# ReCap: Reprogrammable Modular Synthesizer Hardware Platform

WUSTL ESE 498 Electrical Engineering Capstone Project Spring 2022

Tyler Richards and David Papermaster

Advised by Ed Richter

# Abstract

Throughout the spring semester of 2022, we took a deep dive into the electronics development process as we designed and implemented a digital signal processing (DSP) system based around the RP2040 microcontroller. Taking advantage of both the guidance of our professor Dorothy Wang, our Advising Professor Ed Ricther, and our previous development experience, we were able to build a functioning programmable synthesizer module as well as example firmware implementations. This report details every step we took in accomplishing our goals as well as the challenges we encountered along the way.

# Table of Contents

# Introduction

## a. Background Information

Music synthesizers, traditionally, have been monolithic systems, essentially large black where a user gives it an input, and are given an output waveform. This is because these traditional music synthesis systems have fixed signal paths. For instances, below is an example signal path of a generic frequency generator:

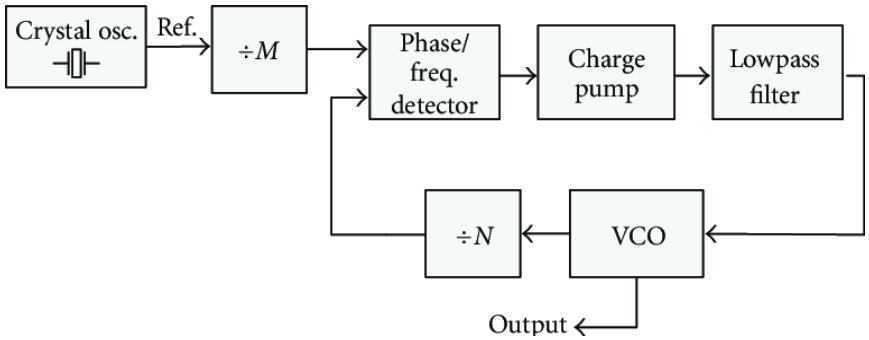



Figure 1: A block diagram of a frequency synthesizer

Modular synthesis is different from this traditional approach, as it allows users to, instead of having a single black box with a fixed signal path, break each of the modules inside that black box up and reconfigure them in any way, allowing users to alter a synthesizer's signal path by simply plugging an audio jack into a different module. One common standard for these interoperable modules is referred to as Eurorack. Electronic music production featuring modular synthesizers challenges producers to explore complex musical relationships to create unique sounds. Independence of these modules creates unlimited potential for music creation. However, a drawback of building a complex system from simple blocks with limited functionality is the necessity of owning many unique blocks (also known as modules).

## b. Problem Statement

Driven by the urge to explore, modular synth producers face the same initial obstacle without exception, absurd costs for limited utility. With some individual modules costing upwards of $2,000, a basic off the shelf system is virtually unattainable for less than $1,500. A common system building tactic used by producers today is to create barebones systems and add more modules as their needs expand. However, the current module market consists of limited functionality modules which are predesigned and non-customizable, so producers are forced to settle with a single new functionality to explore in their new system.

## c. Aims or Objectives

Our capstone project addresses this issue by creating a Eurorack hardware platform which maximizes functionality while minimizing cost. Acting as a high performance module development platform, producers and programmers alike can develop digital signal processing applications to fit their needs.

Utilizing knowledge of analog circuit design, digital signal processing, and embedded computing system design, we aim to build a production-ready eurorack module capable of facilitating user-designed signal processing applications. To enable user-focused design, we plan on developing 5 example firmware applications to accomplish some of the basic functionalities expected in a barebones system.

# Methods

## a. Hardware Design

### i. Hardware Block Diagram

Our design implements a fairly simple digital signal processing architecture. Centered around the RP2040 Microprocessor as our CPU, two analog to digital converters (ADCs) and one digital to analog converter (DAC) allow for interaction between the front panel analog components and the CPU. This relationship is shown in the hardware block diagram shown below.

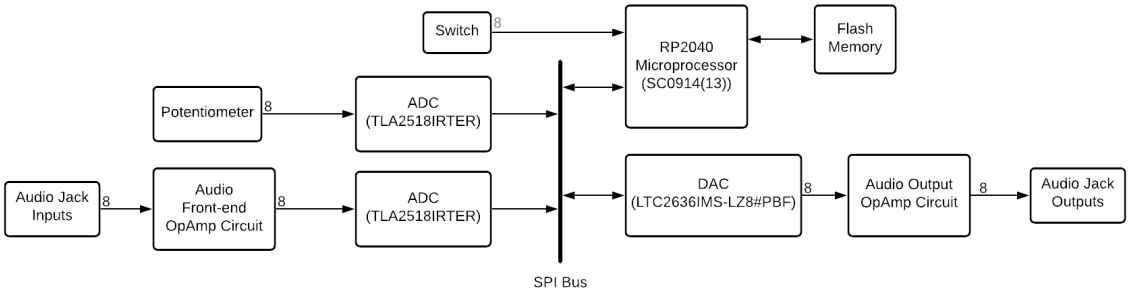



Figure 2: Hardware Block Diagram

### ii. PSPICE Analog Simulation

Designing the Audio Front-end Op Amp circuit as well as the Audio Output Op Amp Circuit, we would use operational amplifiers in different configurations to accomplish the necessary scaling and shifting of the analog signals. Input scaling and shifting circuit utilizes op-amp in the differential configuration to modify a -10V to +10V signal range to 0V to 3.3V. This scale reformats the incoming analog signal to fit the full range of the analog to digital converter placed on the output of this circuit.

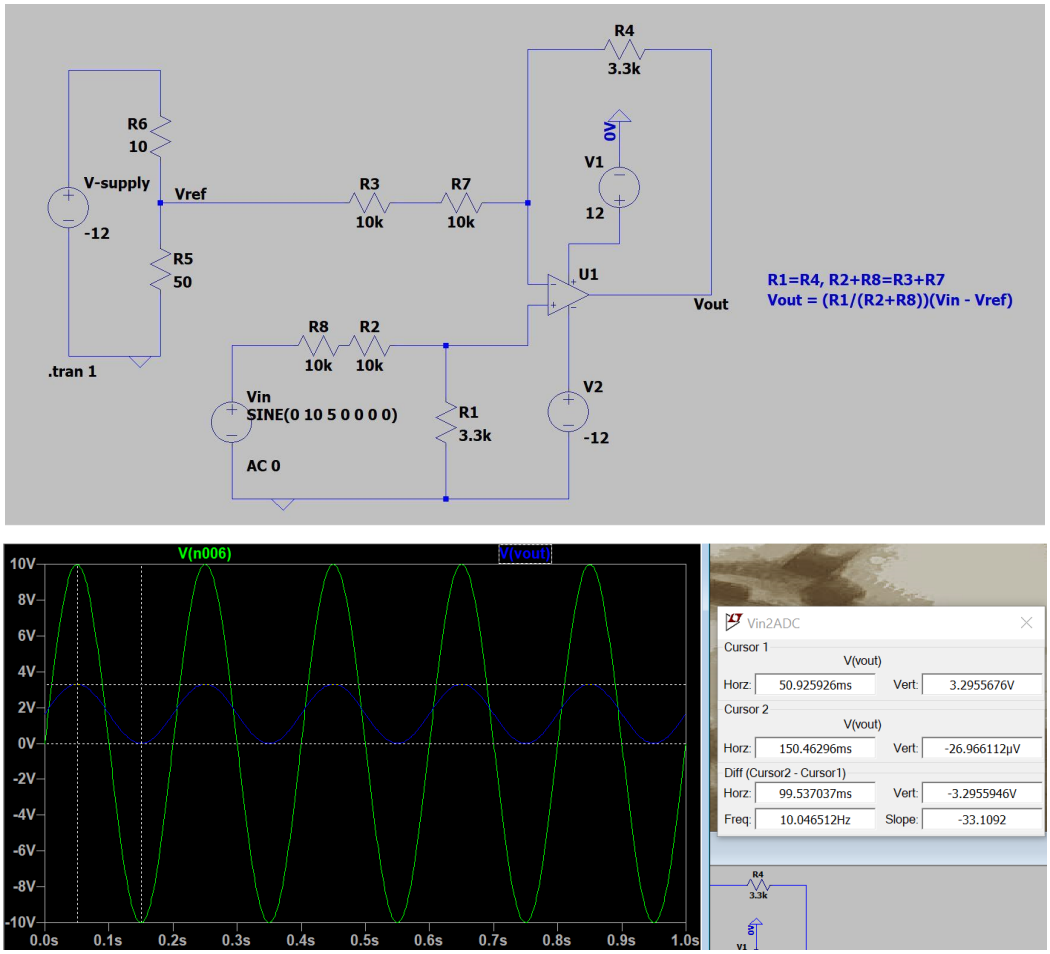



Figure 3: PSPICE Simulations confirming circuit operation

Using a combination of a non-inverting op amp with a differential configuration, the output shifting and scaling circuit reformats the 0V to 3.3V signal from the digital to analog converter to -10V to +10V.

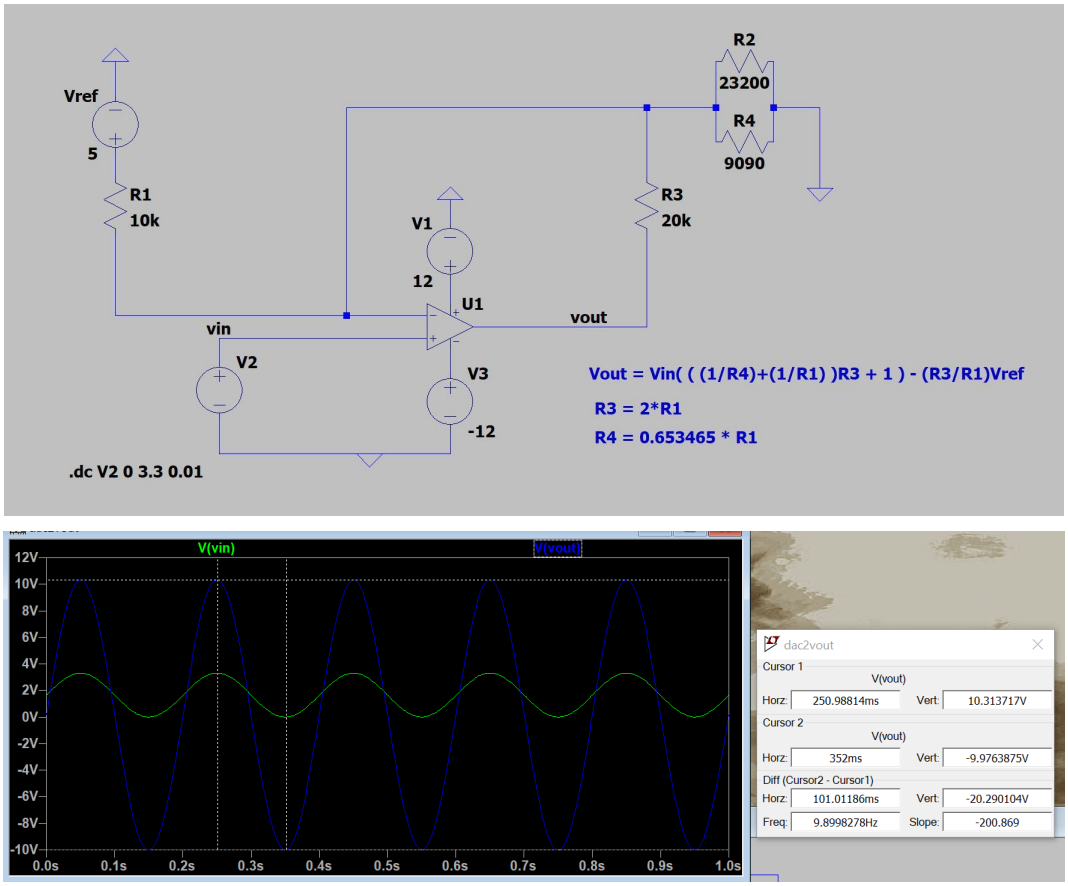



Figure 4: PSPICE Simulations confirming circuit operation

## iii. Component Selection via Digikey

To have the highest possible sample rate, we selected a DAC and ADC that could handle a SPI clock of greater than 50Mhz. Thus we selected the LTC2636CDE-LZ10#PBF for our DAC and the TLA2518IRTER for our ADC. Passives were chosen based on low cost and availability.

During the ordering process, however, our selected DAC LTC2636CDE-LZ10#PBF went out of stock. We researched pin compatible chips while taking into consideration availability, price, bit resolution, and clocking speed. We settled on upgrading our 10-bit DAC to the LTC2636CDE-LMI12#PBF, an in-stock12-bit DAC with a reasonable price.

## iv. Schematic Development

After selecting the essential components, we began researching the corresponding datasheets to develop a basic schematic: essentially setting up our components to be used together. For example, we referred to the minimal design example from the RP2040 datasheet to act as a starting point for our design.

### v. PCB Layout

Following schematic development and completing the rest of part selection, we began the PCB layout process by organizing the parts into functional groupings: op amp ICs with their resistors, decoupling capacitors, flash memory with RP2040, etc. Next, we moved the component groupings onto our defined board size and began routing traces between the parts as necessary. We created two PCBs separating the main digital signal processing components from the front panel interactive components.

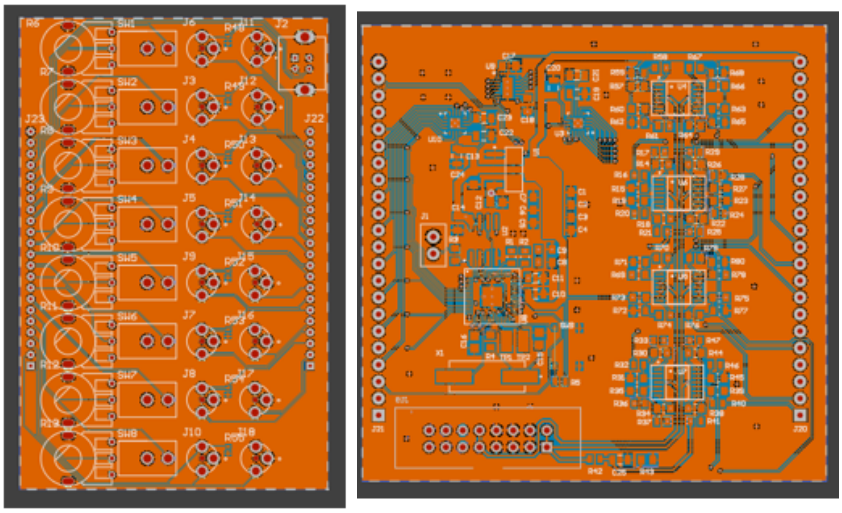


Figure 5: Input panel (Left) and Control board (Right) PCBs

## b. Assembly

### i. JLCPCB Order

Using the gerber files generated by our EDA software Altium CircuitMaker, we were able to create an order with JLBPCB. Using our provided files they are able to manufacture our designed PCBs and have it at our door within a couple of days.

### ii. Digikey Order

Following part selection for all of our components, we created a bill of materials (BOM) for our design. The bill of materials is a list of part numbers, quantities, part labels, etc. organized into a tabular format. Digikey features a BOM creator application on their main website and facilitates an easy ordering process based on the selected components.

### iii. Front Panel Soldering

The front panel circuit consisted of primarily through-hole components such as potentiometers, 3.5mm audio jacks, and switches. Therefore, assembling the front panel involved aligning and soldering 32 individual components with even more individual pins.

### iv. Main Board SMD Reflow Soldering

The main DSP board consisted of mainly surface mounted components. Soldering surface mount components to a circuit board involves first applying small dabs of solder paste onto pads along with rosin flux. Next a hot air gun (set to approximately 220C) is used to melt the solder and secure the parts to their corresponding pads. We found that soldering surface mount components can be very simple to assemble, however due to the increased complexity of the smaller components, it is very important to perform thorough conductivity testing of connection points to avoid accidental shorts between circuits. Some components featured extremely small pitches between pins, so we found it necessary to apply additional flux to certain connections and reapply hot air.

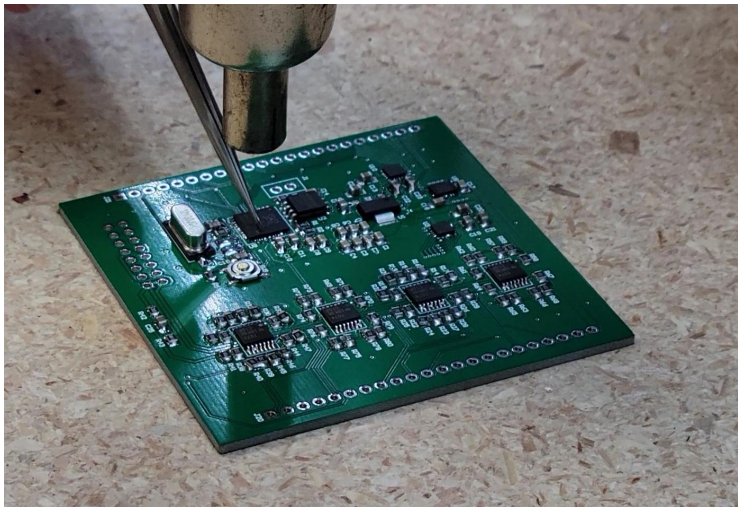


Figure 6: Reflow Soldering the Control Board

## c. Hardware Verification

### i. Microcontroller functionality verification

After assembly, we verified the functionality of the microcontroller by simply plugging it into a computer with the front panel USB-B connector. Upon plugging into the RP2040, we were happy to see the device appear as a storage drive on the computer notification, confirming the microcontroller was booting and communicating with the computer. From this we were able to verify the requirements for the microcontroller were met.

During verification, we made a modification to our board stemming from SPI bus connections on the RP2040. We found that we misplaced the traces for the SCLK and SDI lines after reviewing our design. The RP2040 datasheet confirmed our mistake and we reconnected the traces to their proper pins on the chip.

### ii. Simple DSP Functionality Verification via Oscilloscope + Function Generator

Providing a -10V to 10V, 500 Hz sine wave into the input port we are able to test the functionality of the analog to digital converter as well as the input audio op amp circuit. One of the first issues we encountered was that the analog to digital converter was not properly handling the full scale input. Upon further inspection into the ADC datasheet we realized that the reference voltage of the ADC must be set externally to 3.3V instead of its internal reference of 2.5V. We modified the traces around the ADC to set an external reference of 3.3V. In order to activate external reference mode, an additional setup SPI command was sent during device setup.

## d. Firmware Design

### i. Dual Core Functionality; SPI Handling

We explored many multiprocessing frameworks in an attempt to increase sampling rate. Utilizing multiprocessing would allow us to process SPI transactions and user defined DSP algorithms at the same time.

As proposed by Professor Richter, direct memory access (DMA) would allow our SPI devices to directly access memory, leaving the main processor free to compute data while the SPI peripheral communicated with our IC's. Again, referring to the RP2040 datasheet, we found that there exists DMA hardware internal to the RP2040 chip. During initial development, we began configuring the DMA hardware connecting the SPI bus to the DMA hardware. Due to the fact that three SPI devices are connected to this bus, we found it difficult to properly interact with the three devices. This problem could be alleviated by a circuit redesign, connecting the devices to separate SPI communication bus pins on the RP2040. We decided to move forward with using the dual core capabilities of the RP2040 rather than DMA.

Utilizing both cores of the processor proved to be a relatively simple task. In firmware, we are able to configure a single function to be run on the other core. We decided that the second core would be responsible for the SPI transactions and loading the values to and from memory. A syncing variable is used to communicate to either core that the most recent values are loaded into our buffered input array. Due to the independent nature of the cores, we found that setting and checking this sync variable was not deterministic. This is caused when the sync variable is changed between the call to the variable by a polling loop and the actual access of that variable. To ensure accesses to the variable are mutually exclusive between the two cores, we utilized a mutex (mutual exclusion) lock on the syncing variable. The SPI transaction core processes interactions as follows: put read ADC values into first buffered array, indicate update of array, relocate data into second buffer, and output data from output buffer to the DAC. We utilized the buffered input array in order to allow the DSP core to process the previous ADC samples whilst the new sample is being taken. A diagram of the FSM employed can be seen below, along with a functional block diagram:

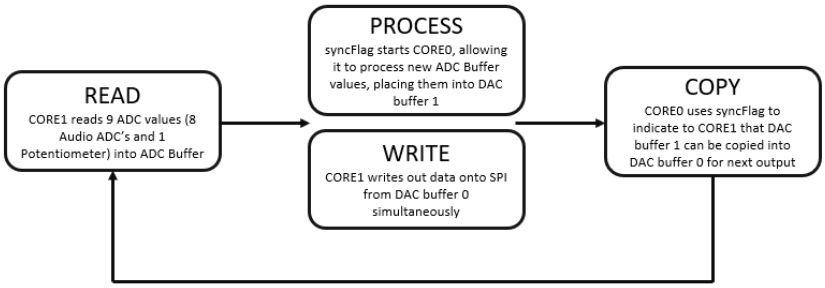



Figure 7: FSM utilized by both CPU cores

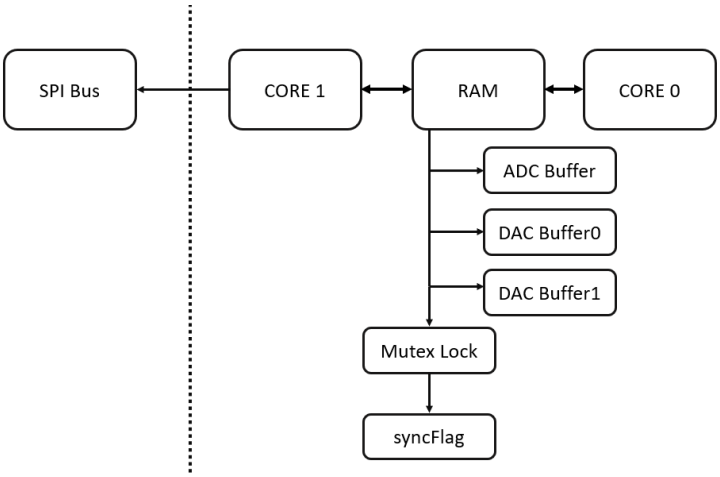



Figure 8: CPU and Peripheral Block Diagram

### ii. Flexibility in Firmware Implementation for Users

A benefit to the separation of processing between two cores is the ease of implementation of different DSP algorithms by the user. In firmware, the SPI commands are handled automatically by the provided code so the user is only responsible for implementing their DSP algorithm. The user is provided with an array of all sampled values from the ADC and an array to store the processed values in for the DAC. During each iteration of the void loop() function in arduino, they process a single sample as desired.

# Data Acquisition and Analysis

## a. Achieved Sample Rate

Using an oscilloscope, we are able to measure the single channel sampling rate for our implementation of a VCO. Using a lookup table, we are able to approximate a sine wave using 65536 samples. When each sample is processed, a corresponding command is sent to the DAC and the output voltage of the desired channel is updated. Sampling rate is the speed at which the system can output different samples therefore we are able to measure this by the time a given sample is maintained before updating. Our measurement is shown below on our oscilloscope.


Figure 9: Single Channel Sample Rate Measurement

The period of time a sample is held (BX-AX) was measured to be 31.60us, therefore the sampling rate for a single channel is 31.64kHz. Compared to the standard lower limit for high definition audio, 44.1kHz, we can qualitatively categorize our system as capable of medium quality audio. When considering the operation of the DAC across all 8 channels, as well as the ADC across all 8 channels, we can approximate the overall sampling rate to be 506kHz.

# Results

## a. Fully Assembled Module

Following assembly and hardware verification we are able to confirm the completion of our hardware. Our completed front panel and main circuit board are shown below connected by headers along the side.

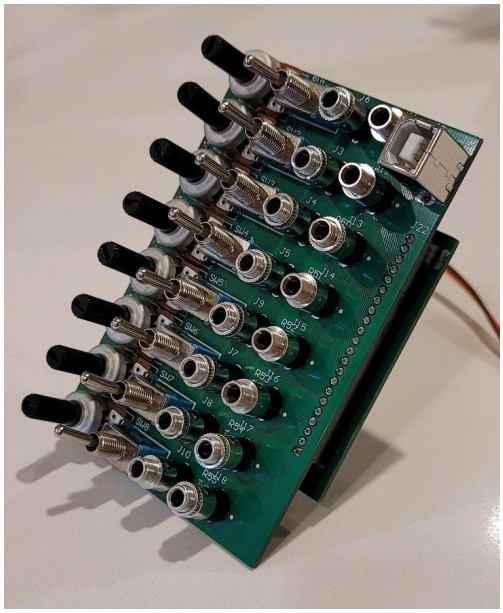


Figure 10: Fully Assembled Module

Using two rows of header pin connectors along the sides of our module both allowed for structural integrity and easier assembly. However, certain hardware debugging tasks proved to be more difficult given the physical inability to access the main components and CPU reset button.

## b. Voltage Controlled Oscillator (VCO) Firmware

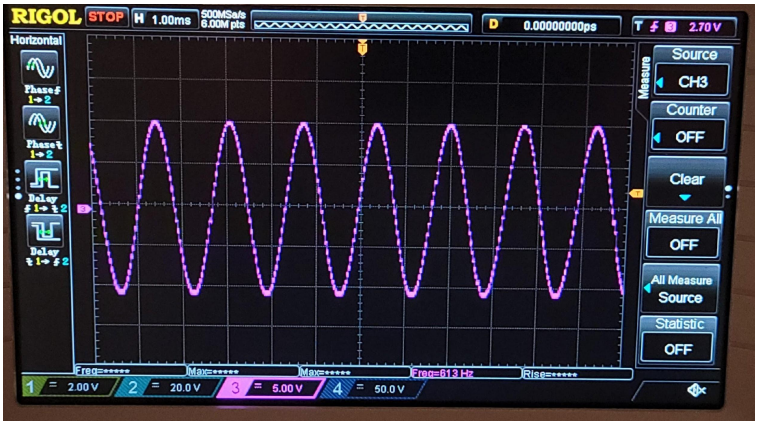
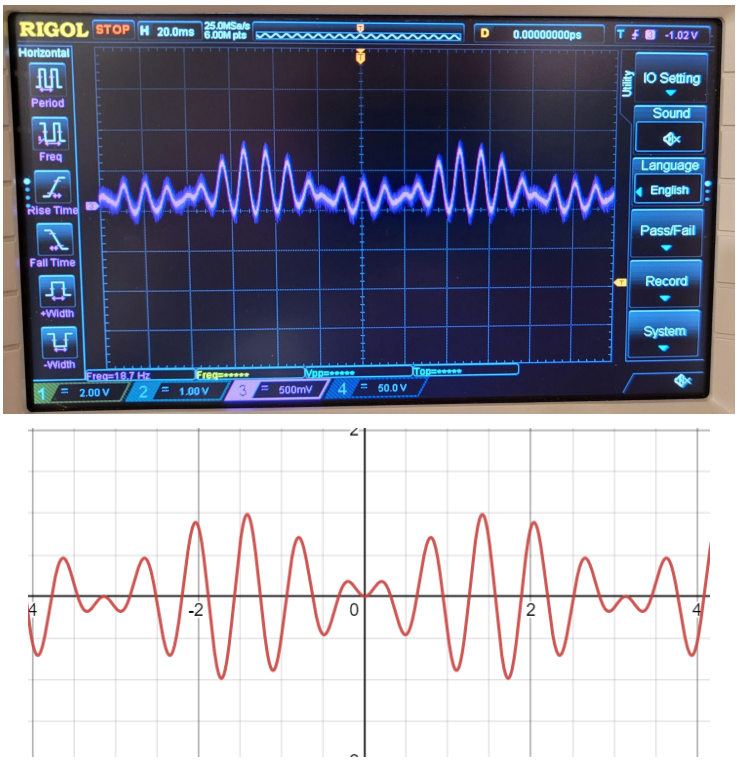



Figure 11: Sine Wave Output of VCO

We verified the functionality of the VCO by first booting the module up and probing the first channel's output. As expected, we measured a sine output, seen above, set to a specific frequency determined by channel 1's potentiometer, known as the center frequency. We then fed an input voltage into channel 1's input, and observed the output frequency of channel 1 increasing and decreasing with the voltage level present at the channel's input, confirming the correct operation of the firmware. This module is functionally equivalent to a frequency modulator.

## c. Voltage Controlled Amplifier (VCA) Firmware

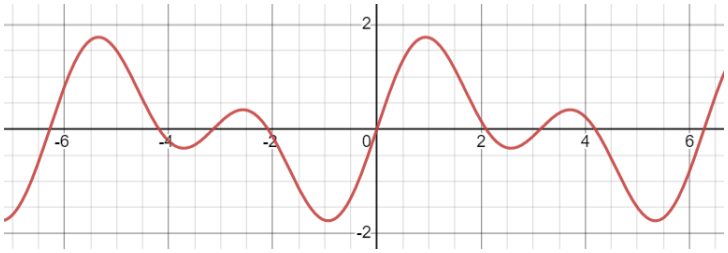



$$\sin(x)\ \sin(10x)$$

Figure 12: Sine Wave Output of VCA compared with expected output and equivalent equation

We verified the functionality of the VCA by feeding sine wave audio signals into channel 1 and channel 2's inputs, with the first channel's frequency being a decade higher than that of the second channel's. As expected, the output of channel 1 saw the first input frequency controlling the amplitude of the second input frequency. This module is functionally equivalent to an amplitude modulator.

## d. Mixer Firmware


$$\sin(x)+\sin(2x)$$

Figure 13: Mixed Sine Wave Output of Mixer

We verified the functionality of the Mixer by feeding sine wave audio signals into channel 1 and channel 2's inputs, with the first channel's frequency being double that of the second channel's. As expected, the output of channel 1 saw the two input frequencies superimposed on one another. Utilizing our oscilloscope's FFT functionality, we were able to observe both frequency peaks present in our output waveform.

## e. Delay Line Firmware

We verified the functionality of the Delay line by feeding a pulse signal into the input of the module, and measuring the time it took to see the pulse on the output. Each channel was given a buffer of 30,000 samples, meaning we expected to measure the pulse approximately 1 second after sending it. This operation was confirmed by listening to the input and output waveforms and measuring the time difference via a stopwatch, which confirmed the 1 second delay time.

## f. IIR Firmware

Unfortunately, due to a COVID illness late in the semester, we did not have enough time to complete the IIR firmware. We were unable to enter the EE laboratory we typically did testing in due to isolation guidelines, and could not test any of our prototype firmwares.

# Discussion

## a. Usability Assessment

The arrangement of components on the front panel is very intuitive in terms of each of their capabilities. Arranged in rows, the four components knob, switch, input, output are referred to as the same channel. This translates very well over to firmware, where accessing the values of these components based on row number corresponds to array indices.

While multicore processing increases overall complexity of the system, by pushing the multicore code to be hidden from users in a library greatly simplifies the development process.

## b. Areas of Improvement

After completing the prototype and firmwares, we have identified 4 areas we could improve and/or expand upon:

- ➢ Fixing PCB layout mistakes
  - During our time troubleshooting our hardware, we identified a handful of routing errors on our physical board. Even though we were able to use bodge wires and extra components to rectify these errors, any further re-designs of the board should be updated to reflect these changes.
- ➢ Reducing ADC signal noise
  - One unexpected obstacle during our firmware development was the large amount of noise present at the input of the ADC's. Further revisions of our board should probably include low pass filters (with a cutoff of around 30kHz) to remove high frequency noise that may affect measurements.
- ➢ Improve sample rates
  - While 31.25kHz per channel is a decent sample rate, most audio circuits are expected to have a bandwidth at a minimum of 44.1kHz. Further improvements could come from overclocking the RP2040 or utilizing a faster SPI algorithm.
- ➢ Utilize Higher Quality ADC's
  - While 12 bit audio is of decent quality, most audio applications expect 16 bit audio. In future board revisions, a 16 bit DAC and ADC will be used.

# Conclusions

Overall, we believe this project was extremely successful. We were able to design, manufacture, and assemble a two-part prototype PCB that utilized both analog and digital hardware. On top of that, we were able to program robust firmwares capable of demonstrating the capabilities of our PCB. 8 Inputs and 8 outputs, all operating at a 31.25kHz sample rate, is, in our eyes, a wonderful starting point to further develop and refine this product. We are confident that the initial firmwares we have released demonstrate the viability and ease of use of our platform over competing products. Utilizing this design as a base, we hope to further develop the concept in the future, with the intent on selling the finalized module as a product.

# Deliverables

a. PCB manufacturing files and schematics (completed)
   i. All manufacturing files can be accessed via Altium365 here.
b. Assembled and functional board (Completed)
c. Firmware files (Mostly completed)
   i. All firmware files can be accessed via GitHub here.
d. Demonstration of firmwares (Completed)

e. Stretch Goals

### i. Arduino Library

We were able to create a working Arduino library which automates the multi core aspect of the firmware. To the user/firmware developer, they are able to access and redefine any of the “private” variables and functions used during the processing of SPI commands. However, most use can be limited to simply accessing the buffered arrays used for sampling data. Having access to the arduino library significantly simplifies the process of firmware development.

### ii. Lab or Module for EE department

Due to time limitations, we have not made significant progress exploring a lab or module for the EE department. We believe the act of designing and assembling a simple circuit board would be an extremely valuable experience for any profession-bound electrical engineer. Whether it is implemented with professionally built circuit boards or more traditional assembly techniques like perfboard, the ability to prototype more complicated circuits outside of a breadboard is very valuable and rewarding. Soldering tasks would also provide a good resume builder (especially utilizing SMD components).

# Schedule/Timeline Adherence

***Proposed Timeline***

| **Task / Date** | 20-Feb | 27-Feb | 6-Mar | 13-Mar | 27-Mar | 3-Apr | 10-Apr | 17-Apr | 24-Apr | 29-Apr | 8-May | 11-May |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BOM Costs | | | | | | | | | | | | |
| Initial Design | | | | | | | | | | | | |
| Received Boards | | | | | | | | | | | | |
| Assemble Boards | | | | | | | | | | | | |
| Validate hardware functionality | | | | | | | | | | | | |
| Firmware development | | | | | | | | | | | | |
| Final Presentation | | | | | | | | | | | | |
| Final Report | | | | | | | | | | | | |
| Webpage | | | | | | | | | | | | |

***Actual Timeline***

| **Task / Date** | 20-Feb | 27-Feb | 6-Mar | 13-Mar | 27-Mar | 3-Apr | 10-Apr | 17-Apr | 24-Apr | 29-Apr | 8-May | 11-May |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BOM Costs | | | | | | | | | | | | |
| Initial Design | | | | | | | | | | | | |
| Received Boards | | | | | | | | | | | | |
| Assemble Boards | | | | | | | | | | | | |
| Validate hardware functionality | | | | | | | | | | | | |
| Firmware development | | | | | | | | | | | | |
| Final Presentation | | | | | | | | | | | | |
| Final Report | | | | | | | | | | | | |
| Webpage | | | | | | | | | | | | |

# References

Sameni, Pedram & Siu, Chris & Mirabbasi, Shahriar & Djahanshahi, Hormoz & Hamour, Marwa & Iniewski, K. & Chana, Jatinder. (2006). Modeling and Characterization of VCOs with MOS Varactors for RF Transceivers. EURASIP Journal on Wireless Communications and Networking. 2006. 32-32. 10.1155/WCN/2006/93712.

Raspberry Pi Foundation. (n.d.). *Colophon - Raspberry Pi datasheets*. Retrieved February 6, 2022, from https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf

Texas Instruments. (n.d.). Retrieved February 6, 2022, from https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Ftla2518

Analog Devices. (n.d.). *LTC2636 (rev. D) - analog devices*. Retrieved February 7, 2022, from https://www.analog.com/media/en/technical-documentation/data-sheets/ltc2636.pdf