from 5.
https://www.st.com/resource/en/application_note/cd00221665-oscillator-design-guide-for-stm8afals-stm32-mcus-and-mpus-stmicroelectronics.pdf

6. St.com. 2021. *Interfacing an STM32L1xx microcontroller with an external I2S audio codec to play audio files*. [online] Available at: https://www.st.com/resource/en/application_note/dm00087544-interfacing-an-stm32l1xx-microcontroller-with-an-external-i2s-audio-codec-to-play-audio-files-stmicroelectronics.pdf [Accessed 2 April 2021].
7. <https://www.st.com/resource/en/datasheet/stm32f446re.pdf>
8. https://www.mouser.com/datasheet/2/122/ecx_3sx-19034.pdf

Appendix B: Copyright Permissions

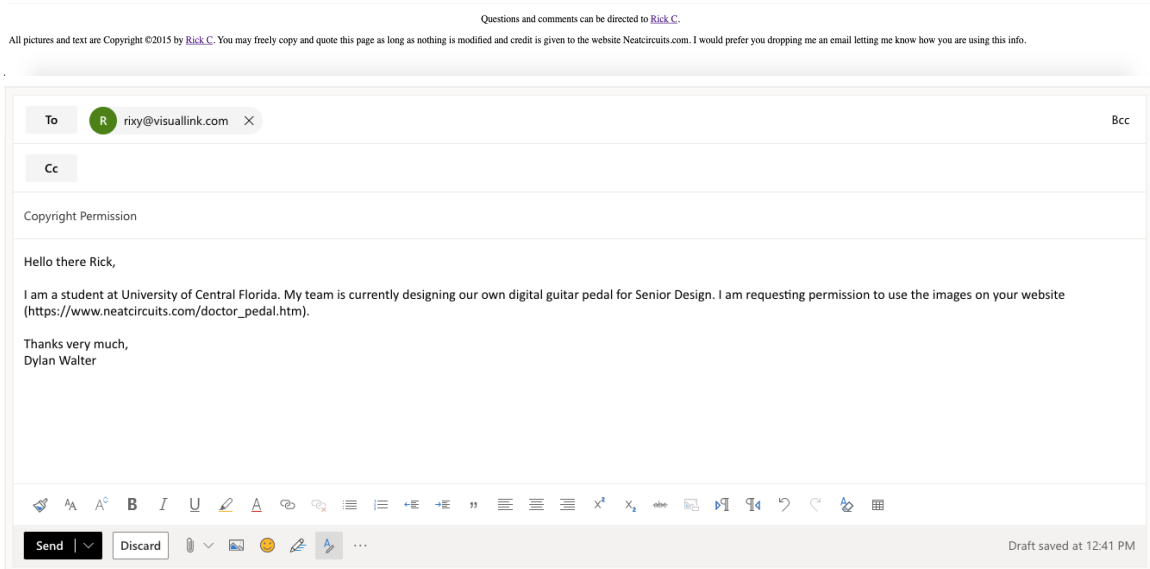


The screenshot shows the contact form on the Pitstop Musicians website. The header includes the logo and navigation links: ABOUT, FIND A MUSICIAN, CREATE A PROFILE, SESSIONS, BLOG, LOGIN, and a LOGIN button. The form fields are:

- Name: Dylan Walter
- Email: walter.dylan@knights.ucf.edu
- Phone: 321-259-8601
- Message: Hello there, I am a student at University of Central Florida. My team is currently designing our own digital guitar pedal for Senior Design. I am requesting permission to use one of the images from a blog post on your website in our paper (https://pitstopmusicians.com/blog/the-ins-outs-of-guitar-pedals). Thanks very much, Dylan Walter
- Account Username, if Applicable: (empty)

Below the message field is a reCAPTCHA widget with the text "I'm not a robot" and a "SEND REQUEST" button.

Pending Permission Request - Cover Photo



Pending Permission Request - Input and Output Buffers

Copyright Permission ☰

RC Rick C. <rixy@visuallink.com> 👍 ↶ ↷ → ...
 Tue 4/6/2021 1:22 PM
 To: Dylan Walter

Be my guest!

Rick

...

Reply | **Forward**

DW Dylan Walter 👍 ↶ ↷ → ...
 Tue 4/6/2021 12:42 PM
 To: rixy@visuallink.com

Hello there Rick,

I am a student at University of Central Florida. My team is currently designing our own digital guitar pedal for Senior Design. I am requesting permission to use the images on your website (https://www.neatcircuits.com/doctor_pedal.htm).

Thanks very much,
 Dylan Walter

Permission Granted - Input and Output Buffers

Submit a request

I'm writing in about ...

Your email address *

Subject *

Suggested articles

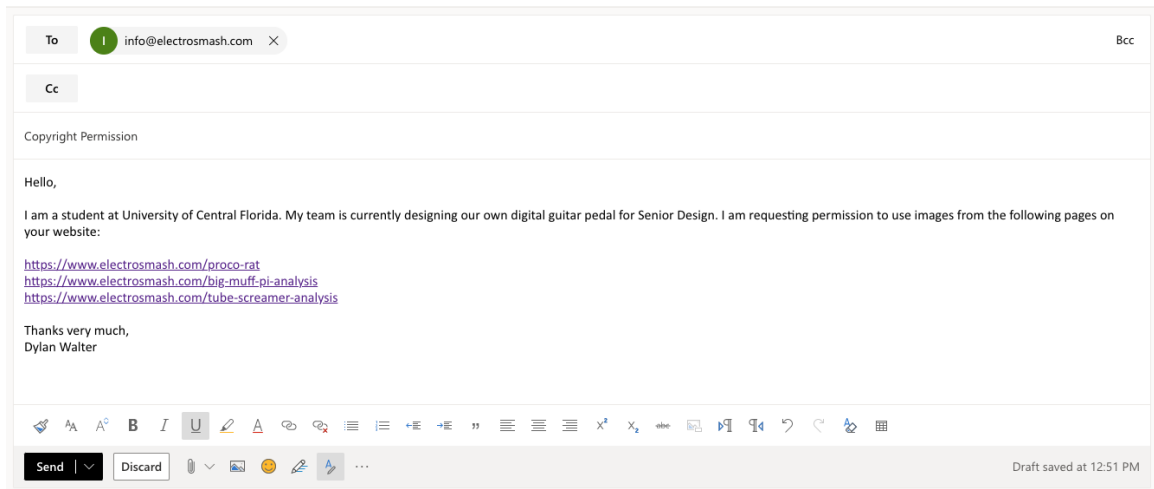
- Does LANDR pay the contributors and collaborators for my recordings?
- Does LANDR own my music?
- What rights do I need to release with LANDR?
- LANDR just asked me to provide a license for my release
- Distribution 101
- Do I own the content I paid for?
- What are LANDR's distribution guidelines?

Description *

 I am a student at University of Central Florida. My team is currently designing our own digital guitar pedal for Senior Design. I am requesting permission to use images from the following pages on your website: (<https://blog.landr.com/eq-cheat-sheet/>)
 Thanks very much,
 Dylan Walter

[Help](#)

Pending Permission Request - Instrument Frequency Chart



Pending Permission Request - Tone Control Analysis Circuits

Info@sixstringsupplies.co.uk Tel: 01902 581 301 Mon-Fri 9am-4pm

Electronics Prewired Harnesses Knobs & Tips Wiring Diagrams Contact


About Us / Returns & Shipping / Privacy Policy / Payment & Security / Terms & Conditions / FAQs

Name

Email

Subject

Message



Pending Permission Request - Linear vs. Logarithmic Potentiometers

Re: Copyright Permission Inbox x



Six String Supplies <info@sixstringsupplies.co.uk>
to me ▾

Sat, Apr 10, 6:22 AM (1 day ago) ☆ ↶ ⋮

No problem Dylan

On Fri, 9 Apr 2021 at 20:27, Contact form for Dylan Walter <sixstringsupplies@email.bigcartel.com> wrote:
From: Dylan Walter (dylemil@gmail.com)

The following message was sent using your Big Cartel contact form at <http://www.sixstringsupplies.co.uk/contact>

Hello,

I am a student at University of Central Florida. My team is currently designing our own digital guitar pedal for Senior Design. I am requesting permission to use images from the following pages on your website:

<https://www.sixstringsupplies.co.uk/audio-or-linear-pots>

Thanks very much,
Dylan Walter

Six String Supplies Ltd

www.sixstringsupplies.co.uk



01902 581 301



Permission Granted - Linear vs. Logarithmic Potentiometers

To: w9xt@unifiedmicro.com X Bcc

Cc:

Copyright Permission - UCF Senior Design

Hello there Gary,

My name is Dylan Walter, and I am an Electrical Engineering student at University of Central Florida. My Senior Design team is currently designing our own digital guitar pedal, and I would like to request permission to use images from the following page on your website:

https://www.w9xt.com/page_microdesign_pt14_ad_pot_control.html

Using a variable resistor with a microcontroller A/D X

Microcontroller Interfacing Part 14. Using a Variable Resistor as a Control . Goals. Push button and other switches are great input devices for some applications.

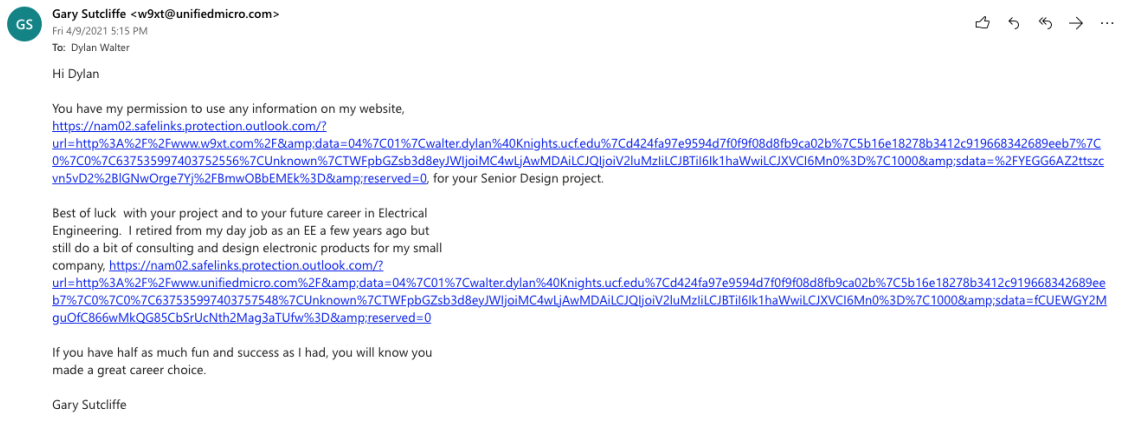
www.w9xt.com

Thanks very much!
Dylan Walter

Are the auto-complete suggestions above helpful? Yes No

Draft saved at 3:36 PM

Pending Permission Request - Resistor MCU Control Diagram



Permission Granted - Resistor MCU Control Diagram

First Name* Diego	Last Name* Conterno
Email* diegoconterno@knights.ucf.edu	Phone (optional) 7863517752

I have a question about... *

Comments/Questions *

Greetings I am a computer engineer student at University of Central Florida. I am currently writing my thesis and I came across one of your post (What are the Differences Between Buffered and True Bypass Pedals? by By Craig Anderton on May 15, 2020, 9:30 AM).

I would like to ask for your permission to use the first image used on this post.

Thank you. |

Items marked with an * are required.

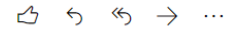
SUBMIT MESSAGE

Pending Permission Request - True Bypass Switch Figure

Copyright Permission Inquiry



Alejandro Porcar
Tue 4/27/2021 1:04 AM
To: support@apple.com



Hello,
I am Alejandro, a student at the University of Central Florida in Orlando. My senior design team is working on designing and building a guitar effects pedal. We found your content of great value and wanted to request permission to use the following in our project report:
[https://people.ok.ubc.ca/mberger/MiscSW/MacTools/MacTraining/FinalCutStudio7/Soundtrack%20Pro%203%20User%20Manual%20\(en\).pdf](https://people.ok.ubc.ca/mberger/MiscSW/MacTools/MacTraining/FinalCutStudio7/Soundtrack%20Pro%203%20User%20Manual%20(en).pdf)
Illustration of Bit Depth, Soundtrack Pro 3 User Manual, Page 499.

Thank you very much!
Alejandro.

Reply | Forward

Pending Permission Request - Digital Signal Resolution Figure

CONTACT US CONTACT US CONTACT US CONTACT US CONT

Product Interest

Support Request – Instruments

Service Request – Test Systems

United States

International

Library

- Brochures
- Application Notes
- Articles

Home > Contact Us > Support Request – Instruments

Support Request – Instruments

Fill out and submit the form to be contacted by a Data Physics representative.

Company University of Central Florida

Company Location Orlando, Florida

Contact Name Alejandro

Email Alejandropc77@gmail.com

Phone 3053840202

Subject Copyright Permission Inquiry

Description request permission to use the following in our project report:
Figure 1:
<https://blog.dataphysics.com>

Pending Permission Request - Aliasing Effect Figure

***What MathWorks products/solutions are you interested in (please include a license number if you have one)?**

hello,
I am Alejandro, a student at the University of Central Florida in Orlando. My senior design team is working on designing and building a guitar effects pedal. We found your content of great value and wanted to request permission to use the following in our project report:
Figure 1:
<https://www.mathworks.com/help/signal/ref/interp.html>

Thank you very much!
Alejandro.

Submit

Pending Permission Request - Interpolation of Discrete signal Figure



Contact Us

Contact the Electronics Tutorials Team

We always encourage you to share your ideas and improvements with us, so if you have any questions about our [Electronics Tutorials](#) website, please feel free to contact us using the form below. Many thanks for your show of support.

Email (required)

Message (required)

is working on designing and building a guitar effects pedal. We found your content of great value and wanted to request permission to use the following in our project report:
Figure 2 in:
<https://www.electronics-tutorials.ws/combination/r-2r-dac.html>

Thank you very much!
Alejandro.

Submit

Pending Permission Request - R-2R Ladder Network Figure

Contact Us

Please visit our [FAQ](#) to find helpful information before submitting your question.

Your name

Alejandro Porcar

Your email

Alejandropc77@knights.ucf.edu

Subject

Copyright Permission Inquiry

Topic *

Other

Feedback *

guitar effects pedal. we found your content of great value and wanted to request permission to use the following in our project report:

Thank you very much!
Alejandro.

Page URL

<https://www.semanticscholar.org/paper/An-FPGA-implemented-24-bit-audio-DAC-with-1-bit-Li-Lee/c96c94fe8ba41823fd1d5925573ff36586e0e444>

Pending Permission Request - Block Diagram DS DACFigure

Contact Us

Contact the Electronics Tutorials Team

We always encourage you to share your ideas and improvements with us, so if you have any questions about our [Electronics Tutorials](#) website, please feel free to contact us using the form below. Many thanks for your show of support.

Email (required)

alejandropc77@knights.ucf.edu

Message (required)

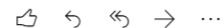
working on designing and building a guitar effects pedal. We found your content of great value and wanted to request permission to use the following in our project report:
Figure 1:
https://www.electronics-tutorials.ws/amplifier/amp_4.html
Thank you very much!
Alejandro.

Pending Permission Request - Distortion Effect Figure

Copyright Permission Inquiry



Alejandro Porcar
Tue 4/27/2021 1:36 AM
To: quebec@softdb.com



Hello,
I am Alejandro, a student at the University of Central Florida in Orlando. My senior design team is working on designing and building a guitar effects pedal. We found your content of great value and wanted to request permission to use the following in our project report:
Figure 1:
<https://www.softdb.com/what-is-reverberation-in-acoustical-analysis/>

What is Reverberation in Acoustical Analysis? | Soft dB

The Bad Side of Reverberated Sound. Although sometimes reverb is a good thing, most of the times it's a nuisance to listeners. Indeed, this phenomenon amplifies the type of sound that a source can produce, including machine noise and ventilation.

www.softdb.com

Thank you very much!
Alejandro.

Pending Permission Request - Softdb Reverb Effect Figure



Michel Pearson <m.pearson@softdb.com>

Tue 4/27/2021 8:38 AM

To: Alejandro Porcar
Cc: quebec@softdb.com



No problem Alejandro.

Have a good day.

Regards,

Michel Pearson, ing. M.Sc

Vice-président / Consultation acoustique & vibrations

T: [18666860993226]1-866-686-0993 x226

E: m.pearson@softdb.com



Permission Granted - Softdb Reverb Effect Figure

Email Us

Name:

Alejandro Porcar

Email Address:

Alejandropc77@knights.ucf.edu

Subject:

Copyright Permission Inquiry

Message:

Hello,

I am Alejandro, a student at the University of Central Florida in Orlando. My senior design team is working on designing and building a guitar effects pedal. We found your content of great value and wanted to request permission to use the following in our project report:

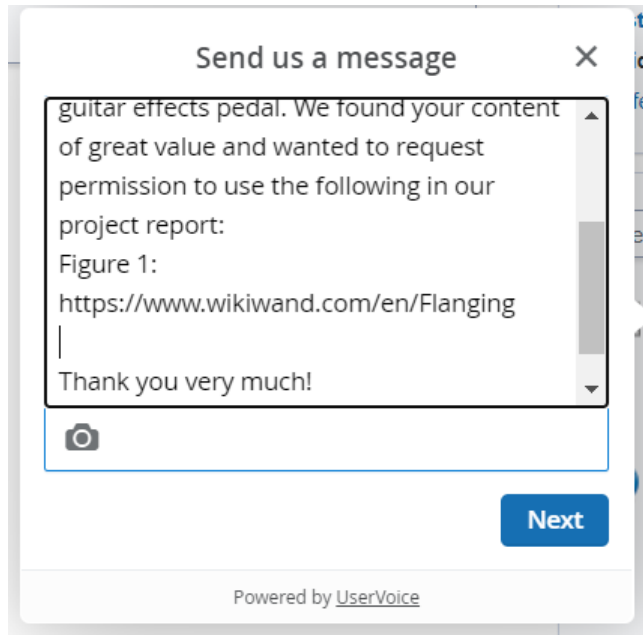
Figure 2:

<https://ledgernote.com/columns/studio-recording/types-of-reverb/>



SEND MESSAGE

Pending Permission Request - Reverb Effect Signal Structure Figure



Pending Permission Request - Flanging vs. Phasing Figure

Permission required to use content >



Diego C <diegoconterno@gmail.com>
to yetanotherelectronicschannel ▾
Hi Markus,

Sun, Nov 28, 1:18 AM (7 days ago)

Thank you once again for all your video lectures. They have made my project possible and I am very happy with the results. I would like to know if it is okay to utilize the content from your video lectures and your github code for my school project. Please let me know.

Thanks,
Diego Conterno

Pending Permission Request - YetAnotherElectronicsChannel (Markus Noll)

Appendix C: Datasheets/Pinout Diagrams

STM32F445RCT7 Datasheet

<https://www.st.com/resource/en/datasheet/stm32f446re.pdf>

LM78L 100-mA Fixed Output Linear Regulator Datasheet

https://www.ti.com/lit/ds/symlink/lm78l.pdf?ts=1617342264702&ref_url=https%253A%252F%252Fwww.google.com%252F

LM317 12v to 9v converter Datasheet

https://www.ti.com/lit/ds/symlink/lm317.pdf?ts=1617306155471&ref_url=https%253A%252F%252Fwww.google.com%252F

2N2222A Transistor Datasheet

<https://www.onsemi.com/pdf/datasheet/p2n2222a-d.pdf>

J201 N-Channel JFET Datasheet

<https://www.interfet.com/jfet-datasheets/jfet-j201-j202-interfet.r00.pdf>

OPA2134 Operational Amplifier Datasheet

https://www.ti.com/lit/ds/symlink/opa4134.pdf?ts=1617270362440&ref_url=https%253A%252F%252Fwww.google.it%252F

LM3671 3.3V DC-DC Buck converter Datasheet

https://www.ti.com/lit/ds/symlink/lm3671.pdf?ts=1617262398822&ref_url=https%253A%252F%252Fwww.google.com%252F

TPS54120 5V DC-DC Buck converter Datasheet

https://www.ti.com/lit/ds/symlink/tps54120.pdf?ts=1617334393122&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTPS54120

PCM186x Datasheet

<https://www.ti.com/lit/ds/symlink/pcm1863.pdf?ts=1617254239399>

PCM3060 Datasheet

https://www.ti.com/lit/ds/symlink/pcm3060.pdf?ts=1617326535836&ref_url=https%253A%252F%252Fwww.google.com%252F

Solid State Relay 24-V AC Switch With Galvanic Isolation:

https://www.ti.com/lit/ug/tidub92b/tidub92b.pdf?ts=1612840577475&ref_url=https%253A%252F%252Fwww.google.com%252F