Spring 2021

Analog & Digital Remote Synthesizer

Adam Brunner
adb151@uakron.edu

Andrew Cihon-Scott
ajc210@zips.uakron.edu

Scott Grisso
sbg28@zips.uakron.edu

Linus Wright
lww9@zips.uakron.edu

Analog + Digital Remote Synthesizer

Project Design Report

Design Team 9

Adam Brunner

Andrew Cihon-Scott

Scott Grisso

Linus Wright

Dr. Robert Veillette

11/25/2020

Table of Contents

List of Figures 4

List of Tables 5

Abstract 7

1. Problem Statement 8

1.1. Need 8

1.2. Objective 8

1.3. Background 8

1.4. Marketing Requirements 14

2. Engineering Analysis 14

2.1. Circuits (LWW) 14

2.1.1. Power Requirements (ADB, ACS) 14

2.2. Electronics 17

2.2.1. Input Select (ACS) 17

2.2.2. Reverb Circuit (ACS) 20

2.2.3. Distortion Circuit (LWW) 21

2.2.4. Tone Control (LWW) 24

2.2.5. Volume Amplifier (ACS) 27

2.3. Signal Processing 27

2.3.1. Synthesizer Signal Processing (ACS) 27

2.3.2. A/D & D/A Conversion (ADB) 28

2.3.3. MIDI Input Processing (ADB) 31

2.4. Communications 33

2.4.1. Remote Host Communication & Data Rates (ADB) 33

2.4.2. Embedded Peripheral Interfacing (ADB) 36

2.5. Computer Networks 41

2.5.1. Wireless Peer-to-Peer Networking (SG) 41

2.5.2. Remote Application Design (SG) 44

2.6. Embedded Systems 45

2.6.1. Systems Specifications (ADB) 45

3. Engineering Requirements Specification 51

4. Engineering Standards Specification 52

5. Accepted Technical Design 53
5.1. Hardware Design 53
5.1.1. Level 0 Functional Decomposition 53
5.1.2. Level 1 Functional Decomposition 54
5.1.3. Level 2 Functional Decomposition 56
5.1.4. Level 3 Functional Decomposition 61
5.1.5. Hardware Schematics 66
5.2. Software Design 90
5.2.1. Level 0 Functional Decomposition 90
5.2.2. Level 1 Functional Decomposition 92
5.2.3. Level 2 Functional Decomposition 94
5.2.4. Systems Integration Code/Pseudocode 99
6. Mechanical Sketch 127
7. Design Team Information 127
8. Parts Information 128
8.1. Cumulative Parts Lists 128
8.2. Materials Budget 131
9. Project Schedules 133
9.1. Final Design Gantt Chart (ADB) 133
10. Conclusions and Recommendations 135
11. References 136
12. Appendices 138

List of Figures

Figure 1: Simulation of a possible oscillator circuit, showing the generated square, triangle and sine waves 19
Figure 2: “Hard” Distortion Method as a Function of Gain 22
Figure 3: “Soft” Distortion Method as a Function of Gain 23
Figure 4: Tone Control at Unity Gain 26
Figure 5: Tone Control at 10x Steps 26
Figure 6: Connection diagram of SPI Master to multiple slave devices [20] 37
Figure 7: Connection diagram of I^2C Master to multiple slave devices (also supports multiple masters) [21] 37
Figure 8: Example of hybrid interfacing with U/D digital potentiometer [22] 39
Figure 9: ATWINC State diagram [24] 43

Figure 10: Host USB interface circuit example [23] 48
Figure 11: Level 0 block diagram of Analog + Digital Remote Synthesizer 53
Figure 12: Level 1 block diagram of Analog + Digital Remote Synthesizer 54
Figure 13: Level 2 block diagram of Digital Control Block 56
Figure 14: Level 2 block diagram of Synthesizer Control Circuits 59
Figure 15: Level 3 block diagram of Input Select 61
Figure 16: Level 3 block diagram of Tone Control 63
Figure 17: Level 3 block diagram of Effects Circuits 65
Figure 18: EagleCAD Schematic of Digital Control Subsystem MCU Data Connections (ADB) 66
Figure 19: EagleCad Schematic of Reverb Driver Circuit (ACS) 72
Figure 20: EagleCad Schematic for Oscillator Circuit (ACS) 73
Figure 21: EagleCad Schematic for Reverb Overvoltage Protection (ACS) 75
Figure 22: Reverb Volume Mixer (ACS) 77
Figure 23: EagleCad Schematic for Tone Control Circuit (LW) 79
Figure 24: EagleCad Schematic for Distortion Circuit (LW) 83
Figure 25: EagleCad Schematic for Power Supply Circuit (LW) 85
Figure 26: Volume Control Circuit (ACS) 87
Figure 27: Input Select Circuit (ACS, LW) 88
Figure 28: Level 0 block diagram describing input/output relationship between software. 90
Figure 29: Level 1 flow chart of Remote Application 92
Figure 30: Level 1 flow chart of Digital Control Processes 93
Figure 31: Level 2 flow chart of System Setup Processes 94
Figure 32: Level 2 flow chart of Recording Processes 95
Figure 33: Level 2 flow chart of Control Profile Save/Load Processes 96
Figure 34: Level 2 flow chart of Synthesizer Control Processes 97
Figure 35: Level 2 flow chart of External Digital Input Processing 98
Figure 36: Webpage render of current remote application user interface. 126
Figure 37: Mechanical sketch of proposed system 127

List of Tables

Table 1: Embedded component typical power requirements 15
Table 2: Digital potentiometer typical power requirements 16
Table 3: MIDI Command Set [18] 32
Table 4: Comparison of various sampling resolutions/rates and their resulting PCM bitrates 34
Table 5: Comparison of USB standards and their transfer speeds [19] 35
Table 6: Power/Current Consumption of ANTWINC1500 for various device states [25] 43
Table 7: Level 0 functional requirements of Analog + Digital Remote Synthesizer 54
Table 8: Level 1 functional requirements of Digital Control Block 55
Table 9: Level 1 functional requirements of Power Supply Unit 55
Table 10: Level 1 functional requirements of Remote Application Host 55
Table 11: Level 1 functional requirements of Synthesizer Control Circuits 56

Table 12: Level 2 functional requirements of Analog-to-Digital Converter 57
Table 13: Level 2 functional requirements of Microprocessor 57
Table 14: Level 2 functional requirements of Synthesizer Controls I/O 57
Table 15: Level 2 functional requirements of Communication Module 58
Table 16: Level 2 functional requirements of Digital-to-Analog Converter 58
Table 17: Level 2 functional requirements of Input Select 59
Table 18: Level 2 functional requirements of Tone Control 60
Table 19: Level 2 functional requirements of Effects Circuits 60
Table 20: Level 2 functional requirements of Volume Control 60
Table 21: Level 3 functional requirements of Digital/Analog Input Toggle 62
Table 22: Level 3 functional requirements of Analog Input Toggle 62
Table 23: Level 3 functional requirements of Voltage Controlled Oscillator 62
Table 24: Level 3 functional requirements of High-Pass Filter 63
Table 25: Level 3 functional requirements of Band-Pass Filter 64
Table 26: Level 3 functional requirements of Low-Pass Filter 64
Table 27: Level 3 functional requirements of Signal Summation 64
Table 28: Level 3 functional requirements of Distortion Circuit 65
Table 29: Level 3 Functional requirements of Reverb Circuit 65
Table 30: Parts List with Reference Designators and Part Descriptions for Digital Control Schematic (ADB) 67
Table 31: Parts for Reverb Drive and Oscillator Circuits (ACS) 74
Table 32: Parts List for Reverb Overvoltage Protection (ACS) 76
Table 33: Reverb Volume Mixer Parts List (ACS) 78
Table 34: Parts List for Tone Control Circuit: (LW) 80
Table 35: Parts List for Distortion Circuit (LW) 84
Table 36: Parts List for Power Supply Circuit (LW) 86
Table 37: Volume Control Parts (ACS) 87
Table 38: Parts list for Input Select Circuit (ACS + LW) 89
Table 39: Level 0 Functional Requirements of Remote Application 91
Table 40: Level 0 Functional Requirements of Digital Control Processes 91
Table 41: Cumulative parts list for Digital Control Schematic (ADB) 128
Table 42: Cumulative parts list for Reverb Drive and Oscillator Schematics (ACS) 129
Table 43: Cumulative parts list for Reverb Overvoltage Protection Schematic (ACS) 129
Table 44: Cumulative parts list for Reverb Volume Mixer Schematic (ACS) 129
Table 45: Cumulative parts list for Tone Control Schematic (LWW) 130
Table 46: Cumulative parts list for Distortion Schematic (LWW) 130
Table 47: Cumulative parts list for Power Supply Schematic (LWW) 130
Table 48: Cumulative parts list for Volume Control Schematic (ACS) 130
Table 49: Cumulative parts list for Input Select Circuit (ACS + LWW) 131
Table 50: Project materials budget list, consolidated from Cumulative Parts Lists 131

Abstract

The purpose of this project is to develop and design an analog synthesizer musical instrument that integrates embedded digital hardware into the design to enable control from a remote source. The use of digital hardware enables the potential for a wide range of convenient features such as sound profile saving and loading, output recording functionality, and the ability to accept digital input from another musical instrument utilizing the Musical Instrument Digital Interface (MIDI). In addition to the synthesizer itself, this project also includes the design of a companion application that can be hosted on a wide variety of consumer computing devices, such as desktop or laptop PCs, tablets, or mobile smart phones. This application provides a clear and easy to use user interface for controlling all aspects of the synthesizer's sound output and can initiate the receipt of an output recording stream for later playback as well as providing the interface for saving and loading sound profiles. With this design the Analog and Digital Remote synthesizer provides both the sound characteristics of traditional analog synthesizers as well as the conveniences of modern digital hardware.

Key Features:

- Analog Audio Synthesis and Effects
- Remote Control via External Application
- Embedded Digital Hardware Interfacing
- Synth Sound Profile Save States
- Accepts Wide Range of External Instrument Inputs
- Output Stream Recording via Remote Application

1. Problem Statement

1.1. Need

Many musicians are nostalgic for the sounds of a pre-digital era, but the challenges of achieving such a sound far exceed the convenience of newer digital recording methods. According to RIAA's 2018 Year-End Music Industry Revenue Report, revenues from shipments of physical products decreased 23% from 2017 while vinyl album revenue increased 8%, reaching the highest level since 1988. [1] As well musicians such as Daft Punk have been famously lauded for using outdated instruments and recording technologies rather than modern digital systems. Thus, a system that could combine the sounds of analog with the ease of use and recording of digital would be well received.

1.2. Objective

The creation of an Analog Synthesizer with added functionality through digital software that is controlled by an external device would combine the appealing aspects of digital and analog instrumentation. By creating sound waveform generators with analog filtering, the classic sounds people are nostalgic for could be created. Integrating the system with an external application for ease of recording, playback, and the creation of live backing tracks for practice would elevate the functionality beyond what analog synthesizers on the market currently offer.

1.3. Background

In recent decades, the traditional methods of frequency synthesis using analog circuits, which were prevalent before the mid-1980s, have been largely substituted with a cheaper & easier to manufacture digital alternative. However, it can be shown that a market still exists for

these analog synthesizers, even in today's digital age. Some music producers seek out the pure, non-discretized waveforms that these devices produce with the hopes of achieving the highest fidelity possible in the music they create. In addition, the unique combinations of analog signal processing and filtering found on these devices can produce sounds that are impossible to perfectly replicate when using their digital counterpart.

Given the advent of digital & software-based synthesis, research into developing new analog synthesis techniques has largely been stifled. This is most likely due to the digital alternative methods being simpler to implement, among other reasons. As N. Temenos, C. Basetas, and P. Sotiriadis say in their paper *Efficient All-Digital Frequency Synthesizer Based on Multi-Step Look-Ahead Sigma-Delta Modulation*, "Pure analog or mixed-signal frequency synthesizers lack the digital accuracy, fine frequency resolution and the fast frequency hopping of digital frequency synthesizers. Furthermore, digital circuits offer additional advantages such as immunity to process, voltage and temperature variations (PVT), scalability, reconfigurability, faster design cycles, smaller chip area, etc."[2] These advances and advantages of digital synthesizers are much of the reason today's analog synthesizers have since stagnated in growth. As a result, many analog synthesizers today are largely made of the same components as those made during the peak of analog synthesizer usage.

Analog synthesizers can be categorized into three different configurations: Normalized, Modular, and Hybrid. These categories specify how the various modules internal to the synth are interconnected, whether that is by toggle switches & potentiometers (Normalized), by physically connecting modules using patch cables (Modular), or a combination of the two (Hybrid). These internal modules consist of a variety of voltage-controlled "building-blocks" that vary depending

on the desired feature set of the synthesizer. The most common & most integral of these building blocks include the following:

- Voltage-Controlled Oscillators (VCO)
- Voltage-Controlled Filters (VCF)
- Envelope Generator
- Low-Frequency Oscillators
- Voltage-Controlled Amplifiers (VCA)

Generally, the control voltages in question for these modules are set via an adjustable resistance value in the form of a potentiometer.[3] These, along with other additional modules such as keyboard encoders, sequencers, and mixers are patched together to make up the "patch" used to generate, alter, and synthesize the analog waveforms and send them to the desired output.[4]

These past designs are what provide the pure output & physical controls that entice users to them still, but they also carry with them some of the same limitations of the outdated technology. The filters and oscillators used in these devices are controlled by potentiometers & switches. Rotary dials, pushbuttons, and toggle switches are common examples of controls found on analog synthesizers. Using these simple controls, a musician using a traditional analog synth would be required to take note of or remember the positions of all these controls if they wanted to reproduce the current sound profile at a later point in time. There are more recent modern designs, such as the 'Prologue' by Korg, that solve this issue by implementing sound profiles stored in on-board memory.

In addition to the difficulty of reproducing a sound profile based solely on the positions of physical controls, the portability of said sound profiles is reliant on the musician's memory &

the usage of a similar model of instrument, and even then it may not result in exact replication due to the nature of analog electronics. As mentioned previously, the Korg Prologue does feature the ability to save sound profiles, but the portability of said profiles is reliant on a dedicated software running on an external system rather than the synthesizer itself.

Beyond sound reproduction limits, there are also noise concerns. There are many different types of noise common in small-signal applications, though according to D. Self, many have no negative effect on audio signals. Some of the main noise concerns come from Johnson Noise, which is intrinsic to resistors, and Shot Noise, which is due to intrinsic variations in any steady current. Though these values are very small, they lead to imperfections in the signal that present technologies have not been able to overcome. [5]

One thing to note about noise in analog synthesizers is that such noise is where some of the difference in sound between digital and analog devices comes from. Much of the noise from digital synthesizers comes from the compression and digital to analog conversion of the signal, according to Jouko Vankka in their book Digital Synthesizers and Transmitters for Software Radio. [6] This noise lends itself to different sound quality than the noise in pure analog devices. As the primary reason to use an analog synth is the difference in sound, the noise and other effects of using a real circuit in the signal modulation is seen as a positive trait by many. While the noise does limit how low the operating signal can be before it is overtaken completely, it can be considered a feature to be watched closely for this project. In contrast, an all-digital design suffers from other challenges intrinsic to the architecture, such as frequency spurs & timing irregularities, that arenare notsent when using analog components.[7]

Our concept hopes to address some of these limitations by integrating modern digital technologies into the design, without affecting the core appeal of the analog synthesizer. Similar

to the profile mapping found on the Korg Prologue, our design will address the issue of reproducible profiles, likely by integrating digital potentiometers into the synthesizer modules while monitoring and controlling their values via both physical encoders as well as a digital microcontroller. In contrast, our design will also address the issue of portability of said profiles by introducing an external control source, something the Korg Prologue does not offer. Given that our design will feature similar circuits to those found in traditional analog designs, the noise intrinsic to those designs will still be present, however as previously mentioned this is a trait that draws some market value to analog synths and is sometimes considered more of a feature than a flaw.

Additionally, the design of the Analog and Digital Remote Synthesizer will differentiate itself from existing designs by implementing options for controlling the synthesizer remotely. This remote control could be implemented by pairing our design with a piece of software run from an external source. This would allow for fine-tuning of the sound generation settings of the device, as well as an ability to integrate recording methods for later playback. This functionality in an analog synthesizer has either not been previously implemented or is exceptionally rare as evidence for existing analog synthesizers with this capability has not been found. The Analog and Digital Remote Synthesizer will be differentiated again through the existence of an analog input that would allow any other instrument with an analog output to be modified and synthesized with our design. This would allow musicians who are unfamiliar with traditional methods of control for a synthesizer to still experience the full capability of the device by using another instrument as the device to create the synthesized signals. For example, a guitarist would be able to plug their guitar into the device, and immediately begin playing without any lengthy learning curve. Another benefit of this is an increase in the variety of sounds that the device

creates because any different instrument plugged into the device would provide a different quality of output sound.

Currently, a multitude of analog and digital synthesizers exist on the market. These synthesizers can range from simple toys unsuitable for complex musical composition to something such as the Korg Kronos synthesizer, a multi-thousand-dollar professional workstation. Fortunately, this means that a large number of patents for existing products exist that can be used as a foundation for iterative improvement and inspiration. US patent 7952395 B2 (Universal CMOS Current-Mode Analog Function Synthesizer) describes a general method of approximating mathematical functions with hardware.[8] This could prove incredibly useful for generating novel sounds as well as creating more complex sounds that are often achieved through digital signal generation. Another existing patent is US20130047824A1. This patent describes a customizable digital synthesizer that achieves many of the marketing requirements that were described in the Analog and Digital Remote Synthesizer Proposal document. The digital synthesizer described also includes a large variety of human interface devices for music generation, real-time filtering of signals for desirable audio characteristics, and the ability to output its generated signal to other devices.[9] While no single patent has been found that completely covers the scope of our marketing requirements, existing patents can provide guidance in understanding the multitude of existing systems. These existing patents will allow the design of the Analog and Digital Synthesizer to be both iterative, in that it does not discard the innovations made by previous synthesizers, and novel in that it will have a set of features that seem to currently be unavailable on any single existing instrument.

1.4. Marketing Requirements

1. The device should produce high-fidelity sound in the form of analog waveforms within the audible frequency spectrum (20 Hz to 20 kHz).
2. The device should have customizable & easily reproducible sound profiles.
3. The device should accept a variety of external inputs.
4. The device should have the ability to record output for future playback.
5. The device should be able to be controlled from a remote source.

2. Engineering Analysis

2.1. Circuits (LWW)

The core components that modulate the input signal are all made up of analog filtering circuits in consideration of our vision for a primarily analog synthesizer. By using op amps and various other passive components, a variety of different effects and control circuits have been designed. All these circuits contain active components and will be detailed in the electronics section.

The primary use of passive components in these systems are to prevent any DC signal from going through the various component circuits, bias the op-amps appropriately, and to tune the passbands of the filters inherent in the tone control subsystem.

2.1.1. Power Requirements (ADB, ACS)

Now that a final design for the synthesizer circuitry has been decided upon, the power this system will require has been ascertained. The voltage rails for the op-amps and most analog power uses will be 15 to negative 15 volts, and the voltage requirements needed for the microprocessor will be covered by a 5 volt and 3.3 volt line. This will be done through a

switching power supply that provides 24 watts with a +15 and -15 rail, along with two voltage regulators for the lower voltages required. The switching supply operates at a fixed 100kHz, which is much higher than the audible range and as such should not affect our system.

An estimate for the current drawn from an op amp biased at 30 volts rail to rail in our system based on simulations is about 5mA per amplifier. The final design has been decided upon, and it is currently using 15 op amps, leading to a draw of 75mA at a high, which is equivalent to 1.591W. The power supply used currently can supply 24 watts, so this is not a problem. One of the sub-circuits is using a much higher power than the rest due to its use of transistors to raise the current entering the reverb tank. This was estimated to require 956mW. The embedded component power requirements are shown below.

In comparison to the power requirements for the analog circuits, the digital control block, and the digital hardware within will require far less power. Analysis of the datasheets of various examples of components considered for the purposes of this project resulted in the values found in the following table.

Table 1: Embedded component typical power requirements

Component	**Typical Supply Voltage (V_{DD}) [V]**	**Typical Supply Current (I_{DD}) [mA]**	**Typical Supply Power (W_{DD}) [mW]**
Microcontroller (PIC24F)	3.3	6.3	20.79
WiFi Module	3.3	289	953.7
24-bit Σ-Δ ADC	3.3	0.25	0.825
24-bit Σ-Δ DAC	3.3	18	59.4
Total		**356.55**	**1034.715**

These results show that typical supply voltage for these embedded components is near universal at 3.3 V, simplifying the digital control blocks input power requirements. This

combined with the relatively low supply currents results in the above components only requiring just over 1 W of power. Given that the power requirements of the analog synthesizer circuits will incur power requirements of roughly 2.6W, an estimated power budget of ~ 3.5W can be made without impacting other aspects of the design. As we are using a converter with a 24W output, this easily falls within the needed parameters.

In addition to the embedded components found within the digital control block, the digital potentiometers and other components used for digital interfacing of the synthesizer circuits must be considered. The interfacing method of choice involves the use of a rotary encoder to generate pulses that will interact with either the microcontroller or the digital potentiometer. The implementation calls for the use of an I^2C digital potentiometer. The specifics of this implementation will be discussed later in the report. The following table describes the power requirements of these components.

Table 2: Digital potentiometer typical power requirements

Component	**Typical Supply Voltage (V_{DD}) [V]**	**Typical Supply Current (I_{DD}) [mA]**	**Typical Supply Power (W_{DD}) [mW]**
U/D Potentiometer	5	0.001	0.005
I^2C Potentiometer	5	0.6	3
4-2 SPDT Switch (Mux)	5	0.001	0.005

In contrast to the components found in the digital control block these components accept a wider supply voltage range with typical values around 5 V. This will require an additional output requirement for the power supply if the 5V rail is not present in the design of the synthesizer circuit. Given that the synthesizer circuit will utilize supply voltages of 15V and -15V, the desired supply voltage for the digital potentiometer components will be produced via a voltage regulator. Aside from the additional voltage requirements the digital potentiometers will

also increase the required power load by 10 μW per U/D potentiometer or 3 mW per I^2C potentiometer. Any additional load incurred by the system due to the digital potentiometers can be safely neglected given the power requirements for the synthesizer circuits will be much higher.

2.2. Electronics

The heart of the analog synthesizer lies in the analog electronics. The process for shaping the sound inputs, to pleasurable and musical sounds will be done almost entirely with analog circuitry. This circuitry is designed such that either an analog input directly from an instrument, or a digital MIDI input that has been converted to an analog signal, can be passed through it. The design of the analog synthesizer controls has been separated into several blocks, as shown in the level 2 hardware design under Figure 15.

2.2.1. Input Select (ACS)

The input select circuitry is the first block. This circuit consists of a simple switching circuit that can select the Analog Input, the Digital Input, or the Oscillators input. The switching circuit must be designed so that it can be controlled by the microprocessor. The analog and oscillator inputs must be buffered, so as the analog input cannot damage or interfere with the correct operation of the circuit. The digital input however does not need to be buffered as the output of the D/A can be easily controlled.

After an input is selected, this input will be sent into the analog effect circuitry. These circuits process the voltage signal, such that the characteristic of the sound is substantially changed. Two standard effects were chosen to be implemented in the effect circuitry. The first was distortion. Distortion makes a sound “fuzzy” or “warm” by both clipping the signal and introducing additional harmonics. This effect is commonly used to create the harsher sounds

found in many more aggressive musical styles. The other effect implemented is the reverb circuit. A reverb effect is a time delayed version of a signal existing in a superposition with the original signal. The effect differs from echo, in that the delay time is typically much smaller, on the order of 50mS or less. Despite the small delay time, the reverb can greatly influence the character of the sound. A reverb effect is often described as making the sound "fuller" and "larger." Because this effect is so desirable, reverb has been a standard effect on nearly every electronic music device for more than fifty years.

2.2.2 Oscillator Circuit (ACS)

Several methods of generating periodic waveforms were considered. Initially Bubba Oscillators and Quadrature oscillators were considered for their ability to generate very low distortion sine waves. However, this design would make it difficult to generate triangle and square waves. Square waves can easily be filtered by RC integrator circuits to become triangle and sine waves. The only limitation on filtering the square wave in this way is that the filters must be tuned to the correct frequency of oscillation. This limitation can easily be solved by including potentiometers controlled by the microcontroller in the design of the RC integrators. To generate the square waves, an op amp implementation of an astable multivibrator, a transistor-based implementation of an astable multivibrator, or a 555 timer based implementation of a square wave generator were considered. All three of these methods would be suitable for the purpose of generating low distortion square waves, however currently the op amp-based implementation is the leading contender due to its simplicity and versatility [10].

It was decided to implement the oscillator circuit using a circuit like the one shown in Figure 2. This circuit consists of an astable multivibrator constructed from a single op amp. The output of this multivibrator is a square wave that is fed into a series of active low pass filters. The

cutoff frequency of the low pass filter is varied to match the frequency of oscillation from the astable vibrator. This ensures that magnitude of the response is the same at all frequencies. The filters ensure that the square wave is filtered to become a triangle wave and a sine wave. Therefore, this circuit has 3 outputs, and can create square, triangle, or sine wave outputs.

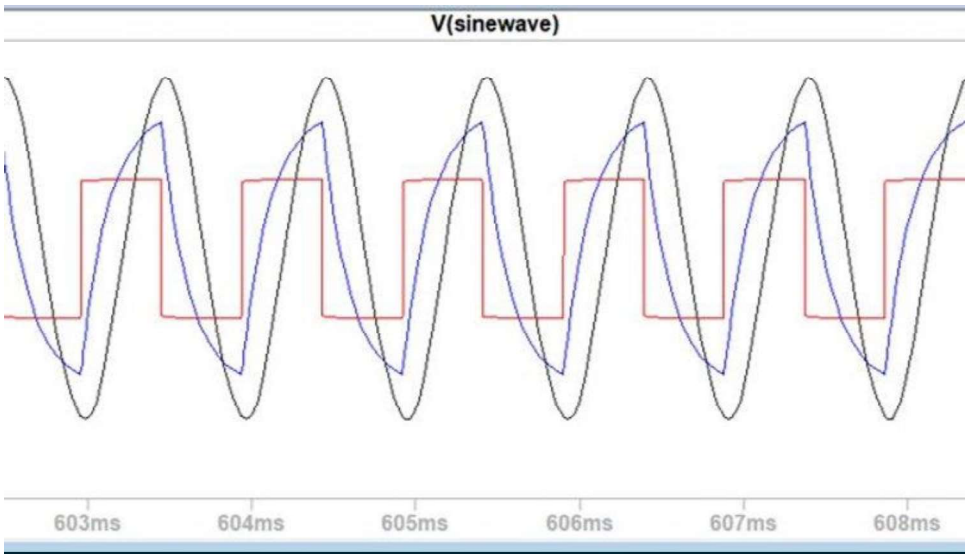


Figure 1: Simulation of a possible oscillator circuit, showing the generated square, triangle, and sine waves

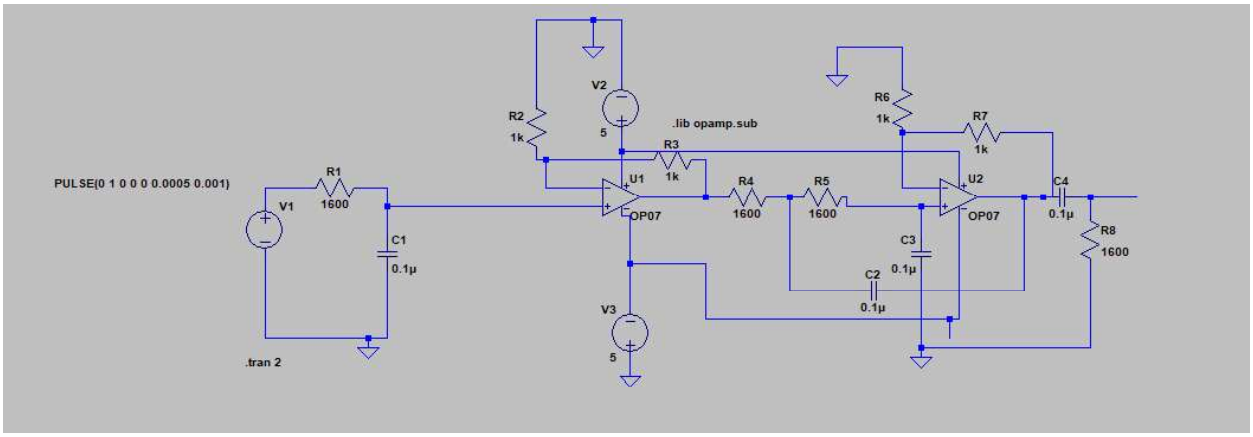


Figure 2: LTSpice Schematic for Simulation

2.2.2. Reverb Circuit (ACS)

Several methods were considered for the implementation of the reverb circuit. Because utilizing analog effects was a design goal, the difficulty of this task was increased. Most modern implementations of this effect are realized digitally. A Bucket Brigade Delay Line (BBD Line) was considered to delay the signal. This device would utilize a series of capacitors tied to a clock. The capacitors would act as a bucket that would hold the value of the signal, and then the clock would cause the value to increment to the next bucket [11]. This method was commonly utilized prior to the rise of cheap digital computing; however, it has fallen out of favor in recent years. Creating a BBD line for our project would not be feasible, as they require many thousands of capacitors in a discrete chip. Finding a BBD line chip proved to be difficult as well, as most are no longer in production, and those that are, are expensive

As an alternative to a BBD, a tape delay was briefly considered, however as tape delay technology is long obsolete, this method was quickly discarded as being prohibitively expensive to implement [12].

The final method for a reverb effect that was considered, was utilizing a reverberation tank. This method involves transducing the electrical signal into mechanical vibration using piezoelectric actuators or voice coils. The mechanical vibration is used to vibrate a long spring, and then this springs vibration is transduced at the opposite end from the input back into electrical signals. Because of how the vibrations reflect along the spring, multiple time shifted signals are superimposed upon the original signal, and a reverb effect is created. This method was chosen as it is simple, cheap to implement, and creates a very desirable sound [13]. This method of implementing the reverb was decided as being the best course of action.

The reverb was implemented using three sub circuits. The first of these being the reverb driver circuit. This driver consists of a basic op amp amplifier circuit, with two power transistors boosting the output. These power transistors are necessary due to the high current draw expected from the reverb coil. Because the reverb coil is an inductive coil, it is vulnerable to saturation. If the coil were driven to saturation the tone could be negatively effected, or even worse the impedance of the coil could change such that the current draw would become too high, and the transducers or the driving circuitry would become damaged. To prevent this from happening, a simple tunable overdrive protection circuit was added in series with the drive circuit. This circuit is simply a passive a set of diodes that pull the output of the drive circuit to ground when the voltage is too high. While the circuit is simple, it should be, if tuned properly, effective at preventing overdriving the reverb coil.

The output of the reverb tank was connected to a reverb recovery circuit. This circuit is simply an amplifying op amp with a filtered input, that connects to a summing op amp. The other input of the summing op amp is the clean unmodified music signal. The two signals are summed together so that the reverb delay is properly incorporated into the output sound.

2.2.3. Distortion Circuit (LWW)

When designing the distortion circuit, it was determined that there were 3 basic methods that are both easily available and able to be added to a simple op-amp circuit. This op-amp circuit will be powered through a positive and negative rail to simplify our power system and to not have a large DC component at any point. All of these methods of distortion involve "clipping" the signal at different points along the signal path.

The first of these methods is clipping through lowering the voltage rails on the op-amp to cause a sharp cutoff in the bounds of the output. This can easily be done through adding a

potentiometer in the supply lines, creating a voltage divider to lower both the upper and lower bound. This method is not being used for this design, however, as it was felt that our limited availability of controllers for the potentiometers and switches could be better used in other places.

The second method of distortion is to add a pair of diodes to the output of the distortion circuit. These will clip the signal sharply if the gain of the amplifier raises the signal above the diode's biasing voltage. Simulations have indicated that it warps the input slightly if the signal does not surpass the threshold, and cuts it off sharply if the signal rises higher. This is useful to achieve "harder" more distorted sounds, and our initial demonstration showed that it had a harsher "attack" on the notes played. [14] A transient analysis of this method at various gains is provided in Figure 3 below.

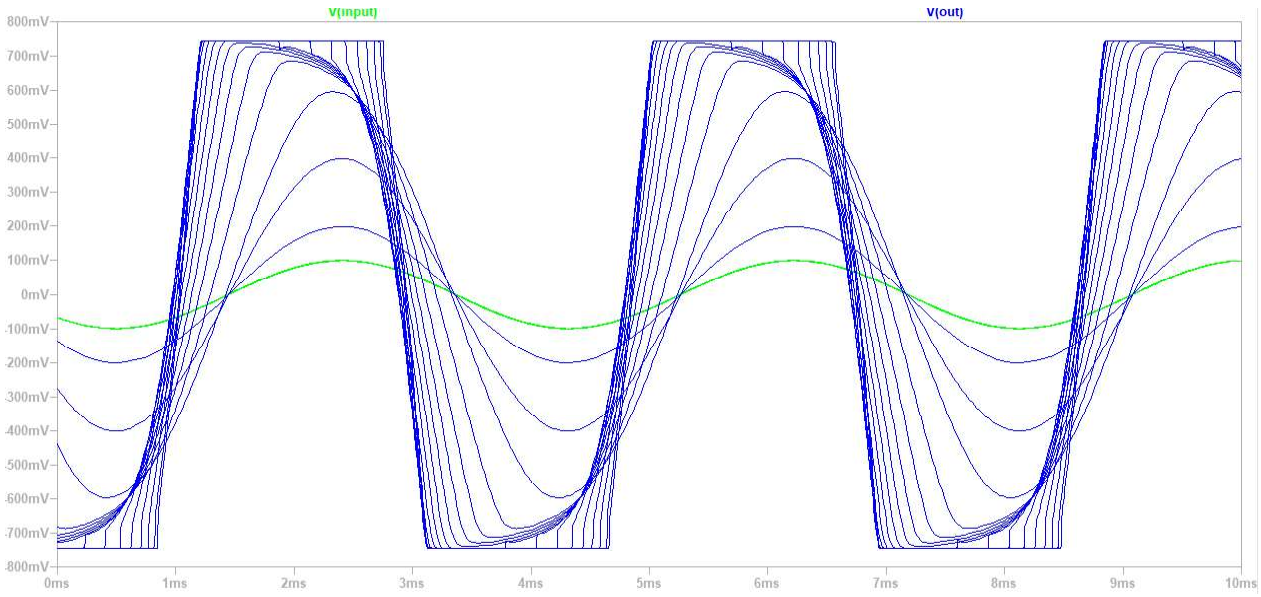


Figure 3: "Hard" Distortion Method as a Function of Gain

The final method of intentional distortion that is included in this subsystem is a pair of diodes in the feedback loop. This will act in much the same way as the diode pair in the output loop but will cause the signal to have a much softer distortion than the former due to their position causing them to “cut off” the feedback resistor, lowering the gain back to the initial value of one. The rounded effect on the signal is entirely due to the diodes put in place, while the warping of the signal into a saw-tooth set up is due to the capacitor placed in parallel with them. While this form of distortion offers a slightly lower maximum gain than the output diode loop, it also allows for a softer, less harsh method of distortion. To see the soft distortion’s effect on the signal at various gains, see Figure 4 below.

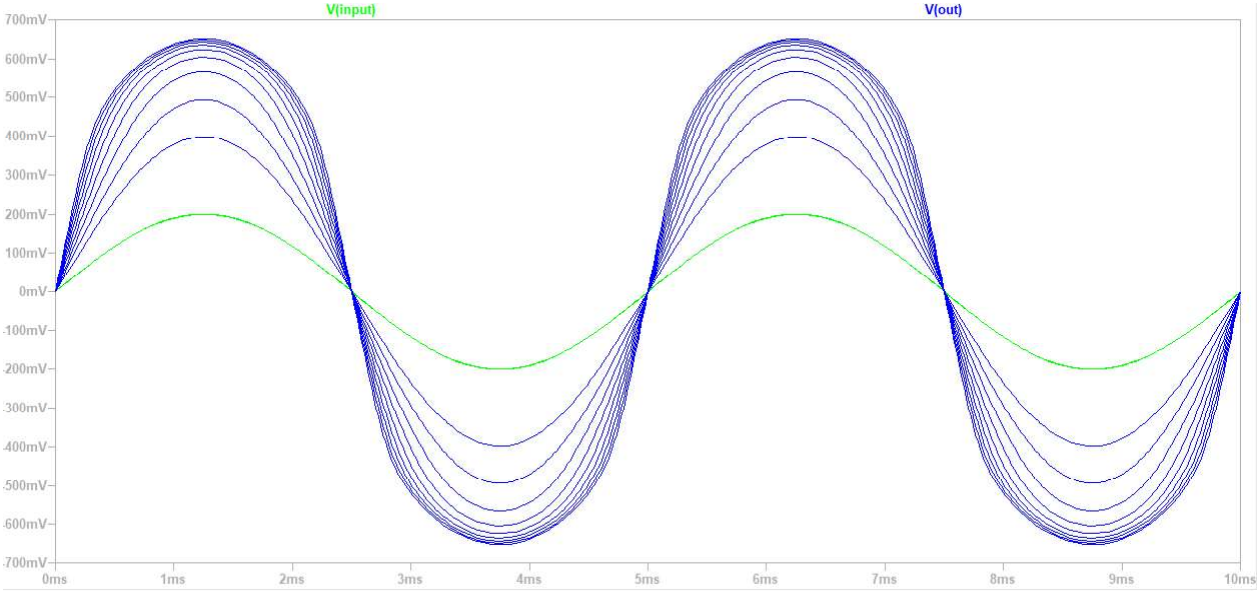


Figure 4: “Soft” Distortion Method as a Function of Gain

It had been found when running simulations and tests that having both the output clipping diodes and the feedback clipping diodes connected at the same time had nearly the same effect as just the feedback diodes on their own, so it was determined that there will only be 2 sets of switches needed to control this circuit: one for toggling the distortion on and off, and one for

toggling the soft distortion on of (switching between hard and soft distortion). In this way if no distortion effects are wanted in the final signal, there is an easy way to ignore the distortion circuit entirely. The feedback distortion diodes will also be able to be disconnected with a switch. In total, 3 switches are being used: 2 for the bypass to ensure that none of the original signal makes it to the output, and 1 for the soft/hard distortion toggle. All final designs can be seen in the Hardware Schematics section.

In order to counteract the amplification of the signal before the main amplification stage (and to keep the signal in line with the inputs that the other subsystems' designs were based on) a voltage divider has been added to the final design. As the maximum possible signal is based on the effects of the diodes, it was be simple to build one to keep the maximum output at roughly the same amplitude as the input.

2.2.4. Tone Control (LWW)

The next major subsystem in the synthesizer block is the tone control circuit. There were 2 different designs considered for this system: an active Baxandall circuit, and a Linkwitz Riley crossover. The Baxandall circuit is a fairly simple design that only requires one op-amp to function and is capable of raising and lowering the gain of both low and high frequency signals. While this is good, there was no method of changing the mid-band in between the two thresholds, and the gain changes in the different frequency bands were very asymmetrical and did not have a uniform response across the spectrum they cover. [15] The Linkwitz Riley crossover, however, uses a much larger amount of op amps (most of which are in a Sallen-Key formation) and has a much flatter response in a neutral position and each pass band has a flat change in dB. In the end it was decided that the Linkwitz Riley was a much better idea, as a flat response in the various bands is important for maintaining the quality of the output signal.

The tone control circuit as broken down into a block diagram in Figure 17 breaks the input signal down into 3 bands: the low band, mid band, and high band. This is done through several Sallen-Key filter networks running in parallel. By using multiple 2^{nd} order filters, the drop off in gain outside of the frequency range of each band is steep, ensuring that the different bands do not affect each other very much and any change in gain only effects the appropriate band.

One concern that was had with the Linkwitz Riley crossover circuit was whether the output signal frequency coming from each branch would line up well enough to not be noticeable. It was found through testing the simulated circuit that the phase of every band dropped by several hundred degrees, which may have a strange reaction when combining the signals together again. When testing the subsystem using .wav files, however, it was determined that there is no substantial change in the sound quality due to this effect.

The tone control system is controlled by a gain stage connected to each filter network. As there is a low, middle, and high band, the gain stages can raise and lower the gain of each segment of the initial signal. They are then added together with an adder circuit. If the gain in each of these bands is set to 1, there is no gain through the rest of the circuit, and the output has a flat gain of 0dB. Raising the gain by a factor of 10 in any given segment raises that section of the output by 20dB, as expected, but lowering the gain only lowers it to roughly -14dB. This is being investigated, but -14 to 20 dB is still a very good range for an amplifier with a consistent output. The output of the ideal tone control circuit described above is shown below in Figure 5 as unity gain and Figure 6 as each band's gain stage differing by a factor of 10.

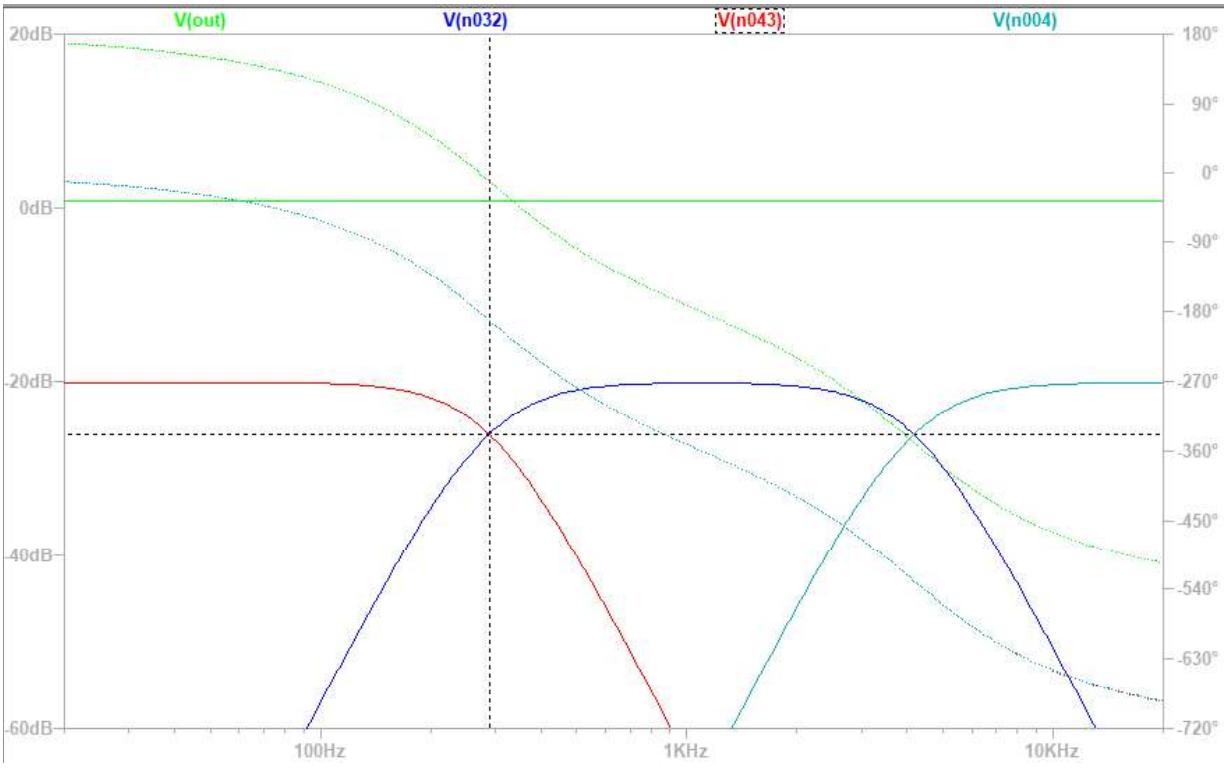


Figure 5: Tone Control at Unity Gain

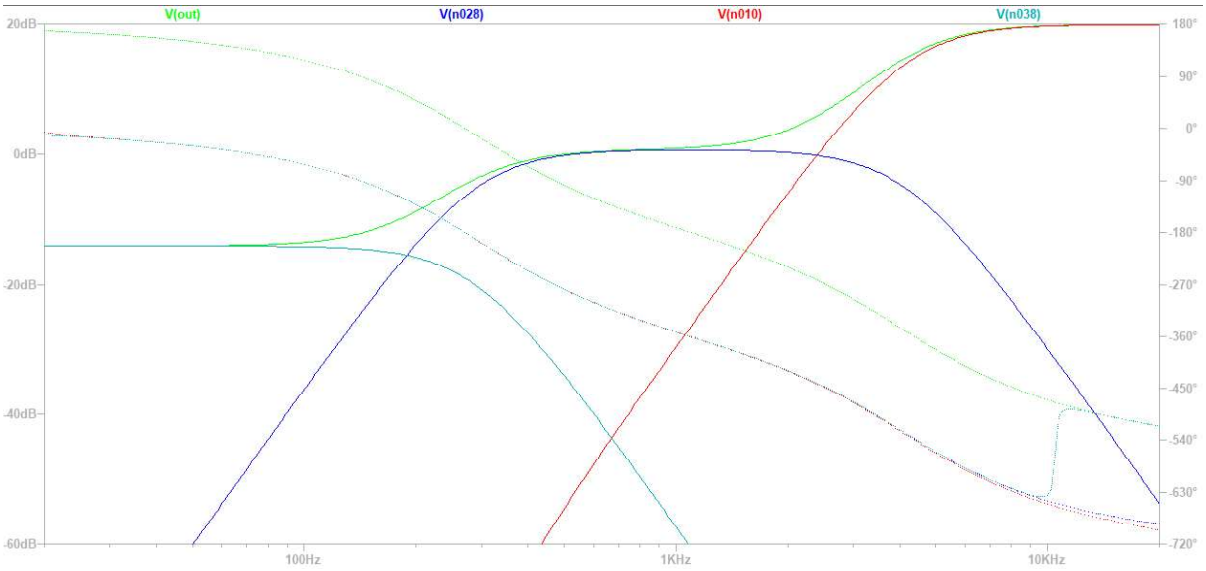


Figure 6: Tone Control at 10x Steps

2.2.5. Volume Amplifier (ACS)

The final major electronic component of the system is the volume amplifier at the output. Initially this was chosen to consist simply of a variable gain op amp inverting amplifier. This amplifier must be able to increase the gain so the output can be listened to at sufficient volume but must also be able to cut the output down to near zero. This is easily achieved with a basic inverting op amp. The basic inverting configuration is considered sufficient because it allows for a gain of less than one and inverting the audio signal should have no effect on the quality of the sound.

An alternative implementation of this volume control circuit could be realized with a Pentode vacuum tube. This implementation would introduce the characteristic harmonics associated with traditional vacuum tube-based audio equipment. This is a desirable characteristic and as such should be considered, however this implementation may be difficult in our design, as vacuum tubes often require high voltages.

It was decided to implement the volume control using a Baxendall Active Volume Op Amp circuit. This circuit uses a simple amplifying op amp, with feedback such that the potentiometer that controls the gain, gives a logarithmic response. This is ideal, as a standard linear potentiometer can be used to control the output of the circuit in a logarithmic fashion that corresponds to human hearing. This configuration was deemed better than the alternatives due to the logarithmic response that it gives from a normal linear potentiometer.

2.3. Signal Processing

2.3.1. Synthesizer Signal Processing (ACS)

The entirety of the synthesizer circuitry can be thought of as an analog signal processing device, that takes voltage inputs, and modifies them to create the desired sound profile. Because

this circuitry was discussed extensively under electronics, an in-depth discussion is not necessary here, however a brief discussion of looking at this circuitry from a signal processing perspective will be included here.

The filters following the astable multivibrator can thought of as a series of integrators, that integrate the square wave input to create both triangle, and sine wave outputs. The creation of these waveforms is desirable because each has a distinct sound and quality that many musicians desire [16].

2.3.2. A/D & D/A Conversion (ADB)

The analog output of the synthesizer circuit must be processed into a digital format to be transmitted to the remote application host. Additionally, any external digital input must undergo D/A conversion before being modified by the Synthesizer Circuit. To meet the requirements set for this project both the A/D and D/A processes must maintain the high-fidelity qualities of the audio signal. The two main variables that could have an impact on the quality of the output are the quantization resolution and the sampling rate of the A/D or D/A converter. Typical digital signal outputs from embedded ADCs will be Pulse-Code Modulated (PCM) signals, where the input analog amplitude is represented by a digital level within the set measurable amplitude range sampled at each sampling period. Typically, this is implemented in hardware using a comparator, and more advanced Δ-Σ converters will super-sample the input signal and convert it to the desired resolution & sample rate using digital decimation filters. These types of converters are common in modern audio applications and provide advantages over traditional conversion such as reducing peak noise.

The sampling rate states how many samples of the analog input are taken in a given amount of time and is typically measured in Hz. To meet the requirements of the project the A/D

converter must sample frequencies within the range of human hearing, known to be from 20 Hz to 20 kHz. The Nyquist-Shannon Sampling Theorem dictates that in order for the digital representation to exactly represent the original analog signal when restored the input must be bandlimited at a certain frequency (in this case 20 kHz) and sampled at a rate greater than twice the highest frequency component in the original signal.

Given that our system works with audible signals, the sampling frequency would need to be greater than 40 kHz to accurately represent the analog input. To this end there are a number of different sampling rates that are commonly used for sampling audio. For example, audio stored on Compact Disc is typically sampled at a rate of 44.1 kHz, and many professional audio applications and high-quality systems sample at an even higher rate of 48 kHz.

The quantization resolution dictates how precisely an A/D converter can measure variations in a signal's amplitude. This in turn determines the dynamic range that the converter is capable of measuring, which is the ratio between the largest amplitude of the analog signal and the smallest amplitude of interest of the analog signal, measured in decibels (dB). For the human ear dynamic range is typically reported as 120 dB, given that the threshold of human hearing is around 0 dB SPL (Sound Pressure Level) and the level at which discomfort is felt is around 120 dB SPL. The number of bits an A/D converter uses to represent the amplitude of the analog signal at each sample determines how precisely each sample can be represented (i.e. quantization resolution), and therefore determines the interpretable dynamic range. The following equation shows the relationship between the quantization resolution, the analog input range in volts and the number of bits N used to represent the range.

$$Quantization\ Resolution = \frac{V_{max} - V_{min}}{2^N}$$

The quantization resolution determines the smallest perceivable signal for the analog input, and therefore the lower bound of the dynamic range. The dynamic range can then be determined by taking the ratio of the largest input signal (V_{max}) and the smallest input signal (Quantization Resolution x 1 bit), as shown below.

$$Dynamic\ Range_{dB} = 20\log_{10}\left(\frac{V_{max} - V_{min}}{Quantization\ Resolution\ \times 1\ bit}\right)$$

Below is a series of equations comparing the quantization resolution and resulting dynamic range of a 16-bit ADC and a 24-bit ADC, assuming an input range of 0-3.3 V.

$$\frac{3.3V - 0V}{2^{16}} \approx 50.3\ {}^{\mu V}/_{bit} \qquad 20\log\left(\frac{3.3V}{50.3\ {}^{\mu V}/_{bit}\ \times 1\ bit}\right) \approx 96.3\ dB$$

$$\frac{3.3V - 0V}{2^{24}} \approx 0.197\ {}^{\mu V}/_{bit} \qquad 20\log\left(\frac{3.3V}{0.197\ {}^{\mu V}/_{bit}\ \times 1\ bit}\right) \approx 144.5\ dB$$

The calculation of dynamic range for an ADC can be estimated by multiplying the number of precision bits by 6 to get the approximate theoretical maximum dynamic range of the ADC in dB. Therefore, to capture the full dynamic range of the human ear the ADC must be at least 20-bit precision. Most A/D and D/A converters used for high-end audio applications are either 24-bit or 32-bit precision both of which exceed a 120 dB dynamic range based on the 6 dB rule, however there are additional factors that restrict high-precision A/D converters from achieving their theoretical maximum range. The primary factor that causes this restriction is a system noise floor higher than the minimum perceivable signal allowed by the ADC. This higher noise floor increases the minimum signal amplitude that can be accurately captured without severe effects from noise, thereby reducing the effective dynamic range. [17] While this may restrict the ability

of a real-world 24-bit ADC to match the dynamic range of the human ear it will still provide a higher dynamic range than what a 16-bit ADC is capable of.

For the purposes of digital-to-analog signal conversion an alternative to utilizing a stand-alone DAC chip would be to utilize the PWM module found on many microprocessors. These modules utilize the timers internal to the microprocessor to set the duty cycle for each pulse, in which the duty cycle represents the amplitude of the output waveform. Given that the resolution of the signal is dependent on the resolution of the timer, this would require at least a 24-bit timer to achieve the same level of quality that the A/D system is producing. Modern 16-bit MCUs such as the PIC24 contain multiple 16-bit timers that can be joined together to form a 32-bit timer, which would be ideal for the uses of the project. The signal output by the PWM module can then be filtered via a simple low-pass filter to remove the carrier frequency, resulting in the desired analog waveform.

2.3.3. MIDI Input Processing (ADB)

To meet the marketing requirements of this project the device must be capable of accepting a variety of inputs from both analog and digital sources. For the purposes of this project digital inputs will take the form of MIDI signals transmitted over a USB connection. MIDI is a standard for communication between digital and digitally enabled musical instruments to allow for control of remote MIDI devices with a separate controller, apply effects to MIDI-generated signals, and record and play back audio. MIDI signals consist of serial bit streams transmitted at a baud rate (frequency) of 31,250 Hz and are separated into byte-sized packets with an additional start and stop bit on either end of the byte. These bytes are then interpreted into 2 hexadecimal digit words ranging from 0x00-0xFF, each representing a different command or value. Any bytes starting with 1 (words 0x80-0xFF) are designated as “Command” bytes, and

any bytes starting with 0 (words 0x00-0x7F) are designated as “Data” bytes. A typical MIDI command will lead with a command byte followed by various numbers of data bytes depending on the command. For example, 0x90 designates a “Note On” event on channel 1 (MIDI signals support up to 16 MIDI channels) which is followed by two data bytes, one determining the pitch of the note and one determining the velocity of the “key press”. The following table describes the full MIDI command set and its associated values.

Table 3: MIDI Command Set [18]

Command Name	**Hex Values (Channels 1-16)**	**Data Bytes (# and representation)**
Note Off	80-8F	2 (note pitch, velocity)
Note On	90-9F	2 (note pitch, velocity)
Key Pressure	A0-AF	2 (note pitch, pressure)
Control Change	B0-BF	2 (controller number, value)
Program Change	C0-CF	1 (program number)
Channel Pressure	D0-DF	1 (pressure)
Pitch Bend	E0-EF	2 (LSB, MSB)
System Exclusive	F0	Variable
System Common	F1-F6	0, 1, or 2
End of System Exclusive	F7	0
System Real Time	F8-FF	0

Additionally, some of these commands can take advantage of a condition known as “Running Status.” This condition is recognized when a MIDI device receives a number of data bytes greater than what the command requires (i.e. a “Note On” command followed by 4 data bytes). When this condition occurs the MIDI device assumes the same command is being issued again with the extra data bytes. This prevents the need from sending repetitive command bytes when the same commands are being set repeatedly. MIDI is able to represent a series of notes being played this way using just a single “Note On” command and releasing the notes by using “Note

On” commands with a velocity value of 0 (equivalent to “Note Off”). This running status condition prevents the bandwidth of the MIDI channel from being taken up by repeating command bytes, thus making more efficient use of the channel.

When designing a device as a MIDI input the device must be able to identify the various command words and respond accordingly. Note that not all commands need to be implemented on a device to make use of the MIDI protocol, but there are certain commands that may be required depending on the application. Once the input device receives and interprets these commands it can then react to them in various ways depending on the application. For the purposes of this project, the MIDI signals will be used to generate waveforms of varying pitch (frequency) via the PWM module built into the microprocessor. These waveforms can then be sent to the synthesizer circuits to be modified by the filters and effects circuits found there.

2.4. Communications

2.4.1. Remote Host Communication & Data Rates (ADB)

In order to prevent unnecessary delay in the digital audio stream sent to the remote host during the recording process the channels and protocols must provide sufficient bandwidth to pass the desired bitrate without bottlenecks. The table below shows a comparison of the bitrates expected at various sampling resolutions and sampling rates and the resulting bitrate of a PCM output stream. The bitrates were derived simply by taking the product of the quantization resolution in bits and the sampling rate in kHz, resulting in the expected PCM bitrate in kbps.

Table 4: Comparison of various sampling resolutions/rates and their resulting PCM bitrates

Quant. Res (# of bits)	Sampling Rate (kHz)	Bitrate - PCM (kbps)
10	44.1	441
10	48	480
12	44.1	529.2
12	48	576
16	44.1	705.6
16	48	768
24	44.1	1058.4
24	48	1152
32	44.1	1411.2
32	48	1536

For the purposes of this project the A/D conversion will be performed at a minimum of 24-bit resolution with a sampling rate of at least 44.1 kHz. This implies that the resulting minimum bitrate expected from the output of the ADC will be approximately 1 Mbps. In turn, the communications channels used to transmit the digital audio stream to the remote host must provide a minimum bandwidth of 1 Mbps, with additional bandwidth allowing for potential fluctuations in the data rate without incurring large delays. The communications channels that must be considered for this requirement are any of those that are responsible for transmitting the digital audio signal from the output of the ADC to the remote host. This includes the communication provided by the communication module between the digital control block and the remote host, and any data transfer communications that occur within the digital control block to transmit the digital audio to the communication module.

Firstly, considering the communications between the digital control block and the remote host, there were multiple protocols available that were considered for the purposes of this project. For wired protocols, RS232 and RS485 communications were not considered for this project due to potential bandwidth limitations in RS232, along with the fact that these protocols

are no longer widely found built-in to modern PC systems. USB is much more prevalent in modern computers, provides sufficient bandwidth levels across all versions of the protocol and can be integrated into microcontroller communications easily. The Table *5* shows a comparison of various modern-day USB standards, when they were introduced and their maximum theoretical data speeds.

Table 5: Comparison of USB standards and their transfer speeds [19]

Standard	**Also Known As**	**Year Introduced**	**Max. Data Transfer Speed**
USB 1.1	Basic Speed USB	1995	12 Mbps
USB 2.0	Hi-Speed USB	2000	480 Mbps
USB 3.2 Gen 1	USB 3.0 USB 3.1 Gen 1 SuperSpeed USB	2008	5 Gbps
USB 3.2 Gen 2	USB 3.1 USB 3.1 Gen 2 SuperSpeed USB 10 Gbps	2013	10 Gbps
USB 3.2 Gen 2x2	USB 3.2 SuperSpeed USB 20 Gbps	2017	20 Gbps

Several different wireless communications protocols were considered for the purposes of this project. These include Bluetooth and its classic, high-speed, and low energy variants, 802.11 WiFi, and Zigbee. Bluetooth LE and Zigbee both provide mesh network capabilities at a very low power cost, however power restrictions on this scale are not a concern for this project and neither are capable of providing the data rates needed to transmit the 1 Mbps data stream efficiently. 802.11 WiFi, specifically the point-to-point WiFi Direct implementation, was also considered as all variants of the 802.11 specification provide data rates greatly in excess of 1 Mbps and it is widely adopted for use in modern devices. The classic implementation of Bluetooth the v5.0 specification is capable of up to 3 Mbps speeds over-the-air, which would

also provide sufficient bandwidth for the transfer of the digital audio stream with minimal delays.

2.4.2. Embedded Peripheral Interfacing (ADB)

In order to transmit and receive data or send configuration commands to peripheral devices the embedded microcontroller must provide a standardized mechanism for communication with said peripherals. While most fully featured MCUs will provide interfaces for both parallel and serial communication, the peripheral devices considered for this project primarily utilize serial communication methods. In serial communications data transfer occurs over a single channel with each binary digit (bit) being sent in sequence via digital signals. These methods are preferred for most peripheral devices as it limits the number of required connections between the master and slave devices. Serial communications take many different forms, however the three most commonly implemented protocols for embedded devices are Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2C), and Universal Asynchronous Receiver Transmitter (UART).

Both SPI and I^2C are synchronous, requiring that a clock signal be supplied to the slave devices. SPI uses dedicated data lines each for input and output data, whereas I^2C uses a bidirectional data line. Additionally, SPI requires a dedicated slave select line for each peripheral connected to the interface, if multiple peripherals are being used. In contrast, I^2C does not require additional lines and instead references each slave device by an address that is transmitted prior to any data being sent, thus indicating the intended recipient. As a result, I^2C only requires 2 lines per interface whereas SPI requires 3 lines plus one line per slave device. The following figures provide a visual comparison of the connections necessary for SPI and I^2C when interfacing with multiple peripherals.

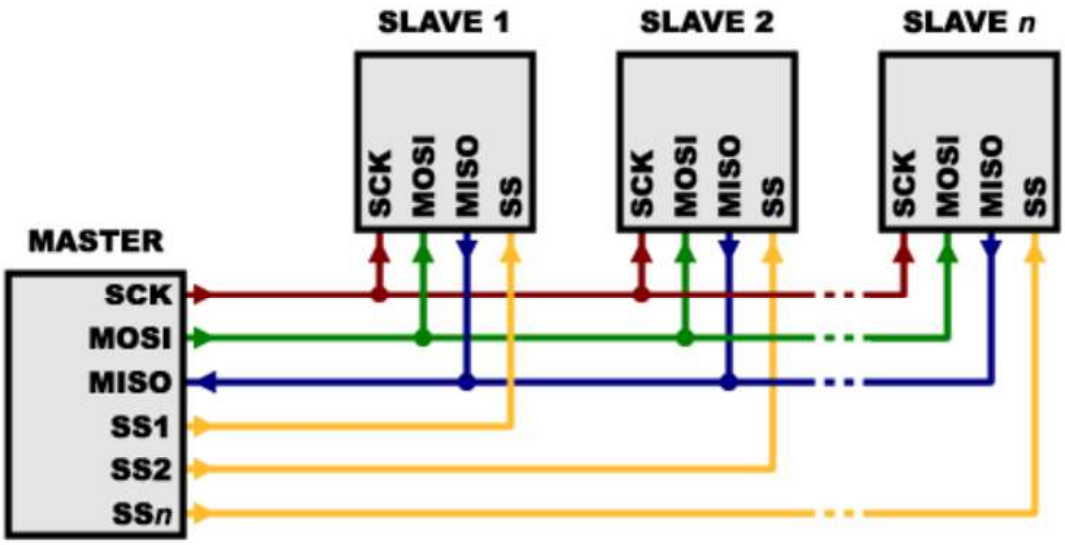


Figure 7: Connection diagram of SPI Master to multiple slave devices [20]

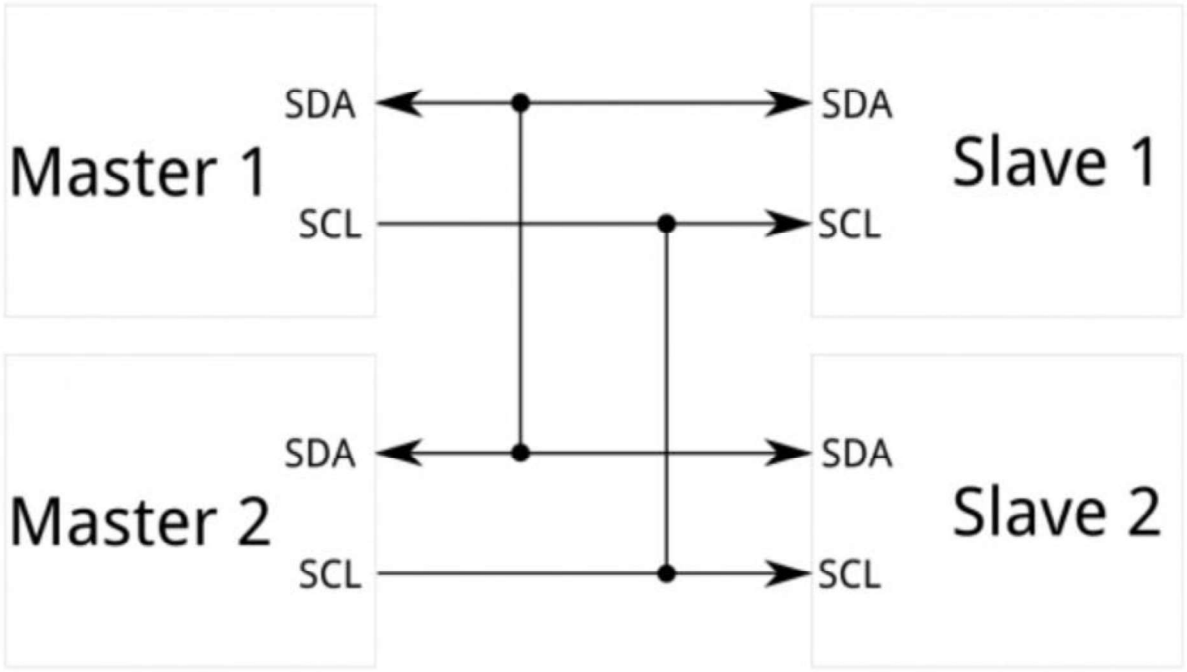


Figure 8: Connection diagram of I^2C Master to multiple slave devices (also supports multiple masters) [21]

While I^2C provides a more simplistic implementation, specifically when connecting to multiple peripherals, it is unable to provide maximum bit rates similar to SPI. The bit rate of a serial interface is largely determined by the shared clock frequency and size of the data words being transferred, but it can be shown that in most implementations the maximum data rate of SPI is on average an order of magnitude (10x) higher. This is in part due to the additional overhead incurred by I^2C from addressing data (device address sent before each data word). In conclusion, SPI is preferred when interfacing a small number of devices that require high data transfer speeds while I^2C is preferred when interfacing with a larger number of devices that are not reliant on high transfer speeds.

A disadvantage of both SPI and I^2C communications is that they are both prone to data corruption if transmitted over a long distance, as the transmitted clock signal is prone to interference and the devices rely on the signal being accurate for timing purposes. UART resolves this issue as it is an asynchronous communications protocol, indicating that it does not need a clock line between devices. Instead the devices are independently configured to communicate at the same baud rate which determines the data period for the communication. This allows for a much longer effective range than either SPI or I^2C making it suitable for communications between remote devices or across PCBs. This does require that the user set the baud rate correctly at the end device, or else the incoming communications will appear corrupted. Similar to I^2C, UART only requires 2 lines between devices, a transmission line and a receiving line.

In addition to these standardized protocols that are suitable for communication between a wide range of devices, there are some more simplistic communications protocols designed with specific use cases in mind. An example of one of these protocols is the U/D communications

used in some digital potentiometers in conjunction with rotary encoders. This simple protocol utilizes the pulses generated by a rotary encoder when rotated to increment or decrement the potentiometer wiper value depending on the direction of rotation. This drastically reduces the complexity of the communication but also limits its usefulness outside of the intended application.

With these communication methods considered, an appropriate protocol for each peripheral device used in the project can be selected. As previously mentioned, some digital potentiometers can utilize U/D communications to simplify interfacing with an encoder, however the project requires that the microcontroller also be able to interface with the potentiometers directly. This hybrid interfacing can be achieved using a switching circuit or multiplexor to toggle control of the potentiometer between the encoder and the MCU. An example of this implementation is shown in the figure below.

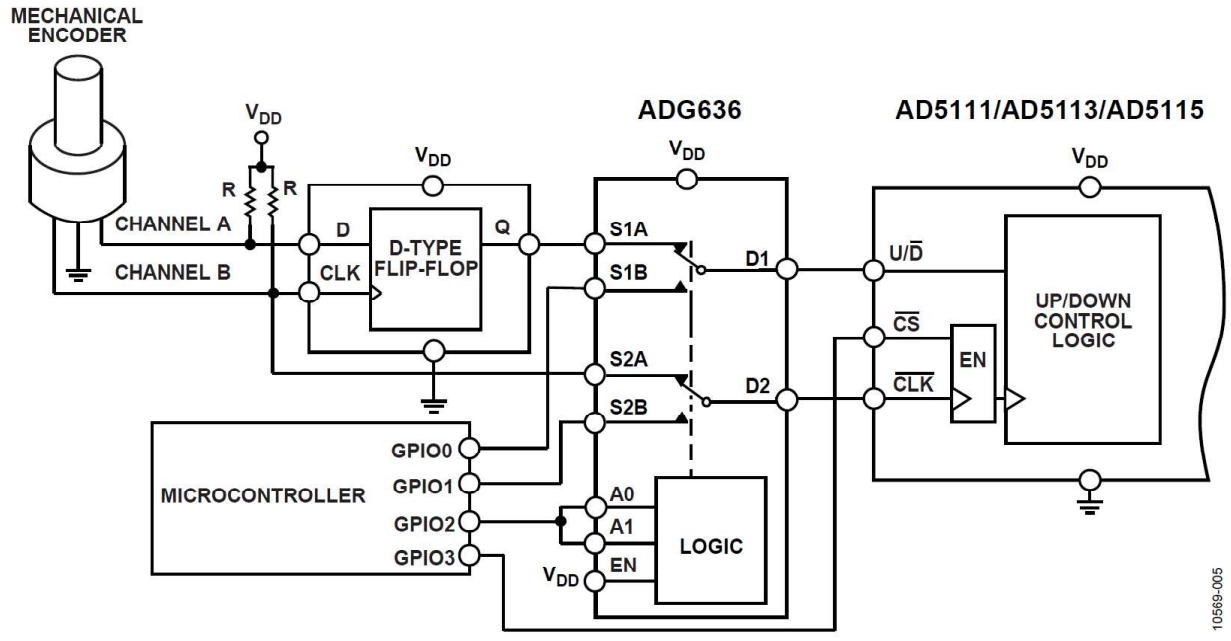


Figure 9: Example of hybrid interfacing with U/D digital potentiometer [22]

While this implementation would provide the hybrid interfacing required for the project, it is difficult to scale up to multiple devices. It also does not provide a simple mechanism for the microcontroller to stay aware of the current wiper position when the circuit is toggled to encoder control. This would require the use of an additional GPIO to sense an output voltage effected by the potentiometer to allow the MCU to interpret the wiper position at all times. Given that the circuit shown requires the use of an additional 4 GPIO pins, this would require 5 GPIO pins be connected per potentiometer, thus introducing scalability concerns.

Turning instead to serial communications standards, it can be shown that I^2C communications are most suitable for interfacing with the digital potentiometers. While the exact number potentiometers needed for the synthesizer control circuit is not yet determined, it is likely that it will be larger than what can be easily implemented in SPI, which would require substantially more circuitry to connect. Data rate is also not of high concern with the digital potentiometer interfacing as any updates can be transmitted in a few data words. Typical manual encoders are not designed to interface directly with I^2C, however by sending their signals to the MCU instead the possibility of hybrid interfacing can be realized.

For communications between the MCU and external ADC/DAC modules SPI is the preferred interface, as the number of peripherals is small and high data rate is of utmost priority. Additionally certain SPI ports such as the ones found on the Microchip PIC24FJ1024GB10 16-bit MCU offer additional modes for interfacing with hardware audio codecs or converters used specifically for audio applications, which may be of interest for the purposes of this project.[23] Similarly SPI is ideal for the communications between the MCU and the communications module which interfaces the synthesizer to the remote host, as high data rate is of high priority specifically during audio stream recording. Depending on the communication method chosen for

communications between the synthesizer and the remote host SPI may not be available for a given communication module or may not be necessary as the module provides an alternative method or mechanism for enabling high data rates.

2.5. Computer Networks

2.5.1. Wireless Peer-to-Peer Networking (SBG)

In order to properly utilize the remote host to control the synthesizer and provide a larger variety of functionality, the communications between the two systems must be strong and secure. To accomplish this, a peer-to-peer communications protocol will be utilized. In comparison to a mesh network communications protocol, which relies on multiple access nodes within the network to communicate amongst each other and pass information along the network, a peer-to-peer communications protocol only has two nodes who share a direct link with each other. In a situation where data is only being transferred between a remote host and a physical peripheral like a synthesizer, a mesh network would be clumsy and difficult to implement. In contrast, a direct connection is designed to operate at full capacity within these restrictions without the hassle of a mesh network.

To accomplish this, a variety of communications protocols were considered, but two stood out amongst the pack; Bluetooth 5.0 and 802.11 WiFi, specifically the WiFi Direct implementation. Both protocols have been found to be excellent peer-to-peer communications protocols and are similar as a result, but they both vary in a few ways. Bluetooth is primarily utilized as Bluetooth Low Energy, which can provide a low power, low throughput utilization for applications looking to save on power. However, Bluetooth still has usage as classic and high-speed variants which boast higher data rates, promising up to 3Mbps over-the-air speeds at a

moderate power consumption rate. The versatility of Bluetooth has allowed it to emerge as the most widely used peer-to-peer communications protocol.

On the other hand, 802.11 WiFi Direct utilizes the standard 802.11 WiFi protocol in a limited peer-to-peer network, allowing for the usage of standard WiFi modules to subvert the standard mesh network structure. As a result of utilizing the existing 802.11 WiFi protocol as a base, the standard WiFi data rates are present, with WiFi Direct boasting speeds up to 250 Mbps. While WiFi Direct appears to outpace Bluetooth, WiFi Direct draws more power than other peer-to-peer communications protocols as WiFi Direct draws up to ten times more power, up to 20 W, than standard Bluetooth communications. As a result of this, the debate between Bluetooth and WiFi Direct can be summarized as data rate vs power consumption. For the project, utilizing the minimum necessary data transfer rate would allow for the power consumption of the project to be limited to a healthy level, improving the general functionality of the project.

WiFi Direct can be implemented on a PIC24 board through the usage of a WiFi board like the ATWINC1500 WiFi breakout board. This breakout board will allow the user to access WiFi communications without needing to completely write TCP/IP connections from scratch. Instead allowing for compatibility with libraries to operate the communications protocols. The ATWINC15000 board operates on a state-based system, switching between different states of operation to allow for a simplification of the communications.

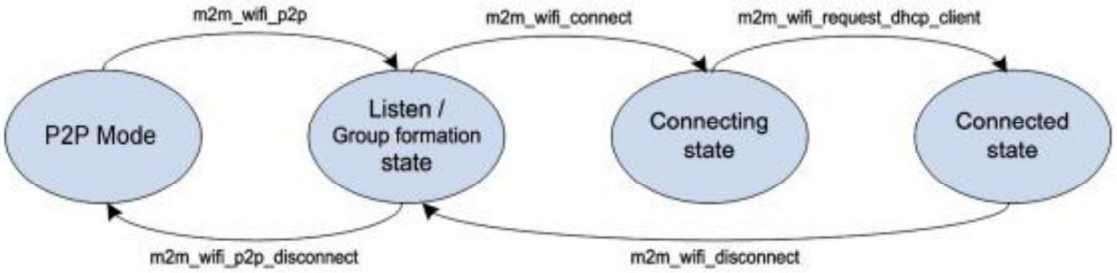


Figure 10: ATWINC State diagram [24]

These states allow the board to become discoverable to other peer-to-peer devices on a predefined listening channel, where the discoverer device will become the group owner of the connection. This connection can then be formed and used in the same way as any conventional connection. As easy as connecting is, disconnecting is also easy, allowing for the safe disconnect of the device without any threat or danger of data faults as a result of disconnecting.

Table 6: Power/Current Consumption of ANTWINC1500 for various device states [25]

Device State	Code Rate	Output power, dBm	Current Consumption [1]	
			IVBATT	IVDDIO
ON_Transmit	802.11b 1Mbps	17.5	268mA	22mA
	802.11b 11Mbps	18.5	264mA	22mA
	802.11g 6Mbps	17.5	269mA	22mA
	802.11g 54Mbps	16.0	266mA	22mA
	802.11n MCS 0	17.0	268mA	22mA
	802.11n MCS 7	14.5	265mA	22mA
ON_Receive	802.11b 1Mbps	N/A	61mA	22mA
	802.11b 11Mbps	N/A	61mA	22mA
	802.11g 6Mbps	N/A	61mA	22mA
	802.11g 54Mbps	N/A	61mA	22mA
	802.11n MCS 0	N/A	61mA	22mA
	802.11n MCS 7	N/A	61mA	22mA

The ATWINC 1500 board is designed to operate with a variety of MCUs with a variety of WiFi communications options. The ATWINC 1500 board consumes a large portion of power

supplied by the power supply to operate, but it manages to consume far less than the high-end estimates for other WiFi modules. The project will continue development and testing with the ATWINC 1500 breakout board as the focal point for the WiFi communications.

2.5.2. Remote Application Design (SBG)

Offering remote control over the synthesizer is a key functionality of the project. This idea is to be accomplished by utilizing an application run on an interchangeable remote device. This remote application will be able to be run on a variety of easily accessible and readily available computer devices. The host is designed to connect to the synthesizer using a peer-to-peer communications protocol across a short distance. To handle the transfer of data and management of advanced settings and options, the remote application is designed to be run locally and connecting directly with the synthesizer, thus allowing for short distance communications with the synthesizer.

The application will allow for the user to remotely control the synthesizer settings on a finer degree than the physical settings can provide. The ability to be more precise and specific allows for the application to more easily recreate specific sound settings that the user might wish to use. Along this line, additional functionality to allow users to save and load specific sound settings as ‘profiles’ is to be provided by the application. The application will also interface with the synthesizer to allow for the recording and playback of the synthesizer output. This application will provide the users with a variety of additional functionality to operate the synthesizer on a much higher and more precise level.

Design of the remote app will focus on implementing an interface like that of a standard digital synthesizer while being more discrete within its functionality. The remote app will be initially developed using JavaScript and HTML to create a user interface. The user interface

settings will be based off the potentiometers and switches that will be present on the physical synthesizer. This will allow the remote application to have the same tonal control over the synthesizer. The additional functionality will be present in several additional buttons to enable and activate the various extended functions the remote app will provide.

2.6. Embedded Systems

2.6.1. Systems Specifications (ADB)

Embedded systems are a key component in the development of the project, as it is primarily responsible for enabling the digital control and interfacing capabilities of the synthesizer. As seen in Figure 14 the Digital Control block consists of various embedded computing components and processes responsible for enabling the digital interfacing with the synthesizer controls and the signal processing of digital inputs and analog outputs for recording. The configuration and control of these devices are performed by the microprocessor block along with processing of input/output signals before forwarding them to their desired output. A few of these peripheral blocks such as the Synthesizer Controls I/O and the DAC may be ultimately realized as interfaces and modules internal to the microcontroller such as GPIO, serial ports, or PWM functionality, depending on the needs of the project. As such determining the interfaces and other specifications needed for the microprocessor to perform the required functions is key to the project and is dependent on the requirements and resulting implementations of the peripheral blocks.

As discussed in the Signal Processing section of the analysis the ADC block responsible for converting the analog output of the synth circuit into a digital signal to be sent to the remote host for recording must sample the incoming audio at a rate of at least 44.1 kHz with a resolution of at

least 24-bit to ensure a high-quality output. The on-board ADCs commonly found on most 16-bit microcontrollers are not capable of sampling at the required resolution, with most only capable of a maximum of 12-bit resolution. High-resolution ADCs are more commonly found in standalone packages, using serial communications with a central MCU to transmit data and receive configuration signals. For example, the MCP3561 from Microchip offers 24-bit Δ-Σ A/D conversion of a single differential input or two single-ended inputs at a maximum data rate of 153,600 samples per second and communicates with a MCU via SPI to receive control commands and send the digital output. The AD7768-1 from Analog Devices offers similar functionality with additional output filtering options to accommodate different latency and bandwidth requirements. Such devices offer the precision and throughput required for the project and use a serial interface commonly found on a wide variety of microprocessors. [26]

The communication module block allows the remote host application to communicate with the synthesizer by sending commands to the microcontroller and receiving data from it. As discussed in the communications analysis there are several potential options for communications between the synth and the remote host. Embedded Bluetooth implementations typically consist of a standalone module that provides the Bluetooth radio and interfaces with a microcontroller via UART for data exchange and control. Depending on the implementation the Bluetooth software stack may run on the module itself or on the host microcontroller for HCI implementations. These implementations typically allow for higher throughput than using the separate Bluetooth stack, and may be required for the purposes of this project. An example of such a module would be the BT860 from Laird Connectivity, which features the aforementioned HCI-UART interface and up to 3 Mbps over the air transfer speeds. [27]

Similarly, embedded WiFi modules exist in standalone packages to integrate wireless communications into an embedded system and can also function as a network controller with the TCP/IP software stack running on the module or as a link controller with the TCP/IP stack running on the host MCU. An advantage to embedded WiFi over embedded Bluetooth is that transfer speeds are not reliant on where the TCP/IP stack is running which allows for high throughput even with all-in-one network controller modules, reducing MCU requirements. These embedded WiFi modules typically use SPI for communication and data transfer with some offering an additional UART interface for debugging purposes. An example of one of these modules is the ATWINC15x0 series modules from Microchip which feature on-board TCP/IP stacks, SPI interfaces and 802.11 b/g/n wireless providing a maximum over-the-air throughput of up to 72 Mbps. [25]

Turning attention now towards wired options for remote host communication, USB enables wired interfacing with a wide range of devices and can be implemented into an embedded design relatively easily. While USB functionality can be implemented via a standalone controller certain microcontrollers feature said controller integrated into their design negating the need for additional hardware. This allows a USB connector to be wired directly to the MCU via a simple circuit to provide an external interface with the embedded system. The figure below shows an example of such a circuit for connecting a micro-USB connector to a host PIC24 MCU.

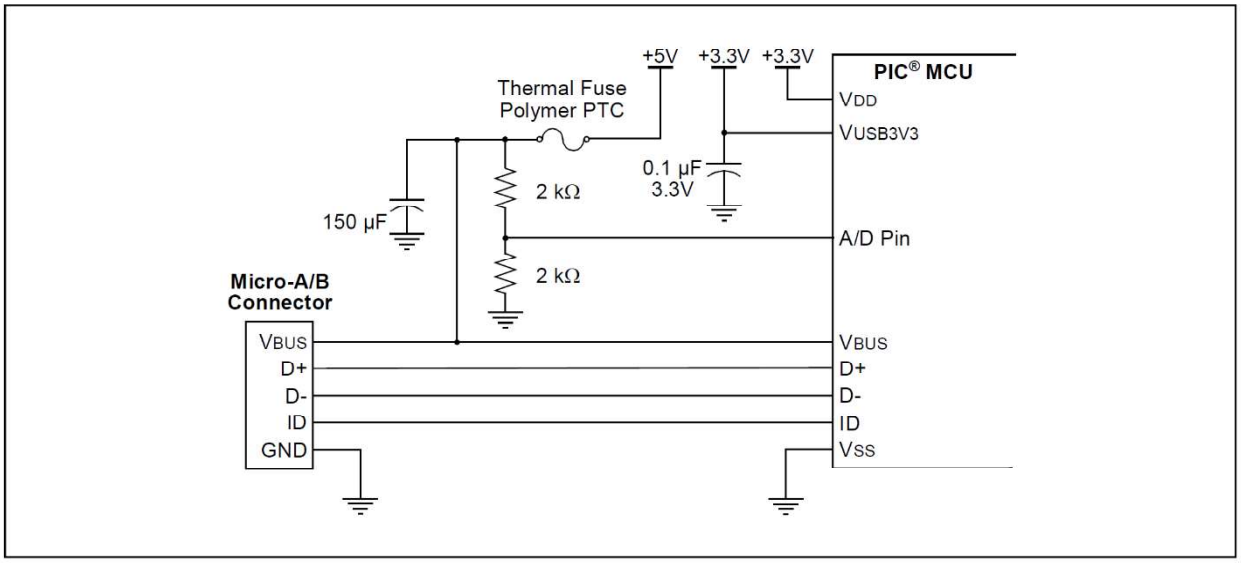


Figure 11: Host USB interface circuit example [23]

The maximum data rates over an embedded USB interface are dependent on the configuration of the controller and the maximum speeds allowed by it. For example the USB controller integrated into the PIC24FJ1024GB10 allows for both low and full speed modes offering maximum speeds of 1.5 Mbps and 12 Mbps respectively. [23] Referring back to the analysis found in the Communications section it can be shown that the three communications methods listed are capable of supporting the 1 Mbps data rate requirement of the ADC output stream, however it is yet to be determined if an HCI implementation of Bluetooth would require a more feature-rich MCU. Both WiFi and USB implementations previously mentioned offer data rates well in excess of this 1 Mbps requirement which may be desired in the event that data rate requirements are higher than anticipated. While USB does provide a more than sufficient data rate it also limits the ease of use of the final product somewhat by requiring a physical connection and could prevent the remote host from taking certain forms such as a mobile phone. Such considerations will need be watched closely and analyzed further for the purposes of this project.

The synthesizer controls I/O block enables communications between the MCU and the digital potentiometers used to control the synthesizer circuits. If a more simplistic MCU is used with limited interfacing options this block may require its own hardware module to enable the communications required for the project. However, given a more fully featured MCU many of the interfacing options are available directly on the controller itself, negating the need for additional hardware. By recalling the analysis of embedded communications protocols found in the Communications section the two most suitable interfacing options for the digital potentiometers are the U/D communication implementation and the I^2C implementation. As previously mentioned, designing an implementation similar to that found in Figure 9 require 5 GPIO lines per digital potentiometer. While this is somewhat scalable with more advanced MCUs the number of lines poses challenges in the wiring of these devices. Additionally, each potentiometer would require a 4-2 switching multiplexor to enable the hybrid interfacing required for the project, thus adding additional complexity.

In contrast to the U/D implementation, using I^2C to interface with the digital potentiometers would only require two leads for each I^2C port used, which can potentially support several dozen devices each. This along with the lack of high data rate requirements may make the I^2C implementation more desirable. While some more basic MCUs may not feature I^2C capabilities, many more fully featured versions will offer multiple ports built-in to the controller. For example, the PIC24FJ1024GB10 includes 3 I^2C modules on-board, each capable of either 7-bit or 10-bit addressing depending on the connected peripherals. This results in a possible 384 total slave devices for 7-bit addressing, or 3072 total slave devices for 10-bit, both of which will likely greatly exceed the number of digital potentiometers required for this project.

Given the requirements described, a microcontroller selected for central communication with these embedded peripherals must meet the following requirements:

- Multiple SPI interfaces for data transfer & communication with ADC & DAC
- Large number of GPIO (over 75) or multiple I^2C interfaces for DigiPot communication
- Various additional requirements for remote host communications
    - Bluetooth: UART interface, HCI-UART capabilities
    - WiFi: SPI interface, additional UART for debugging
    - USB: On-board USB controller

In addition to these interfacing requirements the MCU must feature a sufficiently fast instruction cycle time and sufficient program memory to store the program data required to perform the required functions, both of which have yet to be analyzed and determined. Given this analysis the PIC24FJ1024GB10 is a potential candidate for the chosen MCU, given its wide range of interfacing options and large program memory. While this selection may provide features that are not utilized during the project it meets the specifications previously determined in this analysis.

3. Engineering Requirements Specification

Marketing Requirements	Engineering Requirements	Justification
1	A/D and D/A conversion of audio signals should be performed at a minimum resolution of 24-bits.	24-bit conversion allows for dynamic range that matches or exceeds that of the human ear.
1	A/D and D/A conversion of audio signals should be performed at a minimum sample rate of 44.1 kHz.	44.1 kHz sampling captures all frequencies within the range of human hearing and is the lowest widely adopted sampling rate for music or full-range audio.
2	Sound profile options should be able to be saved and loaded as external files.	Being able to save files externally from the system and remote app will allow for the profile options to be easily transferable between uses of the product.
4	Communications involved in transporting recorded output must provide throughput of more than 1 Mbps.	Digital audio output stream with given resolution and sampling rate requirements produces a bit rate of about 1 Mbps.
4	Recorded output should be packaged in an uncompressed audio format that is playable via media player software.	Uncompressed audio is desirable for professional audio applications. Future playback of recording is enabled through media player software on remote host.
1	Tone control filtering should have a gain deviation of less than 3 dB at equal amplifier gain on each band.	Having the controls cause a consistent effect on the gain of each band is desirable for ease of use.
2	The device should support at least 2 industry standard sound effects (distortion, reverb, etc.).	Without effects being applied the device is simply a complicated audio amplifier and does not provide any music creation functionality.
1,3	The frequency response must be stable on the input range of 20Hz – 20kHz.	This ensures that the device will have a stable and controlled response over the expected input range.
2,5	Any tunable aspects of synthesizer circuits must provide manual control and digital control from MCU.	This enables the remote control and profile save/load functionality intended for the design.
5	The remote application should be able to be hosted on a common consumer device.	Remote control on remote application does not require the purchase of a dedicated device.

Marketing Requirements

1. The device should produce high-fidelity sound in the form of analog waveforms within the audible frequency spectrum.
2. The device should have customizable & easily reproducible sound profiles.
3. The device should accept a variety of external inputs.
4. The device should have the ability to record output for future playback.
5. The device should be able to be controlled from a remote source.

4. Engineering Standards Specification

	Standard	Usage
Communications	Bluetooth, 802.11 WiFi, SPI, I^2C, USB, MIDI	Remote Host Communication, Embedded Peripheral Interfacing, Digital Instrument Input
Data Formats	MIDI, PCM	Digital Inst. Input, Digital Audio Recording
Programming Languages	C, HTML, Javascript	Digital Control Programming, Remote Application
Connector Standards	USB, ¼" TRS, USB-MIDI, RCA	Analog/Digital Input/Output

5. Accepted Technical Design

5.1. Hardware Design

5.1.1. Level 0 Functional Decomposition

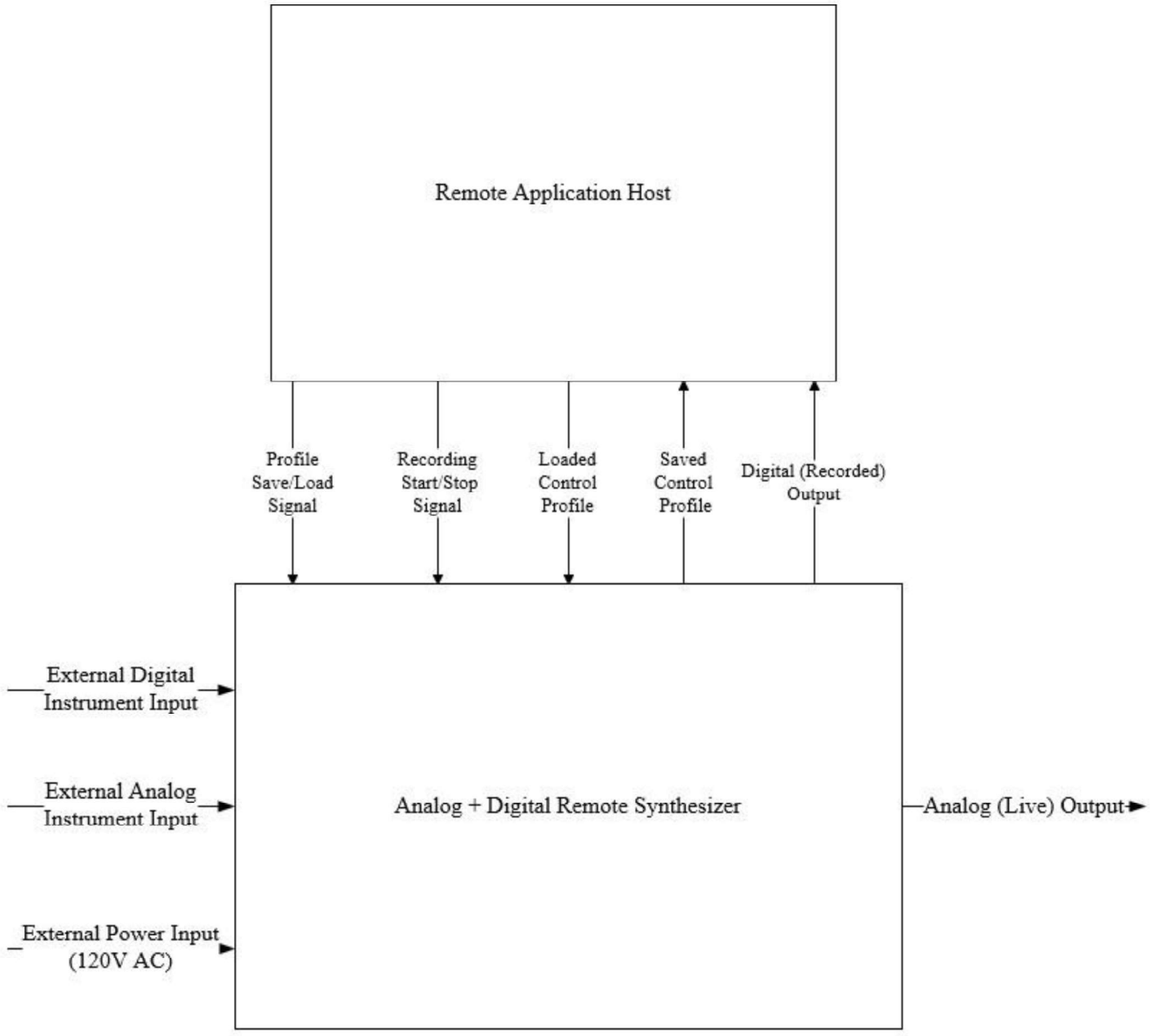


Figure 12: Level 0 block diagram of Analog + Digital Remote Synthesizer

Table 7: Level 0 functional requirements of Analog + Digital Remote Synthesizer

Module	Analog + Digital Remote Synthesizer
Designers	Adam Brunner, Andrew Cihon-Scott, Scott Grisso, Linus Wright
Inputs	External Digital Instrument Input, External Analog Instrument Input, External Power Input (120V AC), Profile Save/Load Signal, Recording Start/Stop Signal, Loaded Control Profile
Outputs	Analog (Live) Output, Digital (Recorded) Output, Saved Control Profile
Description	This system is an Analog Synthesizer that is capable of being controlled by digital settings and inputs. Most synthesizers are either entirely analog and lack the repeatability of digital solutions, or entirely digital and do not produce the sound quality that analog synthesizers provide. The synthesizer will be mostly tuned with digital potentiometers. The connections between the modules will be read by sensors and saved to aid in sound reproduction.

5.1.2. Level 1 Functional Decomposition

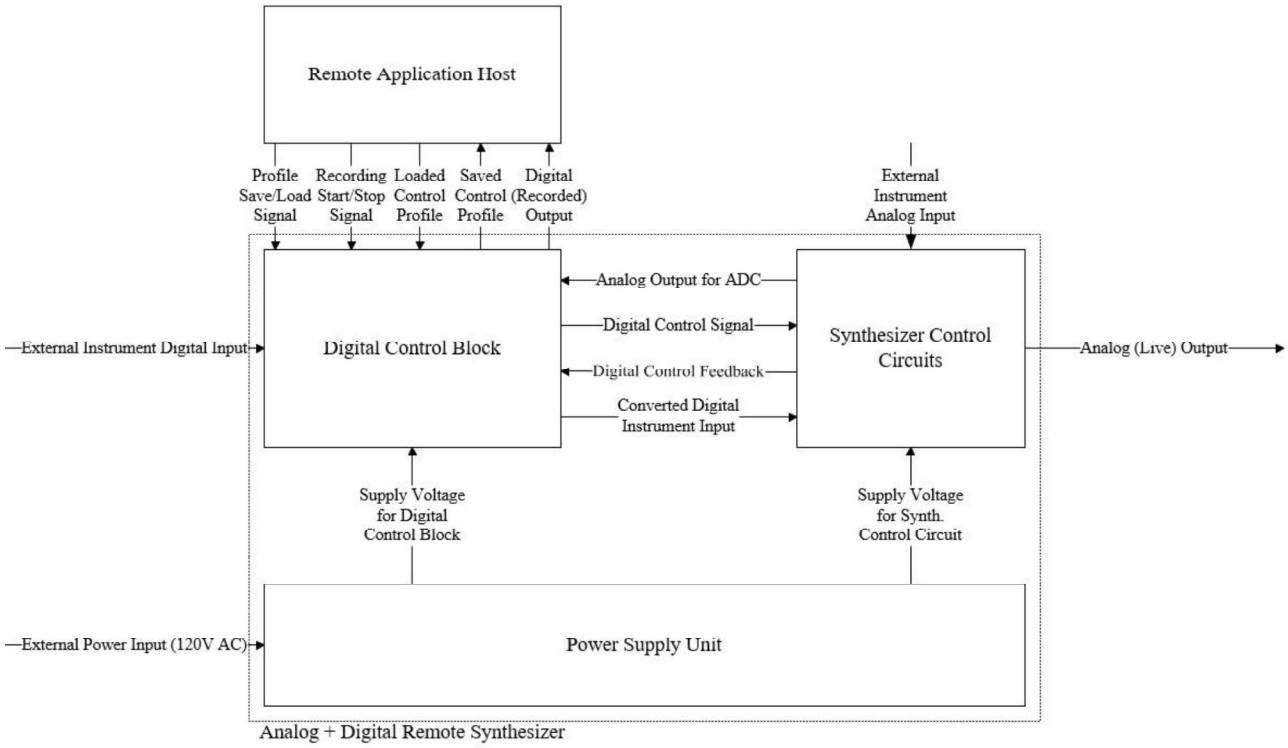


Figure 13: Level 1 block diagram of Analog + Digital Remote Synthesizer

Table 8: Level 1 functional requirements of Digital Control Block

Module	Digital Control Block
Designer	Adam Brunner
Inputs	External Instrument Digital Input, Profile Save/Load Signal, Recording Start/Stop Signal, Loaded Control Profile, Analog Output for ADC, Digital Control Feedback, Supply Voltage
Outputs	Saved Control Profile, Digital (Recorded) Output, Digital Control Signal, Converted Digital Instrument Input
Description	The digital control block contains components responsible for digital signal processing of external digital inputs, D/A conversion of digital input, A/D conversion of analog output, communication with the remote application host, and setting / monitoring of control signals for digital components in synth. control circuit block.

Table 9: Level 1 functional requirements of Power Supply Unit

Module	Power Supply Unit
Designer	Andrew Cihon-Scott, Linus Wright
Inputs	External Power Input (120V AC)
Outputs	Various DC Supply Voltages
Description	This unit is meant to convert the wall power coming into it into stable DC outputs needed in the digital control block and the synthesizer control circuits. The input wall voltage is expected to be 120V AC and Voltage outputs of 5V and 3.3V DC will be needed.

Table 10: Level 1 functional requirements of Remote Application Host

Module	Remote Application Host
Designer	Scott Grisso
Inputs	Saved Control Profile, Digital (Recorded) Output
Outputs	Profile Save/Load Signal, Recording Start/Stop Signal, Loaded Control Profile
Description	The Remote Application is responsible for providing instructions to the Digital Control Processes for the implementation of associated control interactions.

Table 11: Level 1 functional requirements of Synthesizer Control Circuits

Module	Synthesizer Control Circuits
Designer	Andrew Cihon-Scott, Linus Wright
Inputs	Analog Music Signal, Digitally Converted Music Signal, Voltage Oscillator
Outputs	Analog Waveform
Description	The Synthesizer Control Circuitry accepts inputs from the instrument, the DAC, or the voltage oscillator, and then uses analog electronics to shape and process the sound. The output is then a voltage signal that corresponds to the final tone of the music. The circuit is responsible for giving the notes the tone characteristic to a synthesizer.

5.1.3. Level 2 Functional Decomposition

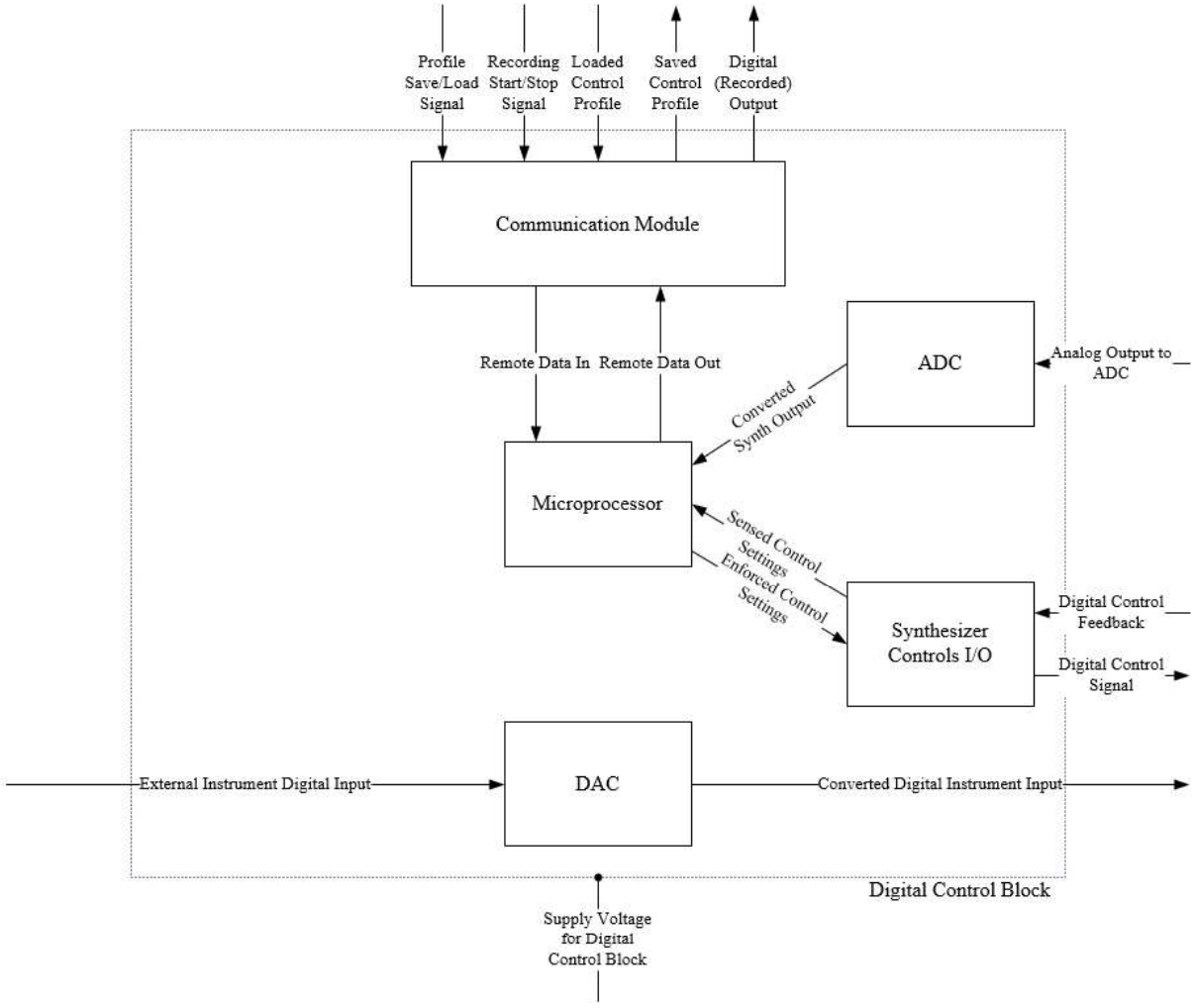


Figure 14: Level 2 block diagram of Digital Control Block

Table 12: Level 2 functional requirements of Analog-to-Digital Converter

Module	Analog-to-Digital Converter
Designer	Adam Brunner
Inputs	Synthesizer Circuit Output, Supply Voltage
Outputs	Digitized Synthesizer Circuit Output
Description	Module is responsible for the conversion of the analog output of the synthesizer circuit into a digital form to be recorded and stored on the remote application host. Sampling rate must be at minimum 44.1 kHz and feature at least 20-bit precision to produce the desired high-quality output. May exist as an on-board peripheral on the Microprocessor.

Table 13: Level 2 functional requirements of Microprocessor

Module	Microprocessor
Designer	Adam Brunner
Inputs	Converted Synth Output, Sensed Control Settings, Remote Data In, Supply Voltage
Outputs	Enforced Control Settings, Remote Data Out
Description	Microprocessor will host the code needed to interface with the other embedded components and route incoming/outgoing data to the proper modules. May also feature some of the other modules as on-board peripherals.

Table 14: Level 2 functional requirements of Synthesizer Controls I/O

Module	Synthesizer Controls I/O
Designer	Adam Brunner
Inputs	Digital Control Feedback, Enforced Control Settings, Supply Voltage
Outputs	Sensed Control Settings, Digital Control Signal
Description	Module is responsible for monitoring and adjusting the states of the digital controls implemented within the Synthesizer Controls block. Will likely be a realized as an on-board communication module built into the microprocessor.

Table 15: Level 2 functional requirements of Communication Module

Module	Communication Module
Designer	Adam Brunner
Inputs	Remote Data Out, Loaded Control Profile, Profile Save/Load Signal, Recording Start/Stop Signal, Supply Voltage
Outputs	Remote Data In, Saved Control Profile, Digital (Recorded) Output
Description	Module is responsible for enabling communication with remote application host to transmit required data, including initiating save/load commands, sending & receiving of control profiles, initiating/halting recording of output and the streamed/recorded output of the Synthesizer Circuit.

Table 16: Level 2 functional requirements of Digital-to-Analog Converter

Module	Digital-to-Analog Converter
Designer	Adam Brunner
Inputs	External Instrument Digital Input, Supply Voltage
Outputs	Converted Ext. Digital Input
Description	Module is responsible for converting digital input from an external instrument digital interface and processing it into an analog signal to be sent to the Synthesizer Controls circuit. May be realizable using a standalone DAC or via PWM functionality found on the microcontroller. Sampling and precision constraints for ADC also apply here for the inverse process.

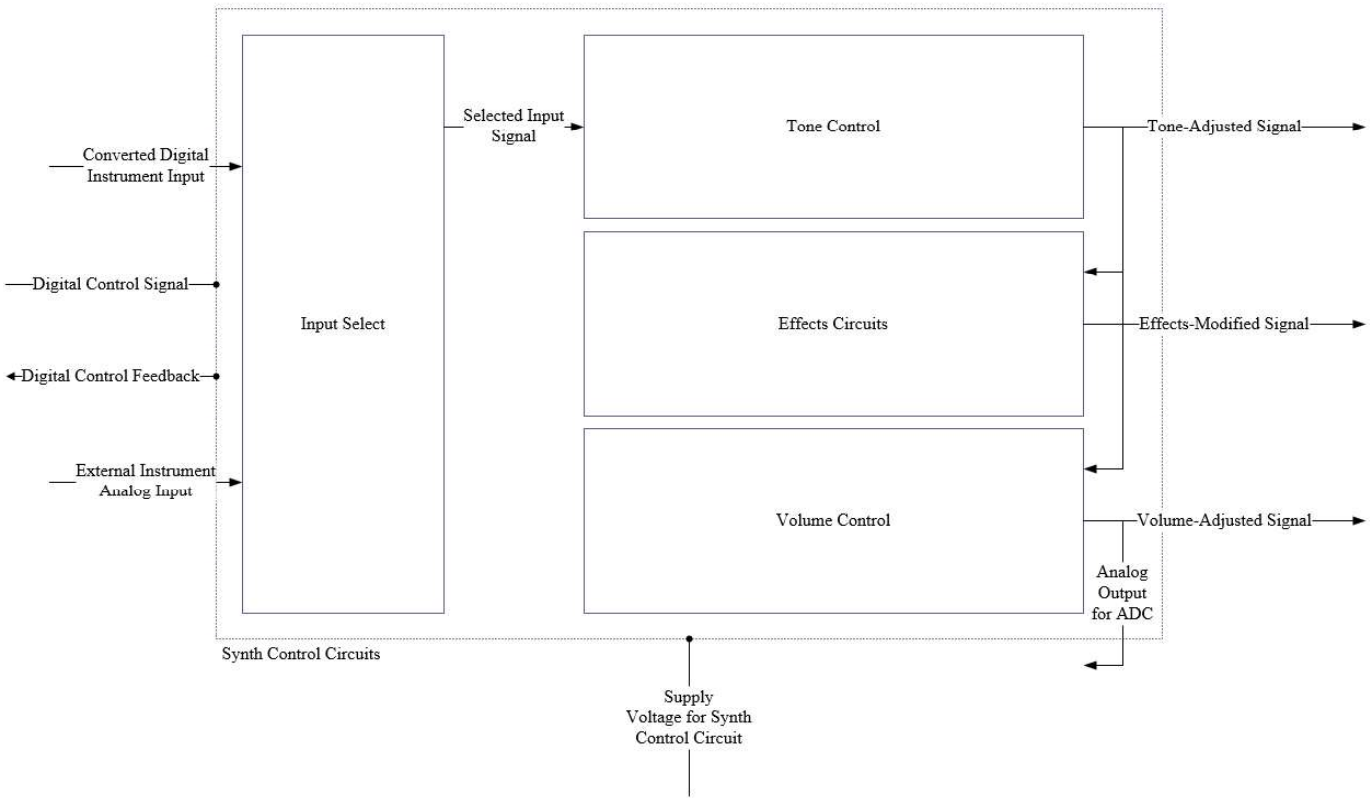


Figure 15: Level 2 block diagram of Synthesizer Control Circuits

Table 17: Level 2 functional requirements of Input Select

Module	Input Select
Designer	Andrew Cihon-Scott
Inputs	Voltage Oscillator, DAC Signal, Analog Instrument Signal
Outputs	Voltage Signal
Description	This circuit takes an input from the instrument, the DAC, or the voltage oscillator, and chooses which one to send to the rest of the synthesizer circuitry.

Table 18: Level 2 functional requirements of Tone Control

Module	Tone Control
Designer	Linus Wright
Inputs	Modulated Input Signal (V)
Outputs	Tone Adjusted Signal (V)
Description	This control circuit rebalances the input signal by amplifying different bands of the signal. These bands are split into a 0-200Hz band, a 200-4000Hz band, and a 4000-20000 Hz band. Each band is amplified to different controlled levels and then added back together to form the output signal.

Table 19: Level 2 functional requirements of Effects Circuits

Module	Effects Circuits
Designer	Andrew Cihon-Scott, Linus Wright
Inputs	Voltage from Input Select
Outputs	Voltage
Description	These circuits take their input voltage waveforms and modify them to provide a change in the perceived quality of the sound. The reverb creates a delayed version of the signal imposed upon itself, while the distortion introduces clipping and harmonics to the signal.

Table 20: Level 2 functional requirements of Volume Control

Module	Volume Control
Designer	Andrew Cihon-Scott
Inputs	Voltage from Tone Control
Outputs	Voltage Signal
Description	This circuit is a simple amplifier used to control the output volume of the system.

5.1.4. Level 3 Functional Decomposition

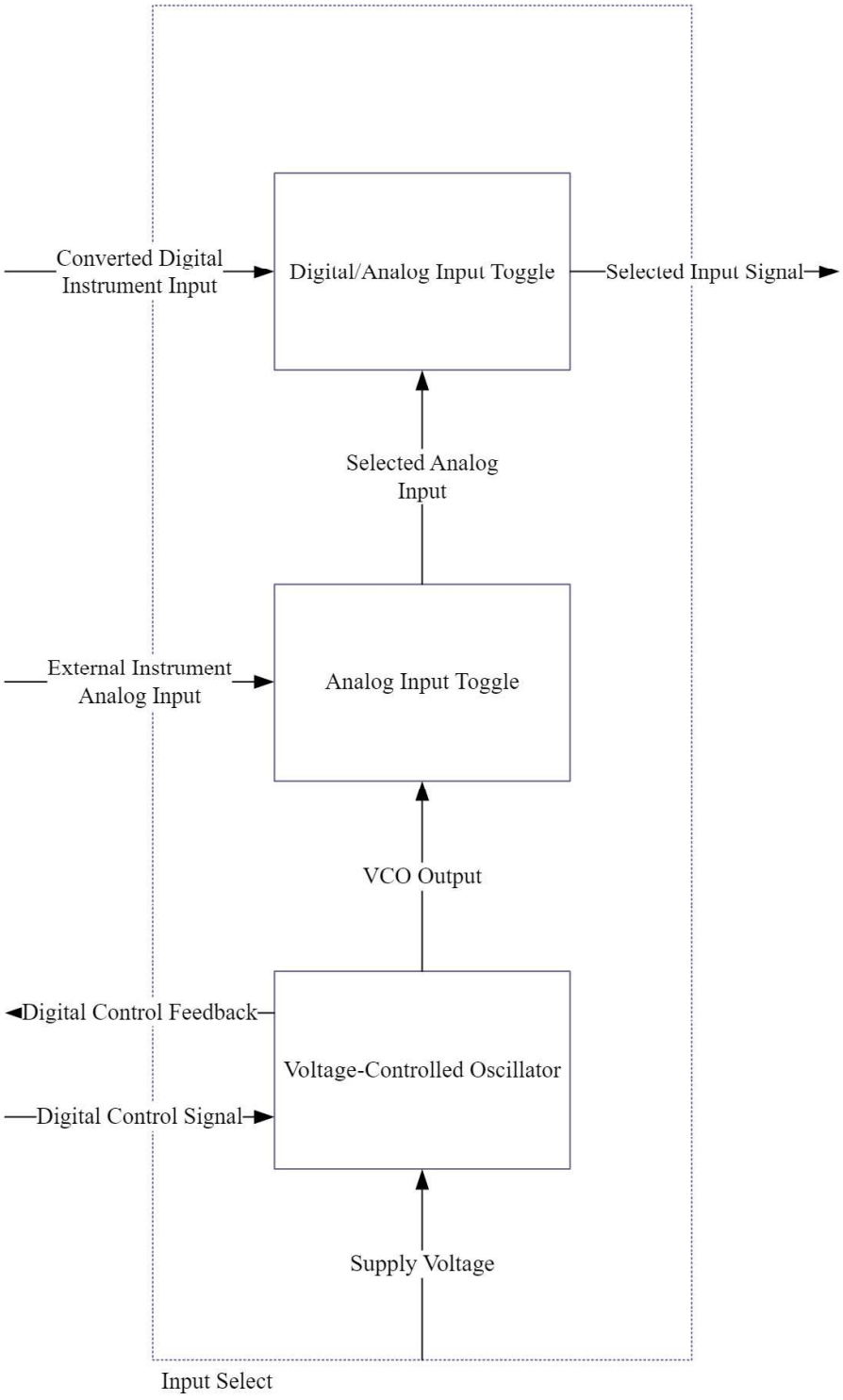


Figure 16: Level 3 block diagram of Input Select

Table 21: Level 3 functional requirements of Digital/Analog Input Toggle

Module	Digital/Analog Input Toggle
Designer	Andrew Cihon-Scott
Inputs	DAC Instrument Input
Outputs	Voltage Signal
Description	The input toggle circuit selects between the multiple sources of instrumental input available and passes their signal to the effect's circuitry.

Table 22: Level 3 functional requirements of Analog Input Toggle

Module	Analog Input Toggle
Designer	Andrew Cihon-Scott
Inputs	Voltage Oscillator, Analog Instrument Input
Outputs	Voltage Signal
Description	The input toggle circuit selects between the multiple sources of instrumental input available and passes their signal to the effect's circuitry.

Table 23: Level 3 functional requirements of Voltage Controlled Oscillator

Module	Voltage Controlled Oscillator
Designer	Andrew Cihon-Scott
Inputs	Frequency Control Signal
Outputs	Voltage Output
Description	The voltage-controlled oscillator can generate square, triangle, or sine waves at arbitrary frequencies. These waves are one of the sources available to the input select.

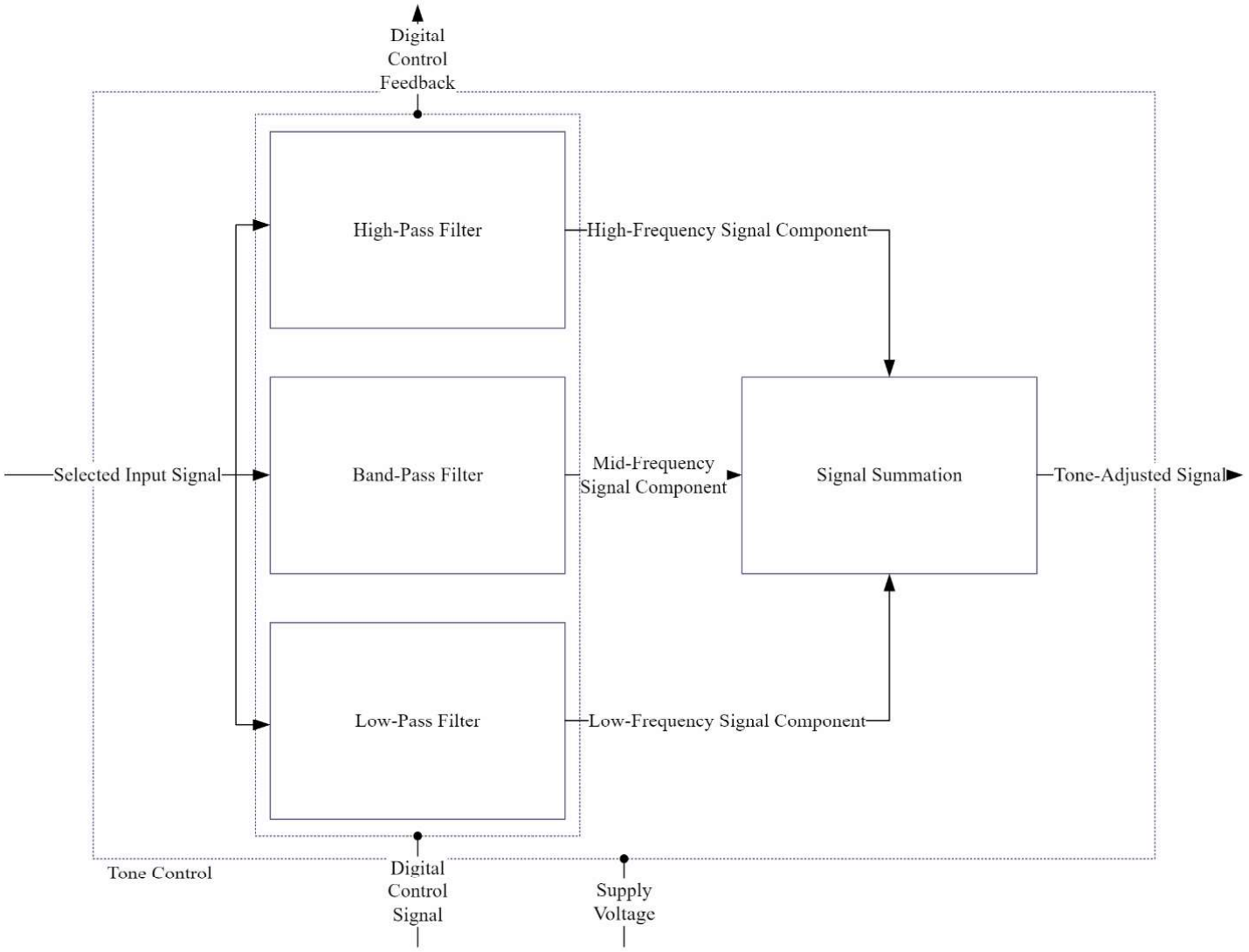


Figure 17: Level 3 block diagram of Tone Control

Table 24: Level 3 functional requirements of High-Pass Filter

Module	High-Pass Filter
Designer	Linus Wright
Inputs	Effects Modified Signal
Outputs	High Frequency Input Band
Description	This sub-circuit is a high pass filter with a cutoff frequency at 4000 Hz. Unity gain is necessary for convenient conditioning of the signal.

Table 25: Level 3 functional requirements of Band-Pass Filter

Module	Band-Pass Filter
Designer	Linus Wright
Inputs	Effects Modified Signal
Outputs	Mid Frequency Input Band
Description	This sub-circuit is a band pass filter with a cutoff frequency at 200 Hz and 4000 Hz. Unity gain is necessary for convenient conditioning of the signal.

Table 26: Level 3 functional requirements of Low-Pass Filter

Module	Low-Pass Filter
Designer	Linus Wright
Inputs	Effects Modified Signal
Outputs	Low Frequency Input Band
Description	This sub-circuit is a low pass filter with a cutoff frequency at 200 Hz. Unity gain is necessary for convenient conditioning of the signal.

Table 27: Level 3 functional requirements of Signal Summation

Module	Signal Summation
Designer	Linus Wright
Inputs	High, Mid, and Low Frequency Input Bands, Digital Control Signals
Outputs	Tone-Adjusted Signal
Description	The Signal Summation circuit will take each of the signals, amplify each input using a simple amplifier, and then use an Adder amplifier to recombine all of the now amplified signals.

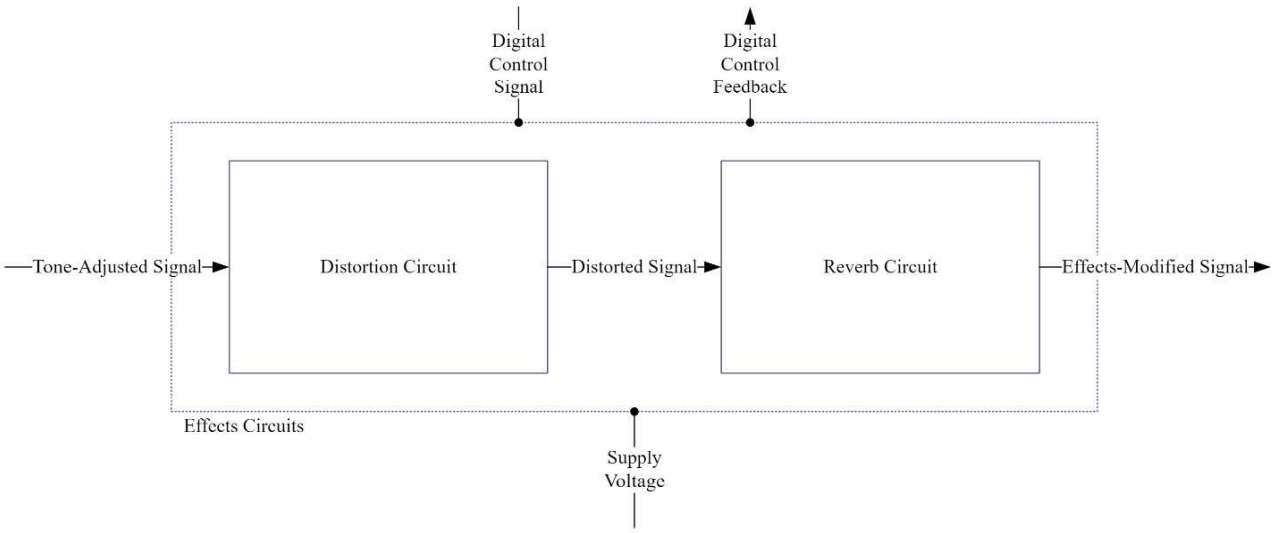


Figure 18: Level 3 block diagram of Effects Circuits

Table 28: Level 3 functional requirements of Distortion Circuit

Module	Distortion Circuit
Designer	Linus Wright
Inputs	Selected Input Signal, Digital Control Signal
Outputs	Distorted Signal
Description	This circuit takes the selected input and warps it using a variety of different methods. Based on the input from the digital control signals, the Hard Distortion, Soft Distortion, Rail Clipping, or Bypass Line will be implemented. The bypass connects the input of the circuit directly to the output so that this effect can be turned off.

Table 29: Level 3 Functional requirements of Reverb Circuit

Module	Reverb Circuit
Designer	Andrew Cihon-Scott
Inputs	Voltage Signal
Outputs	Voltage Signal
Description	The reverb circuit takes an analog input voltage and passes it to the output. The output signal also contains a time delayed version of the input superimposed on itself.

5.1.5. Hardware Schematics

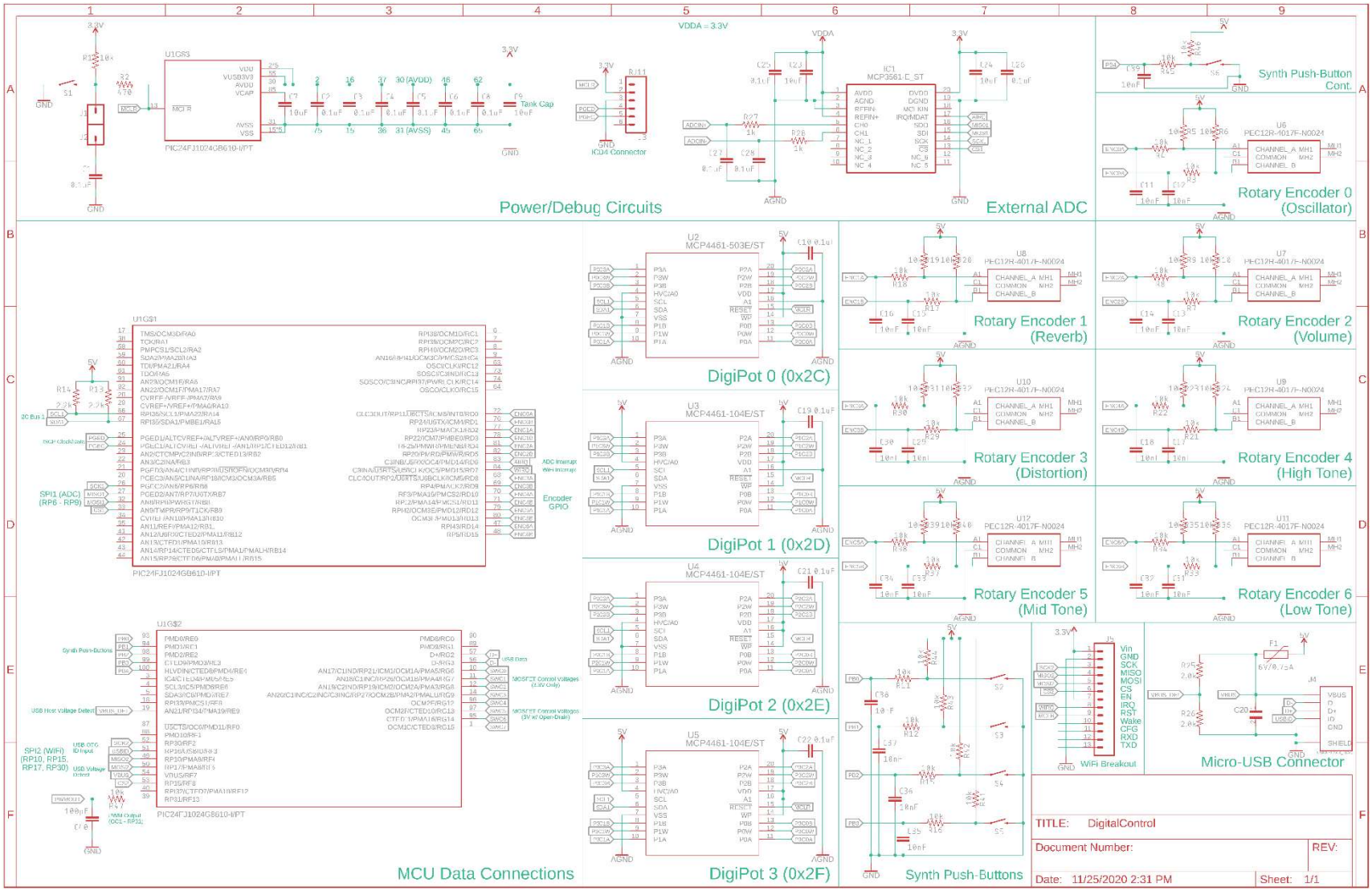


Figure 19: EagleCAD Schematic of Digital Control Subsystem MCU Data Connections (ADB)

Table 30: Parts List with Reference Designators and Part Descriptions for Digital Control Schematic (ADB)

Ref. Des.	Value	Device	Package	Description
C1	0.1uF	0.1UF-0603-25V-5%	0603	0.1µF ceramic capacitors
C2	0.1uF	0.1UF-0603-25V-5%	0603	0.1µF ceramic capacitors
C3	0.1uF	0.1UF-0603-25V-5%	0603	0.1µF ceramic capacitors
C4	0.1uF	0.1UF-0603-25V-5%	0603	0.1µF ceramic capacitors
C5	0.1uF	0.1UF-0603-25V-5%	0603	0.1µF ceramic capacitors
C6	0.1uF	0.1UF-0603-25V-5%	0603	0.1µF ceramic capacitors
C7	10uF	10UF-0805-25V-10%	0805	10.0µF ceramic capacitors
C8	0.1uF	0.1UF-0603-25V-5%	0603	0.1µF ceramic capacitors
C9	10uF	10UF-0805-25V-10%	0805	10.0µF ceramic capacitors
C10	0.1uF	0.1UF-0603-25V-5%	0603	0.1µF ceramic capacitors
C11	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C12	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C13	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C14	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C15	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C16	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C17	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C18	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C19	0.1uF	0.1UF-0603-25V-5%	0603	0.1µF ceramic capacitors
C20	150uF	GRM31CR60J157ME11L	1206	CAPACITOR, MLCC, X5R, 150UF, 6.3V, 1206
C21	0.1uF	0.1UF-0603-25V-5%	0603	0.1µF ceramic capacitors
C22	0.1uF	0.1UF-0603-25V-5%	0603	0.1µF ceramic capacitors
C23	10uF	10UF-0603-6.3V-20%	0603	10.0µF ceramic capacitors
C24	10uF	10UF-0603-6.3V-20%	0603	10.0µF ceramic capacitors
C25	0.1uF	0.1UF-0603-25V-(+80/-20%)	0603	0.1µF ceramic capacitors
C26	0.1uF	0.1UF-0603-25V-(+80/-20%)	0603	0.1µF ceramic capacitors

C27	0.1uF	0.1UF-0603-25V-(+80/-20%)	0603	0.1µF ceramic capacitors
C28	0.1uF	0.1UF-0603-25V-(+80/-20%)	0603	0.1µF ceramic capacitors
C29	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C30	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C31	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C32	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C33	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C34	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C35	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C36	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C37	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C38	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C39	10nF	10NF-0603-50V-10%	0603	0.01uF/10nF/10,000pF ceramic capacitors
C40	100pF	100PF-0603-50V-5%	0603	100pF/0.1nF ceramic capacitors
F1	6V/0.75A	PPTC_0.75A	1206	Resettable Fuse PPTC
IC1		MCP3561-E_ST	TSSOP65P640X120-20N	Microchip Technology MCP3561 24-bit Σ-Δ ADC TSSOP-20
J1		CONN_01	1X01	Single connection point. Often used as Generic Header-pin footprint for 0.1 inch spaced/style header connections
J2		CONN_01	1X01	Single connection point. Often used as Generic Header-pin footprint for 0.1 inch spaced/style header connections
J3	RJ11	RJ11-6	RJ11-6	RJ11 Jack - 6 pin
J4		USB3070-XX-X_REVE	USB3070-XX-X_REVE	USB Type Micro-B Connector
J5		CONN_13	1X13	Multi connection point. Often used as Generic Header-pin footprint for 0.1 inch spaced/style header connections
R1	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R2	470	470OHM-0603-1/10W-1%	0603	470Ω resistor

R3	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R4	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R5	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R6	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R7	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R8	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R9	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R10	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R11	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R12	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R13	2.2k	2.2KOHM-0603-1/10W-1%	0603	2.2kΩ resistor
R14	2.2k	2.2KOHM-0603-1/10W-1%	0603	2.2kΩ resistor
R15	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R16	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R17	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R18	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R19	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R20	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R21	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R22	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R23	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R24	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R25	2.0k	2.0KOHM-0603-1/10W-5%	0603	2kΩ resistor
R26	2.0k	2.0KOHM-0603-1/10W-5%	0603	2kΩ resistor
R27	1k	1KOHM-0603-1/10W-1%	0603	1kΩ resistor
R28	1k	1KOHM-0603-1/10W-1%	0603	1kΩ resistor
R29	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R30	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R31	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R32	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R33	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R34	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor

R35	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R36	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R37	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R38	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R39	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R40	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R41	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R42	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R43	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R44	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R45	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R46	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
R47	10k	10KOHM-0603-1/10W-1%	0603	10kΩ resistor
S1		MOMENTARY-SWITCH-SPST-PTH-6.0MM-KIT	TACTILE_SWITCH_PTH_6.0MM_KIT	Momentary Switch (Pushbutton) - SPST
S2		MOMENTARY-SWITCH-SPST-PTH-6.0MM-KIT	TACTILE_SWITCH_PTH_6.0MM_KIT	Momentary Switch (Pushbutton) - SPST
S3		MOMENTARY-SWITCH-SPST-PTH-6.0MM-KIT	TACTILE_SWITCH_PTH_6.0MM_KIT	Momentary Switch (Pushbutton) - SPST
S4		MOMENTARY-SWITCH-SPST-PTH-6.0MM-KIT	TACTILE_SWITCH_PTH_6.0MM_KIT	Momentary Switch (Pushbutton) - SPST
S5		MOMENTARY-SWITCH-SPST-PTH-6.0MM-KIT	TACTILE_SWITCH_PTH_6.0MM_KIT	Momentary Switch (Pushbutton) - SPST
S6		MOMENTARY-SWITCH-SPST-PTH-6.0MM-KIT	TACTILE_SWITCH_PTH_6.0MM_KIT	Momentary Switch (Pushbutton) - SPST
U1	PIC24FJ1024GB610-I/PT	PIC24FJ1024GB610-I/PT	QFP40P1400X1400X120-100N	PIC PIC® 24F Microcontroller IC 16-Bit 32MHz 1MB (341.5K x 24) FLASH 100-TQFP (12x12)
U2	MCP4461-503E/ST	MCP4461-503E/ST	TSSOP20_MC_MCH	Microchip Technologies MCP4461 4-Channel 257 Tap Digital Potentiometer (50 kΩ)
U3	MCP4461-104E/ST	MCP4461-104E/ST	TSSOP20_MC_MCH	Microchip Technologies MCP4461 4-Channel 257 Tap Digital Potentiometer (100 kΩ)

U4	MCP4461-104E/ST	MCP4461-104E/ST	TSSOP20_MC_MCH	Microchip Technologies MCP4461 4-Channel 257 Tap Digital Potentiometer (100 kΩ)
U5	MCP4461-104E/ST	MCP4461-104E/ST	TSSOP20_MC_MCH	Microchip Technologies MCP4461 4-Channel 257 Tap Digital Potentiometer (100 kΩ)
U6	PEC12R-4017F-N0024	PEC12R-4017F-N0024	PEC12R4017FN0024	Bourns 24 Pulse Incremental Mechanical Rotary Encoder with a 6 mm Flat Shaft (Not Indexed), Through Hole
U7	PEC12R-4017F-N0024	PEC12R-4017F-N0024	PEC12R4017FN0024	Bourns 24 Pulse Incremental Mechanical Rotary Encoder with a 6 mm Flat Shaft (Not Indexed), Through Hole
U8	PEC12R-4017F-N0024	PEC12R-4017F-N0024	PEC12R4017FN0024	Bourns 24 Pulse Incremental Mechanical Rotary Encoder with a 6 mm Flat Shaft (Not Indexed), Through Hole
U9	PEC12R-4017F-N0024	PEC12R-4017F-N0024	PEC12R4017FN0024	Bourns 24 Pulse Incremental Mechanical Rotary Encoder with a 6 mm Flat Shaft (Not Indexed), Through Hole
U10	PEC12R-4017F-N0024	PEC12R-4017F-N0024	PEC12R4017FN0024	Bourns 24 Pulse Incremental Mechanical Rotary Encoder with a 6 mm Flat Shaft (Not Indexed), Through Hole
U11	PEC12R-4017F-N0024	PEC12R-4017F-N0024	PEC12R4017FN0024	Bourns 24 Pulse Incremental Mechanical Rotary Encoder with a 6 mm Flat Shaft (Not Indexed), Through Hole
U12	PEC12R-4017F-N0024	PEC12R-4017F-N0024	PEC12R4017FN0024	Bourns 24 Pulse Incremental Mechanical Rotary Encoder with a 6 mm Flat Shaft (Not Indexed), Through Hole

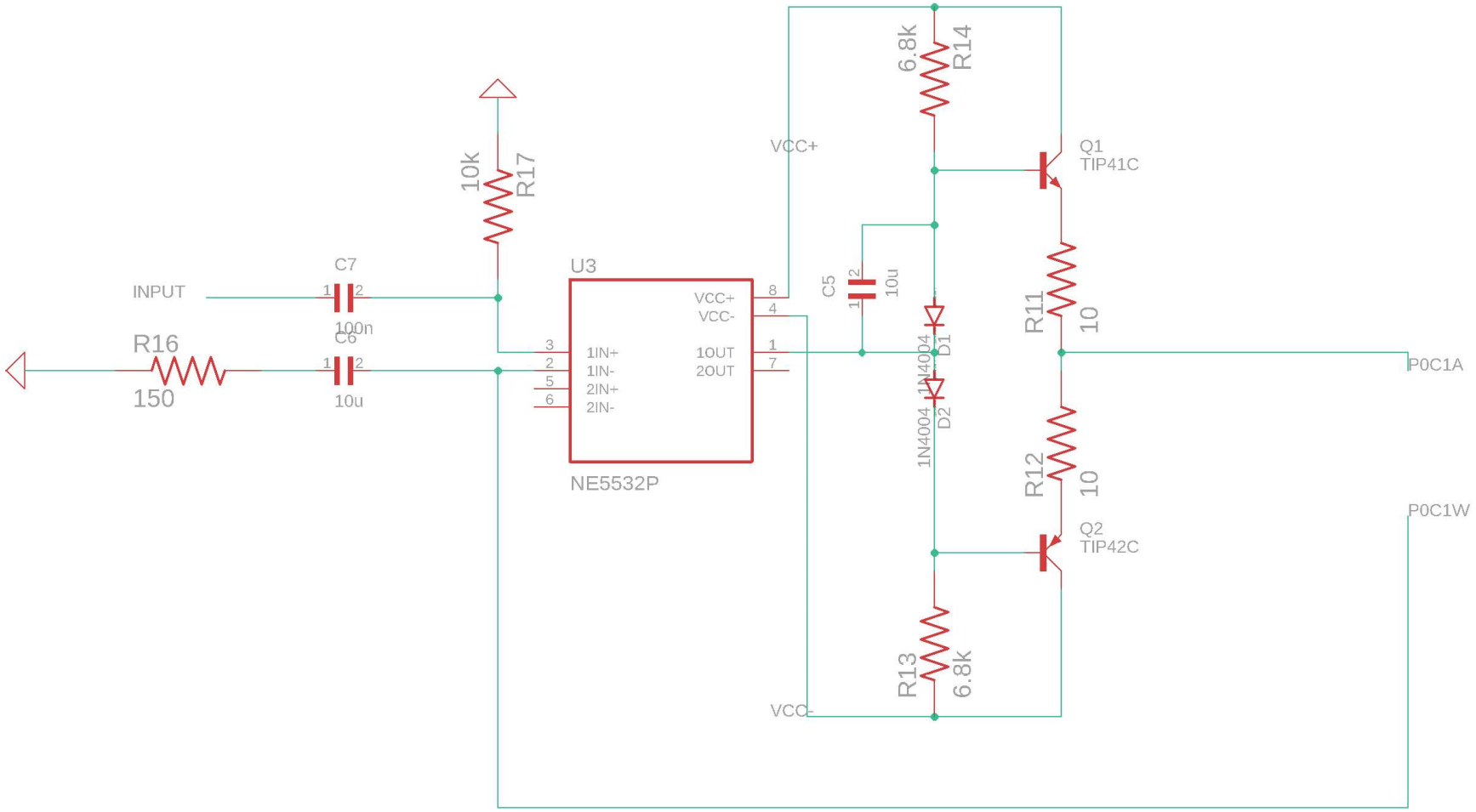


Figure 20: EagleCad Schematic of Reverb Driver Circuit (ACS)

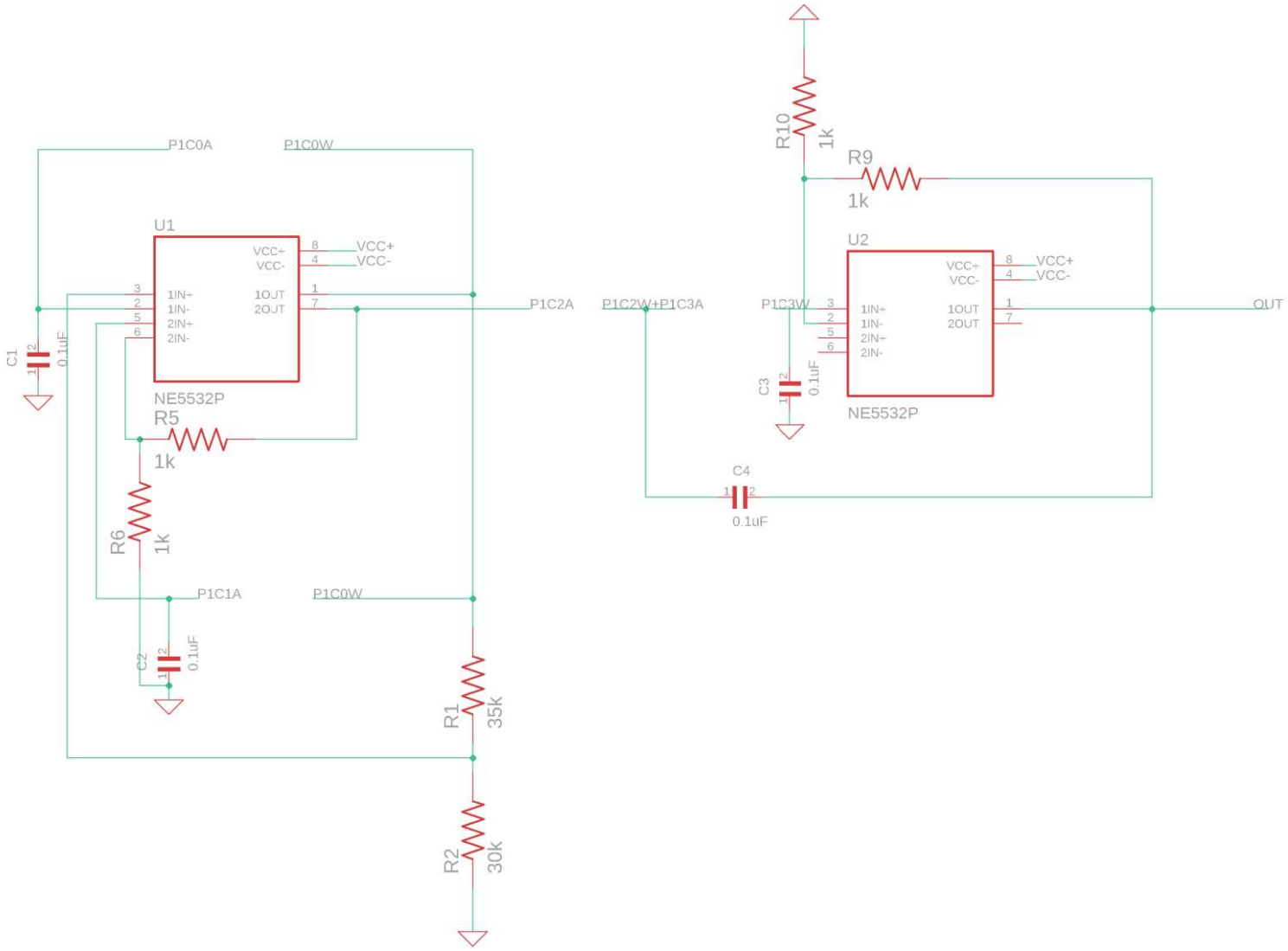


Figure 21: EagleCad Schematic for Oscillator Circuit (ACS)

Table 31: Parts for Reverb Drive and Oscillator Circuits (ACS)

Part	Value	Device	Package
C1	0.1uF	Capacitor	603
C2	0.1uF	Capacitor	603
C3	0.1uF	Capacitor	603
C4	0.1uF	Capacitor	603
C5	10u	Capacitor	603
C6	10u	Capacitor	603
C7	100n	Capacitor	603
D1	1N4004	1N4004	DO41-10
D2	1N4004	1N4004	DO41-10
Q1	TIP41C	TIP41C	TO220AV
Q2	TIP42C	TIP42C	TO220AV
R1	35k	Resistor	603
R2	30k	Resistor	603
R5	1k	Resistor	603
R6	1k	Resistor	603
R9	1k	Resistor	603
R10	1k	Resistor	603
R11	10	Resistor	603
R12	10	Resistor	603
R13	6.8k	Resistor	603
R14	6.8k	Resistor	603
R16	150	Resistor	603
R17	10k	Resistor	603
U1	NE5532P	NE5532P	DIP794W45P254L959
U2	NE5532P	NE5532P	DIP794W45P254L959
U3	NE5532P	NE5532P	DIP794W45P254L959

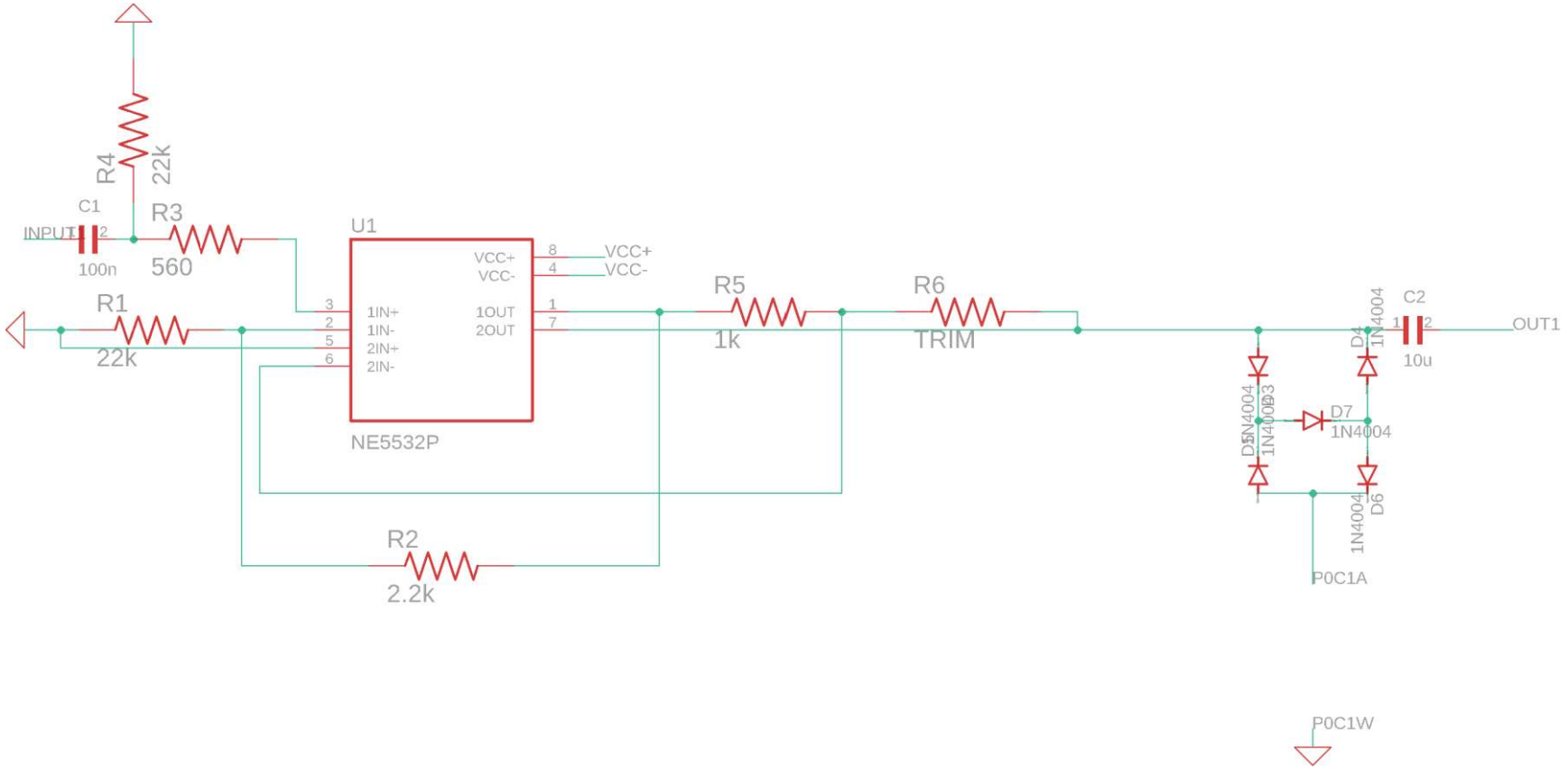


Figure 22: EagleCad Schematic for Reverb Overvoltage Protection (ACS)

Table 32: Parts List for Reverb Overvoltage Protection (ACS)

Part	Value	Device	Package
C1	100n	Capacitor	0603
C2	10u	Capacitor	0603
D3	1N4004	Diode	DO41-10
D4	1N4004	Diode	DO41-10
D5	1N4004	Diode	DO41-10
D6	1N4004	Diode	DO41-10
D7	1N4004	Diode	DO41-10
R1	22k	Resistor	0603
R2	2.2k	Resistor	0603
R3	560	Resistor	0603
R4	22k	Resistor	0603
R5	1k	Resistor	0603
R6	TRIM	Resistor	0603
U1	NE5532P	NE5532P	DIP794W45P254L959

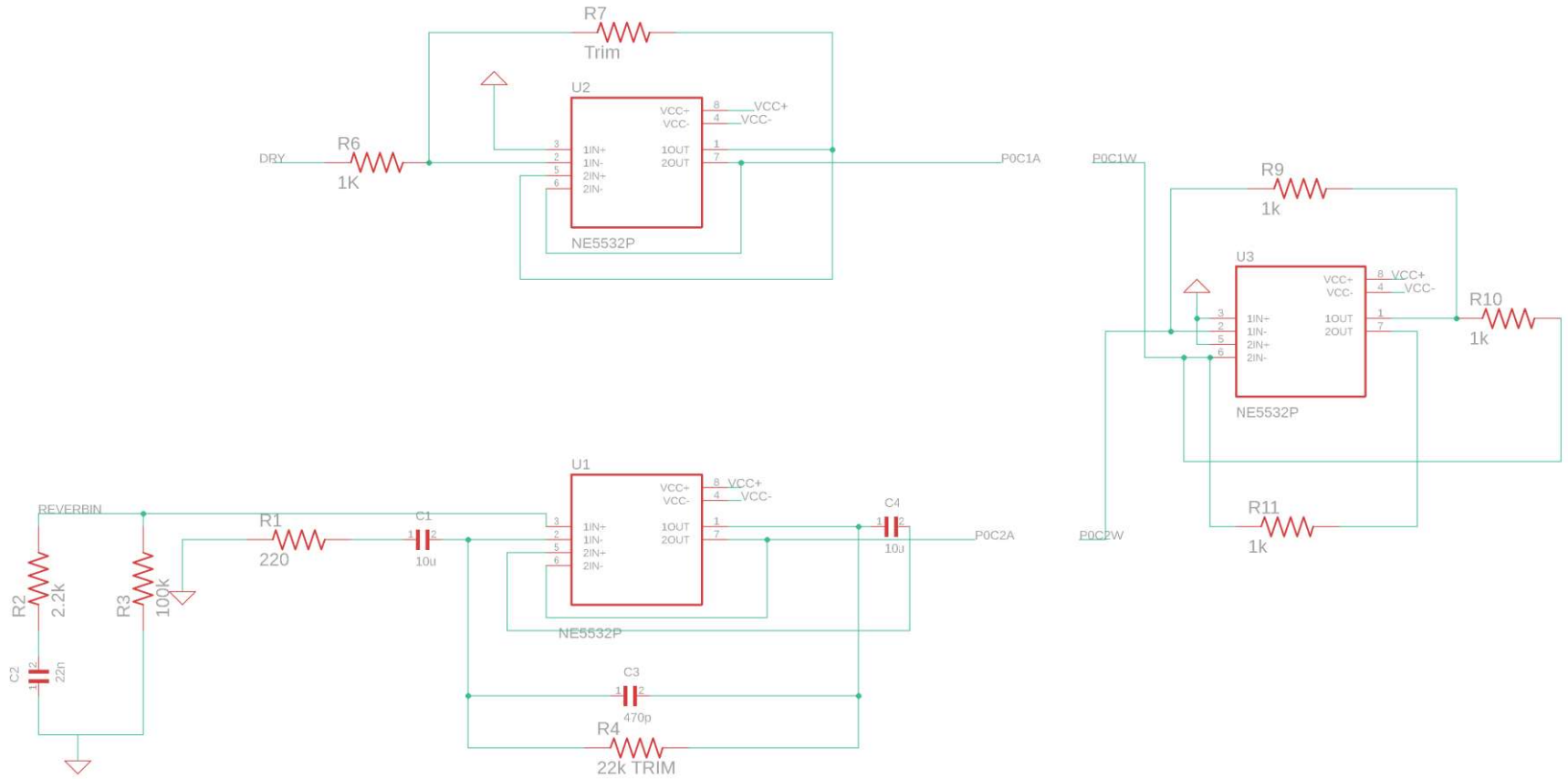


Figure 23: Reverb Volume Mixer (ACS)

Table 33: Reverb Volume Mixer Parts List (ACS)

Part	Value	Device	Package
C1	10u	Capacitor	CAPC0603X33N
C2	22n	Capacitor	CAPC0603X33N
C3	470p	Capacitor	CAPC0603X33N
C4	10u	Capacitor	CAPC0603X33N
R1	220	Resistor	0603
R2	2.2k	Resistor	0603
R3	100k	Resistor	0603
R4	22k TRIM	Resistor	0603
R6	1k	Resistor	0603
R7	Trim	Resistor	0603
R9	1k	Resistor	0603
R10	1k	Resistor	0603
R11	1k	Resistor	0603
U1	NE5532P	NE5532P	DIP794W45P254L959H508Q8
U2	NE5532P	NE5532P	DIP794W45P254L959H508Q8
U3	NE5532P	NE5532P	DIP794W45P254L959H508Q8

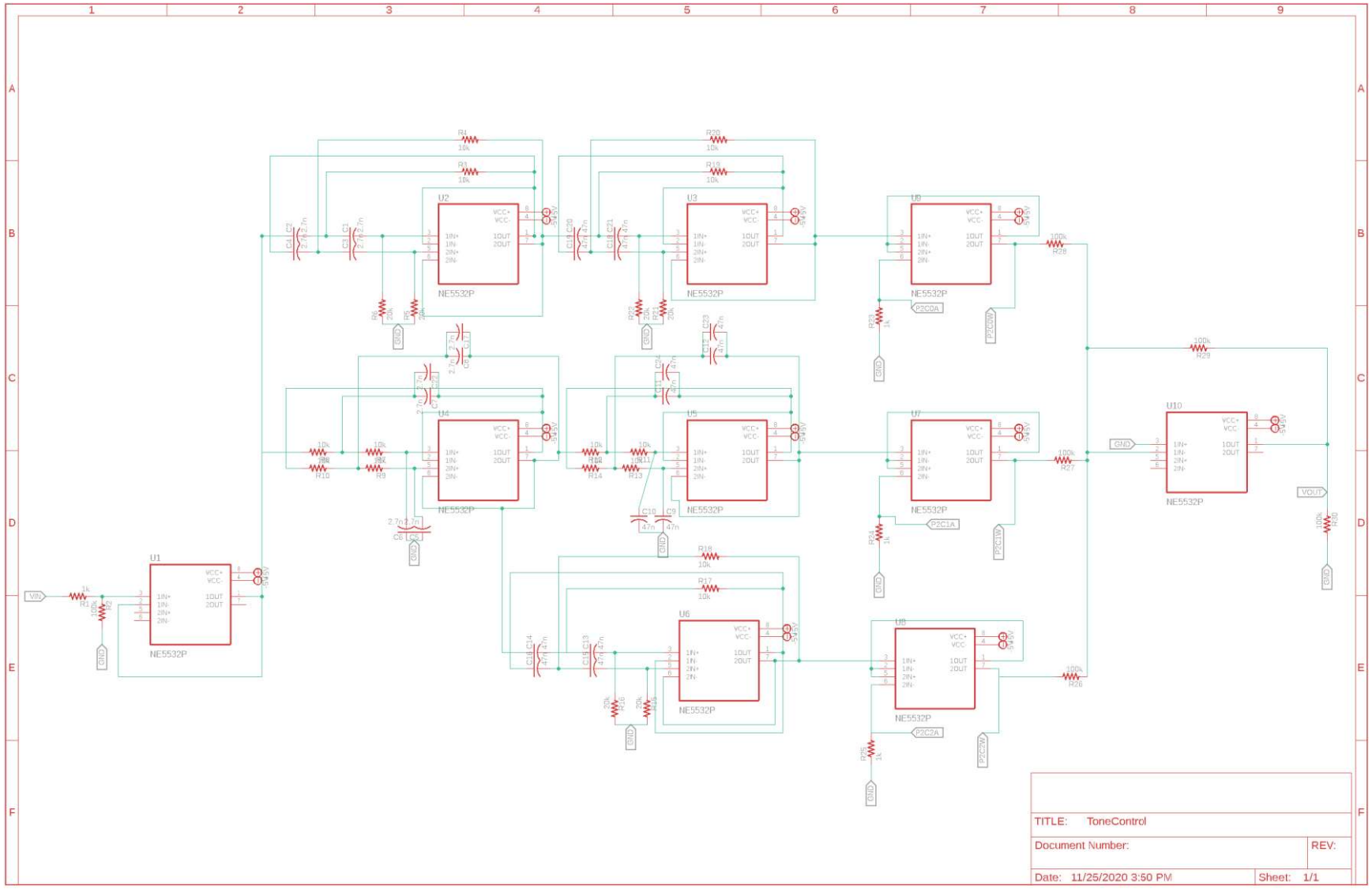


Figure 24: EagleCad Schematic for Tone Control Circuit (LW)

Table 34: Parts List for Tone Control Circuit: (LW)

Part	Value	Device	Package
C1	2.7n	C0603C272F8HAC7867	C0603K
C2	2.7n	C0603C272F8HAC7867	C0603K
C3	2.7n	C0603C272F8HAC7867	C0603K
C4	2.7n	C0603C272F8HAC7867	C0603K
C5	2.7n	C0603C272F8HAC7867	C0603K
C6	2.7n	C0603C272F8HAC7867	C0603K
C7	2.7n	C0603C272F8HAC7867	C0603K
C8	2.7n	C0603C272F8HAC7867	C0603K
C9	47n	CGA3E2X7R1H472K080AA	C0603K
C10	47n	CGA3E2X7R1H472K080AA	C0603K
C11	47n	CGA3E2X7R1H472K080AA	C0603K
C12	47n	CGA3E2X7R1H472K080AA	C0603K
C13	47n	CGA3E2X7R1H472K080AA	C0603K
C14	47n	CGA3E2X7R1H472K080AA	C0603K
C15	47n	CGA3E2X7R1H472K080AA	C0603K
C16	47n	CGA3E2X7R1H472K080AA	C0603K
C17	2.7n	C0603C272F8HAC7867	C0603K
C18	47n	CGA3E2X7R1H472K080AA	C0603K
C19	47n	CGA3E2X7R1H472K080AA	C0603K
C20	47n	CGA3E2X7R1H472K080AA	C0603K
C21	47n	CGA3E2X7R1H472K080AA	C0603K
C22	2.7n	C0603C272F8HAC7867	C0603K
C23	47n	CGA3E2X7R1H472K080AA	C0603K
C24	47n	CGA3E2X7R1H472K080AA	C0603K
R1	1k	R-US_0309/10	0309/10
R2	100k	R-US_0309/10	0309/10
R3	10k	R-US_0309/10	0309/10
R4	10k	R-US_0309/10	0309/10
R5	20k	R-US_0309/10	0309/10
R6	20k	R-US_0309/10	0309/10

R7	10k	R-US_0309/10	0309/10
R8	10k	R-US_0309/10	0309/10
R9	10k	R-US_0309/10	0309/10
R10	10k	R-US_0309/10	0309/10
R11	10k	R-US_0309/10	0309/10
R12	10k	R-US_0309/10	0309/10
R13	10k	R-US_0309/10	0309/10
R14	10k	R-US_0309/10	0309/10
R15	20k	R-US_0309/10	0309/10
R16	20k	R-US_0309/10	0309/10
R17	10k	R-US_0309/10	0309/10
R18	10k	R-US_0309/10	0309/10
R19	10k	R-US_0309/10	0309/10
R20	10k	R-US_0309/10	0309/10
R21	20k	R-US_0309/10	0309/10
R22	20k	R-US_0309/10	0309/10
R23	1k	R-US_0309/10	0309/10
R24	1k	R-US_0309/10	0309/10
R25	1k	R-US_0309/10	0309/10
R26	100k	R-US_0309/10	0309/10
R27	100k	R-US_0309/10	0309/10
R28	100k	R-US_0309/10	0309/10
R29	100k	R-US_0309/10	0309/10
R30	100k	R-US_0309/10	0309/10
U1	NE5532P	NE5532P	DIP794W45P254L959H508Q8
U2	NE5532P	NE5532P	DIP794W45P254L959H508Q8
U3	NE5532P	NE5532P	DIP794W45P254L959H508Q8
U4	NE5532P	NE5532P	DIP794W45P254L959H508Q8
U5	NE5532P	NE5532P	DIP794W45P254L959H508Q8
U6	NE5532P	NE5532P	DIP794W45P254L959H508Q8
U7	NE5532P	NE5532P	DIP794W45P254L959H508Q8
U8	NE5532P	NE5532P	DIP794W45P254L959H508Q8

U9	NE5532P	NE5532P	DIP794W45P254L959H508Q8
U10	NE5532P	NE5532P	DIP794W45P254L959H508Q8

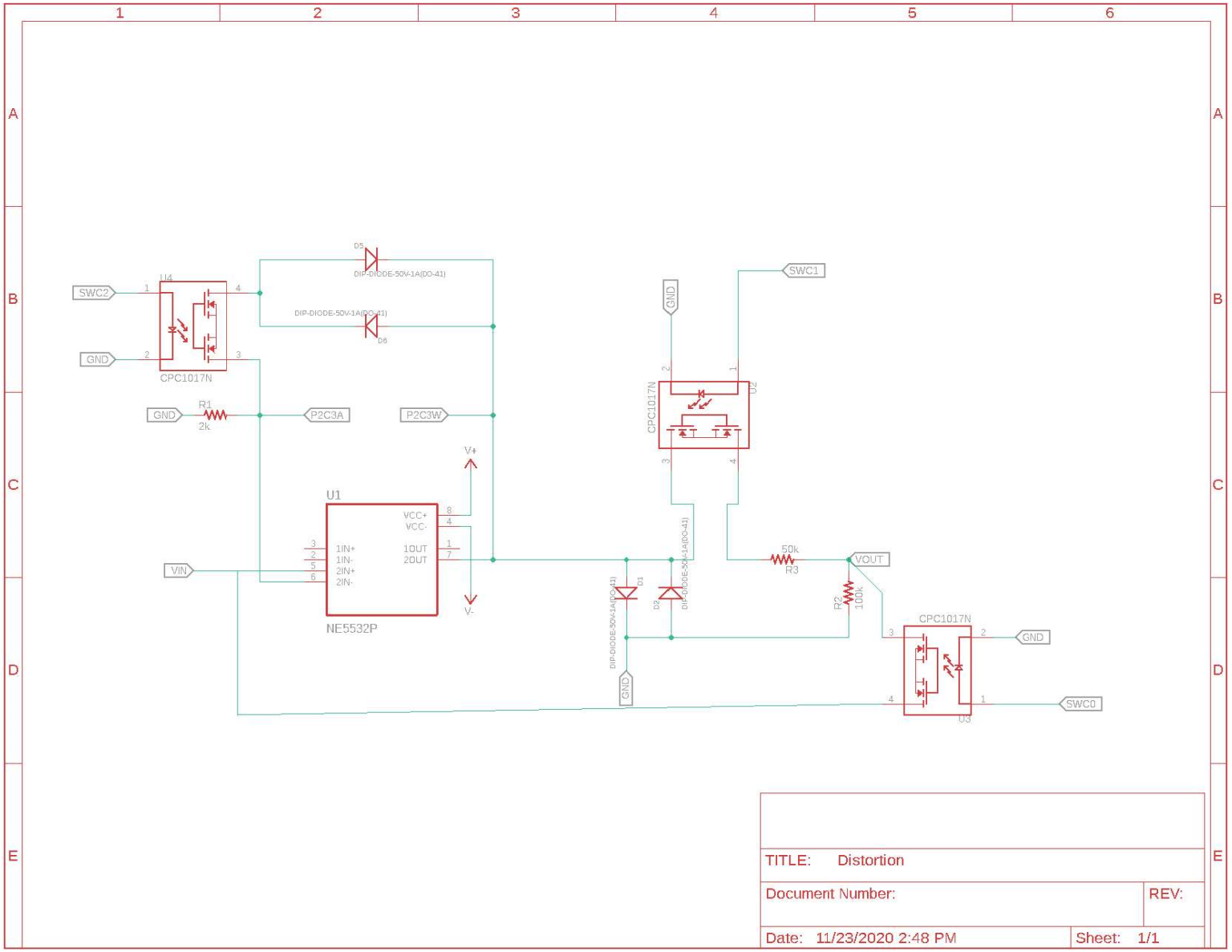


Figure 25: EagleCad Schematic for Distortion Circuit (LW)

Table 35: Parts List for Distortion Circuit (LW)

Part	Value	Device	Package	Description
D1	DIP-DIODE-50V-1A(DO-41)	Diode	DO-41	Diode used for clipping at the threshold voltage.
D2	DIP-DIODE-50V-1A(DO-41)	Diode	DO-41	Diode used for clipping at the threshold voltage.
D5	DIP-DIODE-50V-1A(DO-41)	Diode	DO-41	Diode used for clipping at the threshold voltage.
D6	DIP-DIODE-50V-1A(DO-41)	Diode	DO-41	Diode used for clipping at the threshold voltage.
R1	2k	R-US_0411/15	0411/15	Resistor
R2	100k	R-US_0411/15	0411/15	Resistor
R3	50k	R-US_0411/15	0411/15	Resistor
U1	NE5532P	NE5532P	DIP794W45P254L959 H508Q8	Dual Low-Noise High-Speed Audio Operational Amplifier
U2	CPC1017N	CPC1017N	SOIC254P610X218-4N	Solid State Relay SPST
U3	CPC1017N	CPC1017N	SOIC254P610X218-4N	Solid State Relay SPST
U4	CPC1017N	CPC1017N	SOIC254P610X218-4N	Solid State Relay SPST

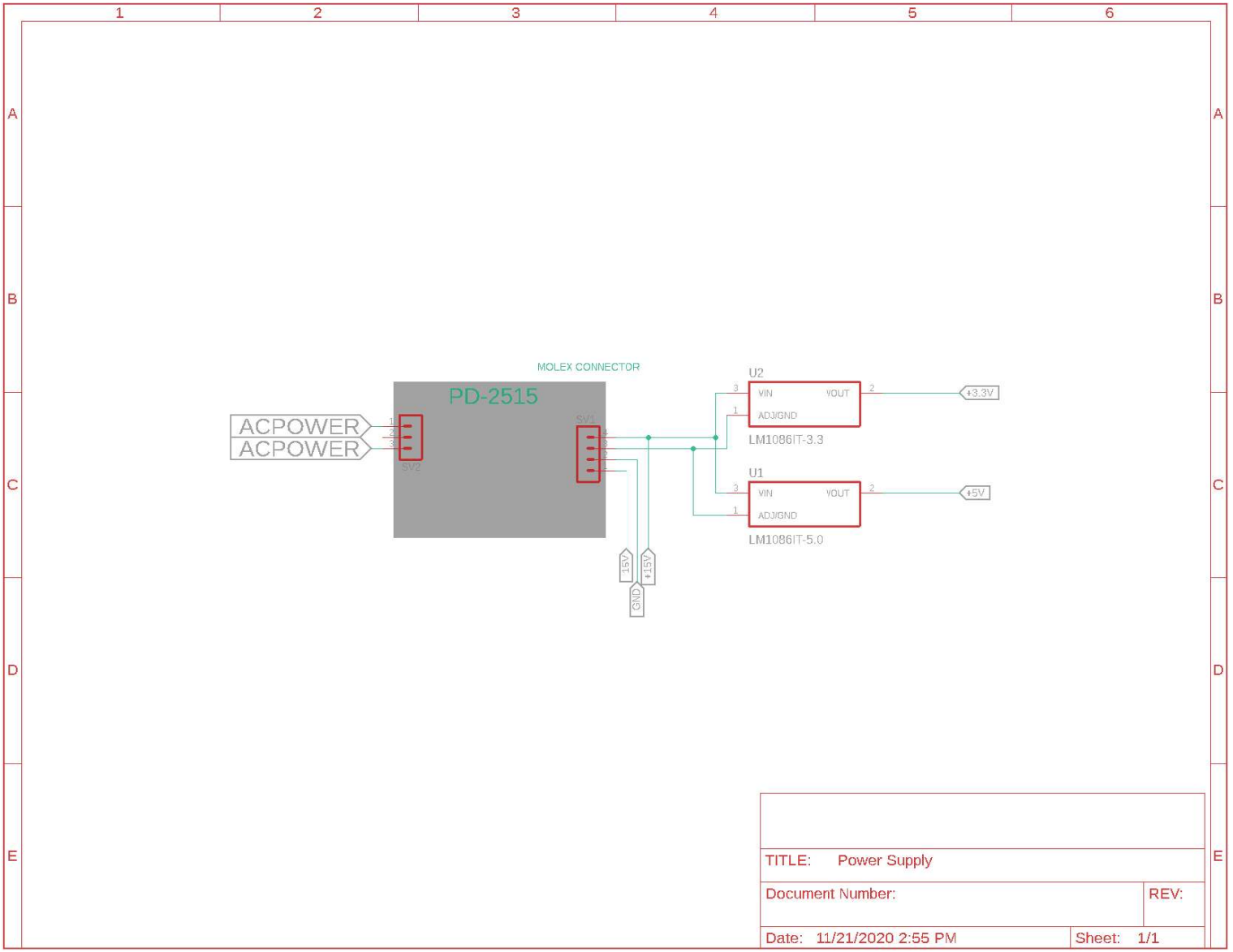


Figure 26: EagleCad Schematic for Power Supply Circuit (LW)

Table 36: Parts List for Power Supply Circuit (LW)

Part	Value	Device	Package	Description
SV1		MA04-1	MA04-1	4 Pin Out to PD2515
SV2		MA03-1	MA03-1	3 Pin from PD2515 to Wall Supply
U1	LM1086IT-5.0	LM1086IT-5.0	TO254P1054X470X1955-3	5V Voltage Regulator
U2	LM1086IT-3.3	LM1086IT-3.3	TO254P1054X470X1955-3	3.3V Voltage Regulator

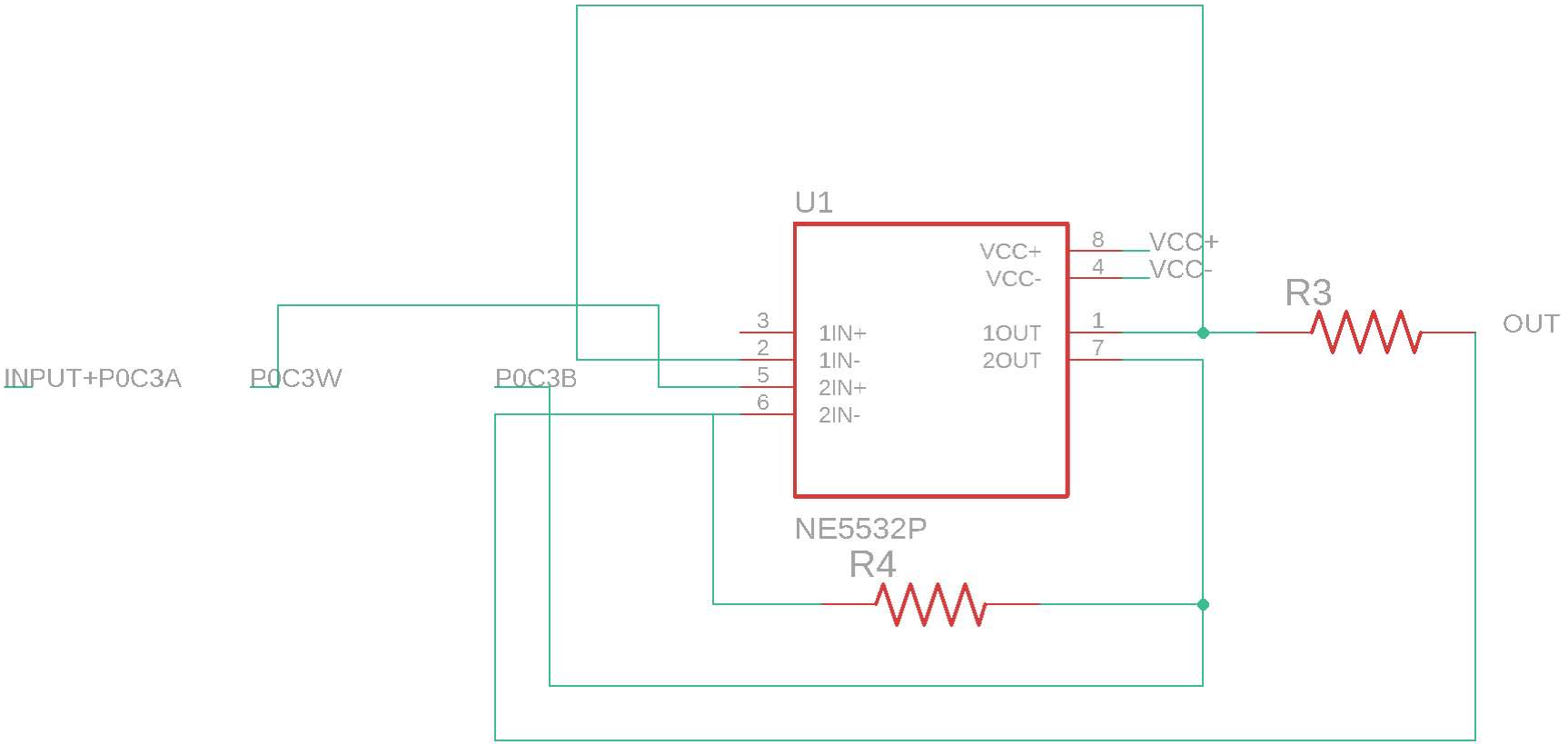


Figure 27: Volume Control Circuit (ACS)

Table 37: Volume Control Parts (ACS)

Part	Value	Device	Package
R3	1k	Resistor	0603
R4	1k	Resistor	0603
U1	NE5532P	High-Speed	DIP794W45P254L959H508Q8

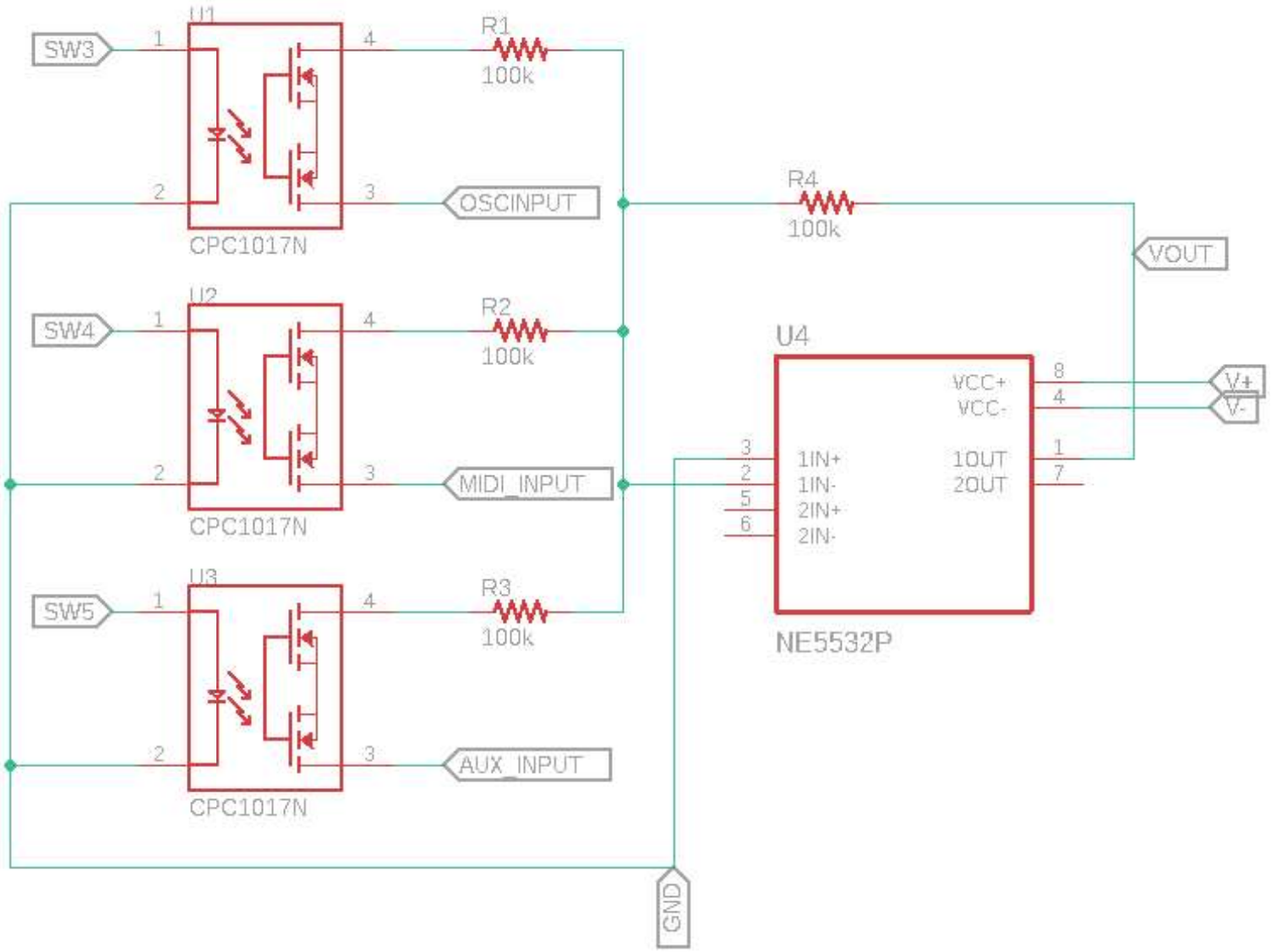


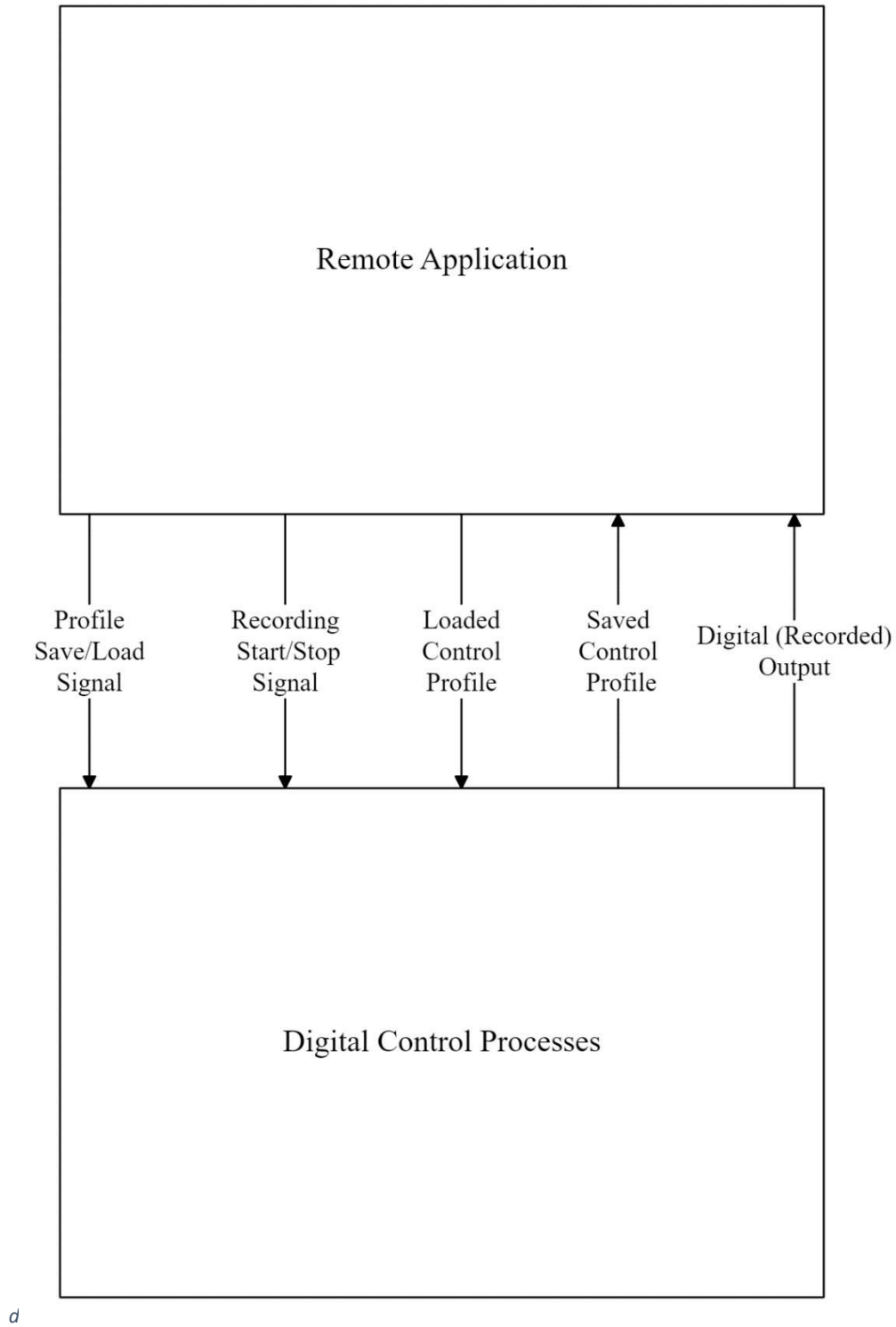
Figure 28: Input Select Circuit (ACS, LW)

Table 38: Parts list for Input Select Circuit (ACS + LW)

Part	Value	Device	Package
R1	100k	R-US_R0603	R0603
R2	100k	R-US_R0603	R0603
R3	100k	R-US_R0603	R0603
R4	100k	R-US_R0603	R0603
U1	CPC1017N	CPC1017N	SOIC254P610X218-4N
U2	CPC1017N	CPC1017N	SOIC254P610X218-4N
U3	CPC1017N	CPC1017N	SOIC254P610X218-4N
U4	NE5532P	NE5532P	DIP794W45P254L959H508Q8

5.2. Software Design

5.2.1. Level 0 Functional Decomposition



d

Figure 29: Level 0 block diagram describing input/output relationship between software.

Table 39: Level 0 Functional Requirements of Remote Application

Module	Remote Application
Designer	Scott Grisso
Inputs	Saved Control Profile, Digital (Recorded) Output
Outputs	Profile Save/Load Signal, Recording Start/Stop Signal, Loaded Control Profile
Description	The Remote Application is responsible for providing instructions to the Digital Control Processes for the implementation of associated control interactions.

Table 40: Level 0 Functional Requirements of Digital Control Processes

Module	Digital Control Processes
Designer	Adam Brunner
Inputs	Profile Save/Load Signal, Recording Start/Stop Signal, Loaded Control Profile
Outputs	Saved Control Profile, Digital (Recorded) Output
Description	Digital control processes are responsible for receiving and responding to interactions from the remote application, as well as interacting with embedded peripheral devices and performing processing of digital instrument inputs.

5.2.2. Level 1 Functional Decomposition

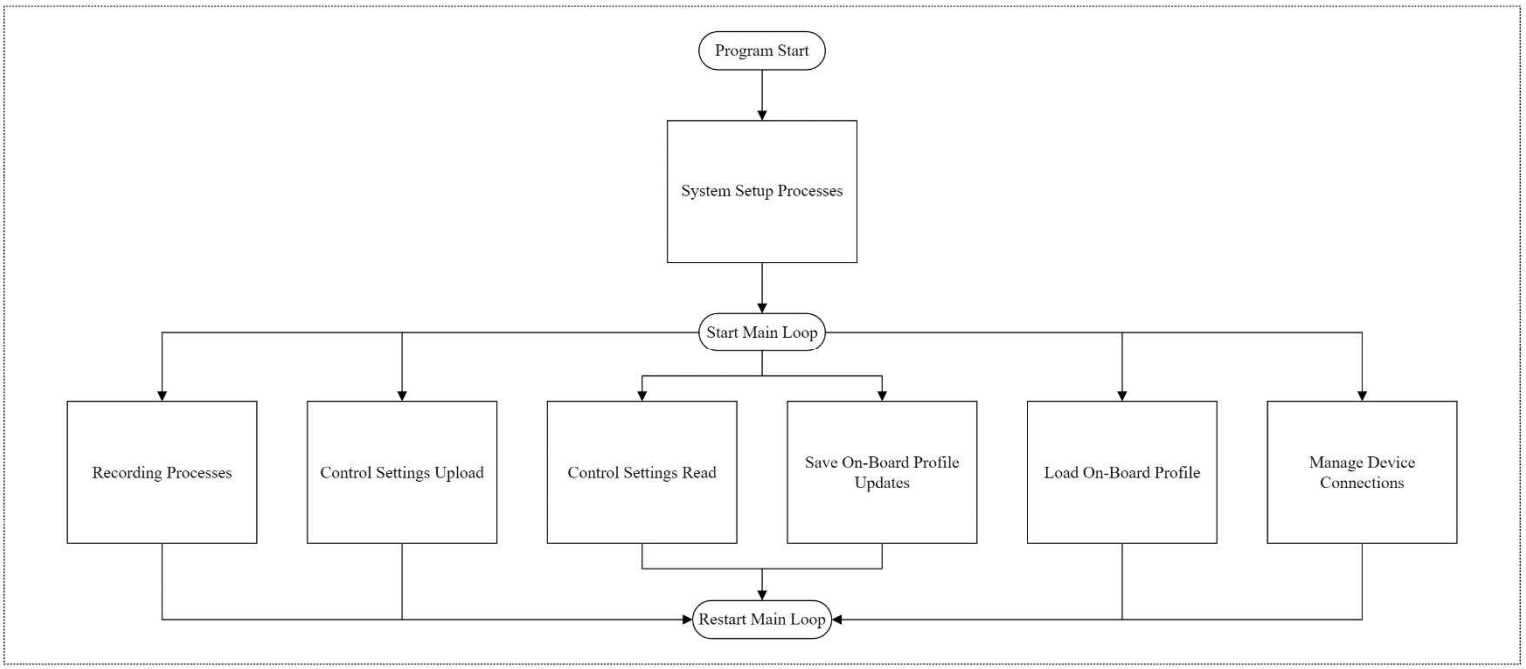


Figure 30: Level 1 flow chart of Remote Application (SBG)

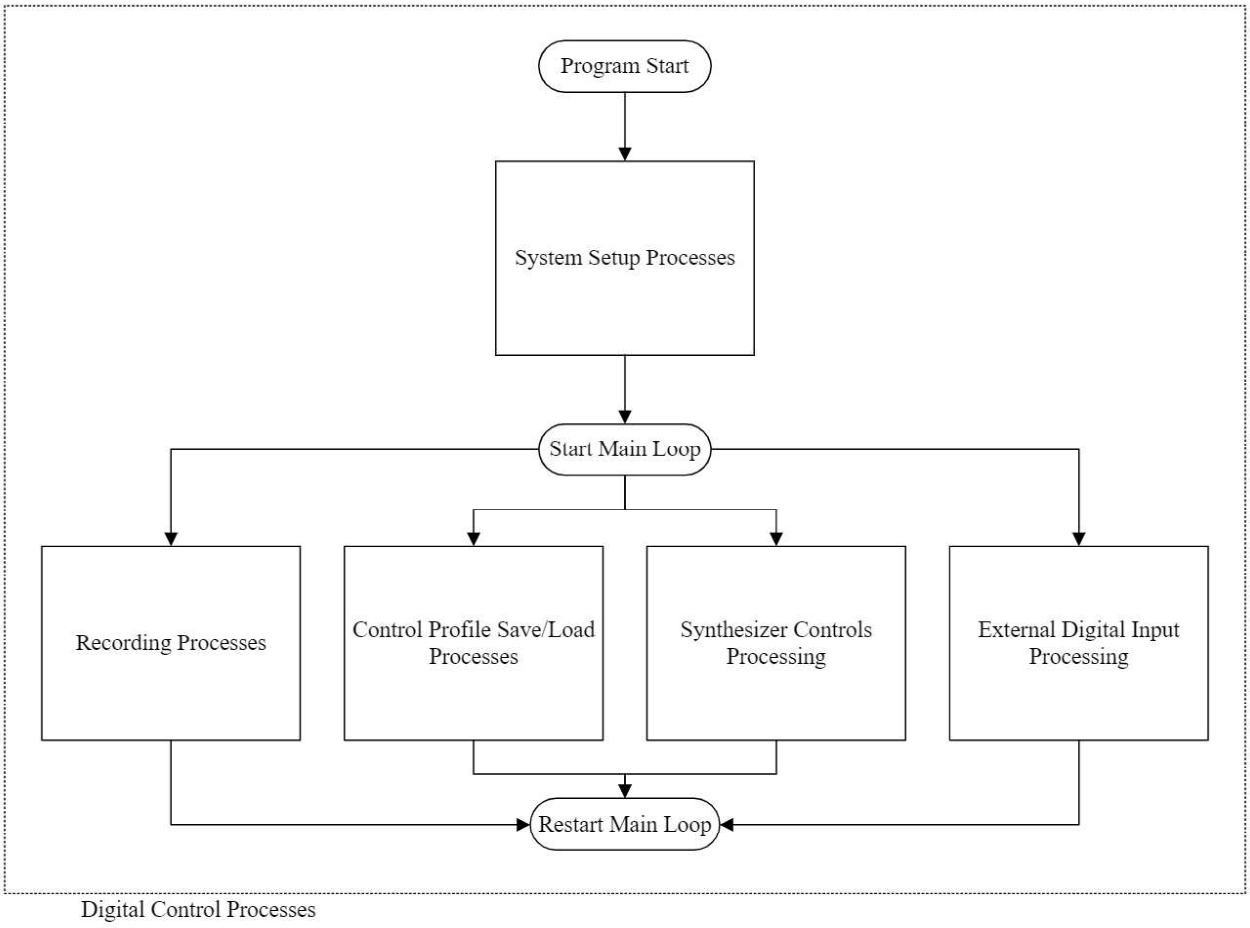


Figure 31: Level 1 flow chart of Digital Control Processes (ADB)

5.2.3. Level 2 Functional Decomposition

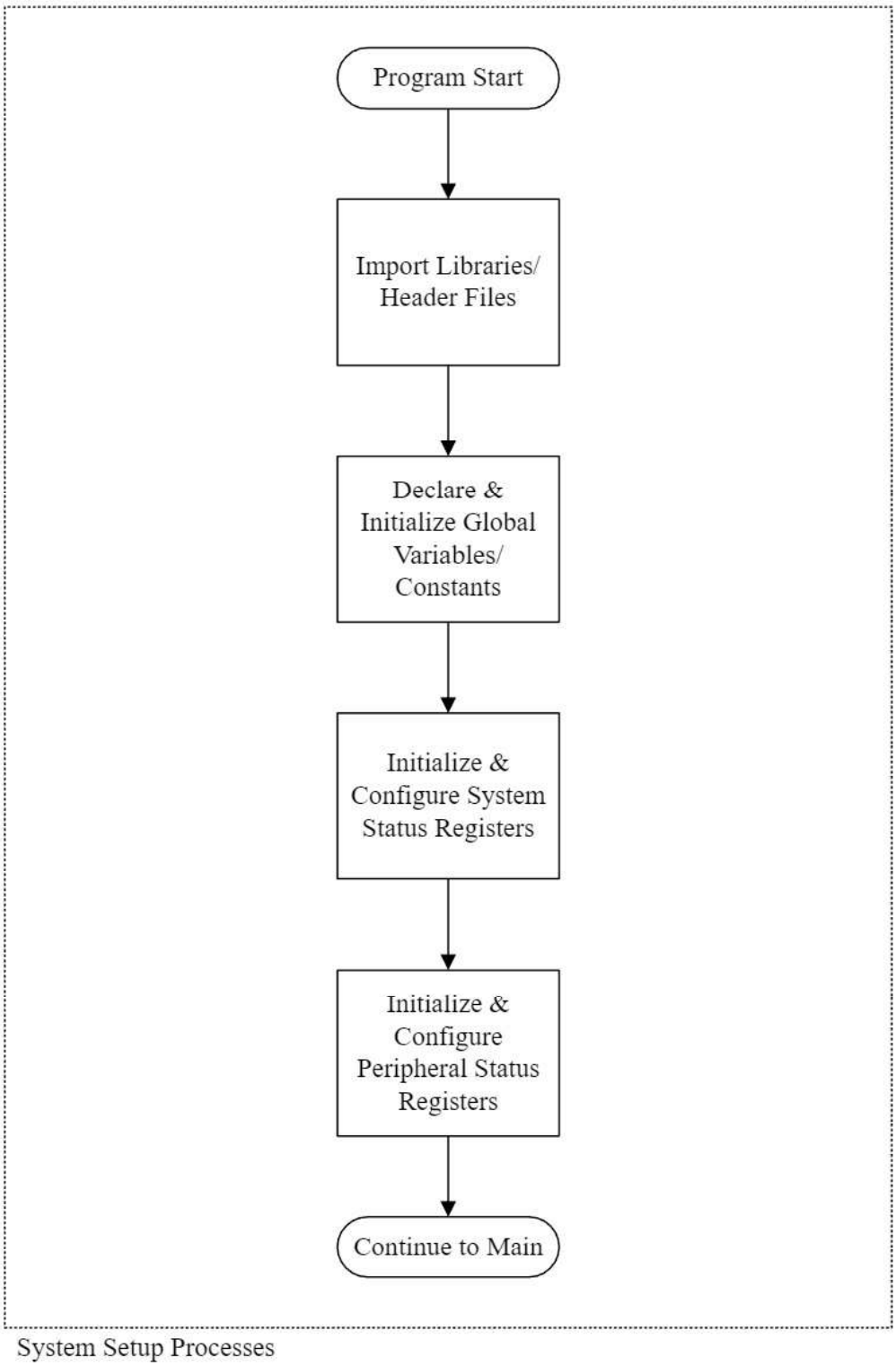


Figure 32: Level 2 flow chart of System Setup Processes (ADB)

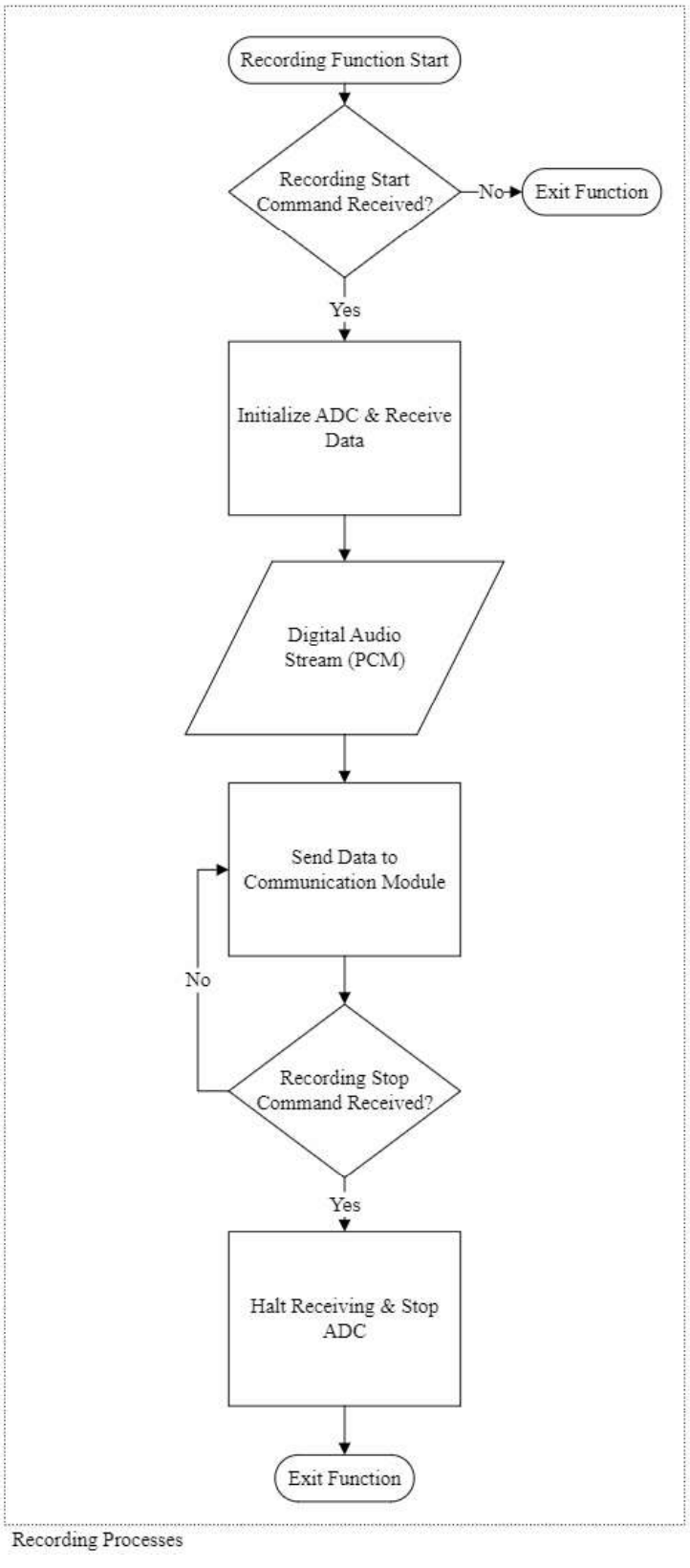


Figure 33: Level 2 flow chart of Recording Processes (ADB)

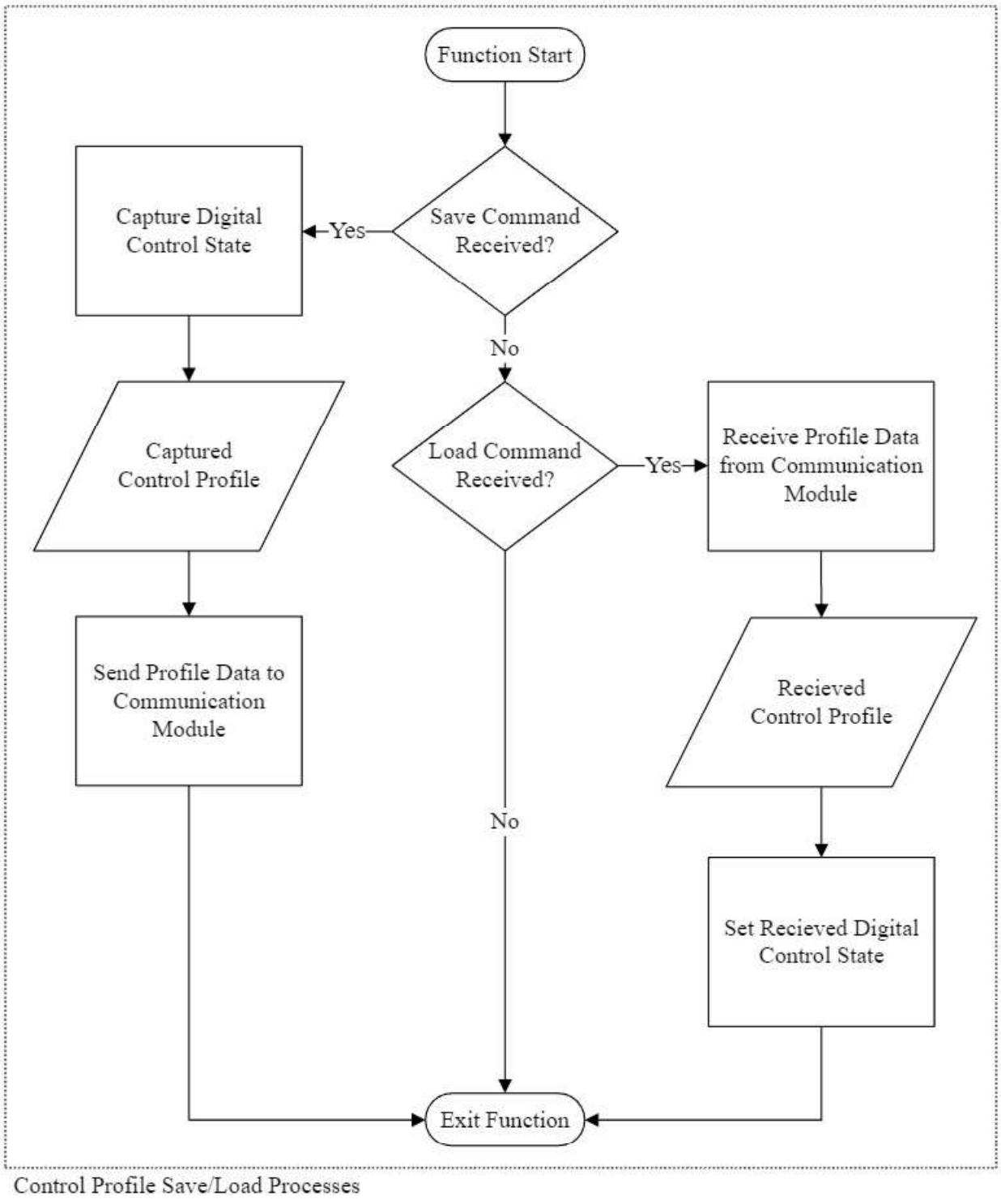


Figure 34: Level 2 flow chart of Control Profile Save/Load Processes (ADB)

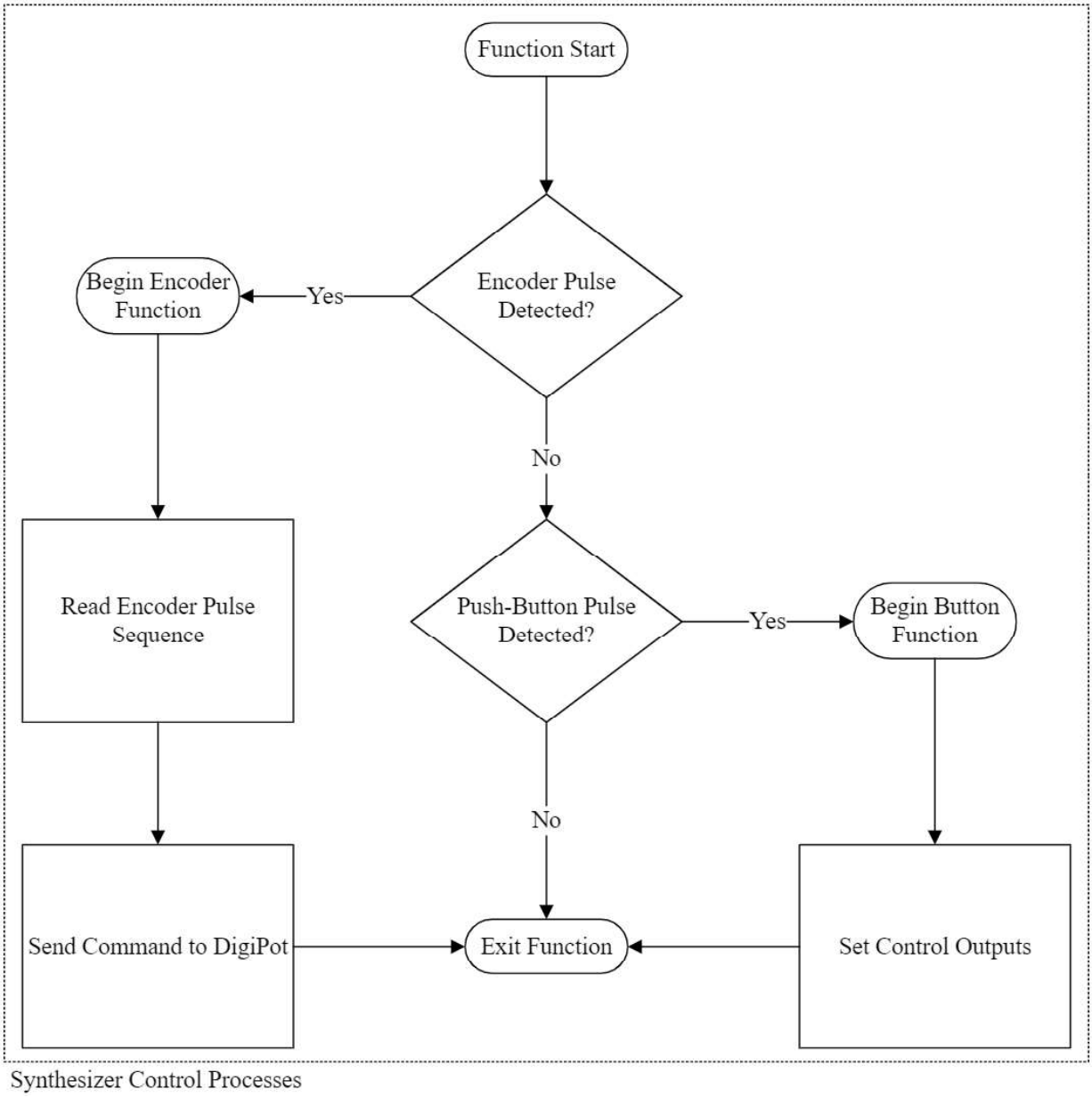


Figure 35: Level 2 flow chart of Synthesizer Control Processes (ADB)

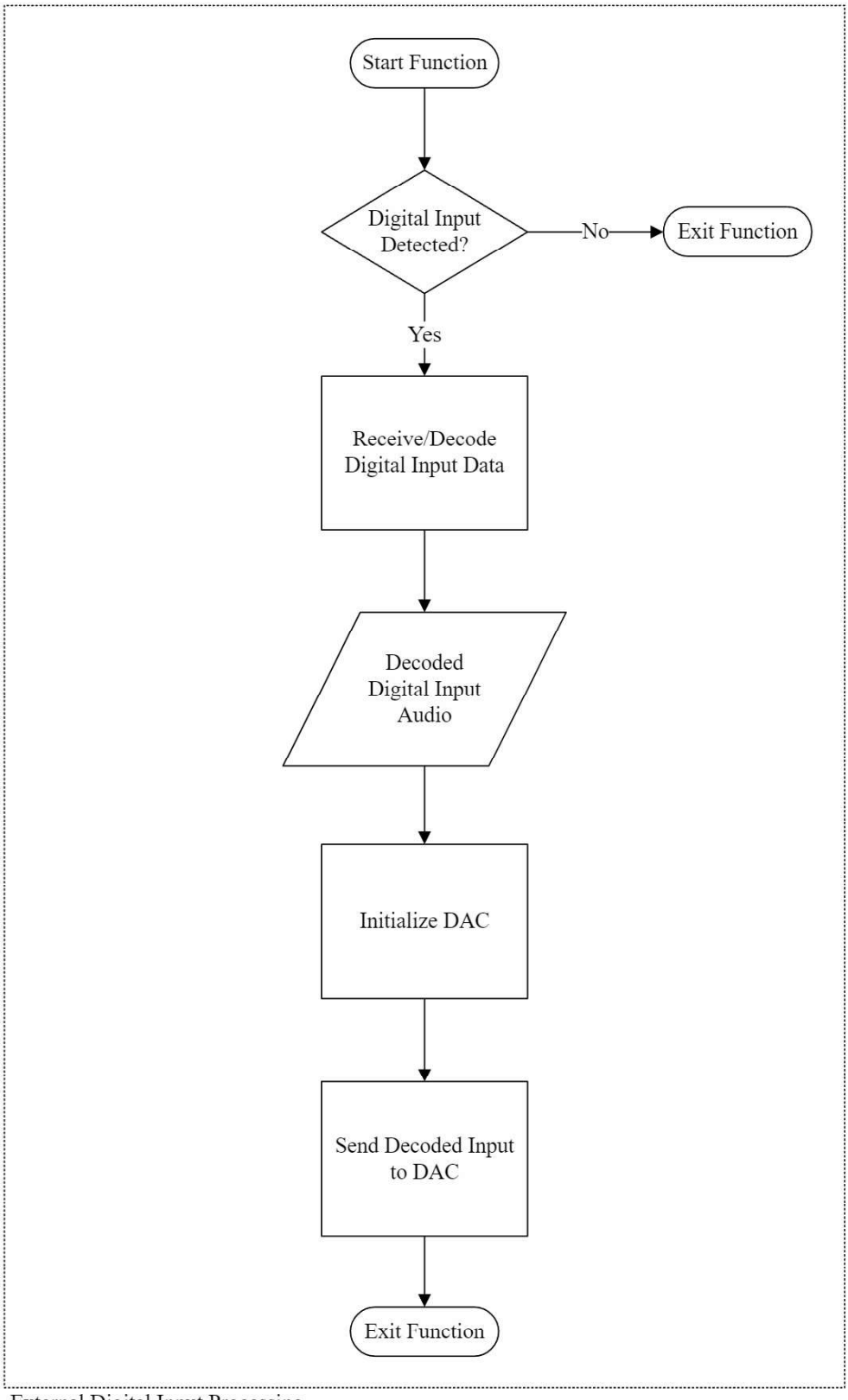


Figure 36: Level 2 flow chart of External Digital Input Processing (ADB)

5.2.4. Systems Integration Code/Pseudocode

5.2.4.1.Digital Control – System Setup Pseudocode (ADB)

The pseudocode below describes the processes necessary to initialize the microcontroller and connected peripherals in order to properly execute the remainder of the program. This includes the importing of necessary header files and libraries which contain definitions of variables and functions used in the program. These header files may also include the configuration of configuration registers on both the microcontroller and connected peripherals, which may require the use of serial communications. While the exact extent to which the program will be broken up into separate headers has not fully been determined yet, the code listed in section 5.2.4.4 shows examples of how separate headers can be created to contain function definitions, configuration register assignment, and more.

```
#include <--header files-->
#include <--libraries-->

int main (void)
{
    Configure GPIO for input/output (set TRISx registers)
    Configure IoC registers to enable IOC functionality
    Configure SPI busses on peripheral pin select pins
    Establish connection with ADC module via SPI
    Establish connection with Remote Communications module via SPI
    Send SPI signals to Remote Communications module to establish connection
with remote host
    Configure and initialize I2C bus for digital potentiometer communication
    Configure and initialize USB port for MIDI data input
    Configure and initialize 32-bit Timer for use with PWM module
    Configure and initialize PWM module

    while(1)
    {
            //Main loop
    }
}
```

5.2.4.2.Digital Control – Recording Process Pseudocode (ADB)

The following pseudocode describes the framework for the processes involved within the Recording Processes portion of the software design. This consists of the receipt of a command from the remote application to signal the initiation of the recording sequence followed by the initialization and receipt of data from the ADC, which is then forwarded to the communication module to be sent to the remote host. This process continues until a stop command is received from the remote host, at which point the MCU will cease transmission of the digital audio stream. Note that this pseudo code assumes the initialization and configuration of the SPI ports and peripherals (Comms. Module & ADC) is handled by the system setup processes, which is the intended outcome of the design.

```
if(SPI2 MISO == Start Command)  //If start received from remote host...
{
    int recordFlag = 1;  //... Begin Recording
}

while(recordFlag)
{
    digitalAudio = SPI1 MISO Data   //Capture ADC output data
    SPI2 MOSI Data = digitalAudio   //Forward to communications module
    if(SPI2 MISO == Stop Command)   //If stop received from remote host...
    {
        recordFlag = 0;  //... Stop Recording
    }
}
```

5.2.4.3.Digital Control – Control Profile Save/Load Pseudocode (ADB)

This pseudocode depicts the general processes involved with the receipt and transfer of synthesizer control profiles to and from the remote host. Similarly to the recording process, this portion of the program is initialized by receiving either a load or save command from the remote host via the communications module. Depending on which command is received the program will either retrieve the current state of the various synthesizer controls interfaced with the MCU

and transmit this information to the remote host, or the MCU will decode and assign the control values to be loaded based on the profile received from the remote host.

```
if(SPI2 MISO == Save Command)    //If save command received from remote
host...
{
    int saveFlag = 1     //... Begin save routine
}

while(saveFlag)
{
    potValues = I2CRead(Channel Data) //Retrieve current state of digital
potentiometers
    switchValues = ControlVoltages    //Retrieve current state of control
voltage outputs
    SPI2 MOSI Data = potValues        //Forward to digipot state to
communications module
    SPI2 MOSI Data = switchValues     //Forward to switch state to
communications module

    saveFlag = 0     // End save routine
}

if(SPI2 MISO == Load Command)    //If load command received from remote
host...
{
    int loadFlag = 1     //... Begin save routine
}

while(loadFlag)
{
    potValues = SPI2 MOSI Data        //Retrieve state of digital
potentiometers from received profile
    switchValues = SPI2 MOSI Data     //Retrieve state of control voltages
from received profile
    I2CWrite(potValues)               //Send I2CWrite to digipots to set
desired values
    ControlVoltageGPIO = switchValues //Set control voltage GPIO to desired
values

    loadFlag = 0     // End load routine
}
```

5.2.4.4.Digital Control – Synthesizer Control Encoder Subprocess Code (ADB)

The code below was developed for the purposes of subsystem demonstration of the interfacing between the rotary encoder, microcontroller and I^2C Digital Potentiometer. The

program was developed in MPLab X IDE and consists of the main source file (main.c) and several header files (config.h, delay.h, I2C.h, MCP444X6X.h and IOC_ISR.h). The specific functionality of these files is described within the source code, but as a general overview the processes involved with detecting and interpreting the encoder pulse sequence and outputting the desired I^2C commands are contained within the IOC_ISR.h header file, with the other files contain the definitions and commands necessary for system configuration and initialization.

```
#include "config.h"
#include "I2C.h"
#include "delay.h"
#include "MCP444X6X.h"
#include "IOC_ISR.h"

int main(void)
{
    TRISA = 0x0000; //Set GPIO Port A to output
    TRISD  = 0xFFFF; //Set GPIO Port D to input (push-buttons)
    I2C1Init(78); //Initialize I2C Bus 1 w/ 100kHz clock rate

    IEC1bits.IOCIE = 1;
    PADCONbits.IOCON = 1; //Enable Interrupt-on-Change functionality
    IFS1bits.IOCIF = 0; //Clear IOC Flag
    IOCND = 0xFFFF; //Enable IoC on falling edge for port D
    IOCFD = 0x0000; //Clear IoC Port D flag bits
    IOCPDF = 0; //Clear IoC Port D flag bit in IOCSTAT register

    while(1)
    {
     //Main loop
    }
}


/*
 * File:   config.h
 * Author: Adam Brunner
 *
 * Created on October 14, 2020, 5:22 PM
 * Description: Recreation of ESI config.h file with corresponding configs
for
 *              PIC24FJ1024SB610
 */

#ifndef CONFIG_H
#define CONFIG_H

#ifdef __cplusplus
extern "C" {
```

```
#endif

#include <xc.h>

    //_CONFIG1
    #pragma config JTAGEN = OFF     //Disable JTAG interface (FICD Register)
    #pragma config GSS = DISABLED   //Disable general code protection (FSEC
Register)
    #pragma config GWRP = OFF       //Disable program mem. write protection
(FSEC Register)
    #pragma config ICS = PGD2       //Emulator pin placement select (FICD
Register)
    #pragma config FWDTEN = OFF     //Disable Watchdog timer (FWDT Register)

    //_CONFIG2
    #pragma config IESO = OFF       //Starts device w/ user-selected
oscillator source (FOSCSEL Register)
    #pragma config FCKSM = CSDCMD   //Clock Switching/Monitor disabled (FOSC
Register)
    #pragma config FNOSC = PRIPLL   //Primary Oscillator w/ PLL selected
(FOSCSEL Register)
    #pragma config POSCMOD = HS     //Primary Oscillator: HS Mode (FOSC
Register)

#ifdef  __cplusplus
}
#endif

#endif  /* CONFIG_H */


/*
 * File:   I2C.h
 * Author: Adam Brunner
 *
 * Created on October 14, 2020, 8:38 PM
 */

#ifndef I2C_H
#define I2C_H

#ifdef  __cplusplus
extern "C" {
#endif
#include <xc.h>
#include "delay.h"

    void I2C1Init(int BRG)
    {
        I2C1BRG = BRG; //See PIC24FJ1024GB610 data sheet, Table 18-1 pg. 249
        while (I2C1STATbits.P); // Check bus idle, see 5.1 of I2C document.
        // It works, not sure its needed.
        I2C1CONLbits.A10M = 0; // 7-bit address mode
        I2C1CONLbits.I2CEN=1; // enable module
    }

    void I2C1Start(void)
```

```
    {
        us_delay(10); // delay to be safe
        I2C1CONLbits.SEN = 1; // Initiate Start condition see pg. 21 of I2C
man. DS70000195F
        while (I2C1CONLbits.SEN); // wait for Start condition complete See
sec. 5.1
        us_delay(10); // delay to be safe
    }

    void I2C1Stop(void)
    {
        us_delay(10); // delay to be safe
        I2C1CONLbits.PEN = 1; // see 5.5 pg. 27 of Microchip I2C manual
DS70000195F
        while (I2C1CONLbits.PEN); // This is at hardware level, & I suspect
fast.
        us_delay(10); // delay to be safe
    }

    void I2C1SendByte(char data)
    {
        while (I2C1STATbits.TBF) ; //wait if buffer is full
        I2C1TRN = data; // pass data to transmission register
        us_delay(10); // delay to be safe
    }

    char I2C1GetByte(void)
    {
        I2C1CONLbits.RCEN =1; // Set RCEN, Enables I2C Receive mode
        while (!I2C1STATbits.RBF) ; //wait for byte to shift into I2C1RCV
register
        I2C1CONLbits.ACKEN = 1; // Master sends Acknowledge
        us_delay(10); // delay to be safe
        return(I2C1RCV);
    }


#ifdef  __cplusplus
}
#endif

#endif  /* I2C_H */



/*
 * File:   delay.h
 * Author: Adam Brunner
 *
 * Created on October 14, 2020, 7:53 PM
 */

#ifndef DELAY_H
#define DELAY_H

#ifdef  __cplusplus
extern "C" {
```

```
#endif
#include <xc.h>

    void ms_delay (int N)
    {
        T1CON = 0x8030;          // Enable Timer 1 w/ 1:256 pre-scale

        unsigned long delay = N * 62.5;    // 1 Millisecond =>
.001/(256*(1/16,000,000)) = 62.5
        TMR1 = 0;
        while (TMR1 < delay);
    }

    void us_delay (int N)
    {
        T1CON = 0x8000;         // Enable Timer 1 w/ 1:1 pre-scale

        unsigned long delay = N * 16;     // 1 Microsecond =>
.000001/(1/16,000,000) = 16
        TMR1 = 0;
        while (TMR1 < delay);
    }

#ifdef  __cplusplus
}
#endif

#endif  /* DELAY_H */



/*
 * File:   MCP444X6X.h
 * Author: Adam Brunner
 * Description: Definitions & functions for Microchip MCP444X/6X 4-Channel
 *              digital potentiometers.
 *
 * Created on November 10, 2020, 7:12 PM
 */

#ifndef MCP444X6X_H
#define MCP444X6X_H

#ifdef  __cplusplus
extern "C" {
#endif

    //I2C Addressing Definitions

    #define POT0W 0x58       //DigiPot 0 (0x2C) Write Address [A1,A0] = [0,0]
    #define POT0R 0x59       //DigiPot 0 (0x2C) Read Address [A1,A0] = [0,0]
    #define POT1W 0x5A       //DigiPot 1 (0x2D) Write Address [A1,A0] = [0,1]
    #define POT1R 0x5B       //DigiPot 1 (0x2D) Read Address [A1,A0] = [0,1]
    #define POT2W 0x5C       //DigiPot 2 (0x2E) Write Address [A1,A0] = [1,0]
    #define POT2R 0x5D       //DigiPot 2 (0x2E) Read Address [A1,A0] = [1,0]
    #define POT3W 0x5E       //DigiPot 3 (0x2F) Write Address [A1,A0] = [1,1]
    #define POT3R 0x5F       //DigiPot 3 (0x2F) Read Address [A1,A0] = [1,1]
```

```
    //Command Byte Definitions

    //Volatile Wiper 0
    #define V0W0 0x00 //Write w/ Data Bit D8 = 0 (Follow w/ I2C Write D[7:0])
    #define V0W1 0x01 //Write w/ Data Bit D8 = 1 (Follow w/ I2C Write D[7:0])
    #define V0RR 0x0C //Random Read - !!!MUST be followed by I2C Start bit,
sending read control byte & I2C read!!!
    #define V0INC 0x04          //Increment
    #define V0DEC 0x08          //Decrement

    //Volatile Wiper 1
    #define V1W0 0x10
    #define V1W1 0x11
    #define V1RR 0x1C
    #define V1INC 0x14
    #define V1DEC 0x18

    //Nonvolatile Wiper 0
    #define NV0W0 0x20
    #define NV0W1 0x21
    #define NV0RR 0x2C

    //Nonvolatile Wiper 1
    #define NV1W0 0x30
    #define NV1W1 0x31
    #define NV1RR 0x3C

    //Volatile Wiper 2
    #define V2W0 0x60
    #define V2W1 0x61
    #define V2RR 0x6C
    #define V2INC 0x64
    #define V2DEC 0x68

    //Volatile Wiper 3
    #define V3W0 0x70
    #define V3W1 0x71
    #define V3RR 0x7C
    #define V3INC 0x74
    #define V3DEC 0x78

    //Nonvolatile Wiper 2
    #define NV2W0 0x80
    #define NV2W1 0x81
    #define NV2RR 0x8C

    //Nonvolatile Wiper 3
    #define NV3W0 0x90
    #define NV3W1 0x91
    #define NV3RR 0x9C


#ifdef  __cplusplus
}
#endif
```

```
#endif  /* MCP444X6X_H */



/*
 * File:   IOC_ISR.h
 * Author: Adam Brunner
 *
 * Created on November 12, 2020, 4:41 PM
 */

#ifndef IOC_ISR_H
#define IOC_ISR_H

#ifdef  __cplusplus
extern "C" {
#endif

#include "MCP444X6X.h"
#include "I2C.h"

#define ENC0A PORTDbits.RD0 //Rotary Encoder 0 Pin A
#define ENC0B PORTDbits.RD1 //Rotary Encoder 0 Pin B

//GPIO Interrupt-on-Change Interrupt Service Routine
void __attribute__((interrupt, auto_psv)) _IOCInterrupt(void)
{
    /*
     Rotary Encoder 0 I2C Interfacing
     */
    char Flag_CWSTART = 0;
    char Flag_CW2 = 0;
    char Flag_CW3 = 0;

    char Flag_CCWSTART = 0;
    char Flag_CCW2 = 0;
    char Flag_CCW3 = 0;

    //Initial Condition Check
    if (!ENC0A && ENC0B)        //[A,B] = [0,1] - CW Step 1
    {
        Flag_CWSTART = 1;       //Start clockwise sequence
    }
    else if (ENC0A && !ENC0B)   //[A,B] = [1,0] - CCW Step 1
    {
        Flag_CCWSTART = 1;      //Start counter-clockwise sequence
    }
    else {}

    //Clockwise Turn Sequence
    while(Flag_CWSTART == 1)
    {
        if (!ENC0A && !ENC0B)   //[A,B] = [0,0] - CW Step 2
        {
            Flag_CW2 = 1;       //Move to step 2
            Flag_CWSTART = 0;
        }
```

```
    }
    while(Flag_CW2 == 1)
    {
        if (ENC0A && !ENC0B)     //[A,B] = [1,0] - CW Step 3
        {
            Flag_CW3 = 1;        //Move to step 3
            Flag_CW2 = 0;
        }
    }
    while(Flag_CW3 == 1)
    {
        if (ENC0A && ENC0B)       //[A,B] = [1,1] - Cycle complete
        {
            I2C1Start();
            I2C1SendByte(POT0W);//Send control (addressing) byte
            us_delay(100);
            I2C1SendByte(V0DEC);//Decrement volatile wiper 0 register by 1
(W->B)
            us_delay(100);
            I2C1SendByte(V1DEC);//Decrement volatile wiper 1 register by 1
            us_delay(100);
            I2C1SendByte(V2DEC);//Decrement volatile wiper 2 register by 1
            us_delay(100);
            I2C1SendByte(V3DEC);//Decrement volatile wiper 3 register by 1
            us_delay(100);
            I2C1Stop();

            Flag_CW3 = 0;        //Clean up flags
        }
    }

    //Counter-Clockwise Turn Sequence
    while(Flag_CCWSTART == 1)
    {
        if (!ENC0A && !ENC0B)    //[A,B] = [0,0] - CCW Step 2
        {
            Flag_CCW2 = 1;       //Move to step 2
            Flag_CCWSTART = 0;
        }
    }
    while(Flag_CCW2 == 1)
    {
        if (!ENC0A && ENC0B)     //[A,B] = [0,1] - CCW Step 3
        {
            Flag_CCW3 = 1;       //Move to step 3
            Flag_CCW2 = 0;
        }
    }
    while(Flag_CCW3 == 1)
    {
        if (ENC0A && ENC0B)      //[A,B] = [1,1] - Cycle complete
        {
            I2C1Start();
            I2C1SendByte(POT0W);//Send control (addressing) byte
            us_delay(100);
            I2C1SendByte(V0INC);//Increment volatile wiper 0 register by 1
(W->A)
```

```
            us_delay(100);
            I2C1SendByte(V1INC);//Increment volatile wiper 1 register by 1
            us_delay(100);
            I2C1SendByte(V2INC);//Increment volatile wiper 2 register by 1
            us_delay(100);
            I2C1SendByte(V3INC);//Increment volatile wiper 3 register by 1
            us_delay(100);
            I2C1Stop();

            Flag_CCW3 = 0;      //Clean up flags
        }
    }

    // Clear interrupt flags
    IOCFD = 0x0000;
    IOCPDF = 0;
    IFS1bits.IOCIF = 0;
}


#ifdef  __cplusplus
}
#endif

#endif  /* IOC_ISR_H */
```

5.2.4.5. Digital Control – Synthesizer Control Button Subprocess Pseudocode (ADB)

The pseudocode below describes the pushbutton subprocesses within the Synthesizer control process. Similarly to the encoder, the pulse generated from pressing the pushbutton will trigger an Interrupt-on-Change interrupt on the microcontroller, which will then run the IoC Interrupt Service Routine. This routine will consist of the encoder interfacing code seen above, as well as code derived from the following pseudocode. The code will detect which pushbutton was pressed by reading the values of the connected GPIO and will set the corresponding output voltages to trigger the switching of the MOSFET relays used in the synthesizer circuits. Note that which buttons toggle which voltages has not been fully determined yet and that the voltages listed below are arbitrary examples.

```
--Within IOC ISR function--

if(Button 0 pressed)
{
    Toggle Control Voltage 0,1
}
else if(Button 1 pressed)
{
    Toggle Control Voltage 2
}
else if(Button 2 pressed)
{
    Toggle Control Voltage 3,4,5
}
else if(Button 3 pressed)
{
    Toggle Control Voltage 6,7
}
else{}
```

5.2.4.6. Digital Control – External Digital Input Processing Pseudocode (ADB)

The external digital input processing routine of the MCU program is responsible for detecting, receiving and interpreting MIDI communications being input from an external MIDI device. The interpretation of the MIDI signals is performed by comparing the incoming signal to the defined command set and producing the corresponding response. The results of the interpretation are then used to generate an analog signal via the internal PWM module within the MCU, which will be passed to the synthesizer circuits. Note that use of the entire MIDI command set mentioned earlier in the report is not required for a proper MIDI implementation, and as such the defined command set will be developed to include only the commands necessary for producing the desired analog waveforms.

```
#define <MIDI Command Set>

if(USB MIDI Input Detected)
{
    midiIN = USB Input Data //Receive MIDI command byte from USB
}

--Compare midiIN to Command Set--
```

```
if(midiIN == MIDI Command)
{
    --Update PWM based on matched command--
    Examples:
    if(midiIN == Note On)    //Command byte = 0x90-9F
    {
        notePitch = USB Input Data  //Receive pitch data byte
        PWM Output Frequency = notePitch    //Set PWM output frequency to
pitch frequency
    }
    if(midiIN == Note Off)
    {
        PWM Output Frequency = 0
    }
}
```

5.2.4.7. Remote Application – Recording Process Pseudocode (SG)

The pseudocode below describes the general processes necessary for the remote app to communicate with the MCU within the synthesizer. The recording process will involve the remote app initiating a conversation with the synthesizer and keeping an open channel as the synthesizer responds back. The synthesizer will continue to send back to the remote app until either the remote app sends the stop command or until a timer within the synthesizer clocks out or the connection is disconnected in some way. This will allow for the remote app to have full control over the recording process while allowing the synth to have a failsafe mechanism should something occur to the remote app and its connection.

--Recording Process Pseudo Code

```
if(!CONNECTED)
 {
    Try
    {
        Open_Connection();
    };
    Catch(e)
    {
        print('Error: %d', e);
        End;
    };
 }
```

```
Open_File(File);
 Send_Start_Signal();
 While(!STOP_RECORDING_BUTTON)
 {
    Listen_to_Connection();
    Input -> File;
 }
 Send_Stop_Signal();
While(!ACK) {}
 Close_File(File);
```

5.2.4.8. Remote Application – Syncing Remote Application and Synth Pseudocode (SG)

The pseudocode below describes the processes for the remote application to upload the current configuration on the remote application to the synthesizer, synchronizing the two. As well as the pseudocode for synchronizing the remote application to the settings on the synthesizer. The specifics of the communications will be utilized from the WiFi ATWINC libraries. To accommodate data integrity, data transactions will be designed as an 'all-or-nothing' principle. This principle states that if data is to be edited, all the edits will be made at the same time. This will operate as a safety procedure to accommodate the possibility of error in data transmission and communication, so that the program will not be left with incomplete data.

--Control Profile Read from Synth Pseudocode

```
If(!CONNECTED)
 {
    Try
    {
        Open_Connection();
    };
    Catch(e)
    {
        print('Error: %d', e);
        End;
    };
 }
 Send_Read_Signal();
 While(!END_OF_DATA)
 {
```

```
    Signal -> Temp_Variables;
 }
 If(No Error)
 {
    Temp_Variables[] -> UI.vars[];
 }
 Else
 {
    print('Error %d', 'Error handling variables');
    end;
 }
```

--Upload Current Configuration to Synth Pseudocode

```
if(!CONNECTED)
 {
    Try
    {
        Open_Connection();
    };
    Catch(e)
    {
        print('Error %d',e);
        End;
    };
 }
UI.vars[] -> temp_vars[];
 Send_Signal_for_MCU_to_listen();
 for(int I = 0; I < sizeof(temp_vars);I++)
 {
    Transmit_MSG(temp_vars[I]);
 }
end_transmission_signal();
```

5.2.4.9. Remote Application – Save/Load Profile Pseudocode (SG)

The pseudocode below describes the processes for the remote application to save and load different sound profiles to the synthesizer. This will allow for the user to load sound profiles from an external source, save the current configuration displayed on the remote application and save the current configuration of the synthesizer. The communication specifics of operating the WiFi Direct communication will be handled by the ATWINC1500 libraries used to interface

with the ATWINC1500 breakout board. The Save/Load Profile functionality will be designed to work together with the data synchronization functions mentioned in the previous section.

--Control Profile Save From UI Pseudocode

```
Open_file(file); //for Storage
 while(!file.eof())
 {
    Read_UI_variables();
    Write_values_to_File_with_formatting(file);
 }
 Close_file(file);
```

--Control Profile Save From Synth Pseudocode

```
try
{
    [Control Profile Read From Synth]
    [Control Profile Save From UI]
}
catch(e)
{
    Print('Error %d',e);
    End;
}
```

--Control Profile Load Pseudocode

```
Validate_File_Type(inFile); //for Storage

int I = 0;

try

{

    while(!inFile.eof())

    {

        inFile.line -> UI.vars[I];

        I++;

    }

}

catch(e)

{

    print('Error %d',e);
```

```
    end;
}
[Upload Current Config to Synth];
```

5.2.4.10. Remote Application – User Interface Server File (SG)

The code below is the web-server creation file. The code below is run to start the local server to host the associated HTML files and use any attached JavaScript file. Using JavaScript, particularly the HTTP module in Node.js, the code opens a connection to a web browser on the localhost network, allowing for the local host to access the content in the form of a web-page. This local host will be primarily used in the testing of the functionality of the associated code as well as being the main process through which the remote application is run.

```
/* File: serverFile.js
 * Author: Scott Grisso
 *     Taken from a Node.js web server example and modified for test use
 * Description: A JavaScript main file to run that will open the
HomePage.html on the server
 * http://localhost:1337/
*/
const http = require("http");
const fs = require('fs').promises;
const host = 'localhost';
var port = process.env.PORT || 1337;
let indexFile;
const requestListener = function (req, res) {
    res.setHeader("Content-Type", "text/html");
    res.writeHead(200);
```

```
    res.end(indexFile);
};
const server = http.createServer(requestListener);
fs.readFile(__dirname + "/HomePage.html")
    .then(contents => {
        indexFile = contents;
        server.listen(port, host, () => {
            console.log(`Server is running on http://${host}:${port}`);
        });
    })
    .catch(err => {
        console.error(`Could not read index.html file: ${err}`);
        process.exit(1);
    });
```

5.2.4.11. Remote Application – User Interface HTML Main Code (SG)

The code below was developed to implement a first iteration of the remote application user interface to demonstrate the functionality and processes that will be present within the project. The program was developed using Visual Studio to run a web-page example built using HTML and scripted using JavaScript. The attached code is a sample of some of the more unique implementations of HTML and CSS used to implement unique functionality. The HTML code features three unique elements that occur in multiple places throughout the main program, the sliders, toggle switches, and three-way radio switches. All the elements used are objects of testing referenced from examples and may not be representative of the finalized product.

These elements operate as a mixture of HTML formatting with CSS styling and JavaScript functions to achieve the desired result of a user interface similar in function to that of

a physical interface for a synthesizer. The sliders are constructed using specialized CSS styling to create the individual parts of the slider that interact together to create a functional slider for the user. The sliders consist of three individual elements, the slide container, the slider, and the slide wedge. The Slide container is an invisible box that tells the other elements what the bounds of the slier will be. The slider is the slider itself, appearing as a rail that the slide wedge will move along. Finally, the slide wedge is the interactable part of the slider which sets determines the value along the slider, never exceeding the bounds of the slider.

The toggle switches operate on a similar level to the sliders but on a fixed binary output instead of a more continuous slider from point a to point b. The toggle switch acts whenever the user clicks on the switch instead of having the user drag an element along the background. The toggle switch consists of three elements with a handful of additional stylings to add motion and animation to the element. Like the slider, the toggle switch has an invisible container to hold the elements as well as a visible container akin to the slider rail in the sliders. Instead of a moveable wedge however, the switch consists of a button that moves from one side to another depending on which side is active.

The three-way switch used in the Oscillator Wave setting operates more conventionally to the other two primary elements. The three-way wave setting works with three different radio buttons that check which button is pressed on an update/click, shutting off the buttons that are not selected and setting the appropriate one as selected. Despite operating differently to the other elements, it maintains its consistency of only allowing one option to be pressed and one value to be present at a time keeping data consistent and minimizing error.

```
<!--Slider Styling-->
<style>
```

```
    .slidecontainer {
        width: 100%;
    }
    .slider {
        -webkit-appearance: none;
        width: 50%;
        height: 10px;
        border-radius: 5px;
        background: #d3d3d3;
        outline: none;
        opacity: 0.7;
        -webkit-transition: .2s;
        transition: opacity .2s;
    }
    .slider:hover {
        opacity: 1;
    }
    .slider::-webkit-slider-thumb {
        -webkit-appearance: none;
        width: 15px;
        height: 16px;
        border-radius: 50%;
        background: #4CAF50;
        cursor: pointer;
    }
</style>
<!--Switch Styling-->
<style>
    .toggle-container {
```

```
    position: relative;
    display: inline-block;
    width: 60px;
    height: 34px;
}
.toggle-container input {
    opacity: 0;
    width: 0;
    height: 0;
}
.toggle {
    position: absolute;
    cursor: pointer;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background-color: #ccc;
    -webkit-transition: .4s;
    transition: .4s;
}
.toggle:before {
    position: absolute;
    content: "";
    height: 26px;
    width: 26px;
    left: 4px;
    bottom: 4px;
    background-color: white;
```

```
        -webkit-transition: .4s;
        transition: .4s;
    }
    input:checked + .toggle {
        background-color: #2196F3;
    }
    input:focus + .toggle {
        box-shadow: 0 0 1px #2196F3;
    }
    input:checked + .toggle:before {
        -webkit-transform: translateX(26px);
        -ms-transform: translateX(26px);
        transform: translateX(26px);
    }
</style>
<!--Frequency Slider-->
<div class="slidecontainer col-6">
    <div>
        <div class="setting col-3">Frequency</div>
        <div class="output col-3" id="OSC_Frequency"></div>
    </div>
    <input type="range" min="0" max="100" value="50" class="slider"
id="OSC_Frequency_Slider">
</div>
<!--Three-Way Wave Switch-->
<div class="switch-toggle switch-3 switch-candy row col-6">
    <div class="col-2">
        <input id="on" onclick="setValue('OSC_Wave','Triangle Wave')"
name="state-d" type="radio" checked="" />
```

```
        <label for="on">Triangle</label>
    </div>
    <div class="col-2">
        <input id="na" onclick="setValue('OSC_Wave','Square Wave')"
name="state-d" type="radio" />
        <label for="na">Square</label>
    </div>
    <div class="col-2">
        <input id="off" onclick="setValue('OSC_Wave','Sine Wave')"
name="state-d" checked="checked" type="radio" />
        <label for="off">Sine</label>
    </div>
    <label class="output col-3" id="OSC_Wave">Sine Wave</label>
</div>
<!--Toggle Switch-->
<div class="col-1">
    <label class="toggle-container">
        <input onClick="toggleEN('DIS_Enable')" type="checkbox">
        <span class="toggle"></span>
    </label>
</div>
```

5.2.4.12. Remote Application – User Interface JavaScript Main Code (SG)

The code below was developed to implement a first iteration of the remote application user interface to demonstrate the functionality and processes that will be present within the project. The program was developed using Visual Studio to run a web-page example built using

HTML and scripted using JavaScript. The functionality of this code is to illustrate the interactions between the remote application and the synthesizer potentiometers.

The attached code is the JavaScript associated with the HomePage.html file for the current iteration of the remote application. The JavaScript controls the different UI elements that the user can interact with on the remote application. The sliders call a function to update the value associated with them whenever the user slides the slider. A similar process is used for the toggle switches, which activate on click to toggle the value. These values are then grabbed and used whenever a function is called that reads the UI, allowing for the user to see clearly what the current values of each setting are. The code below is subject to change and may not be indicative of the final product.

```
<!--

    File: Home.html

    Author: Scott Grisso

    Description: Initial draft of User Interface for Remote Application

    Used in project Demo on 11/16/2020

-->

<!DOCTYPE html>
<script type="text/javascript" src="include.js"></script>
<body>…</body></html>
<!--SLIDERS-->
<div>
    <script>
        var OSC_Frequency_Slider =
```

```
document.getElementById("OSC_Frequency_Slider");
        var OSC_Frequency_Output = document.getElementById("OSC_Frequency");
        OSC_Frequency_Output.innerHTML = OSC_Frequency_Slider.value;
        OSC_Frequency_Slider.oninput = function () {
            OSC_Frequency_Output.innerHTML = this.value;
        }
    </script>
    <script>
        var OSC_Wave_Slider = document.getElementById("OSC_Wave_Slider");
        var OSC_Wave_Output = document.getElementById("OSC_Wave");
        OSC_Wave_Output.innerHTML = OSC_Wave_Slider.value;
        OSC_Wave_Slider.oninput = function () {
            OSC_Wave_Output.innerHTML = this.value;
        }
    </script>
    <!--REPEAT THE ABOVE SCRIPT WITH EACH SLIDER-->
</div>
<!--TOGGLE SWITCHES-->
<script>
    function setValue(locationID, Value) {
        var output = document.getElementById(locationID);
        output.innerHTML = Value;
    }
    function toggleDISLevel() {
        var output = document.getElementById("DIS_Level");
        curVal = output.innerHTML;
        if (curVal == "Soft Distortion")
            output.innerHTML = "Hard Distortion";
        else
```

```
            output.innerHTML = "Soft Distortion";
        }
        function toggleEN(outputID) {
            var output = document.getElementById(outputID);
            curVal = output.innerHTML;
            if (curVal == "Off") {
                output.innerHTML = "On";
                output.hidden = false;
            }
            else {
                output.innerHTML = "Off";
                output.hidden = true;
            }
        }
    </script>
    <!--Show Values of All Settings-->
    <script>
        function showResults() {
            var popup = document.getElementById("Results");
            var OSC_Frequency =
    document.getElementById("OSC_Frequency").innerHTML;
            popup.innerHTML = OSC_Frequency;
            var OSC_Wave = document.getElementById("OSC_Wave").innerHTML;
            var VC_Volume = document.getElementById("VC_Volume").innerHTML;
            var REV_Total = document.getElementById("REV_Total").innerHTML;
            var REV_In = document.getElementById("REV_In_Filter").innerHTML;
            var REV_Out = document.getElementById("REV_Out_Filter").innerHTML;
            var TONE_Low = document.getElementById("TONE_Low").innerHTML;
            var TONE_Mid = document.getElementById("TONE_Mid").innerHTML;
```

```
        var TONE_High = document.getElementById("TONE_High").innerHTML;
        var DIS_Gain = document.getElementById("DIS_Gain").innerHTML;
        var REV_Enable = document.getElementById("REV_Enable").innerHTML;
        var DIS_Enable = document.getElementById("DIS_Enable").innerHTML;
        var DIS_Soft_Hard = document.getElementById("DIS_Level").innerHTML;
        var results = "Frequency: " + OSC_Frequency + "<br />";
        results = results + "Wave: " + OSC_Wave + "<br />";
        results = results + "Volume: " + VC_Volume + "<br />";
        results = results + "Reverb: " + REV_Enable + "<br />";
        results = results + "      : " + REV_Total + "<br />";
        results = results + "Reverb In: " + REV_In + "<br />";
        results = results + "Reverb Out: " + REV_Out + "<br />";
        results = results + "Tone Low: " + TONE_Low + "<br />";
        results = results + "Tone Mid: " + TONE_Mid + "<br />";
        results = results + "Tone High: " + TONE_High + "<br />";
        results = results + "Distortion: " + DIS_Enable + "<br />";
        results = results + "Distortion Gain: " + DIS_Gain + "<br />";
        results = results + "Soft Or Hard Distortion: " + DIS_Soft_Hard +
"<br />";
        popup.innerHTML = results;
    }
```

```
</script>
```

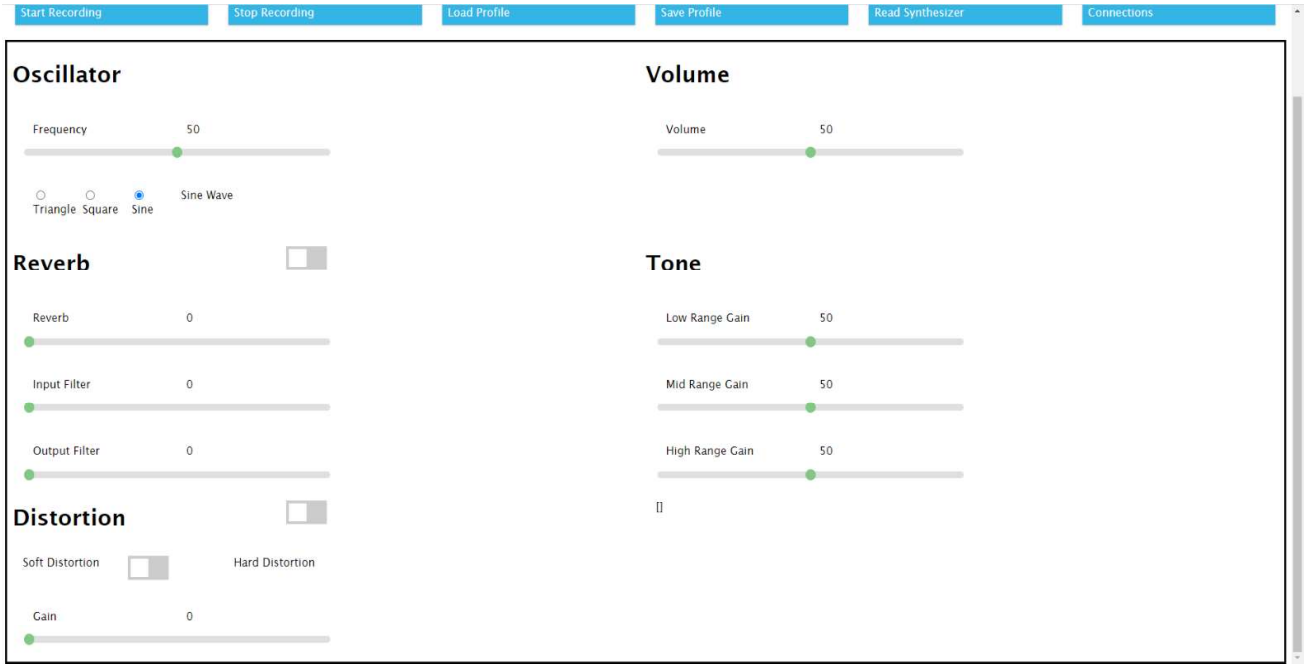


Figure 37: Webpage render of current remote application user interface.

6. Mechanical Sketch (ADB)

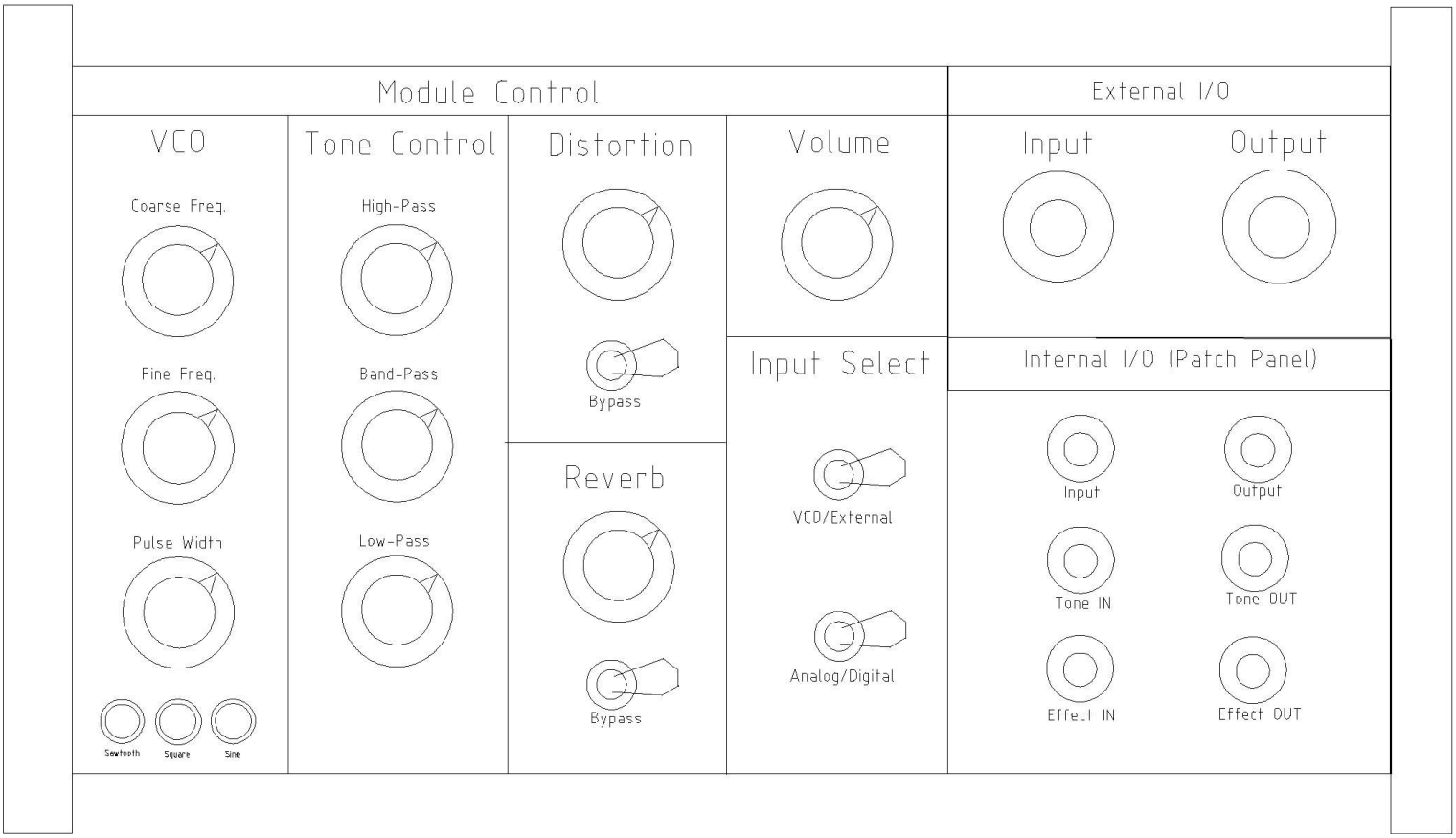


Figure 38: Mechanical sketch of proposed system

7. Design Team Information

- Adam Brunner, Computer Engineering
- Andrew Cihon-Scott, Electrical Engineering
- Scott Grisso, Computer Engineering
- Linus Wright, Electrical Engineering

8. Parts Information

8.1. Cumulative Parts Lists

Table 41: Cumulative parts list for Digital Control Schematic (ADB)

Quantity	Ref. Des	Part Number	Description
1	C40	CC0603JRX7R9BB101	100pF 0603 Ceramic Capacitor
15	C1-C6, C8, C10, C19, C21, C22, C25-C28	CC0603JRX7R8BB104	0.1uF 0603 Ceramic Capacitor
2	C7, C9	CL21A106KAYNNNG	10uF 0805 Ceramic Capacitors
19	C11-C18, C29-C39	CL10B103KB8NNNC	10nF 0603 Ceramic Capacitors
1	C20	GRM31CR60J157ME11L	150uF 1206 Ceramic Polarized Capacitor
2	C23, C24	CL10A106MQ8NNNC	10uF 0603 Ceramic Capacitors
1	F1	NANOSMDH075F-02	Resettable 1206 0.75A Fuse
1	IC1	MCP3561-E/ST	24-bit Σ-Δ ADC TSSOP-20
1	J3	MTJ-661X1	6-pin RJ11 Jack
1	J4	USB3070-30-A	USB Type Micro-B Connector
40	R1, R3-R12, R15-R24, R29-R47	RC0603FR-0710KL	10kΩ 0603 Resistors
1	R2	RC0603FR-07470RL	470 Ω 0603 Resistor
2	R13, R14	RC0603FR-072K2L	2.2kΩ 0603 Resistors
2	R25, R26	RC0603FR-072KL	2.0kΩ 0603 Resistors
2	R27, R28	RC0603FR-071KL	1.0kΩ 0603 Resistors
6	S1-S6	COM-00097	SPST Pushbutton Switches
1	U1	PIC24FJ1024GB610-I/PT	PIC24FJ1024GB610 100-TQFP MCU
1	U2	MCP4461-503E/ST	50kΩ 8-bit I^2C 4-Channel Potentiometers
3	U3-U5	MCP4461-104E/ST	100kΩ 8-bit I^2C 4-Channel Potentiometers
7	U6-U12	PEC12R-4017F-N0024	24-Pulse Mechanical Rotary Encoders
1	J5	PRT-00115	Female Pin Headers
1	J1, J2	PRT-00116	Male Pin Headers
1	J1, J2	PRT-09044	2-Pin Jumper
1	-	2999	Adafruit ATWINC1500 WiFi Breakout

Table 42: Cumulative parts list for Reverb Drive and Oscillator Schematics (ACS)

Quantity	Ref. Des	Part Number	Description
7	C1 – C7	CC0603JRX7R8BB104	0.1uF 0603 Ceramic Capacitor
2	R11, R12	CRGCQ0603F10R	Fixed Resistor CRGCQ 0603 (10)
1	R17	CRGCQ0603F10K	Fixed Resistor CRGCQ 0603 (10k)
1	R16	CRGCQ0603F150R	Fixed Resistor CRGCQ 0603 (150)
2	D1, D2	DO41-10	Diode Standard 100V 1A Through Hole DO-41
4	R5, R6, R9	CRGCQ0603F1K0	Fixed Resistor CRGCQ 0603 (1k)
1	R2	RC0603FR-0730KL	30kΩ 0603 Resistors
1	R1	CRGCQ0603F35K	Fixed Resistor CRGCQ 0603 (35k)
2	R13, R14	CRGCQ0603F6K8	Fixed Resistor CRGCQ 0603 (6.8k)
3	U1, U2, U3	NE5532P	Dual-Channel NE5532 Op-Amp
1	Q1	ZXTN25100DGTA	Bipolar (BJT) Transistor NPN 100V 3A 175MHz 3W Surface Mount SOT-223
1	Q2	MJD32CT4G	Bipolar (BJT) Transistor PNP 100V 3A 3MHz 1.56W Surface Mount DPAK

Table 43: Cumulative parts list for Reverb Overvoltage Protection Schematic (ACS)

Quantity	Ref. Des	Part Number	Description
9	R11, R1	CRGCQ0603F22K	Fixed Resistor CRGCQ 0603
3	C5, C6,	GRM033C71A104KE14D	CAPC0603X33N
1	C1	GRM033C71A104KE14D	CAPC0603X33N
1	C2	GRM033C71A104KE14D	CAPC0603X33N
7	D1, D2,	1N4004	DO41-10
1	R5	CRGCQ0603F1K	Fixed Resistor CRGCQ 0603
1	R2	CRGCQ0402F2K2	Fixed Resistor CRGCQ 0603
1	R3	CRGCQ0402F560	Fixed Resistor CRGCQ 0603
2	U1, U3	NE5532P	DIP794W45P254L959H508Q8
1	Q1	TIP41C	TO220AV
1	Q2	TIP42C	TO220AV
1	R6	CRGCQ0603FxxK	TBD

Table 44: Cumulative parts list for Reverb Volume Mixer Schematic (ACS)

Quantity	Ref. Des	Part Number	Description
1	R3	CRGCQ0402F100K	Fixed Resistor CRGCQ 0603
2	C1, C4	GRM033C71A104KE14D	CAPC0603X33N
1	R6	CRGCQ0402F1K	Fixed Resistor CRGCQ 0603
3	R9, R10, R11	CRGCQ0402F1K	Fixed Resistor CRGCQ 0603
1	R2	CRGCQ0402F2k2	Fixed Resistor CRGCQ 0603
1	R1	CRGCQ0402F220	Fixed Resistor CRGCQ 0603

1	R4	CRGCQ0402FxxK	TBD
1	C2	GRM033C71A104KE14D	CAPC0603X33N
1	C3	GRM033C71A104KE14D	CAPC0603X33N
3	U1, U2, U3	NE5532P	DIP794W45P254L959H508Q8
1	R7	CRGCQ0402FxxK	TBD

Table 45: Cumulative parts list for Tone Control Schematic (LWW)

Quantity	Ref. Des	Part Number	Description
10	C1-C8, C17, C22	C0603C272F8HAC7867	2.7n 0603 Capacitors
14	C9-C16, C18-C21, C23, C24	CGA3E2X7R1H472K080AA	47n 0603 Capacitors
4	R1, R23-R24	RC0603FR-071KL	1k 0603 Resistors
6	R2, R26-R30	RC0603FR-07100KL	100k 0603 Resistors
14	R3-R4, R7-R14, R17-R20	RC0603FR-0710KL	10k 0603 Resistors
6	R5-R6, R15-R16, R21-R22	RC0603FR-0720KL	20k 0603 Resistors
10	U1-U10	NE5532P	Dual-Channel NE5532 Op-Amp

Table 46: Cumulative parts list for Distortion Schematic (LWW)

Quantity	Ref. Des	Part Number	Description
4	D1, D2, D5, D6	1N4007	Diodes
1	R1	RC0603FR-072KL	2.0kΩ 0603 Resistors
1	R2	RC0603FR-07100KL	100kΩ 0603 Resistors
1	R3	RT0603BRD0750KL	50kΩ 0603 Resistors
1	U1	NE5532P	Dual-Channel NE5532 Op-Amp
3	U2-U4	CPC1017N	SPST Normally Open Relay

Table 47: Cumulative parts list for Power Supply Schematic (LWW)

Quantity	Ref. Des	Part Number	Description
1	SV1-SV2	PD2515	+15, -15V Switching Supply
1	U1	LM1086IT-5.0	5V Voltage Regulator
1	U2	LM1086IT-3.3	3.3V Voltage Regulator

Table 48: Cumulative parts list for Volume Control Schematic (ACS)

Quantity	Ref. Des	Part Number	Description
2	R3, R4	CRGCQ0402F1K	0603 1kΩ Resistors
1	U1	NE5532P	Dual-Channel NE5532 Op-Amp

Table 49: Cumulative parts list for Input Select Circuit (ACS + LWW)

Quantity	Ref. Des	Part Number	Description
4	R1, R2, R3, R4	CRGCQ0603F100K	100k 0603 Resistors
3	U1, U2, U3	CPC1017N	State Relay SMT
1	U4	NE5532P	Op Amp

8.2. Materials Budget (ADB)

The following list describes the estimated budget required for the components given in the Cumulative Parts Lists to produce the presented schematics. Note that some of the part numbers listed in the Cumulative Parts Lists are not shown here due to the consolidation and substitutions made from those lists. Any parts omitted in the following table have been substituted with equivalent replacements.

Table 50: Project materials budget list, consolidated from Cumulative Parts Lists (ADB)

Quantity	Part Number	Description	Unit Cost	Total Cost
1	2999	Adafruit ATWINC1500 WiFi Breakout	$24.95	$24.95
7	1N4004RLG	Diode Standard 400V 1A Through Hole DO-41	$0.20	$1.40
4	1N4007RLG	Diode Standard 1000V 1A Through Hole DO-41	$0.18	$0.72
2	22AR10KLFTR	10 kOhms 0.125W, 1/8W J Lead Surface Mount Trimmer Potentiometer Cermet 1 Turn Top Adjustment	$1.26	$2.52
1	3314J-1-223E	22 kOhms 0.25W, 1/4W J Lead Surface Mount Trimmer Potentiometer Cermet 1 Turn Top Adjustment	$1.98	$1.98
1	AC0603KRX7R8BB223	0.022µF ±10% 25V Ceramic Capacitor X7R 0603 (1608 Metric)	$0.13	$0.13
2	CC0603JPX7R9BB104	100nF 0603 Ceramic Capacitors	$0.11	$0.22
1	CC0603JRNPO8BN471	470pF ±5% 25V Ceramic Capacitor C0G, NP0 0603 (1608 Metric)	$0.11	$0.11

19	CC0603JRX7R8BB104	0.1uF 0603 Ceramic Capacitors	$0.11	$2.09
1	CC0603JRX7R9BB101	100pF 0603 Ceramic Capacitors	$0.11	$0.11
7	CL10A106MQ8NNNC	10uF 0603 Ceramic Capacitors	$0.11	$0.77
19	CL10B103KB8NNNC	10nF 0603 Ceramic Capacitors	$0.10	$1.90
10	CL10B272KB8NNNC	2.7nF 0603 Capacitors	$0.10	$1.00
14	CL10B473JB8NNNC	47nF 0603 Capacitors	$0.10	$1.40
2	CL21A106KAYNNNG	10uF 0805 Ceramic Capacitors	$0.20	$0.40
6	COM-00097	SPST Pushbutton Switches	$0.35	$2.10
6	CPC1017N	Solid State SPST-NO (1 Form A) 4-SOP (0.150", 3.81mm)	$0.72	$4.32
1	GRM31CR60J157ME11L	150uF 1206 Ceramic Polarized Capacitor	$1.14	$1.14
1	LD1117S50TR	5V Voltage Regulator	$0.41	$0.41
1	LM1086IT-3.3	3.3V Voltage Regulator	$0.41	$0.41
1	MCP3561-E/ST	24-bit Σ-Δ ADC TSSOP-20	$2.95	$2.95
3	MCP4461-104E/ST	100kΩ 8-bit I2C 4-Channel Potentiometers	$1.88	$5.64
1	MCP4461-503E/ST	50kΩ 8-bit I2C 4-Channel Potentiometers	$1.88	$1.88
1	MJD32CT4G	Bipolar (BJT) Transistor PNP 100V 3A 3MHz 1.56W Surface Mount DPAK	$0.58	$0.58
1	MTJ-661X1	6-pin RJ11 Jack	$0.48	$0.48
1	NANOSMDH075F-02	Resettable 1206 0.75A Fuse	$2.21	$2.21
20	NE5532P	Dual-Channel NE5532 Op-Amp	$0.56	$11.20
1	PD2515	+15, -15V Switching Supply	$17.22	$17.22
7	PEC12R-4017F-N0024	24-Pulse Mechanical Rotary Encoders	$0.89	$6.23
1	PIC24FJ1024GB610-I/PT	PIC24FJ1024GB610 100-TQFP MCU	$4.70	$4.70
1	PRT-00115	Female Pin Headers	$1.50	$1.50
1	PRT-00116	Male Pin Headers	$1.50	$1.50
1	PRT-09044	2-Pin Jumper	$0.35	$0.35
12	RC0603FR-07100KL	100k 0603 Resistors	$0.10	$1.20
1	RC0603FR-07220RL	220Ω 0603 Resistors	$0.10	$0.10
55	RC0603FR-0710KL	10kΩ 0603 Resistors	$0.10	$5.50
2	RC0603FR-0710RL	10Ω 0603 Resistors	$0.10	$0.20
1	RC0603FR-07150RL	150Ω 0603 Resistors	$0.10	$0.10
17	RC0603FR-071KL	1.0kΩ 0603 Resistors	$0.10	$1.70
2	RC0603FR-0722KL	22kΩ 0603 Resistors	$0.10	$0.20
4	RC0603FR-072K2L	2.2kΩ 0603 Resistors	$0.10	$0.40
1	RC0603FR-0734K8L	34.8kΩ 0603 Resistors	$0.10	$0.10
1	RC0603FR-07560RL	560Ω 0603 Resistors	$0.10	$0.10
2	RC0603FR-076K8L	6.8kΩ 0603 Resistors	$0.10	$0.20
6	RC0603FR-0720KL	20k 0603 Resistors	$0.10	$0.60

3	RC0603FR-072KL	2.0kΩ 0603 Resistors	$0.10	$0.30
1	RC0603FR-0730KL	30kΩ 0603 Resistors	$0.10	$0.10
1	RC0603FR-07470RL	470 Ω 0603 Resistor	$0.10	$0.10
1	RC0603FR-0749K9L	49.9kΩ 0603 Resistors	$0.10	$0.10
1	USB3070-30-A	USB Type Micro-B Connector	$0.67	$0.67
1	ZXTN25100DGTA	Bipolar (BJT) Transistor NPN 100V 3A 175MHz 3W Surface Mount SOT-223	$0.56	$0.56
			Total	$116.75

9. Project Schedules

9.1. Final Design Gantt Chart (ADB)

Final Parts Request Form	**11 days**	**Mon 10/12/20**	**Mon 10/26/20**	55	**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
Final Parts Req Due	*1 day*	*Mon 10/26/20*	*Mon 10/26/20*		
Subsystems Demonstrations Begin	***18 days***	***Mon 10/19/20***	***Wed 11/11/20***		
Hardware Demo	**18 days**	**Mon 10/19/20**	**Wed 11/11/20**		**Adam Brunner,Andrew Cihon-Scott,Linus Wright**
Oscillator Circuit	18 days	Mon 10/19/20	Wed 11/11/20		**Andrew Cihon-Scott**
Distortion Circuit	18 days	Mon 10/19/20	Wed 11/11/20		**Linus Wright**
Digital Pot Interfacing	18 days	Mon 10/19/20	Wed 11/11/20		**Adam Brunner,Andrew Cihon-Scott**
Software Demo	**18 days**	**Mon 10/19/20**	**Wed 11/11/20**		**Scott Grisso**
Remote App	18 days	Mon 10/19/20	Wed 11/11/20		**Scott Grisso**
Final Design Report	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**	4	
Frontal Material	37 days?	Tue 10/6/20	Wed 11/25/20		**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
Cover page	37 days	Tue 10/6/20	Wed 11/25/20		
Table of Contents	37 days	Tue 10/6/20	Wed 11/25/20		
List of Figures	37 days	Tue 10/6/20	Wed 11/25/20		
List of Tables	37 days	Tue 10/6/20	Wed 11/25/20		
Abstract	37 days?	Tue 10/6/20	Wed 11/25/20		**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
Problem Statement	37 days?	Tue 10/6/20	Wed 11/25/20		**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
Need	37 days	Tue 10/6/20	Wed 11/25/20		
Objective	37 days	Tue 10/6/20	Wed 11/25/20		
Background	37 days	Tue 10/6/20	Wed 11/25/20		
Marketing Requirements	37 days	Tue 10/6/20	Wed 11/25/20		
Engineering Analysis	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**		**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
Circuits (DC, AC, Power, ...)	37 days?	Tue 10/6/20	Wed 11/25/20		**Linus Wright**
Electronics (analog and digital)	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**		**Andrew Cihon-Scott,Linus Wright**
Input Select	37 days?	Tue 10/6/20	Wed 11/25/20		**Andrew Cihon-Scott**
Reverb Circuit	37 days?	Tue 10/6/20	Wed 11/25/20		**Andrew Cihon-Scott**
Distortion Circuit	37 days?	Tue 10/6/20	Wed 11/25/20		**Linus Wright**
Tone Control	37 days?	Tue 10/6/20	Wed 11/25/20		**Linus Wright**
Volume Amplifier	37 days?	Tue 10/6/20	Wed 11/25/20		**Andrew Cihon-Scott**
Signal Processing	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**		**Adam Brunner,Andrew Cihon-Scott**
Synthesizer Signal Processing	37 days?	Tue 10/6/20	Wed 11/25/20		**Andrew Cihon-Scott**
A/D & D/A Conversion	37 days?	Tue 10/6/20	Wed 11/25/20		**Adam Brunner**
MIDI Input Processing	37 days?	Tue 10/6/20	Wed 11/25/20		**Adam Brunner**
Communications (analog and digital)	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**		**Adam Brunner**
Remote Host Communication & Data Rate	37 days?	Tue 10/6/20	Wed 11/25/20		**Adam Brunner**
Embedded Peripheral Interfacing	37 days?	Tue 10/6/20	Wed 11/25/20		**Adam Brunner**
Computer Networks	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**		**Scott Grisso**
Wireless Peer-to-Peer Networking	37 days?	Tue 10/6/20	Wed 11/25/20		**Scott Grisso**
Remote Application Design	37 days?	Tue 10/6/20	Wed 11/25/20		**Scott Grisso**
Embedded Systems	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**		**Adam Brunner**
System Specifications	37 days?	Tue 10/6/20	Wed 11/25/20		**Adam Brunner**
Engineering Requirements Specification	37 days	Tue 10/6/20	Wed 11/25/20		**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
Engineering Standards Specification	37 days?	Tue 10/6/20	Wed 11/25/20		**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
Safety	35 days	Tue 10/6/20	Mon 11/23/20		
Communication	35 days	Tue 10/6/20	Mon 11/23/20		
Data Formats	35 days	Tue 10/6/20	Mon 11/23/20		
Design Methods	35 days	Tue 10/6/20	Mon 11/23/20		
Programming Languages	35 days	Tue 10/6/20	Mon 11/23/20		
Connector Standards	35 days	Tue 10/6/20	Mon 11/23/20		

Accepted Technical Design	37 days?	Tue 10/6/20	Wed 11/25/20	**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
Hardware Design: Phase 2	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**	**Adam Brunner,Andrew Cihon-Scott,Linus Wright**
Schematics	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**	**Adam Brunner,Andrew Cihon-Scott,Linus Wright**
Digital Control	37 days	Tue 10/6/20	Wed 11/25/20	Adam Brunner
Oscillator/Input Select	37 days?	Tue 10/6/20	Wed 11/25/20	Andrew Cihon-Scott
Tone Control	37 days?	Tue 10/6/20	Wed 11/25/20	Linus Wright
Distortion	37 days?	Tue 10/6/20	Wed 11/25/20	Linus Wright
Reverb In/Out Conditioning	37 days?	Tue 10/6/20	Wed 11/25/20	Andrew Cihon-Scott
Volume Control	37 days?	Tue 10/6/20	Wed 11/25/20	Andrew Cihon-Scott
Power Supply	37 days?	Tue 10/6/20	Wed 11/25/20	Linus Wright
Software Design: Phase 2	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**	**Adam Brunner,Scott Grisso**
Code (Working Subsystems)	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**	Adam Brunner
Digital Potentiometer Interfacing	37 days	Tue 10/6/20	Wed 11/25/20	Adam Brunner
Pseudo Code (Systems Integration)	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**	**Adam Brunner,Scott Grisso**
Digital Control Processes	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**	**Adam Brunner**
Push-Button Synth Control	37 days	Tue 10/6/20	Wed 11/25/20	Adam Brunner
External Input Processing	37 days	Tue 10/6/20	Wed 11/25/20	Adam Brunner
Profile Save/Load Processes	37 days	Tue 10/6/20	Wed 11/25/20	Adam Brunner
Recording Processes	37 days	Tue 10/6/20	Wed 11/25/20	Adam Brunner
Remote Application	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**	**Scott Grisso**
Recording Processes	37 days	Tue 10/6/20	Wed 11/25/20	Scott Grisso
Control Settings Read/Upload	37 days	Tue 10/6/20	Wed 11/25/20	Scott Grisso
Device Connection Management	37 days	Tue 10/6/20	Wed 11/25/20	Scott Grisso
On-Board Profile Save/Load	37 days	Tue 10/6/20	Wed 11/25/20	Scott Grisso
Parts Lists	37 days?	Tue 10/6/20	Wed 11/25/20	**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
Parts List(s) for Schematics	**37 days?**	**Tue 10/6/20**	**Wed 11/25/20**	**Adam Brunner,Andrew Cihon-Scott,Linus Wright**
Digital Control	37 days	Tue 10/6/20	Wed 11/25/20	Adam Brunner
Oscillator/Input Select	37 days?	Tue 10/6/20	Wed 11/25/20	Andrew Cihon-Scott
Tone Control	37 days?	Tue 10/6/20	Wed 11/25/20	Linus Wright
Distortion	37 days?	Tue 10/6/20	Wed 11/25/20	Linus Wright
Reverb In/Out Conditioning	37 days?	Tue 10/6/20	Wed 11/25/20	Andrew Cihon-Scott
Volume Control	37 days?	Tue 10/6/20	Wed 11/25/20	Andrew Cihon-Scott
Power Supply	37 days?	Tue 10/6/20	Wed 11/25/20	Linus Wright
Materials Budget list	37 days?	Tue 10/6/20	Wed 11/25/20	**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
Project Schedules	37 days?	Tue 10/6/20	Wed 11/25/20	**Adam Brunner**
Final Design Gantt Chart	37 days?	Tue 10/6/20	Wed 11/25/20	**Adam Brunner**
Proposed Implementation Gantt Chart	37 days?	Tue 10/6/20	Wed 11/25/20	**Adam Brunner**
Conclusions and Recommendations	37 days?	Tue 10/6/20	Wed 11/25/20	**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
References	37 days?	Tue 10/6/20	Wed 11/25/20	**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**
Appendices	37 days?	Tue 10/6/20	Wed 11/25/20	**Adam Brunner,Andrew Cihon-Scott,Linus Wright,Scott Grisso**

10. Conclusions and Recommendations

With the schematics, pseudocode and theory of operation established the implementation phase of the project can begin. This will consist of the development of the hardware layout which contains all included schematics, the acquisition of the parts listed in the Parts Information section, and the development and testing of code from the developed pseudocode. The designs proposed in this report are subject to change as the development of the implementation progresses, but any changes should adhere to the general framework and theory of operation laid out here. Arriving at the conclusion of the implementation process should produce a functioning prototype of the proposed system that follows the proposed theory of operation and satisfies the marketing and engineering requirements set forth in this report.

11. References

[1] J. P. Friedlander, “Mid Year 2019 RIAA Music Revenues Report,” 2019. Accessed: Feb. 26, 2020. [Online]. Available: https://es.scribd.com/document/427022365/Mid-Year-2019-RIAA-Music-Revenues-Report#download&from_embed.

[2] N. Temenos, C. Basetas, and P. P. Sotiriadis, “Efficient All-Digital Frequency Synthesizer Based on Multi-Step Look-Ahead Sigma-Delta Modulation.”

[3] R. Wilson, *Make: Analog Synthesizers*. Make:Community, 2013.

[4] W. M. Hartman, *Principles of Musical Acoustics*. New York: Springer Science+Buisness Media, 2013.

[5] D. Self, *Small Signal Audio Design*. 2014.

[6] J. Vankka, *Digital Synthesizers and Trasmitters for Software Radio*, 2nd ed. Springer US, 2005.

[7] P. P. Sotiriadis and K. Galanopoulos, “Direct all-digital frequency synthesis techniques, spurs suppression, and deterministic jitter correction,” *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 59, no. 5, pp. 958–968, 2012, doi: 10.1109/TCSI.2012.2191875.

[8] D. Muhammad Taher Abuelma’atti, Dhahram, Nawal Mansour Al-Tahia, “US7952395B2,” US7952395B2, 2011.

[9] L. F. Ludwig, “US 2013 / 0296499 A1,” US 2013 / 0296499 A1, 2013.

[10] R. Mancini and R. Palmer, “Sine-Wave Oscillator,” 2001. doi: 10.1080/00207217008900249.

[11] “SAD-1024 Dual Analog Delay Line.” http://www.geofex.com/sad1024.htm (accessed

Oct. 02, 2020).

[12] Sweetwater, “Tape Delay,” 2003. https://www.sweetwater.com/insync/tape-delay/ (accessed Oct. 02, 2020).

[13] AmplifiedParts, “Spring Reverb Tanks Explained and Compared.” https://www.amplifiedparts.com/tech-articles/spring-reverb-tanks-explained-and-compared (accessed Feb. 10, 2020).

[14] D. Morrin, “Op Amp Distortion and Overdrive Topologies.” https://sites.google.com/a/davidmorrin.com/www/home/trouble/troubleeffects/distortion-overdrive/op-amp-distortion-and-overdrive.

[15] R. Elliott, “Designing with Opamps - Part 2.” https://sound-au.com/dwopa2.htm.

[16] G. Fonte, “Making Waves,” *Nuts and Volts*.

[17] D. Katz and R. Gentile, “Embedded Media Processing,” *Embed. Media Process.*, pp. 149–188, 2006, doi: 10.1016/B978-0-7506-7912-1.X5000-8.

[18] P. D. Lehrman, “What Is MIDI?,” *Midi Man.*, pp. 1–12, 2020, doi: 10.4324/9781315670836-1.

[19] “USB Types_ Ports, Speeds and Standards _ Tripp Lite.” https://www.tripplite.com/products/usb-connectivity-types-standards.

[20] MIKEGRUSIN, “Serial Peripheral Interface (SPI),” *Sparkfun*. https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi.

[21] SFUPTOWNMAKER, “I2C,” *Sparkfun*. https://learn.sparkfun.com/tutorials/i2c.

[22] M. Usach, “Controlling the AD5111/AD5113/AD5115 Using a Traditional Dial Interface,” no. AN-1150, pp. 2–3.

[23] Microchip, “PIC24FJ1024GA610 / GB610 FAMILY 16-Bit Microcontrollers with Large , Dual Partition Flash Program Memory and USB On-The-Go (OTG) PIC24FJ1024GA610 / GB610 FAMILY,” 2018.

[24] “Wi-Fi Network Controller Software Design Guide,” 2018.

[25] Microchip, “ATWINC15x0 Datasheet,” 2018.

[26] Microchip, “MCP3561/2/4 - Two/Four/Eight-Channel, 153.6 ksps, Low Noise 24-Bit Delta Sigma ADCs,” 2020.

[27] L. Connectivity, “BT860-Sx Datasheet,” pp. 1–36.

12. Appendices

http://ww1.microchip.com/downloads/en/DeviceDoc/30010074e.pdf

https://ww1.microchip.com/downloads/en/DeviceDoc/MCP3561-2-4-Two-Four-Eight-Channel-153.6-ksps-Low-Noise-24-Bit-Delta-Sigma-ADCs-DS20006181B.pdf

http://ww1.microchip.com/downloads/en/DeviceDoc/22265a.pdf

https://www.mouser.com/datasheet/2/54/PEC12R-777795.pdf

https://ww1.microchip.com/downloads/en/DeviceDoc/ATWINC15x0-MR210xB-IEEE-802.11-b-g-n-SmartConnect-IoT-Module-Data-Sheet-DS70005304C.pdf