

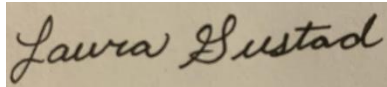
The Harmonic Triad - Vocal Harmonizer

Laura Gustad, Nate Hunter, and Noah Mills

December 10, 2020

Capstone Design ECE 4440 / ECE 4991

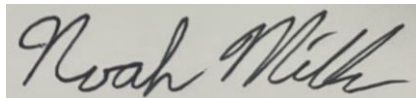
Signatures



Laura Gustad



Nate Hunter



Noah Mills

Statement of work:

Laura Gustad

My main contributions to the project were the circuit design, board layout, and PCB testing. I designed the circuit schematic in Multisim, which included calculations for resistor and capacitor values, creating footprints, and researching the Codec communication configurations. From the schematic design, I handled the board layout in Ultiboard and organized the signoff checks before submitting the Gerber files to be printed. Noah and I worked closely together on PCB construction, testing individual subsystems, and debugging to resolve issues with the board. In designing the second iteration of the PCB, I corrected the issues we encountered with the first PCB design, including spacing and correcting the biasing in the filters.

My secondary responsibilities for the project included both software testing and smaller non-technical roles. As I was the most familiar with the Codec datasheet, I designed the SPI commands necessary to configure the Codec, both for smaller tests of SPI function and our full system. For the SPI and I2S communication testing, I manned the VirtualBench to obtain screenshots and confirm frequencies. Among my non-technical roles, I was responsible for ordering the microphone and audio chords/converters. More specific to board construction, I was also the contact person and transportation to and from WWW Electronics both times they soldered the Codec to our PCB.

Nate Hunter

I created the top-level design for the harmonizer, illustrated in a block diagram. I configured my laptop and the myRIO to run LabVIEW. I proceeded to design the DSP algorithm, first on a conceptual level, then in LabVIEW code. The initial algorithm code accepted voice and keyboard WAV files as inputs, producing a harmonized output. I then determined the codec communication requirements for SPI and I2S and developed the LabVIEW FPGA code interface between the myRIO real-time processor and the codec. During this phase I also led the testing of the SPI and I2S communication. Finally, I synthesized the DSP algorithm and the FPGA FIFO code into a program for the real-time processor and deployed it on the myRIO. I led the testing of the final program. In secondary roles, I also helped out with some hardware testing and selecting SPI commands.

Noah Mills

My main contributions to this project were performed with the physical circuit board. I did all of the soldering/stuffing of our PCBs, as well as lead testing before and after adding components and subsystems to the boards. I worked closely alongside Laura during the construction of the PCBs, since she did the designing of them with Multisim and Ultiboard. In addition to soldering components onto the boards, I created the documents required to have select components stuffed by WWW Electronics. For the test plan, I used YEd to design a test plan from scratch that would allow us to individually test each subsystem before and after connecting to our whole system, to ensure that each component worked on its own. By doing so, we were able to reduce the need to search for errors in our completed project.

My secondary roles for this project included less technical jobs, such as setting up orders from Digi-Key by creating bills of materials (BOM), arranging meetings/reserving lab time, and documenting our progress with screenshots and pictures. In addition, I was responsible for designing the 3D printed container that would eventually hold our board on the keyboard. To do this, I downloaded Autodesk AutoCAD 2021 and designed the holder to be printed by the MECH-E Department.

Table of Contents

Signatures	1
Statement of work:	1
Table of Contents	4
Table of Figures	5
Abstract	6
Background	7
Constraints	8
Design Constraints	8
Economic and Cost Constraints	9
External Standards	9
Tools Employed	10
Ethical, Social, and Economic Concerns	11
Environmental Impact	11
Sustainability	11
Health and Safety	11
Manufacturability	11
Ethical Issues	12
Intellectual Property Issues	12
Detailed Technical Description of Project	13
Schematic	13
PCB Layout	17
Software Design	18
Mechanical Chassis	27
Problems and Design Modifications	29
Project Timeline	32
Test Plan	33
Final Results	36
Costs	37
Future Work	38
References	40
Appendix	43
Appendix A: PCB Bill of Materials (BOM)	43
Appendix B: Codec Block Diagram and Timing Requirements	43
Appendix C: Software Data Communication Testing	45
Appendix D: Full System Testing	46
Appendix E: Vowel Comparisons	48
Appendix F: Budget	51

Table of Figures

Figure 1: Block Diagram for Vocal Harmonizer	13
Figure 2: Top Level Schematic for PCB.....	14
Figure 3: VMID from myRIO 3.3 V Supply	14
Figure 4: Input Signals Subcircuit	15
Figure 5: Anti-Alias Filter Subcircuit	16
Figure 6: Codec Communication, myRIO DIO, and Crystal.....	16
Figure 7: Ultiboard Layout for PCB (GND Plane Hidden)	18
Figure 8: Populated PCB	18
Figure 9: File structure of Harmonizer.lvproj.....	19
Figure 10: Main.vi front panel in action	19
Figure 11: Main.vi block diagram	20
Figure 12: Mic 2 Harmonics.vi block diagram left: fundamental computation	21
Figure 13: Mic 2 Harmonics.vi block diagram right: harmonic extraction	22
Figure 14: Keys 2 Notes.vi block diagram left: amplitude computation.....	22
Figure 15: Keys 2 Notes.vi block diagram right: note determination.....	23
Figure 16: Synthesis.vi block diagram.....	24
Figure 17: Main.vi block diagram, SPI.....	25
Figure 18: SPI message interpretations.....	25
Figure 19: FPGA 2.vi block diagram, SPI.....	26
Figure 20: FPGA 2.vi block diagram, I2S left channel	26
Figure 21: FPGA 2.vi block diagram, I2S right channel	27
Figure 22: Chassis design in AutoCAD.....	28
Figure 23: Final Printed Chassis	28
Figure 24: First Iteration PCB with Deadbugging.....	29
Figure 25: Phase Clipping Between Two Audio Sections.....	31
Figure 26: Original Gantt Chart.....	32
Figure 27: Final Gantt Chart	33
Figure 28: Input Signal Test Plan	34
Figure 29: Codec Connection Test Plan	34
Figure 30: Full System Test - C3 + C4 on Keyboard, “Oo” Vowel	36
Figure 31: Initial Proposal Deliverables	37
Figure 32: Table of Costs.....	38
Figure 33: Bill of Materials (BOM) for PCB Components	43
Figure 34: SPI Timing Requirements for Codec	43
Figure 35: I2S Timing Requirements for Codec.....	43
Figure 36: Codec Block Diagram	44
Figure 37: SPI Send Codec Configuration VirtualBench Test	45
Figure 38: I2S Read 44.1 kHz DOUT on myRIO VirtualBench Test	45
Figure 39: myRIO Read In for 44.1 kHz DOUT	45
Figure 40: LabView WaveForm and FFT for Keyboard C Chord Test	46
Figure 41: LabView WaveForm and FFT for No Voice Input Test	46
Figure 42: LabView WaveForm and FFT for Soft Ah Test	47

Figure 43: LabView WaveForm and FFT for Loud Ah Test.....	47
Figure 44: Mic to I2S, “Ah”	48
Figure 45: Mic to I2S, “Ee”	48
Figure 46: Mic to I2S, “Eh”	48
Figure 47: Mic to I2S, “Oh”	48
Figure 48: Mic to I2S, “Oo”	49
Figure 49: Mic to Chord Output, “Ah”	49
Figure 50: Mic to Chord Output, “Ee”	49
Figure 51: Mic to Chord Output, “Eh”	50
Figure 52: Mic to Chord Output, “Oh”	50
Figure 53: Mic to Chord Output, “Oo”	50
Figure 54: Total Project Budget Breakdown	51

Abstract

Our project is a real-time vocal harmonizer. The idea is that someone could sing or play an instrument into a microphone while simultaneously pressing down a chord on a keyboard, producing the output of their voice reharmonized to the chord that is played. The signal processing required to execute this transformation can be accomplished with a field-programmable gate array (for the digital side) and a printed circuit board with an amplifier and a codec (for the analog side). This project will enable musicians to simulate their own choir, thus unlocking new realms of creative expression. Such a capability could be particularly valuable during the solitary times of a pandemic.

Background

As three students with musical backgrounds, this project is both useful to us for personal musical projects and also is recognized as helpful for both new and experienced musicians. This project is especially useful during times of quarantine, in which musical groups are unable to meet together in person. It both simulates the effects of being with other musicians, and encourages musicians to practice on their own.

There have been a number of similar projects and products to ours that have been done in the past, both in academic settings and in the commercial market. At the University of Virginia (UVA), one former capstone project was to identify the pitch of a microphone input and to harmonize it in the output [1]. Students at the University of Michigan developed a device to harmonize microphone input according to various preset scales as well as to keyboard MIDI input [2]. A device known as a vocoder can be used to harmonize vocal input according to keyboard input, typically to produce a synthetic or electronic sound. Vocoderes such as the Roland JD-Xi are commonly used and are available on the market [3]. The musician Jacob Collier uses an advanced vocal harmonizer that was developed at MIT to harmonize his voice with his keyboard [4].

There are a number of elements that our project has combined in such a way that none of the aforementioned projects have done. First, the notes of the output chord are exclusively defined by a keyboard, and not by presets or by the voice itself. Second, our device produces an output that mirrors the natural timbre of the voice, instead of producing a synthetic timbre. Third, we have created a product that, with modifications, could be widely available and reasonably affordable on the market. The combination of these three factors define our project uniquely, and make for a device that musicians of all levels could enjoy in a way that no other device exactly captures.

This project greatly relied on knowledge from previous courses that our group members have taken. The programming of the MyRIO was completed in LabView, which Laura has become familiar with through Independent Research. The construction of the electronic system, schematic design and PCB layout design, drew on all three ECE Fundamentals classes (ECE 2630, ECE 2660, ECE 3750), which all three of us have taken. In addition, all three of us took Intro to Embedded Systems (ECE 3430), which gave us the background knowledge to interface between the codec and the MyRIO system using serial peripheral interface (SPI) protocol.

Constraints

Design Constraints

Parts Availability

As we borrowed a MyRIO and keyboard from the university, the availability of parts constraints can be narrowed down to the Codec, microphone, and 3D printing materials. Also, the piano amplifier used for the signal output was borrowed from Nate Hunter. The Stereo Audio Codec from Texas Instruments is stocked in the thousands by Digikey [5]. The microphone did not have to be a specific brand and we could not find a microphone with a detailed datasheet; therefore, we chose a dynamic microphone from Arctic Violet that was in stock [6]. 3D printing was available at multiple locations in the Engineering School as well as in the Fab Lab at the Architecture School, so the time slots and access were more of a concern than material availability. We chose to have the PCB holder 3D printed by the mechanical engineering department, since they had a quick turnaround time and were able to easily deliver our completed product to the ECE department [7].

Manufacturing Limitations

Professor Powell ordered our board as a larger sheet with multiple boards from other teams to save on shipping costs, therefore the board designs needed to match with respect to certain layering and design requirements. The origin of the board outline was kept at the bottom left and the number of copper layers was kept to two. Of the 2-layer board requirements imposed by the PCB manufacturer, Advanced Circuits, the following were relevant to the PCB design for this project [8]:

- Maximum board size: 30 sq. in.
- Minimum 0.005 in. line/space
- Minimum 0.010 in. hole size
- Maximum 50 drilled holes per sq. in.

Software Availability

The softwares used in this project were Matlab [9] and National Instruments' LabView [10], Multisim [11], and Ultiboard [12]. The University of Virginia provides student licenses for Matlab 2019, NI LabView 19.0, NI Multisim 14.1, and NI Ultiboard 14.1.

CPU Limitations

The software options were limited by the myRIO compatibility requirements. The processor on the myRIO can be programmed in C or C++, but the FPGA can only be customized using National Instruments' LabView [10]. The firmware on the myRIO was difficult to update without direct help from National Instruments, but the second myRIO we adopted already had firmware for LabView 2019, which was the version we were all able to download to our laptops.

Economic and Cost Constraints

The vocal harmonizer fit comfortably within the cost constraints of the capstone budget of \$500. We borrowed a Yamaha keyboard and MyRIO from the Electrical and Computer Engineering departments, which considerably reduced the larger cost variables. The circuit components were not a major cost variable as some of the components needed for the PCB were included in the kits from the Fundamentals classes, but the components that needed to be ordered were each in the range of \$0.10-\$5.05. The cost of 3D printing the casing for the myRIO and PCB was \$15 per cubic inch of material by the Mechanical Engineering Department. The only purchased items were the microphone, codec, circuit components, and the PCB.

If this device were to go into production, the economic constraints would be considerably higher. The potential customers would be universities, musical education programs, and individuals interested in music, so the cost would need to be kept reasonable. As this device would be a keyboard attachment, the cost of the keyboard would not need to be included in the price. A new myRIO typically costs between \$500-\$1000, but developing a more specified FPGA board with just the necessary hardware for this device would reduce the cost. More professional-grade encasing than 3D printing would add to the initial development cost either with out-sourcing or purchasing plastic molds. After development, the device cost could reasonably be estimated between \$25-\$50.

External Standards

The following external regulatory standards are relevant to this project.

1. *IPC Standards for PCB Design* - Characteristics of PCB designs are defined in the IPC standards. IPC-A-600 defines the acceptable levels for plating, hole size, material, solder mask quality, internal and external characteristics, etc. for printed circuit boards [13]. IPC 2221 sets the specifications and electrical testing protocol for the materials used in rigid printed circuit boards, including nickel, polymer, copper, dielectrics, etc. [14]. In order to meet these requirements, we used the DRC and netlist check alongside the connectivity check. We also submitted the Gerber files to the FreeDFM tool from Advanced Circuits to make sure our board had no showstoppers [15].
2. *FCC Standards* - Part 15 of the FCC standards regulate devices operating over 9 kHz, requires all digital and peripheral devices to accept interference from licensed sources, and defines the acceptable frequency operation bands [16]. The codec would be defined as a peripheral device under the FCC §15.3 [17]. The myRIO that was used within the vocal harmonizer complies with Part 15 of the FCC restrictions on Class B devices that act as unintentional radiators [18]. Should we have any issues with interference related to harmful radio communications, we will use the guidance in the “FCC Interference Statement” of the NI myRIO manual [18].
3. *SPI (Serial peripheral Interface) Communication Protocol* - SPI is a synchronous communication protocol used between master and peripheral devices with modes defined by clock polarity and phase [19]. For this project, SPI was used to communicate in the

Control Interface between the myRIO and the codec with CPOL = 0, CPHA = 0, and an approximately 1 MHz clock.

4. *I2S (Inter-IC Sound) Communication Protocol* - I2S is a synchronous protocol for communication between digital audio devices [20]. In this project, I2S was used to read and write digital audio between the myRIO and the codec in the Digital Audio Interface. The clock input provided by the myRIO was approximately 1.66 MHz.

Tools Employed

Different design, simulation, software, and testing tools were used to complete each section of this project. The tools used in each subsection of the project are described in detail below.

Math Analysis

In designing the schematic, the resistor and capacitor values were checked using Matlab [9]. Laura already had experience programming in Matlab and a bank of typical resistor and capacitor values saved from Fun I and Fun II projects.

Hardware Design

For the schematic and PCB design, Laura used National Instruments' Multisim [11] and Ultiboard [12]. These tools were necessary to simulate subcircuits, design the schematic, and develop the PCB layout. Laura, the primary on Multisim schematic design and Ultiboard layout, had experience with Multisim and Ultiboard from Fun I, II, and III.

Software and FPGA Design

For the FPGA and processor design, National Instruments' graphical programming software LabView 2019 [10] was used. LabView was required to develop the serial peripheral interface (SPI) and signal calculations on the myRIO processor as well as the I2S (Inter-IC Sound) on the myRIO FPGA. Within LabView, the more specified toolkits needed for this project were: Advanced Signal Processing Toolkit [21], LabView FPGA [22], and myRIO Toolkit [23]. Nate, the primary on algorithm and software development, learned LabView from scratch for this project.

3D Printing Design

For the 3D printed holder design, Noah used Autodesk AutoCAD [24]. With AutoCAD, the holder was designed from scratch based on measurements made of the piano slot (to insert the holder) and the myRIO. After designing the holder, the files were exported to .stl and .igs formats, and they were sent to the Mechanical Engineering Department [7] to be printed.

Testing

For testing voltages and digital inputs/outputs, the National Instruments' VirtualBench [25] and accompanying software were used. These tools were necessary for testing the voltage

supplies and subcircuits before and after sections of the PCB were stuffed. During software development, the Digital Input/Output tool was also necessary to confirm communication timing on the myRIO before testing on the PCB. All three students were involved in testing the system, and each had individual experience with the VirtualBench hardware and software from the ECE Fundamentals I, II, and III classes.

Ethical, Social, and Economic Concerns

Environmental Impact

The environmental impact of this device is heavily dependent on the responsibility of the consumer if and when they dispose of it. E-Waste can be safely recycled at designated locations, but the PCB and myRIO components could release toxic chemicals into the ground if placed in a landfill. The myRIO should only be disposed of according to the Directive 2012/19/EU for waste electrical and electronic equipment to prevent leaching toxic chemicals into the ground or atmosphere [18].

Sustainability

The myRIO has a warranty of 1 year after the shipment date, but the electronics within should last much longer than that. The keyboard, speaker, and microphone should all have at least 5+ year lifetimes, and parts that are not easily repaired could be replaced easily. If this device entered the market, the warranty for repairs on the custom board should be around a year as well.

Health and Safety

The myRIO has radio exposure risk if not operated according to the FCC limits, antennas kept more than 20 cm from a person's body [18]. Unauthorized product changes or modifications could also lead to bodily harm or injury. Signals of recklessly excessive amplitude-at or above 85 dB- input into a speaker could cause ear damage [26].

Due to the small parts involved in the system, this project could be hazardous if used by/around small children, due to the potential of swallowing/choking on the small pieces. Additionally, the device is not waterproof, so caution should be used if using the device around water or any other liquid (such as drinks). The chassis does not enclose all electrical components and wiring, but 3.3 V is not enough to overcome the resistance in skin, which prevents a risk of electrical shock [27].

Manufacturability

The only manufactured parts of the system are the PCB and the casing, all other components and cords will be purchased off-the-shelf. There is no specified power source requirement outside of the standard wall supply. The PCB and casing would be easy to reproduce with access to the design files. As the myRIO has more capabilities than are needed for this system and are not often stocked in high numbers, a custom FPGA would need to be designed if this project were to go into market production. With an FPGA (field programmable gate array)

design and designated code file to configure the FPGA, the LabView code would most likely need to be translated to VHDL or Verilog, but a custom FPGA would be easily programmed for manufacturing once the translated code file was written.

For manufacturing the PCB, the major skills required would be soldering and reading a PCB layout for component directionality. Component directionality is only applicable for the operational amplifiers and the stereo audio codec. The board requires both surface mount and through-hole components, but the board could be redesigned to use surface mount components to allow for machine soldering.

Ethical Issues

The ethical issues surrounding a real-time vocal harmonizer relate to the possession of voice recordings without permission. We are not planning to record the voice inputs to the system, and permission and human testing forms are not required as we are our own test subjects. However, if the myRIO system were to be compromised, it is possible that a malicious entity could modify the code to record a user's voice without their consent. To avoid this possibility, the myRIO should be kept in a secure location, and it should not be connected to the internet, since that is not a necessary feature to use the vocal harmonizer anyway.

Intellectual Property Issues

[28] is a patent for a software harmonizer that automatically harmonizes pitch inputs into chords. It does not use user input to select notes the way our project does, but automatically selects tones to be played in harmony with the input. The Harmonizer “harmonizes a melody in accordance with the rules using an iterative technique of chord selection, permutation and submission to the rules until solutions are found”. In addition, this patent is for an exclusively software design.

[29] is a patent for a system that listens to a pre-recorded song, analyzes to find the key, then is run during a live vocal performance (such as karaoke). During this live run, the system takes in a “melody note” from the performer, and generates a harmony that fits both the song and the melody note. Once again, this system is different from our project due to the fact that it is an automatic harmonizer, and only relies on the user voice input and previously recorded notes to generate harmony.

Finally, [30] is a patent for a harmony generator based on a melody note and an accompaniment note. This is a software system that would take in MIDI files or live audio to analyze and output the harmonies digitally. This system is able to process quickly in real-time to be used for an audio performance. This, like the previous two patents, generates its own harmony and does not require all harmony to be generated by the user. Therefore, our Vocal Harmonizer should be patentable, since it appears to be unique in our method of harmony generation.

Detailed Technical Description of Project

The purpose of our project was to develop a vocal harmonizer, which would create a chord from microphone and keyboard input signals. Once activated, the microphone takes input from voice or a musical instrument of the user's choice. The keyboard was then played at any note or chord chosen by the user. The piano amplifier output the user's original vocal or instrumental input, modulated to match the pitch or pitches selected on the keyboard.

According to the requirements of the capstone project, our project involves a microcontroller, the myRIO [18], and a custom PCB. The overall block diagram for our system is shown in Figure 1.

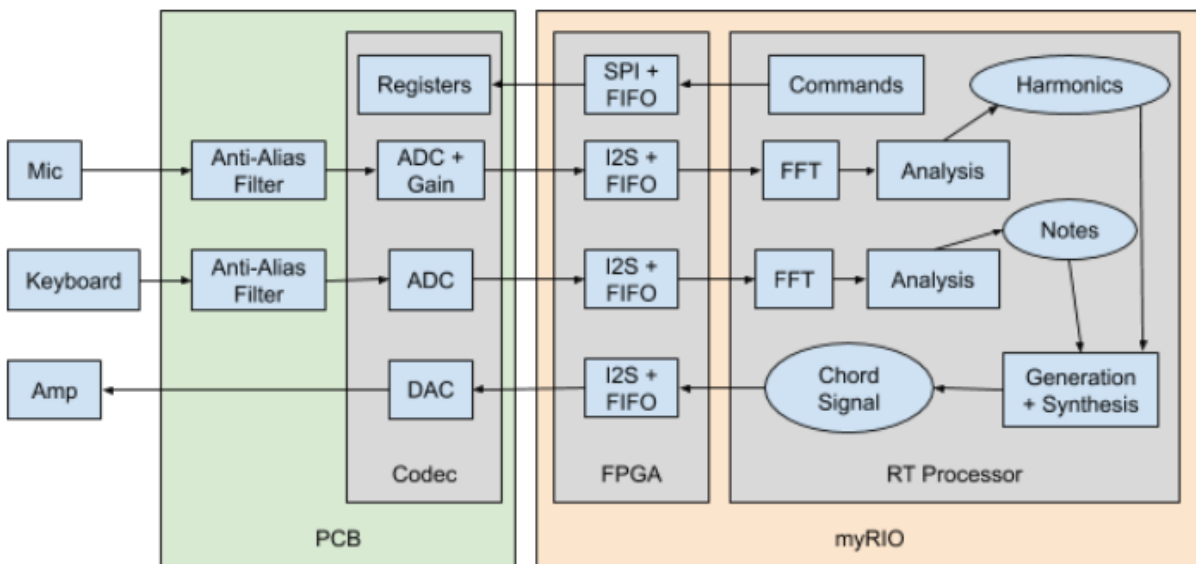


Figure 1: Block Diagram for Vocal Harmonizer

Schematic

The schematic for the PCB was developed using NI Multisim. The schematic design began with understanding the components needed to fulfill the block diagram and signal path necessary for the vocal harmonizer: real-time (RT) processor, FPGA, ADC/DAC function, filtering, and audio jacks. The myRIO [18] was chosen for the FPGA, processor, numerous digital input/output pins (DIO), and SPI capabilities. The TLV320AIC23BIPWR Stereo Audio Codec [31] was chosen for its 16-bit ADC/DAC, voltage supply requirements, left/right channel and mic inputs, and simple footprint. Finally, the 35RASMT2BHNTRX 3.5 mm audio jack [32] was chosen because Laura had prior experience wiring it from the class project in Fun II.

The top level design for this project is shown below in Figure 2. Smaller operating sections of the schematic are described in more detail in the following sections. A full bill of materials (BOM) for the components used for the PCB can be found in Appendix A.

Audio Jacks, Biasing, and Anti-Aliasing Filters

The Input Signals subcircuit from the top level schematic is shown in Figure 4. The audio jacks for the microphone and keyboard were wired such that the tip signal is biased to VMID before being input to the anti-alias filter. Test pins were included for both the raw signals and the biased, filtered signals to allow for easier debugging of the keyboard and microphone inputs.

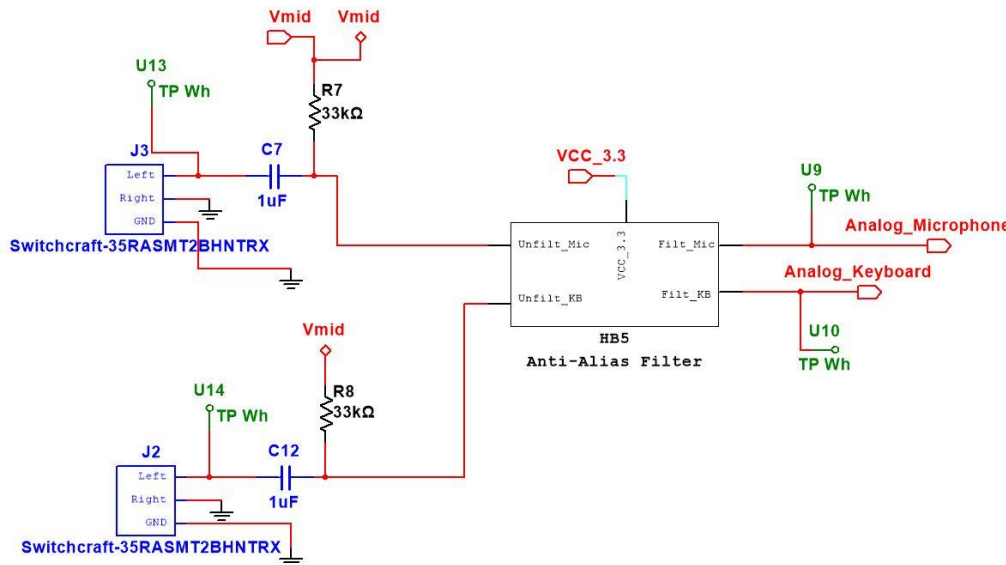


Figure 4: Input Signals Subcircuit

The Anti-Alias Filter subcircuit is shown in Figure 5, depicting the Sallen Key architecture filters for the keyboard and microphone. According to the Nyquist sampling theorem, the anti-aliasing filters were designed to have a 22 kHz cutoff frequency as the sampling frequency was 44.1 kHz. The gain was selected to be 1 for both filters and the quality factor was designed to be as close to 0.707 as possible.

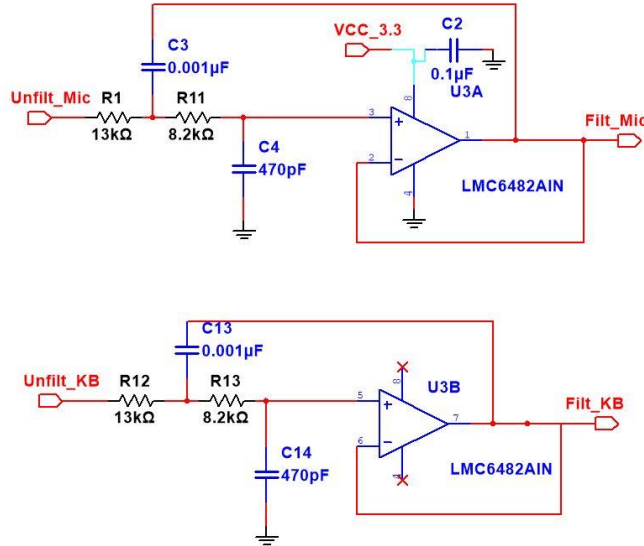


Figure 5: Anti-Alias Filter Subcircuit

The appropriate resistor and capacitor values were calculated in Matlab using the Sallen Key equations shown below. With R1, R11, C3, and C4 values of 13kΩ, 8.2 kΩ, 0.001μF, and 470 pF, respectively, the cutoff frequency is 22484 Hz and the quality factor is 0.71.

$$f_c = \frac{1}{2\pi RC\sqrt{mn}} \quad Q = \frac{\sqrt{mn}}{m+1}$$

The biasing, cutoff frequency, and quality factor were all confirmed using Multisim simulation and breadboard prototyping before the final PCB was printed.

Codec Communication, myRIO DIO, and Crystal

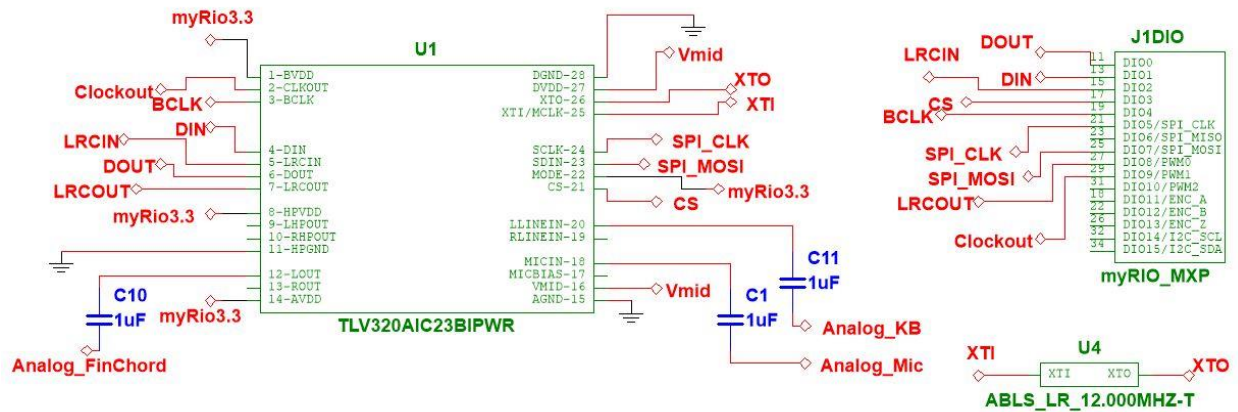


Figure 6: Codec Communication, myRIO DIO, and Crystal

The Codec connections were designed based on the codec datasheet [31] specifications for the datapath and configuration required for this project. The BVDD, AVDD, HPVDD,

DVDD, and VMID were routed to 3.3 V, 3.3 V, 3.3 V, and 1.65 V, respectively, as outlined in the Codec datasheet. The ABL5_LR_12.000MHZ-T Crystal [33] was chosen to fulfill the XTI/MCLK requirements for a 44.1 kHz sampling frequency. The CLKOUT signal was wired to a myRIO DIO pin in case it was needed as a reference in LabView but was not used during programming.

There were two major communication systems between the myRIO and the codec, SPI for the Control Interface and I2S for the Digital Audio Interface, as shown in the Codec block diagram in Appendix B. First, the Control Interface consists of the SDIN, SCLK, CS, and MODE pins. The MODE pin was hard-wired to 3.3 V to set the interface to SPI communication. The SDIN and SCLK were connected to the appropriate SPI pins on the myRIO with the CS signal provided by a DIO pin. Second, the Digital Audio Interface consists of the BCLK, LRCIN, LRCOUT, DIN, and DOUT pins on the Codec. With the Codec in slave mode, each of these signals needed to be provided by a DIO pin on the myRIO.

The analog inputs for the keyboard and microphone were routed to LLINEIN and MICIN, respectively. The MICIN was originally chosen over RLINEIN for internal gain features, but the microphone trace had to be soldered to the RLINEIN on the final PCB due to limitations of the ADC select. The capacitors on the analog inputs, C1 and C11, were used to remove the DC biasing added in the anti-alias filtering. Because the codec does not require external filtering on the LOUT pin, C10 was used as a buffer between the codec output and the audio jack for the piano amp.

PCB Layout

For the design of the PCB, the most important design considerations came from the myRIO connector, the input/output jacks, and the minimization of space used. For the myRIO connector, our initial design did not give enough room for the myRIO to connect to the board without pressing up against other components, so it was placed on one edge by itself to avoid this in the final design. For the input jacks, we chose to place them next to each other so we could keep the board consistent in its organization. We decided to keep all three jacks on the same size to make it easier to place the myRIO/PCB in the chassis. The final design in Ultiboard [12] for the PCB is shown in Figure 7. After the PCB was printed by Advanced Circuits [34], we had the codec soldered on by WWW Electronics [35], due to the precision required. After this, we soldered the rest of the components onto the board by hand. The final board design can be seen in Figure 8.

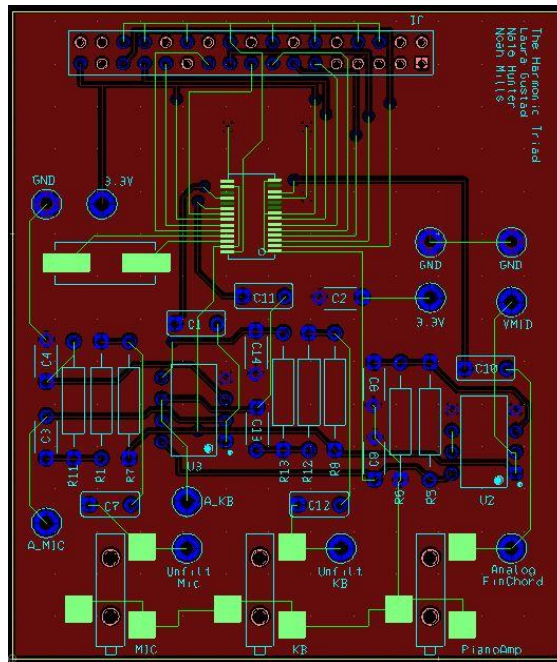


Figure 7: Ultiboard Layout for PCB (GND Plane Hidden)



Figure 8: Populated PCB

Software Design

Overview

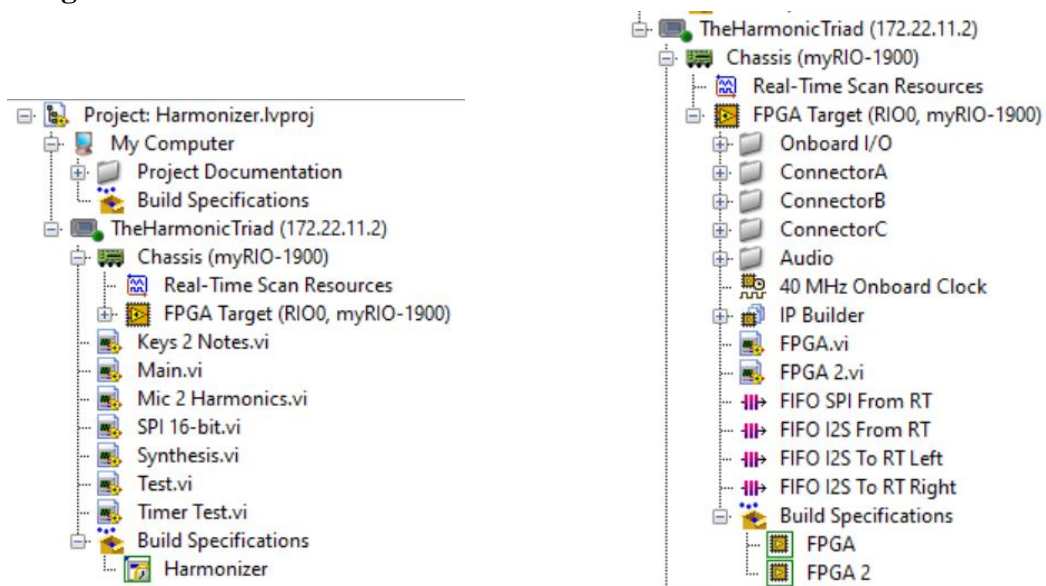


Figure 9: File structure of Harmonizer.lvproj

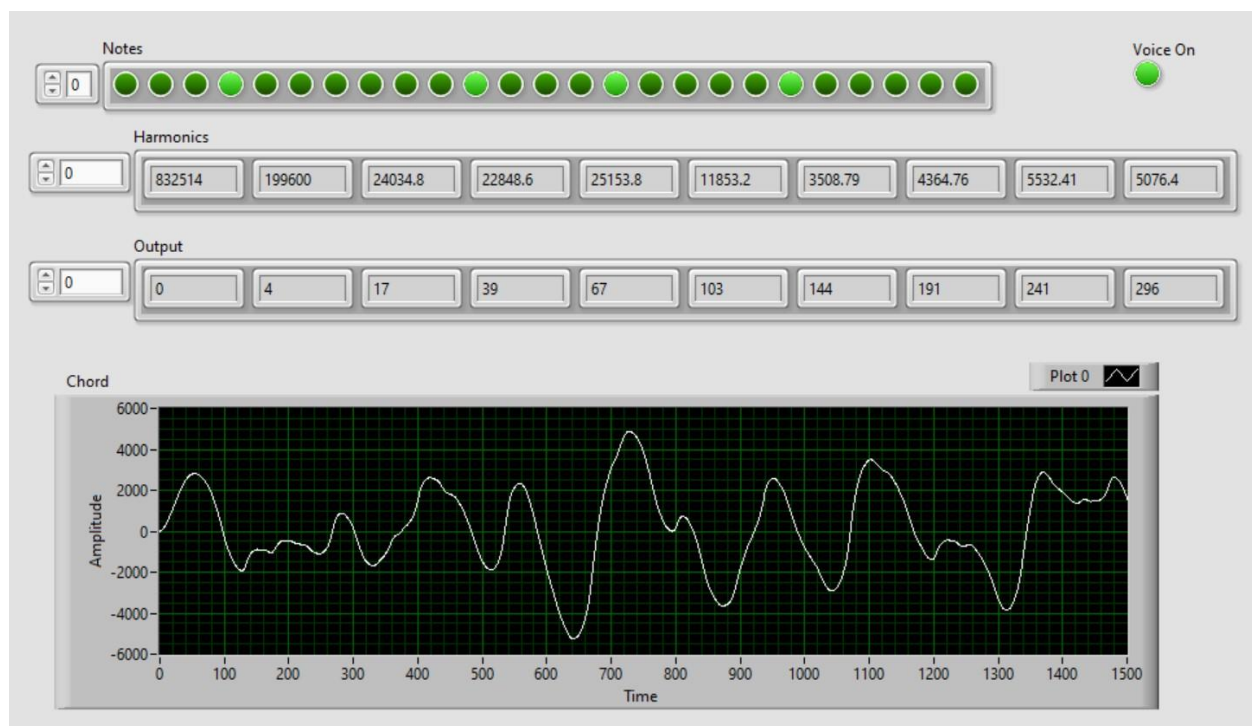


Figure 10: Main.vi front panel in action

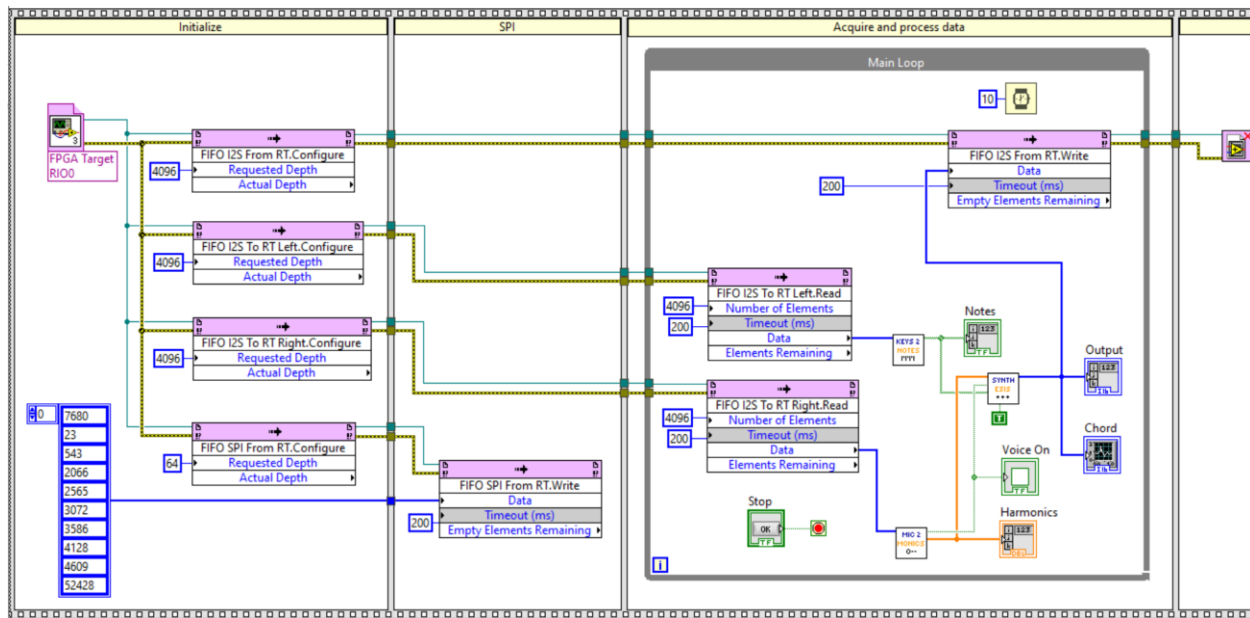


Figure 11: Main.vi block diagram

The software for the harmonizer is contained in a LabVIEW [10] project called Harmonizer.lvproj. This project contains LabVIEW VIs for the real-time processor and the FPGA. The main program running on the processor is *Main.vi*. In the Initialize step, a reference to the FPGA program, *FPGA 2.vi*, is opened, and FIFOs for communication between the processor and the FPGA are configured. In the SPI step, SPI messages are written to the codec through *FIFO SPI From RT*. In the main loop, I2S communication and digital signal processing occur. This runs indefinitely until the program is stopped, at which point the FPGA reference is closed.

Within the main loop, keyboard and mic audio samples are read through the left (*FIFO I2S To RT Left*) and right (*FIFO I2S To RT Right*) I2S channels respectively. 4096 16-bit samples (about 93 ms of sound) are collected per channel in each loop iteration. The keyboard samples are analyzed in the subVI *Keys 2 Notes.vi*, which returns a boolean array of which notes are being played. The mic samples are analyzed in the subVI *Mic 2 Harmonics.vi*, which returns the amplitudes of the first ten voice harmonics and a “Voice On” indicator. These data are then synthesized in *Synthesis.vi*, which produces the output chord as an array of 4096 16-bit samples. These samples are written back to the codec through *FIFO I2S From RT*.

The VIs that are used in this project are *Main*, *Mic 2 Harmonics*, *Keys 2 Notes*, *Synthesis*, and *FPGA 2*. The first four run on the myRIO processor; the fifth runs on the myRIO FPGA. Having looked at the main, we will proceed to look at the other four VIs.

DSP Algorithm

Mic 2 Harmonics

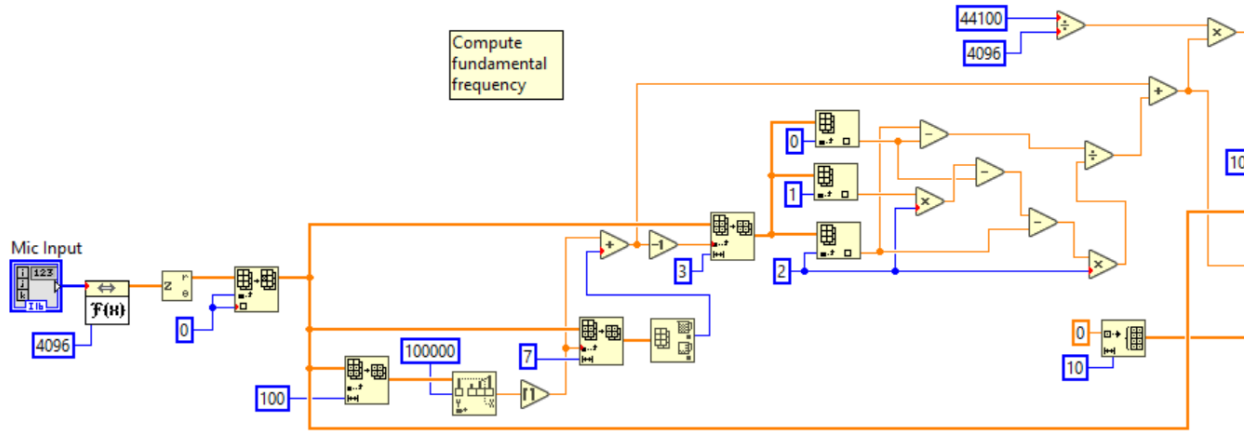


Figure 12: Mic 2 Harmonics.vi block diagram left: fundamental computation

The microphone analysis subVI takes in 4096 samples and runs a Fast Fourier Transform (FFT) on them. This produces an array of 4096 amplitudes at frequencies evenly distributed between 0 kHz and 44.1 kHz (about 10.8 Hz apart). The magnitudes are separated from the phases, and the magnitude at 0 Hz is zeroed out (in testing, we found that the microphone signal had a DC offset). The lowest 100 frequencies (up to 1066 Hz) are isolated and passed through a threshold function, which identifies the first index to cross a magnitude threshold of 100,000 (determined in testing). The maximum magnitude is identified among the next 7 frequencies (determined in DSP development); this index marks the spot closest to the fundamental frequency.

$$\Delta_m = \frac{M[k_m + 1] - M[k_m - 1]}{2(2 * M[k_m] - M[k_m - 1] - M[k_m + 1])}$$

Parabolic interpolation follows, using the formula above [36] to get a more accurate reading of the true fundamental frequency (beyond the 10.8 Hz granularity afforded by the FFT, that is). Finally, the refined index is multiplied by the 10.8 Hz (= 44100 / 4096) bin division to obtain the true fundamental frequency.

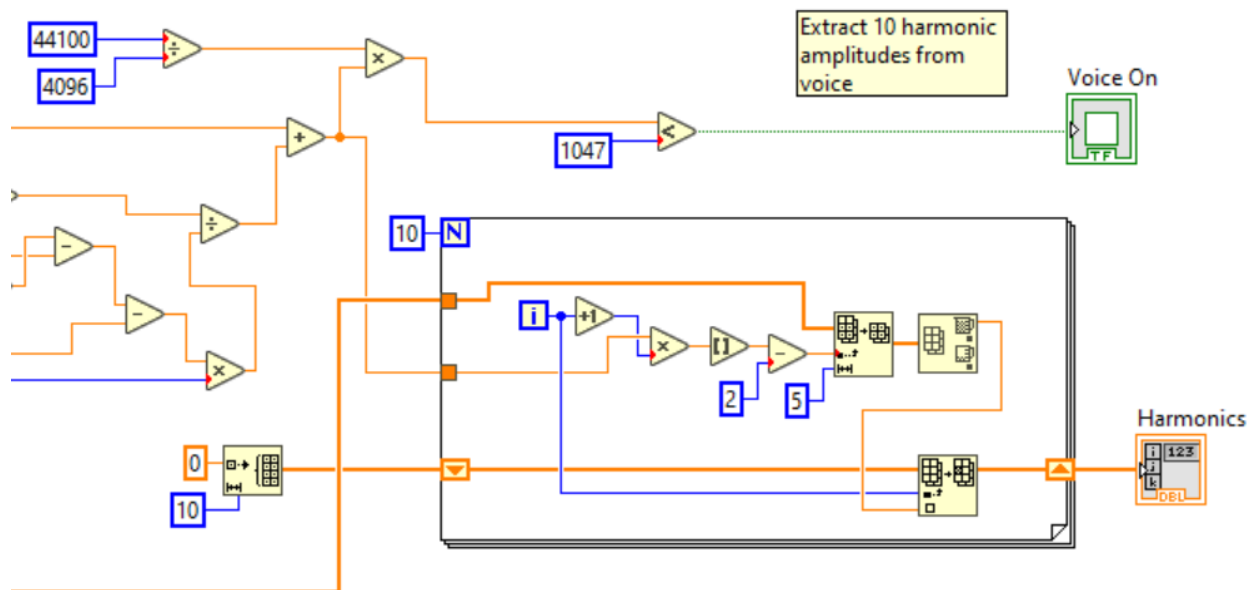


Figure 13: Mic 2 Harmonics.vi block diagram right: harmonic extraction

The fundamental frequency is compared with a maximum at 1047 Hz (determined in DSP development) to establish whether or not the voice is on. This is reasonable, as a soprano's range caps at around 880 Hz [37]. When the voice is off, the 100K threshold is never reached, so the threshold function returns a frequency at the 1066 Hz cap (which may be slightly adjusted by the rest of the computation). This "Voice On" signal is sent to the synthesis VI.

In a for loop, the unscaled fundamental frequency is multiplied by the integers from 1 to 10 to establish ten harmonic frequencies. At each one, a maximum amplitude is found from the nearest 5 (determined in DSP development) FFT bins. These 10 amplitudes are sent to the synthesis VI as "Harmonics".

Keys 2 Notes

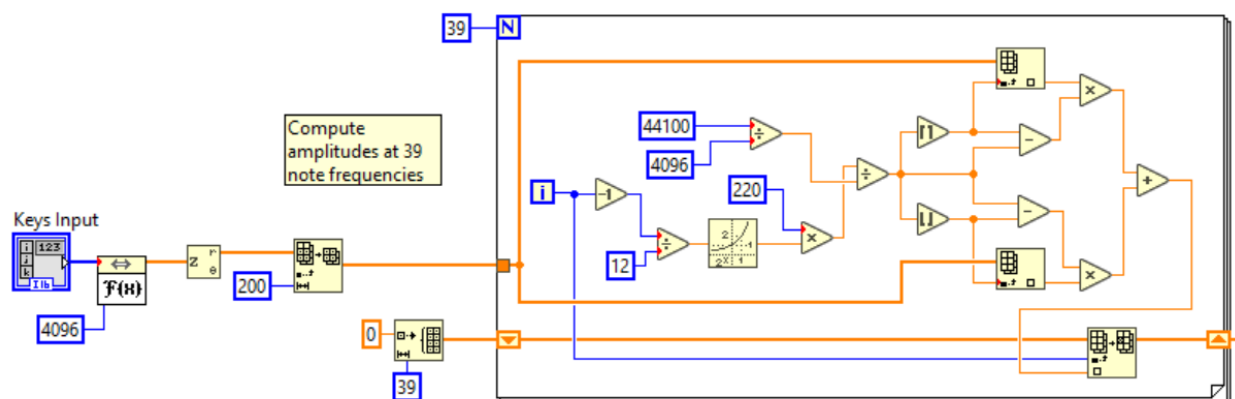


Figure 14: Keys 2 Notes.vi block diagram left: amplitude computation

As with the microphone analysis, the keyboard analysis subVI takes in 4096 samples and runs an FFT on them, producing an array of 4096 amplitudes at frequencies evenly distributed between 0 kHz and 44.1 kHz (about 10.8 Hz apart). The bottom 200 bins are isolated (up to 2143 Hz); higher bins are unneeded. The first for loop runs 39 times to compute 39 note amplitudes in the 3+ octave range of Ab3 (208 Hz) to Bb6 (1865 Hz). Note frequencies are computed with the formula $f = 220 \text{ Hz} \times 2^{n/12}$, where n is the interval between the note and A3 in half steps. Note partial indices are computed by dividing by 10.8 Hz, and the note amplitude is computed with linear interpolation by the formula $A = M[\lfloor p \rfloor] \times (\lceil p \rceil - p) + M[\lceil p \rceil] \times (p - \lfloor p \rfloor)$, where p is the partial index and M[i] is the magnitude at bin i.

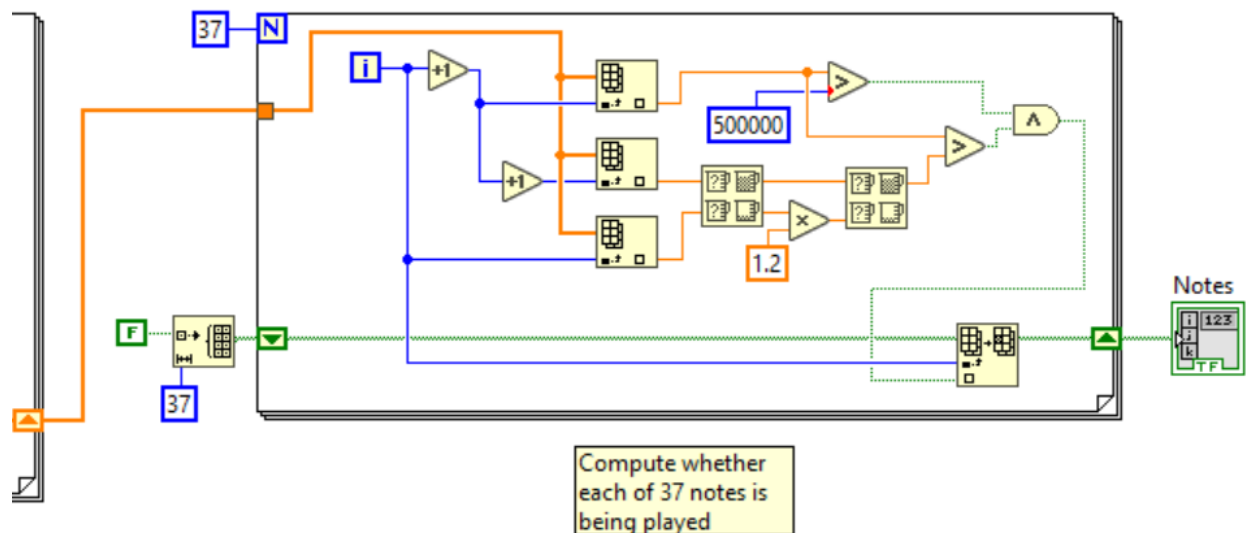


Figure 15: Keys 2 Notes.vi block diagram right: note determination

In the next for loop, the 39 note amplitudes are reduced to 37 boolean signals for whether or not the notes in the 3-octave range from A3 to A6 are being played. For a note to be “on” its amplitude must be greater than 3 numbers: 500,000 (determined in testing), the maximum of its two neighbors, and the minimum of its two neighbors times 1.2 (determined in testing). The neighbor-comparison approach yielded promising results, but with limitations, as explained in “Problems and Design Modifications”. The 37 signals are sent to the synthesis VI as “Notes”.

The synthesis subVI takes in multiple signals: voice harmonics and “Voice On” from *Mic 2 Harmonics.vi*, notes from *Keys 2 Notes.vi*, and a smoothing signal (set to True in the final project). It begins with an empty, 4096-sample array. Looping 37 times, it determines for each of 37 notes (and factoring in “Voice On”) whether or not to add sinusoids. If both the note and the voice are present, it computes a frequency as $f = 110 \text{ Hz} \times 2^{n/12}$, where n is the interval between the note and A2 in half steps. Note that this base frequency is an octave lower than in the keyboard subVI; the reason is that the desired output frequency range is A2-A5, but the higher-frequency notes from A3-A6 have greater frequency resolution in the keyboard FFT. Thus, a musician must play an octave up on the keyboard from the intended output.

Page 24 of 51

reversed when returning to the time domain. We also apply a software gain of 4 here (determined in testing). Thus, we have $\frac{4}{2048} = \frac{1}{512}$.

With each loop, a new sinusoid is generated and added element-wise to the previous chord. At the end of the inner for loop, sinusoids for ten harmonics of one keyboard note are added to the signal, and at the end of the outer for loop, sinusoids for ten harmonics of k keyboard notes (where k is the number of asserted notes) have been summed into a final chord signal.

When the smoothing signal is asserted, a linear fade in/out is added to the 100 samples (determined in DSP development) at each end of the 4096-sample section. This helps to reduce the clipping due to phase differences at the borders between sections. The output chord signal is finally converted to signed 16-bit integers (as the codec expects) and sent to *FIFO I2S From RT* as “Output”.

Data Communication

SPI Messages

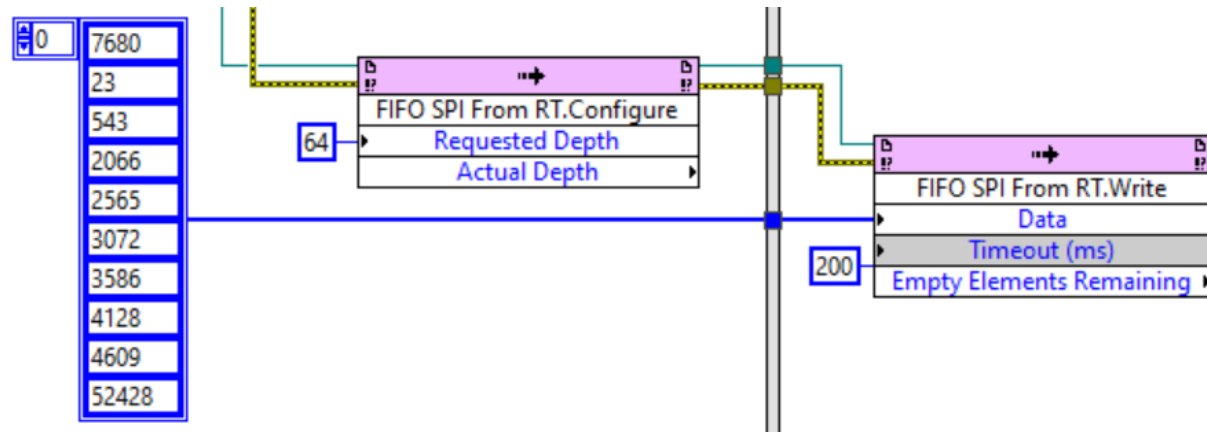


Figure 17: Main.vi block diagram, SPI

The main VI sends 10 16-bit SPI messages to the codec through the FPGA. Each message contains a 7-bit register address followed by a 9-bit configuration setting. Below is the meaning of each message, as determined from the codec datasheet [31].

Message	Address	Input	Register	Function
7680	0001111	000000000	Reset register	Reset all registers
23	0000000	000010111	Left line input channel volume control	Simultaneous gain update disabled, unmuted, no gain
543	0000001	000011111	Right line input channel volume control	Simultaneous gain update disabled, unmuted, +12 dB (x4) gain
2066	0000100	000010010	Analog audio path control	Bypass disabled, DAC selected, mic muted
2565	0000101	000000001	Digital audio path control	DAC mute disabled, de-emphasis disabled, ADC HPF disabled
3072	0000110	000000000	Power down control	Power everything up
3586	0000111	000000010	Digital audio interface format	Slave mode, no swap, no phase, 16-bit word, I2S format
4128	0001000	000100000	Sample rate control	MCLK for CLKOUT and CLKIN, 44.1 kHz sampling rate, normal mode
4609	0001001	000000001	Digital interface activation	Activate interface
52428	1100110	011001100	None	Stop SPI signal (FPGA stops SPI_CLK)

Figure 18: SPI message interpretations

SPI

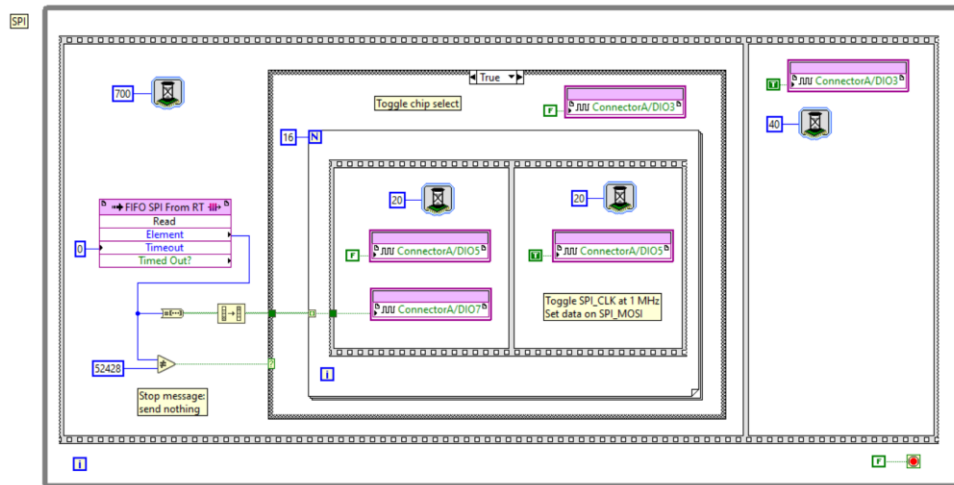


Figure 19: FPGA 2.vi block diagram, SPI

The SPI communication code runs in *FPGA 2.vi*. *FIFO SPI From RT* streams 16-bit messages from the processor to the FPGA. If the stop message (52,428, determined in communication development) is the last message sent, then nothing more happens in the case structure and SPI stops. For other messages, the integer is converted to a Boolean array with the MSB first, and bits are sent in the for loop one by one. While the chip select is low, SPI_CLK is cycled 16 times (once per bit). A cycle is 40 myRIO ticks, which is about a microsecond (one tick is 25 ns); this is well above the codec minimum requirement. The SPI_MOSI pin is set to the message bit on the falling edge, so that it can be read by the codec on the rising edge. At the end of the message, the chip select is asserted, and the codec is ready to read a new message.

I2S

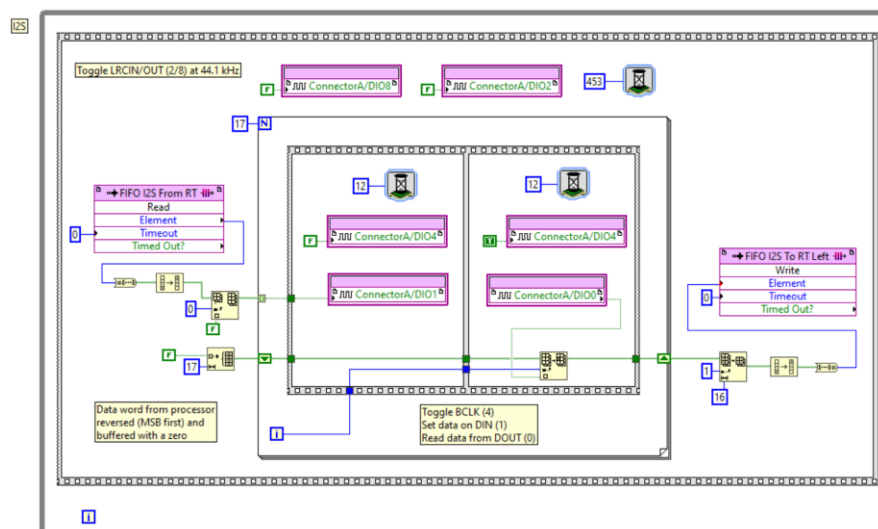


Figure 20: FPGA 2.vi block diagram, I2S left channel

The I2S communication code also runs in *FPGA 2.vi*. LRCIN and LRCOUT are cycled at 44.1 kHz, which works out to approximately 453 myRIO ticks between toggles. While LRCIN and LRCOUT are low, the codec's left channels are written and read, as shown above. For writing the output chord to the codec, 16-bit signed integers are sent to the FPGA through *FIFO I2S From RT*. These are converted to boolean arrays with the MSB first and buffered with a zero, per codec specifications [31]. BCLK is toggled at 24 ticks per cycle, which is the maximum allowable without slowing down the loop (determined in testing). Bits are written on DIN on the falling edge so that the codec can read them on the rising edge.

For reading the input chord from the keyboard, the same clock structure is used. A boolean array is initialized and populated with the value provided by the codec on DOUT on the rising edge. The initial zero is removed and the boolean array is converted to a 16-bit unsigned integer and sent to the real-time processor through *FIFO I2S To RT Left*, which treats the integer as signed. Both the input and output streams run constantly, transferring numbers between the codec and the processor.

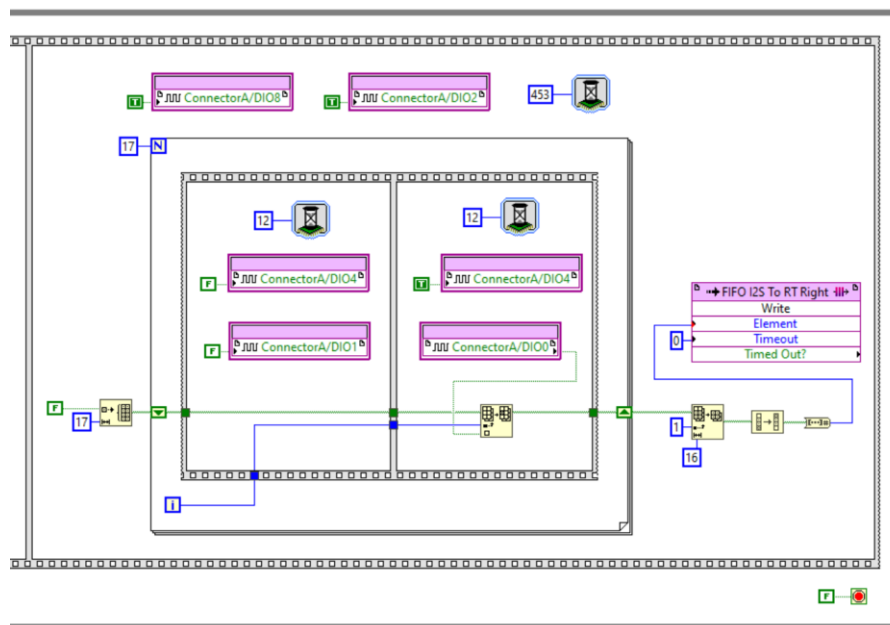


Figure 21: FPGA 2.vi block diagram, I2S right channel

While LRCIN and LRCOUT are high, the codec's right channel is read. The right write channel is unused. A signal from the microphone input is communicated through this channel in the same way that the keyboard signal was communicated through the left channel, except using *FIFO I2S To RT Right*. By alternating in this way, the keyboard signal and the microphone signal can be streamed to the processor simultaneously.

Mechanical Chassis

The mechanical chassis was designed to easily hold the myRIO/PCB on the piano, so that a user of the system did not need to be concerned with them falling while operating the system.

The chassis was designed in AutoCAD [24], which allowed for precise measurements of each dimension, as required by the shape of the myRIO and the hole on the piano in which the chassis would rest. The dimensions of the chassis came out to be 3.0" x 1.46875" x 3.81252". The final AutoCAD design for the chassis is shown in Figure 22.

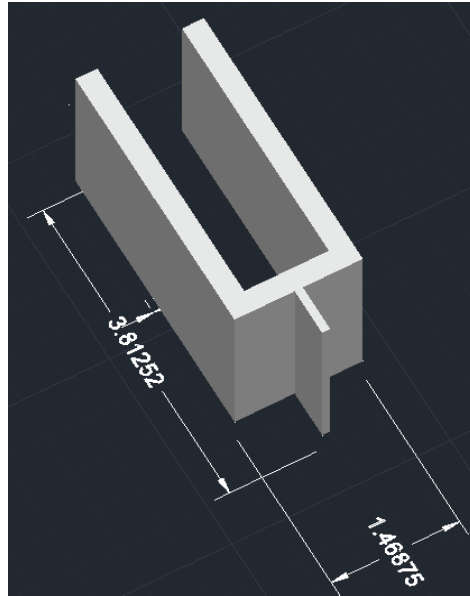


Figure 22: Chassis design in AutoCAD

After the chassis was designed in AutoCAD, the .dwg file was exported to .stl and .igs. By doing this, the chassis could be 3D printed by the Mechanical Engineering Department [7]. The two files were emailed to Professor Powell, he contacted the MECH-E Department on our behalf, and they printed the final product, as shown in Figure 23.

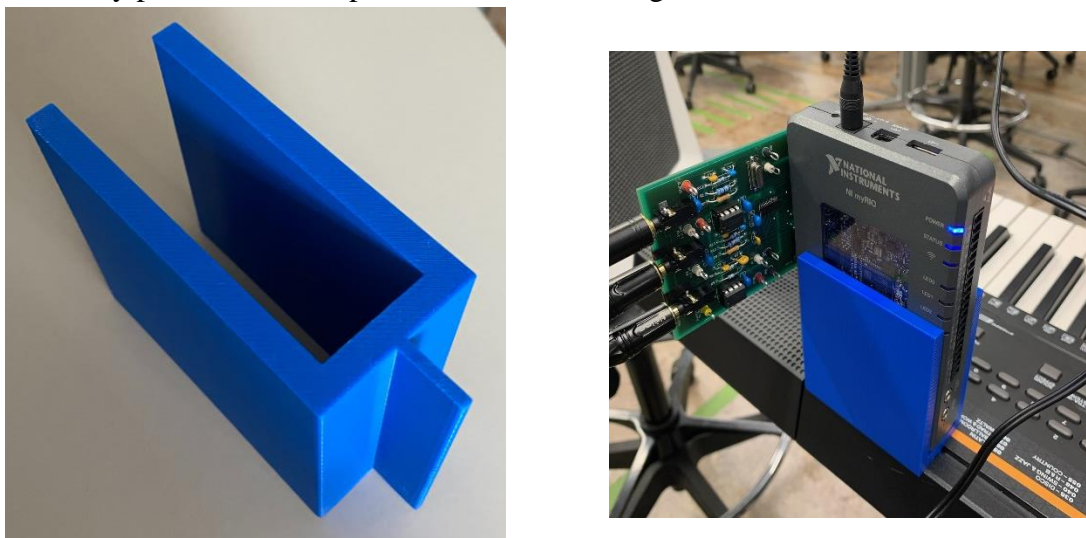


Figure 23: Final Printed Chassis

Problems and Design Modifications

Note: this section will also include discussion of tradeoffs and design decisions made (particularly on the software side).

Hardware

The first iteration of the PCB did not correctly bias the keyboard and microphone signals to 1.65 V, which caused the output from the anti-aliasing filters to rail to one of the op-amp supplies. To fix the railing, we cut the traces causing railing and soldered the non-inverting input and output of the op-amp together. In order to bias the input signals to 1.65 V, we “debugged” the PCB to add the necessary 33 k Ω resistors and 1 μ F capacitors as shown in Figure 24.

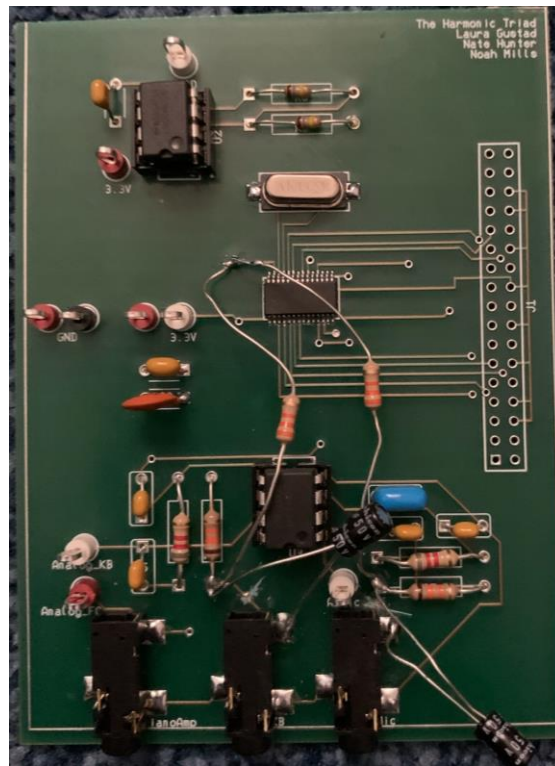


Figure 24: First Iteration PCB with Debugging

The final iteration of the PCB corrected the biasing and anti-aliasing filters, but we discovered an issue with the 3.3 V supply from the myRIO. When connected to the myRIO, the 3.3 V supply dropped to 2.65 V and the VMID generation dropped to 8 mV. The error laid with the ground plane, specifically the myRIO GND did not share the same GND as the rest of the PCB. To rectify this, jumper wires were soldered to connect the myRIO GND references with the ground plane of the PCB.

Finally, the final iteration of the PCB had two other small trace errors that were able to be fixed with soldering. During testing of the keyboard, microphone, and piano amplifier signals on the PCB, the signals from the audio jacks for the keyboard and piano amplifier were unusual. To

correct these signals, the pin for the ring signal from the jack had to be desoldered and electrical tape added between the solder pad and the pin. The second error was discovered when working on the SPI configuration for the datapath to read and write from the Codec in I2S format. According to the Codec datasheet [31], the ADC cannot be used on the MICIN and LLINEIN channels simultaneously due to the fact that the input MUX select that decides mic or line for the ADC is shared between the left and right ADC. In order to read the keyboard and microphone inputs simultaneously, the MICIN pin and RLINEIN pins were soldered together on the Codec.

Software

One of the first tradeoffs we had to consider was how much of the vocal content we would capture. Initially, the goal was to include non-harmonic content. However, it quickly became clear that synthesizing non-harmonic content from a FFT would be extremely sloppy due to spectral leakage [36]. One option would be to use a phase-vocoder technique, which shifts signals in the frequency domain before converting back to the time domain [39], but it was unclear how well this would work with multiple pitches. Thus, we decided to stick with harmonic content only.

The length of the FFT was a major tradeoff to consider. By the nature of the algorithm, the frequency resolution is inversely proportional to the length of time captured [36]. We found 4096 samples to be a good balance, yielding a time period of 93 ms (close enough to real time) and a frequency resolution of 10.8 Hz (small enough to identify the fundamental and later to determine notes on the keyboard). We considered implementing an overlap of $\frac{3}{4}$ to reduce the update delay, but the complexity of this approach, the additional computational demands, and the sufficiency of the existing time period motivated us to avoid this [40].

In identifying the fundamental of the voice, we had to consider interpolation. Leaving out interpolation would give a result that could be off by more than 5 Hz, which was too much. Two options were parabolic interpolation, which is approximate but simpler, and Gaussian interpolation, which is the most precise [36]. Parabolic interpolation turned out to be sufficient, as the error has been shown not to exceed 4% of the bin spacing [41].

For analyzing the keyboard signal, we initially considered a strobe tuner approach, which would compare the input signal to expected note frequencies to establish the pitch [42]. Upon implementing this, however, numerous problems arose: using low-pass filters required more time to detect a DC signal, and using sums failed because AC frequencies from other notes in the chord overpowered the DC difference we were trying to detect. Thus, we decided to use an FFT for the keyboard as well, approximating magnitudes at various notes with interpolation. This worked better; however, it limited us to higher keyboard frequencies where there were more bins between notes. To determine which magnitudes were large enough to indicate the presence of a note, absolute magnitude alone ran into significant issues due to spectral leakage. Thus, we decided to also compare note magnitudes to their neighbors. This worked well, but the downside is that notes are not guaranteed to be detected unless separated by at least a minor third (usually a

whole step is fine too). Another issue is that, due to the low bin resolution at lower frequencies, one of the bottom notes in our input range, Bb3, was mistaken by the algorithm for B3. Thus, instead of an output range of A2 to A5, our correct output range is B2 to A5.

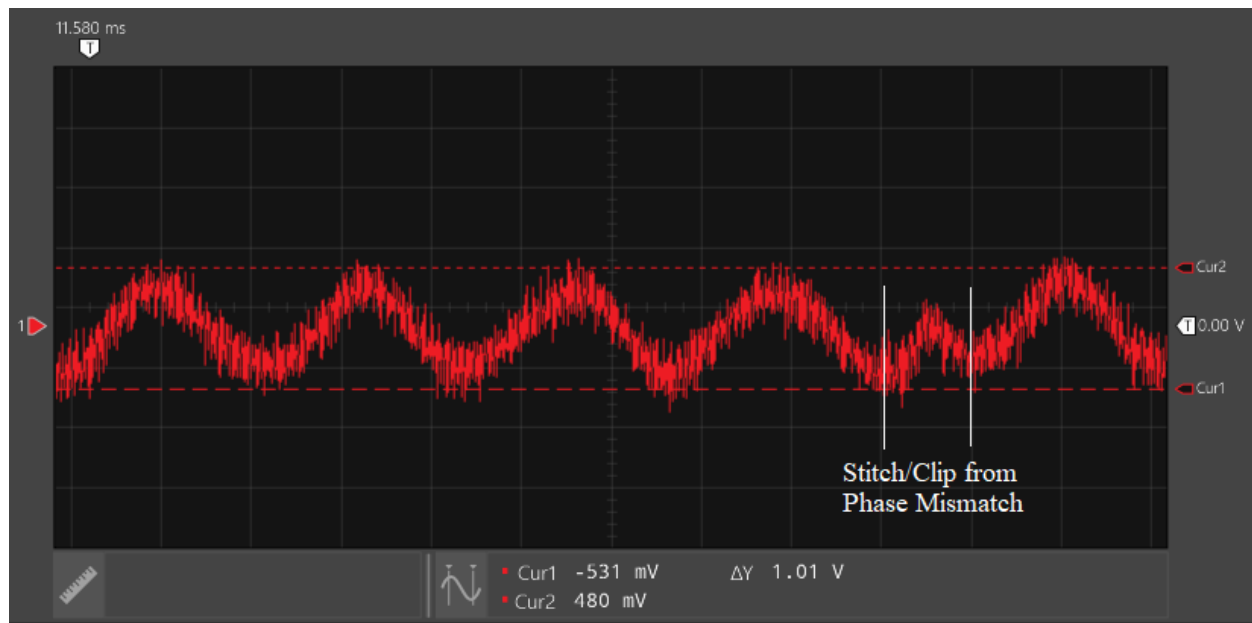


Figure 25: Phase Clipping Between Two Audio Sections

On the algorithm output side, clipping between audio sections became a big problem. A possible solution to this would be to align the phases of the component sinusoids between sections. This step would be a great future improvement. Due to time constraints, we instead implemented smoothing at the ends of each section, which dampened the impact of the clipping. Another output consideration was volume. Unfortunately, due to the codec's inability to transmit mic and line data simultaneously, we did not have much hardware gain to work with on the mic input (the multiplier was about 4). Thus, we added an additional software gain of 4 to the output.

When moving the code to the myRIO, we had to decide what to put on the FPGA and what to put on the processor. We initially intended to put everything on the FPGA in order to process the signals as fast as possible. However, the FPGA code was much more limited than the standard LabVIEW palette, and the FPGA did not work well with storing large arrays of data. We then timed the algorithm on the real time processor, and estimated a maximum delay of only 30 ms, well under the 93 ms period allotted. Thus, the entire DSP algorithm went on the processor.

The last major hurdle was that when we tried to run SPI and I2S communication in the same program, we got no signal whatsoever. We initially thought that the I2S wasn't doing anything, but upon further investigation, we found that the SPI messages were no longer sending. We were using a dedicated subVI for SPI and the FPGA VI for I2S. With Prof. Powell's help, we figured out that these two VIs could not run on the FPGA simultaneously, so we reconfigured

the SPI to run on the same FPGA VI as the I2S. From there, it was (relatively) smooth sailing to the finish line: a working vocal harmonizer.

Project Timeline

To create our timeline, we used a free online Gantt chart tool [43]. The chart in Figure 26 shows our initial prediction for our project timeline. Due to problems such as difficulty configuring software, reordering a new PCB after discovering problems with the initial design, and the pandemic-caused complications with meeting in person, not all sections were completed in accordance with the initial timeline. The Gantt chart in Figure 27 shows a much more accurate timeline, as this was created at the end of the semester.

Tasks that had to be done serially were signal design followed by LabVIEW coding. Circuit design was followed by the PCB layout, which was followed by the PCB assembly. In addition, the PCB assembly and the chassis design had to be complete before beginning the final assembly. Finally, all of the previous tasks were completed before testing performance began.

Tasks that were done in parallel include component purchases, signal design, and circuit design. After those were completed, LabVIEW coding and the PCB layout were done in parallel. Then, the PCB was assembled while the testing system was designed. The chassis design was also completed at this time.

- *Primary Responsibilities:*
Laura: Component purchase, circuit design, PCB layout
Nate: Signal design, LabVIEW coding, final assembly
Noah: PCB assembly, test design, test performance, chassis design
- *Secondary Responsibilities:*
Laura: Signal design, LabVIEW coding, PCB assembly, final assembly
Nate: Component purchase, test design, test performance
Noah: Circuit design, PCB layout, chassis design

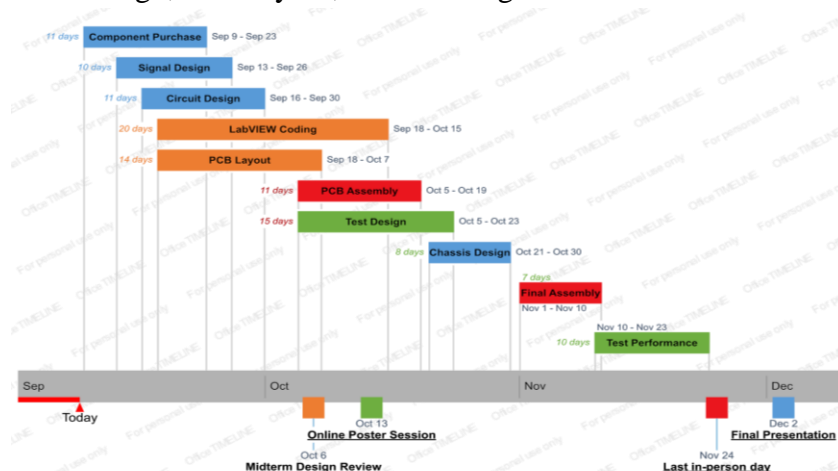


Figure 26: Original Gantt Chart

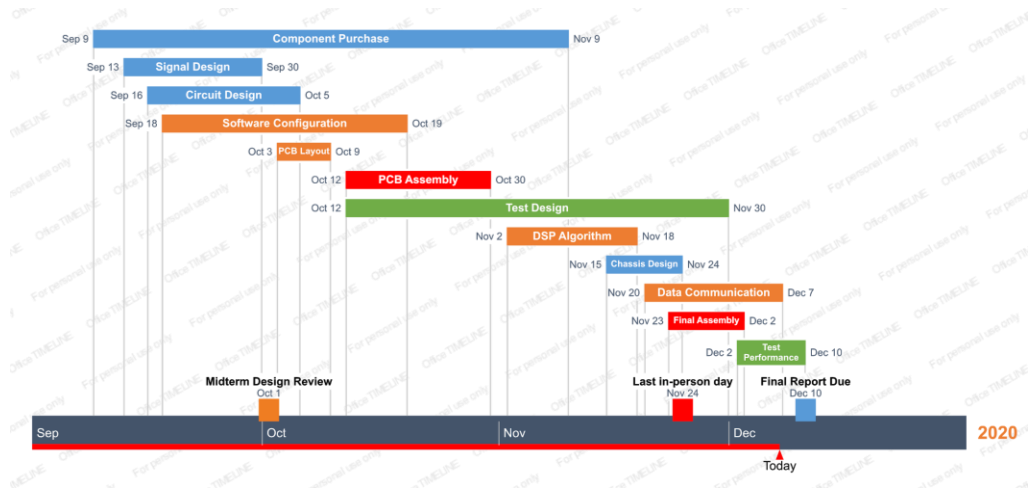


Figure 27: Final Gantt Chart

Test Plan

Hardware

The vocal harmonizer was split into multiple subsections in order to test most effectively. The hardware was divided into input signals, filters, signal analysis, and output signals. During the initial hardware design process, the circuitry was tested in Multisim [11] by performing AC sweeps on filters to confirm that they properly filtered out signals as we designed. After the PCB was printed, each hardware subsystem was tested by using assembling on a breadboard connected to a VirtualBench [25]. The VirtualBench provided input signals while also measuring the output of the subsystem. After successfully testing the subsystem, we soldered the individual components onto the PCB and tested that subsystem again. This process allowed us to ensure that each subsystem functioned on its own, which made debugging the entire system much easier later on. Figure 28 and Figure 29 show the original hardware test plans. The original hardware test plans were followed closely, however, we spent more time analyzing the signals coming out of the anti-alias filters than originally planned. This was due to a biasing problem, which was later fixed.

After testing our original PCB, we came to the realization that our anti-aliasing filters did not bias the input signals to 1.65 V as desired. Because of this, we had to partially redesign the anti-alias filters to achieve the desired signal bias. The original PCB also lacked a ground plane in the circuit, which was added to the final PCB. Additionally, we realized that the first PCB did not have enough room to easily connect to the myRIO via the 34-pin connector, so the final PCB was redesigned to allow enough space for that connection.

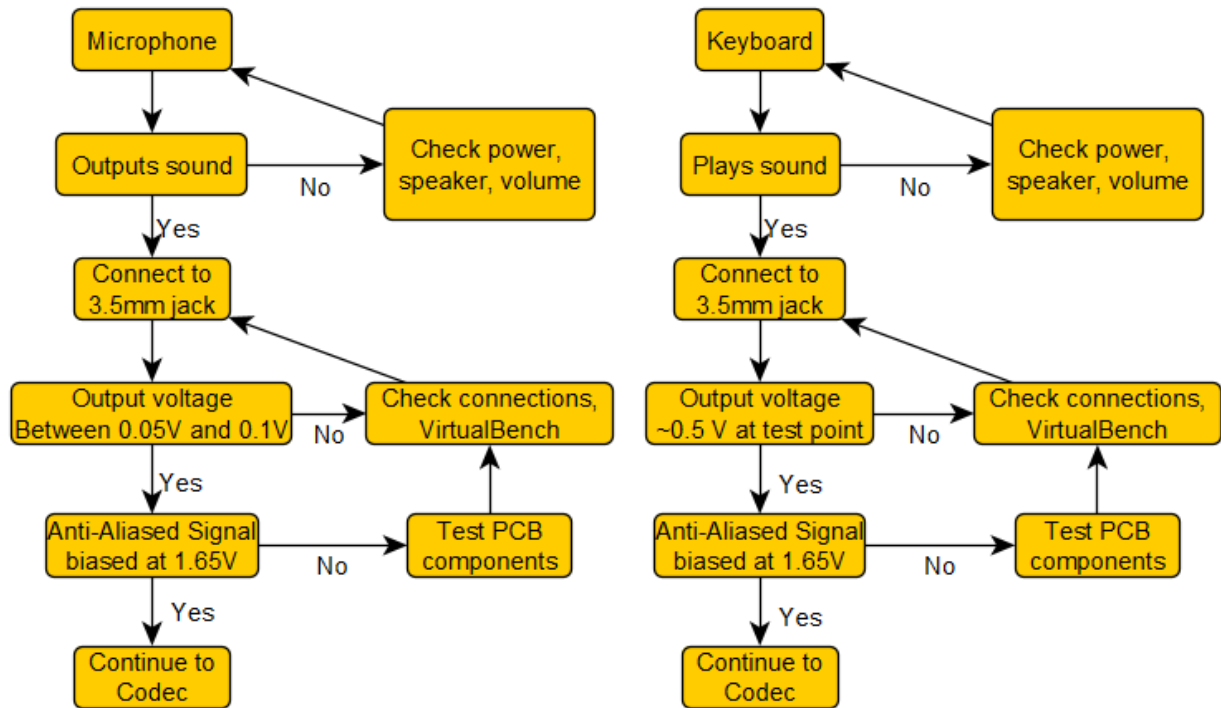


Figure 28: Input Signal Test Plan

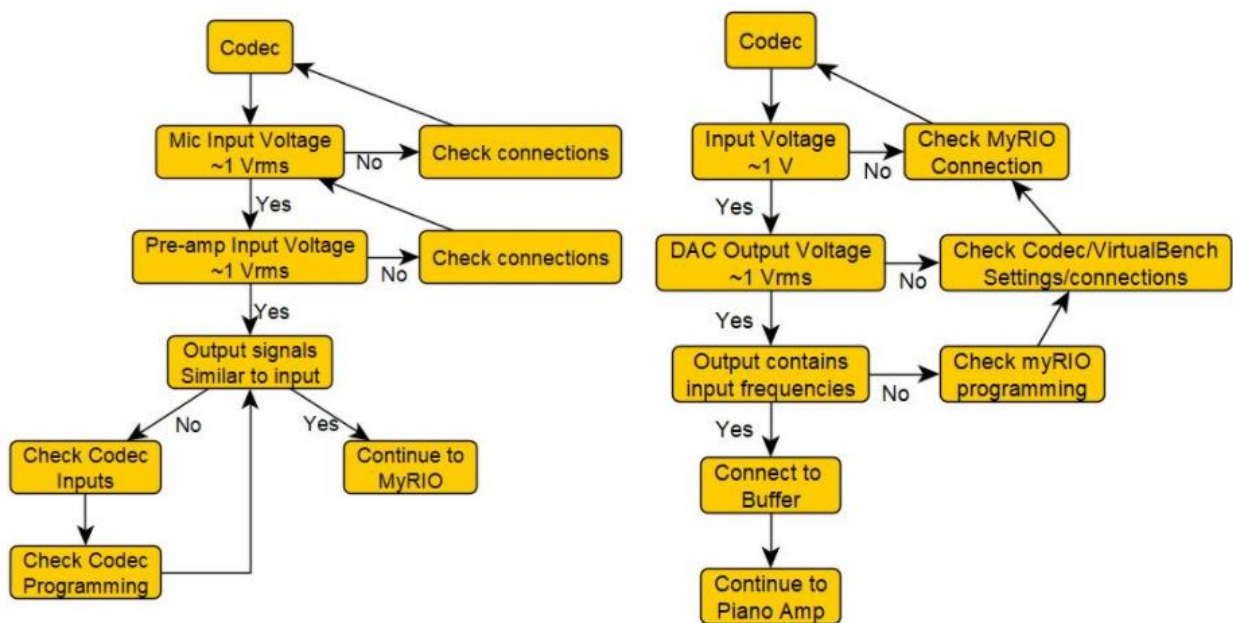


Figure 29: Codec Connection Test Plan

Software

Testing of the DSP algorithm occurred somewhat in parallel with its development (i.e. “code a little, test a little”). First, the FFT was tested with the voice input to determine that the

frequency of the voice was picked up in the appropriate bins. Then, the fundamental frequency identification was tested. The input frequencies were C3 (131 Hz) and F3 (175 Hz), both of which were read appropriately. Next, the harmonic detection was tested by comparing the output with the display of the FFT and seeing that they matched. Additional confirmation of this was seeing the distinct harmonic components of the five different vowel sounds in the input voice WAV file. Having completed the microphone input section, testing proceeded to the keyboard side. Preliminary tests revealed that the strobe tuner approach was not working as expected. After playing around with it, this approach was replaced with an FFT and neighbor comparison approach. Five different keyboard chords, each containing four or five notes, were run against the algorithm, and not a single note was missed in any of them. Testing proceeded to the output. To confirm that the output was adapting properly to the inputs, sections from each of the five vowels in the voice WAV and from each of the five chords in the keyboard WAV were combined. The output chord matched the keyboard chord in each case, and the output timbre also shifted with each new vowel as expected.

The data communication protocols were tested individually to confirm timing and functionality using the DIO tool with the VirtualBench as shown in Appendix C. For the SPI, a DIO trigger was used to detect an SPI message burst and confirm that it matched the shape in the codec datasheet. Then, the necessary Control Interface commands to set the left channel in BYPASS mode and turn on the line input were sent. As a null test, the keyboard was played before the SPI code was run to ensure no sound was emitted from the amp. Then, the sound emitted from the amp after the code was deployed, confirming that the SPI write was successful. For the I2S, the VirtualBench was used to send square waves of known frequencies to DOUT on the myRIO to confirm read capabilities. To test write capabilities in I2S, the DIO was used once again to confirm the shape of the transmission. Then, a 220 Hz digitally-represented sinusoid was sent from the myRIO through DIN to the codec to play from the amp. To test I2S in conjunction with SPI, the codec was configured such that the keyboard input would be sent through the ADC to the myRIO through DOUT, back to the codec through DIN, and through the DAC to be played on the amp.

Full System

To test the entire system, after confirming that both the hardware and software operated as expected individually, we connected the myRIO to the PCB, and the PCB to the microphone and keyboard. We tested the keyboard input first, confirming that the sinusoidal signal of a note could show up on a waveform graph and then that the notes of the keyboard corresponded to the notes lighting up on the front panel. Next, we tested the microphone input, confirming the visibility of a signal in a waveform graph and clear changes in the harmonic amplitudes with the voice both on and off. A selection of the LabView screenshots used to confirm these tests are shown and explained in Appendix D. We also tested that both the mic and the keyboard could be heard when piped directly to the amp. Finally, we hooked these components up to the synthesis subVI and confirmed that the desired output signal was displaying on a waveform graph. As

shown below in Figure 30, an octave on the keyboard shows up on the LED display and in the waveform graph of the output signal (a signal of period ~320 samples plus a signal of period ~160 samples is apparent), while the voice signal is confirmed in the indicator light, the harmonic array, and the relatively smooth triangular shape of the output. Vowel signals can be compared in Appendix E.

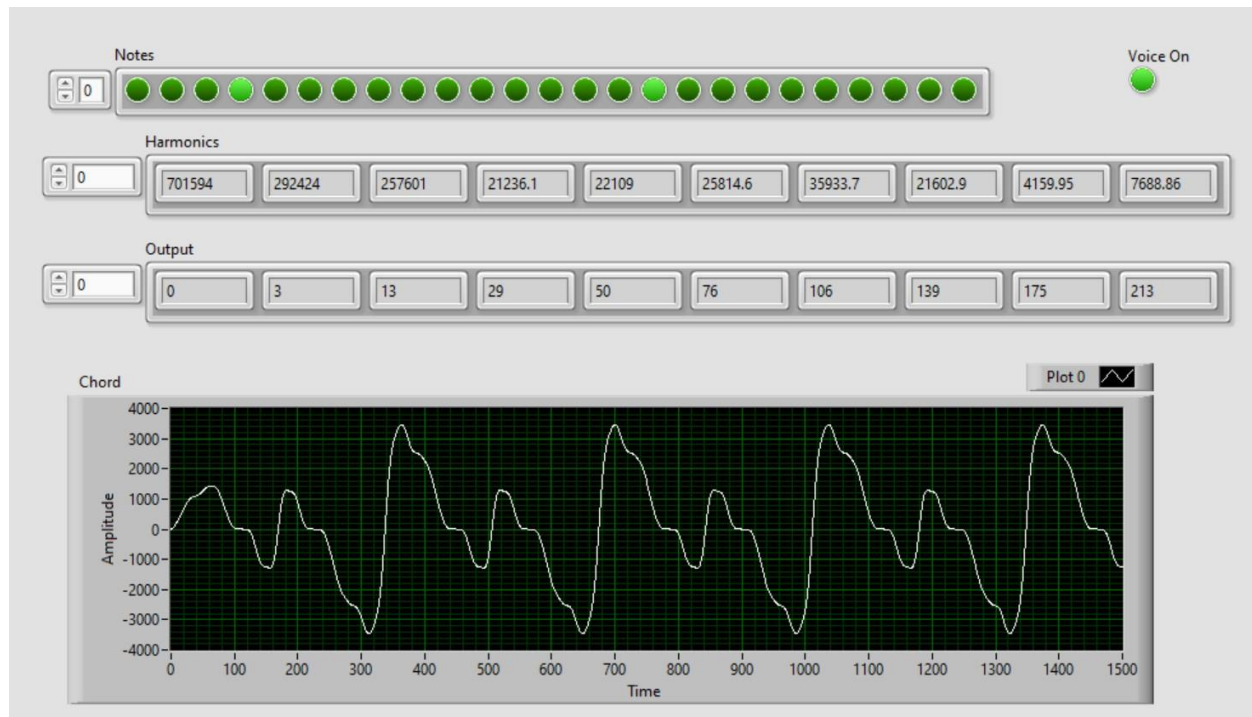


Figure 30: Full System Test - C3 + C4 on Keyboard, “Oo” Vowel

After connecting all of the subVIs and the amp output, we attempted to sing into the microphone while playing the keyboard, and we found that the system worked as expected, harmonizing the voice to the chord of the keyboard. We also tested this after deploying to the myRIO and unplugging from the laptop, and got the same result. To continue testing, we changed volume while singing and playing the piano and found that the output proportionally changed volume, which was the desired result. We then attempted to change singing pitch while maintaining the same notes on the piano, and found that the system worked as planned and the output pitch was not impacted by the microphone. We also spoke/sang different words into the microphone and found that the output could clearly be heard when changing vowel sounds. The words “wow” and “yeah” could be distinguished, but more complex words such as “harmonizer” were much harder to understand. We had anticipated that this was a possibility due to the time resolution of the algorithm, so this was not a problem that we attempted to fix.

Final Results

Our team completed a fully-functional vocal harmonizer. The system functions as its own module, which is to say that the myRIO can run independently from a computer. The system as a

whole contains the keyboard, the microphone, the PCB, the myRIO, and the output amp. When the user sings into the microphone, nothing happens unless they are also pressing keys on the keyboard. Similarly, the keyboard only outputs sounds when the user sings into the microphone. This is exactly what we intended, so we consider the project to be a success.

The system outputs in real-time, which means that the delay between singing into the microphone and hearing output from the system is reasonably small. The most significant problem that we found is that the output is not extremely clear, and it is not easy to hear spoken words into the system. That said, the system still gives the sense of performing within a group.

The text in Figure 31 is an excerpt from the initial project proposal.

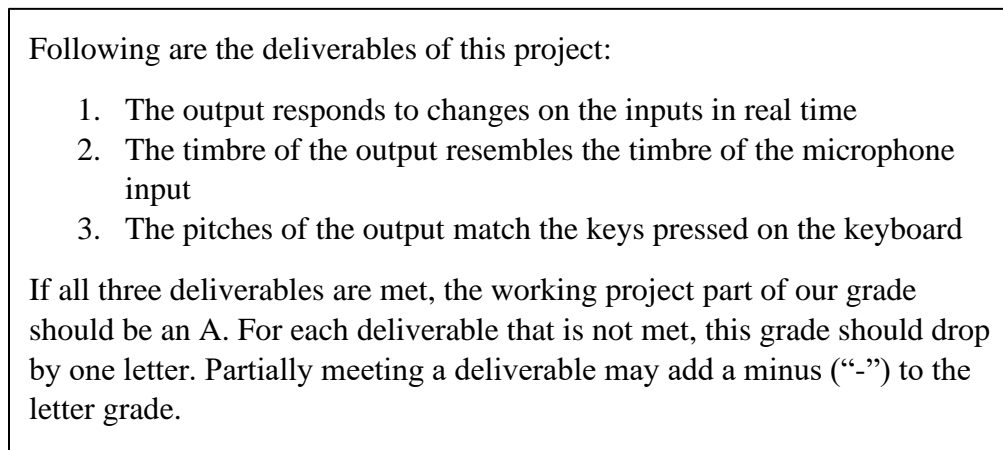


Figure 31: Initial Proposal Deliverables

Figure 31 shows the 3 gradable deliverables for the project. Deliverable 1 stated that the output must respond to changes of the inputs in real time. This was successfully accomplished: at under a tenth of a second, the delay between input and output is not too bothersome when using the system. Deliverable 2 stated that the timbre of the output must resemble the timbre of the microphone input. There is a resemblance, in that vowels can be distinguished from each other and in that the shapes of the input and output waveforms for a particular vowel resemble each other. However, the output is not all that human-sounding, due to the clipping, time resolution, and lack of non-harmonic content. Thus, this deliverable was partially met. Lastly, deliverable 3 states that the pitches of the output match the input keys from the keyboard. This final deliverable was fully met, since the output pitches were based solely on the keyboard input and were faithfully produced in the range of B2 to A5. Since we successfully completed two of the deliverables, and partially completed the third, our finished product has earned an A-.

Costs

The cost of the vocal harmonizer system for our production was \$138.20. This included the PCB printing, PCB parts, and the 3D printed holder. We were provided with the myRIO (FPGA) and a keyboard. A microphone was purchased for testing, in addition to cables to connect to the system. We used an amp for the output that we already had available. A full

breakdown of the budget can be found in Appendix F. To determine the estimated cost of mass production, we received a quote from Protolabs for the holder [44]. For the PCB, the cost estimate for mass production came from a quote from Advanced Circuits [34]. Typically, a myRIO system costs between \$500 and \$700. If the vocal harmonizer were going to be mass produced, an FPGA that is specifically designed for this system (likely an ASIC) would be designed and manufactured. We received a quote from Sigenics [45] to create a custom FPGA with our specifications, and its cost is shown in Figure 32.

Item	Cost for 1	Estimated cost (each) for 10,000
PCB	\$71.20	\$24.80
FPGA	\$578.00	\$9.60
3D Printed Holder	\$67.18	\$3.16
Total	\$716.38	\$37.56

Figure 32: Table of Costs

Future Work

For future projects, there are a few possible topics that could be explored based on our project. We have selected four to discuss in the following section.

The first possible future topic would be to find a method of pre-recording. This idea came from the realization that it would be nearly impossible to play an instrument (trumpet, violin, flute) into the system's microphone while also playing the piano key for input. To account for this, a future iteration of this project could include pre-recording the keys/chords that they want to play. This would involve having a "recording session" in advance, saving the chords on the FPGA, and then playing at a time specified by the user. There are many technical challenges to implementing this expansion, specifically with saving the notes on the FPGA and giving the user the option to switch to "recording mode" with the keyboard.

A second future work could improve on the first idea, and allow for pre-recorded inputs for both chords and the user voice/instrument input. While this idea would allow the most options for a user, it would also face the most challenges. It would require multiple levels of user input to select the type of input, multiple storage systems, and the ability to read memory and output multiple sounds at the same time. This would also face more ethical issues than our current product. When storing user input in memory, the user's voice print will be available to be stolen from the memory. If the system is compromised, the voice print could be taken without the user's consent and used to impersonate them. To account for this, various security measures would need to be taken to confirm the device is being used only as designed.

Another idea for future work involves fixing some problems with the final PCB and improving the software and 3D printed chassis to allow for a more robust and clean system. These fixable problems include fixing the ground references for the 3.5 mm jacks, and wiring the microphone input to the RLINEIN input on the codec. We were able to correct both of these problems in our design by un-soldering the ground references and soldering the RLINEIN input to the LLINEIN input on the codec. However, while these solutions work for this board, if the board were going to be produced again, these problems should be addressed in the initial design. If this project were continued, we could also consider using a different stereo Codec with the capability to read both microphone and keyboard inputs simultaneously. With respect to the software, the current algorithm creates an output signal in 4096 16-bit word increments, which creates a clicking noise at each stitch as the signals are out of phase. In the future, we could explore options for better stitching the signal together.

The final topic to explore for future work involves creating a larger and more secure case for the system. At this point, the holder we designed only cradles the myRIO to keep the system secure on the piano. If the holder were larger and more encasing, the system would be more protected from the environment (dust, falling objects) and also any intentional tampering. To do this, we would consider the dust and water specifications NEMA requires for manufactured mechanical casings in the redesign. This holder would be more costly than the original holder, but would likely be worthwhile to protect the system from any damage.

References

- [1] I. M. Dansey, “ThereFret,” 2013, Charlottesville, VA, 2013.
- [2] *Music Makeathon: Harmonizer*. 2018.
- [3] Roland JD-Xi - *How to use Vocoder and Auto Pitch*. 2016.
- [4] JACOB COLLIER: WTF IS A HARMONISER? | EFG LONDON JAZZ FESTIVAL PREVIEW. 2015.
- [5] “DigiKey Electronics - Electronic Components Distributor.” <https://www.digikey.com/> (accessed Dec. 01, 2020).
- [6] “Arctic Violet,” *Arctic Violet*. <https://arcticviolet.com/> (accessed Dec. 02, 2020).
- [7] “MAE Rapid Prototyping and Machine Labs, U.Va.” <https://rpl.mae.virginia.edu/> (accessed Dec. 01, 2020).
- [8] “PCB Tolerances | Advanced Circuits.” <https://www.4pcb.com/pcb-design-specifications/> (accessed Sep. 08, 2020).
- [9] “MATLAB Documentation.” <https://www.mathworks.com/help/matlab/> (accessed Dec. 02, 2020).
- [10] “What is LabVIEW? - NI,” *NI*. <https://www.ni.com/en-us/shop/labview.html> (accessed Sep. 15, 2020).
- [11] “Multisim Live Online Circuit Simulator,” *NI Multisim Live*. <https://www.multisim.com/> (accessed Dec. 04, 2020).
- [12] “Ultiboard,” *NI*. <https://www.ni.com/en-us/shop/software/products/ultiboard.html> (accessed Sep. 15, 2020).
- [13] “fffb5c2b5dd8c965428a2783fef5a567.pdf.” Accessed: Dec. 10, 2020. [Online]. Available: <http://www.vipcircuit.com/data/download/201810/29/fffb5c2b5dd8c965428a2783fef5a567.pdf>.
- [14] “IPC-2221A(L).pdf.” Accessed: Dec. 10, 2020. [Online]. Available: [http://www-eng.lbl.gov/~shuman/NEXT/CURRENT_DESIGN/TP/MATERIALS/IPC-2221A\(L\).pdf](http://www-eng.lbl.gov/~shuman/NEXT/CURRENT_DESIGN/TP/MATERIALS/IPC-2221A(L).pdf).
- [15] “FreeDFM - A Service of Advanced Circuits.” <https://www.my4pcb.com/net35/FreeDFMNet/FreeDFMHome.aspx> (accessed Dec. 10, 2020).
- [16] “Part 15 - Radio Frequency Devices.” <http://www.arrl.org/part-15-radio-frequency-devices> (accessed Dec. 10, 2020).
- [17] “47 CFR § 15.3 - Definitions,” *LII / Legal Information Institute*. <https://www.law.cornell.edu/cfr/text/47/15.3> (accessed Dec. 10, 2020).
- [18] “myRIO-1900 User Guide and Specifications - National Instruments,” p. 32.
- [19] “SPI Tutorial – Serial Peripheral Interface Bus Protocol Basics.” <https://www.corelis.com/education/tutorials/spi-tutorial/> (accessed Dec. 10, 2020).
- [20] “Inter-IC Sound Bus (I2S),” no. 001, p. 19.
- [21] “LabVIEW Advanced Signal Processing Toolkit - National Instruments.” <https://sine.ni.com/nips/cds/view/p/nid/209055> (accessed Dec. 04, 2020).
- [22] “What Is the LabVIEW FPGA Module.” <https://www.ni.com/en-us/shop/electronic-test-instrumentation/add-ons-for-electronic-test-and-instrumentation/what-is-labview-fpga-module.html> (accessed Dec. 04, 2020).
- [23] “LabVIEW myRIO Toolkit Download.” <https://www.ni.com/en-us/support/downloads/software-products/download.labview-myrio-toolkit.html> (accessed Dec. 04, 2020).

- [24] “AutoCAD for Mac & Windows | 2D/3D CAD Software | Autodesk.” <https://www.autodesk.com/products/autocad/overview?support=ADVANCED> (accessed Dec. 01, 2020).
- [25] “What Is VirtualBench?” <https://www.ni.com/en-us/shop/electronic-test-instrumentation/virtualbench/what-is-virtualbench.html> (accessed Dec. 04, 2020).
- [26] “Noise-Induced Hearing Loss,” *NIDCD*, Aug. 18, 2015. <https://www.nidcd.nih.gov/health/noise-induced-hearing-loss> (accessed Sep. 08, 2020).
- [27] T. R. Kuphaldt and J. Haughery, “ELECTRICAL SAFETY,” in *Applied Industrial Electricity*, Iowa State University Digital Press, 2020.
- [28] A. J. Mack and AU, “United States Patent: 7189914 - Automated music harmonizer,” 7189914, Mar. 13, 2007.
- [29] D. K. Hilderman, CA, and J. Devecka, “United States Patent: 9626946 - Vocal processing with accompaniment music input,” 9626946, Apr. 18, 2017.
- [30] G. A. Rutledge, CA, W. N. Campbell, CA, P. R. Lupini, and CA, “United States Patent: 8168877 - Musical harmony generation from polyphonic audio signals,” 8168877, May 01, 2012.
- [31] “tlv320aic23b.pdf.” Accessed: Dec. 07, 2020. [Online]. Available: <https://www.ti.com/lit/ds/symlink/tlv320aic23b.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607364933278>.
- [32] “35rasmt2bhntrx_cd.pdf.” Accessed: Dec. 10, 2020. [Online]. Available: http://www.switchcraft.com/Drawings/35rasmt2bhntrx_cd.pdf.
- [33] “ABLS-LR_Series_DS.pdf.” Accessed: Dec. 10, 2020. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/Abracon%20Corp%20PDFs/ABLS-LR_Series_DS.pdf.
- [34] “USA PCB Manufacturer & Assembly | Advanced Circuits.” <https://www.4pcb.com/> (accessed Dec. 01, 2020).
- [35] “3W Electronics.” <http://3welec.com/> (accessed Dec. 10, 2020).
- [36] “ImprovingFFTResoltuion.pdf.” Accessed: Dec. 10, 2020. [Online]. Available: <http://www.add.ece.ufl.edu/4511/references/ImprovingFFTResoltuion.pdf>.
- [37] “Vocal Types and Ranges | Music Appreciation.” https://courses.lumenlearning.com/musicappreciation_with_theory/chapter/introduction/ (accessed Dec. 10, 2020).
- [38] “How to Interpret FFT results - complex DFT, frequency bins and FFTShift,” *GaussianWaves*, Nov. 16, 2015. <https://www.gaussianwaves.com/2015/11/interpreting-fft-results-complex-dft-frequency-bins-and-fftshift/> (accessed Dec. 10, 2020).
- [39] “🎵Jens Johansson · The Phase Vocoder: A Tutorial.” <http://www.panix.com/~jens/pvoc-dolson.par> (accessed Dec. 10, 2020).
- [40] “Understanding FFT Overlap Processing Fundamentals | Tektronix.” <https://www.tek.com/document/primer/understanding-fft-overlap-processing-fundamentals-0> (accessed Dec. 10, 2020).
- [41] M. Gasior and J. L. Gonzalez, “Improving FFT Frequency Measurement Resolution by Parabolic and Gaussian Interpolation,” p. 17.
- [42] “Electronic tuner,” *Wikipedia*. Dec. 08, 2020, Accessed: Dec. 10, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Electronic_tuner&oldid=993100835.
- [43] “Office Timeline Online - Interactive Timeline & Gantt Chart Maker,” *Office Timeline Online*. <https://online.officetimeline.com/> (accessed Dec. 02, 2020).

- [44] “Protolabs | Rapid Prototyping & On-demand Production.” <https://www.protolabs.com/> (accessed Dec. 02, 2020).
- [45] S. via PeopleVine, “Custom ASIC Cost Calculator - Sigenics.” <https://www.sigenics.com/page/asic-cost-calculator> (accessed Dec. 02, 2020).

Appendix

Appendix A: PCB Bill of Materials (BOM)

Quantity	Description	RefDes	Package	Type	Obsolete	Vendor	Status	Price	Hyperlink	Manufacturer	Manufacturer Part No.	Vendor Part No.	Group Name
2	CAPACITOR, 0. C3, C13		IPC-2221A Cut Tape (CT)	-		Digikey	Active	0.41000	https://www.murata-electronics.com	Murata Electronics	RDER72E102K1M1H03A	490-9238-1-ND	Harmonic Triad
2	CAPACITOR, 4' C4, C14		IPC-2221A Cut Tape (CT)	-		Digikey	Active	0.48000	https://www.murata-electronics.com	Murata Electronics	RDE5C2A471J0P1H03B	490-8783-1-ND	Harmonic Triad
2	CAPACITOR, 0. C2, C9		IPC-2221A Cut Tape (CT)	-		Digikey	Active	0.41000	https://www.kemet.com	KEMET	C320C104M5U5TA7303	399-9873-1-ND	Harmonic Triad
2	OPAMP, LMC6 U2, U3		IPC-2221A Tube		No	Digikey	Active	5.05	https://www.ti.com	Texas Instruments	LMC6482AIN/NOPB	LMC6482AIN/NOPB-ND	Harmonic Triad
2	RESISTOR, 180 R5, R6		IPC-2221A Cut Tape (CT)	-		Digikey	Active	0.10000	https://www.stackpole.com	Stackpole Electronics	RNMF14FTC180K	S180KCACT-ND	Harmonic Triad
1	CAPACITOR, 0. C8		IPC-2221A Cut Tape (CT)	-		Digikey	Active	0.41000	https://www.kemet.com	KEMET	C320C474M5U5TA7301	399-13979-1-ND	Harmonic Triad
3	AudioJacks, Sv J2, J3, J4		Ultiboard\ Cut Tape (CT)	No		Digikey	Active	1.19000	https://www.switchcraft.com	SwitchCraft	35RASMT2BHNTRX	SC1489-1-ND	Harmonic Triad
11	HCPGeneratec U5, U6, U7		Ultiboard\ Cut Tape (CT)	No		Digikey	Active	0.40000	http://www.keystone.com	Keystone	5012	36-5012-ND	Harmonic Triad
5	RealCAPS, 1uF C1, C7, C11		IPC-2221A Cut Tape (CT)	No		Digikey	Active	1.99000	https://www.unitedchemi-con.com	United Chemi-Con	KTD251B105M55A0T00	565-4656-1-ND	Harmonic Triad
1	myRIO, myRIOJ1		TE Conneq Cut Tape (CT)	No		Digikey	Active	5.28000	https://www.te-connectivity.com	TE Connectivity AMP	1-534206-7	A26466-ND	Harmonic Triad
1	HarmonicTriac U1		Ultiboard\ Cut Tape (CT)	No		Digikey	Active	8.26000	https://www.ti.com	Texas Instruments	TLV320AIC23BIPWR	296-15031-1-ND	Harmonic Triad
1	HarmonicTriac U4		Ultiboard\ Cut Tape (CT)	No		Digikey	Active	1.40000	https://www.abracon.com	Abracon LLC	ABLS-LR-12.000MHZ-T	535-10179-1-ND	Harmonic Triad

Figure 33: Bill of Materials (BOM) for PCB Components

The bill of materials (BOM) shown in Figure 33 was created using the Reports function in NI Multisim. It contains all the components necessary for PCB construction. For clarification as U1 and U4 are described by their footprint name in the table, U1 is the stereo audio codec and U4 is the 12 MHz crystal.

Appendix B: Codec Block Diagram and Timing Requirements

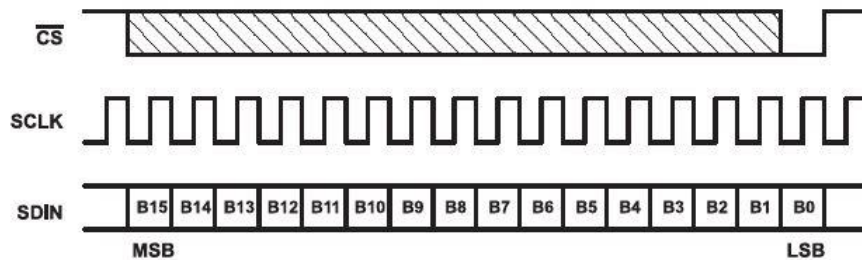


Figure 3-1. SPI Timing

Figure 34: SPI Timing Requirements for Codec

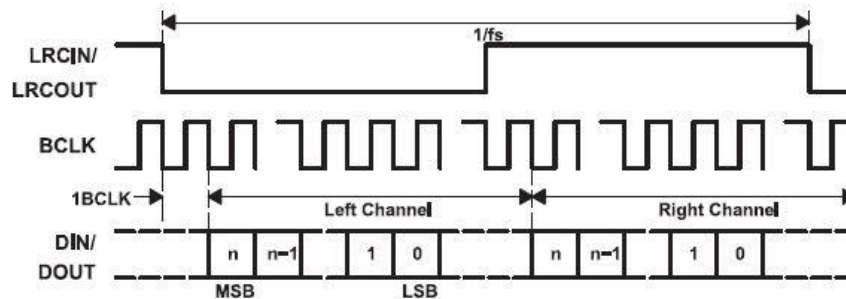


Figure 3-7. I2S Mode Timing

Figure 35: I2S Timing Requirements for Codec

The SPI and I2S timing diagrams shown in Figures 34 and 35 were used to write the SPI and I2S functionality on the myRIO [31]. With respect to the SPI, the diagram was necessary to determine clock polarity and phase. For the I2S, the graphic was used to determine when the

myRIO should “read” the bits of incoming data and whether the signal belonged to the microphone or keyboard input.

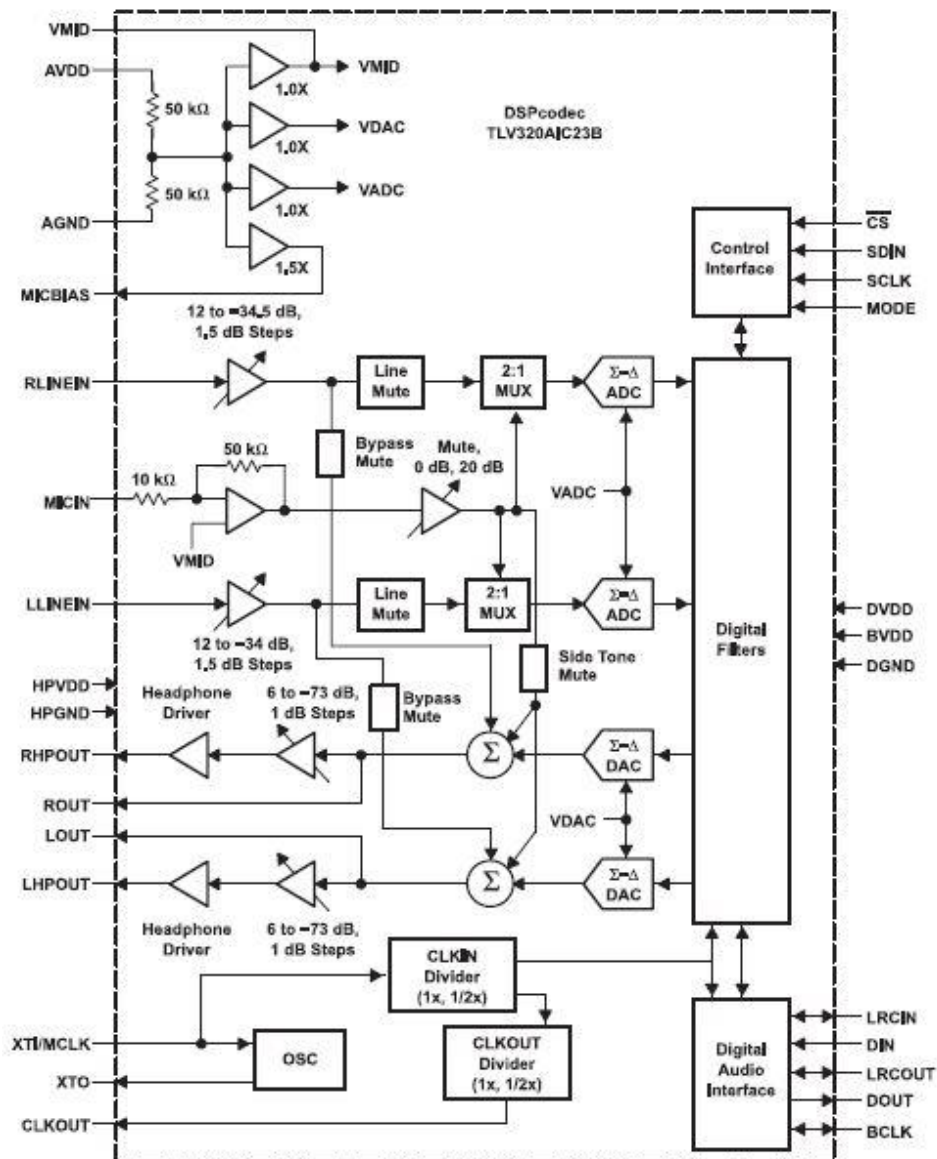


Figure 36: Codec Block Diagram

The block diagram of the codec shown in Figure 36 was taken from the datasheet for the TLV320AIC23BIPWR Stereo Audio Codec [31]. The diagram was used to determine the routing for communication, audio input, and voltage supply for the codec. The datapath information regarding the ADC select, Bypass mute, Sidetone mute, and gain placement were also necessary for deciphering the necessary SPI commands.

Appendix C: Software Data Communication Testing

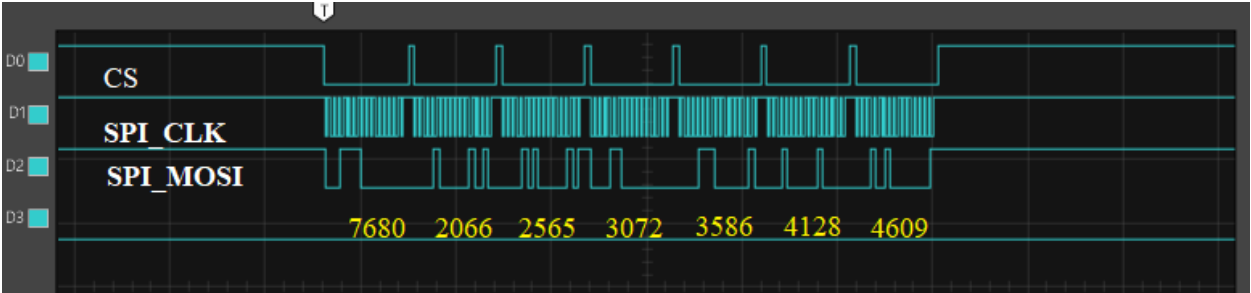


Figure 37: SPI Send Codec Configuration VirtualBench Test

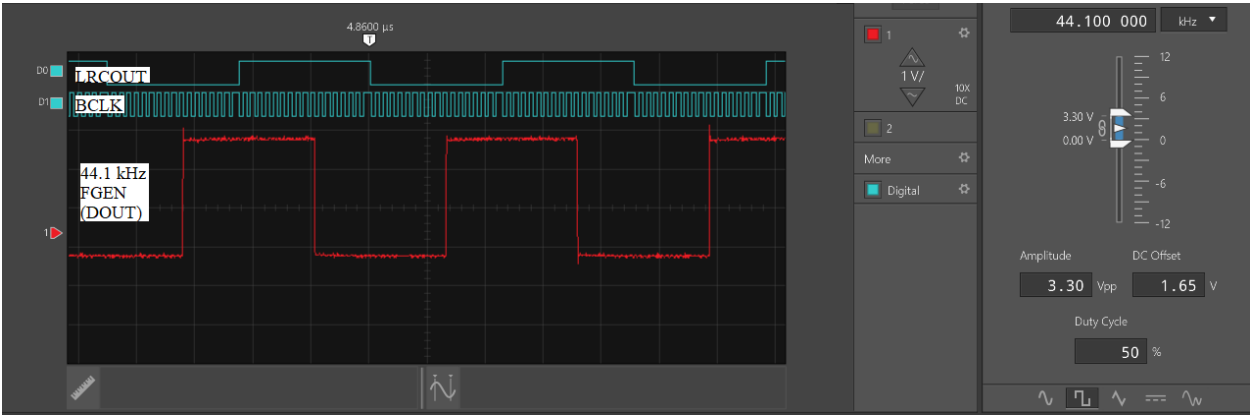


Figure 38: I2S Read 44.1 kHz DOUT on myRIO VirtualBench Test

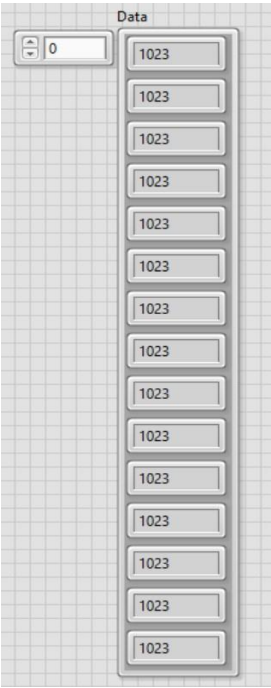


Figure 39: myRIO Read In for 44.1 kHz DOUT

The VirtualBench DIO test screenshot in Figure 37 shows the SPI communication test and timing confirmation. The burst 7 SPI messages shown is the register configuration needed to prepare the codec for the myRIO to write to I2S to be played by the amp. The I2S Read test for a 44.1 kHz square wave from the FGEN is shown in Figures 38 and 39. The VirtualBench DIO for LRCOUT, BCLK, and DOUT (44.1 kHz FGEN) are shown in Figure 38. As the frequency of LRCOUT is approximately 44.1 kHz, the data read in LabView should be the same decimal value each time as shown in Figure 39.

Appendix D: Full System Testing

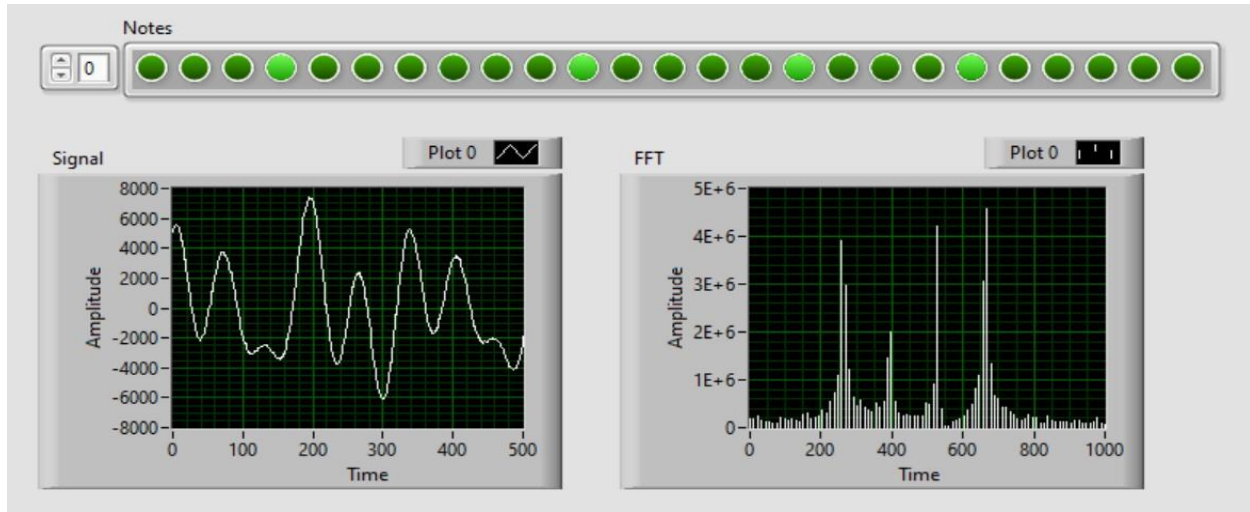


Figure 40: LabView WaveForm and FFT for Keyboard C Chord Test

The test to confirm the FFT capability for identifying the chord from the keyboard is shown in Figure 40. The Waveform of the audio signal read from the DOUT of the codec is given on the left with the corresponding FFT on the right. The notes are again confirmed using the LED strip at the top to depict a C chord.

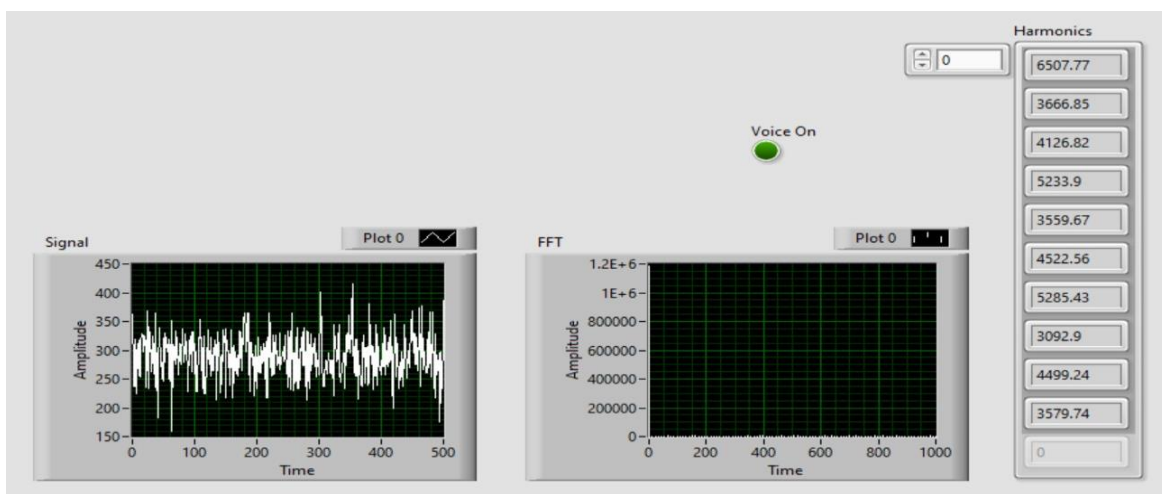


Figure 41: LabView WaveForm and FFT for No Voice Input Test

As there was static present when the microphone was connected, the FFT was run with no voice input as shown in Figure 41. The negligible FFT response confirmed that the noise should not negatively affect the functionality of the FFT for identification of fundamental frequencies and harmonics from the voice inputs.

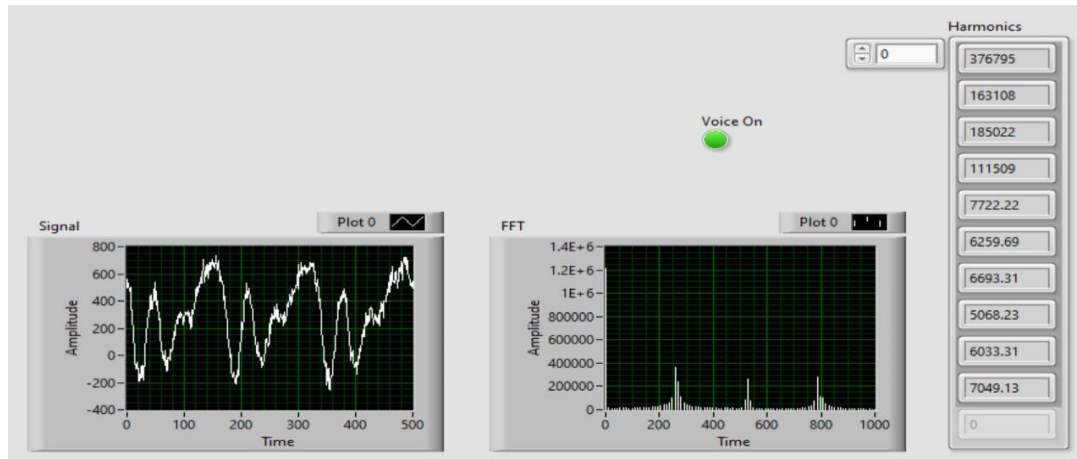


Figure 42: LabView WaveForm and FFT for Soft Ah Test

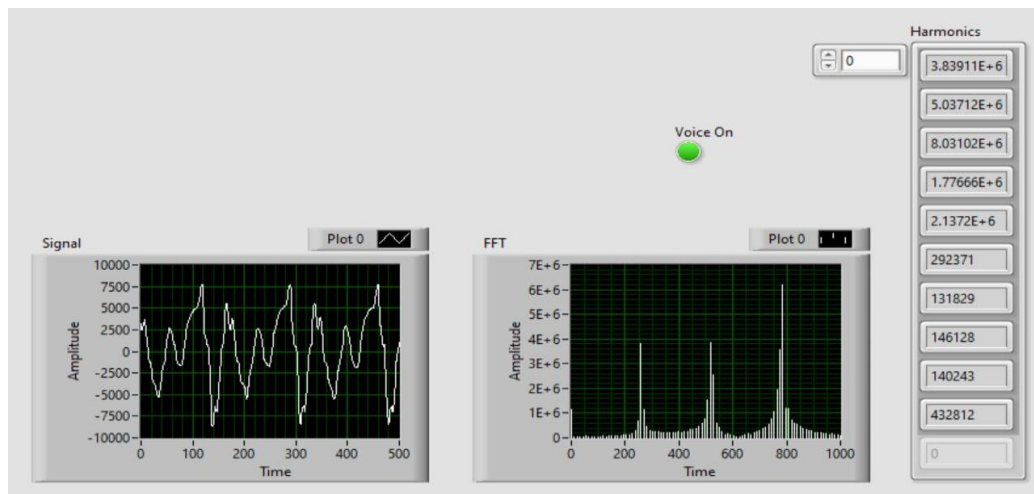


Figure 43: LabView WaveForm and FFT for Loud Ah Test

In order to show that the volume of the output changes with the volume of the voice input, Nate sang the same note softly and loudly with the same vowel sound. The amplitudes on the FFT and Waveform are ten times higher for the loud test than the soft test as shown in the Figures 42 and 43.

Appendix E: Vowel Comparisons

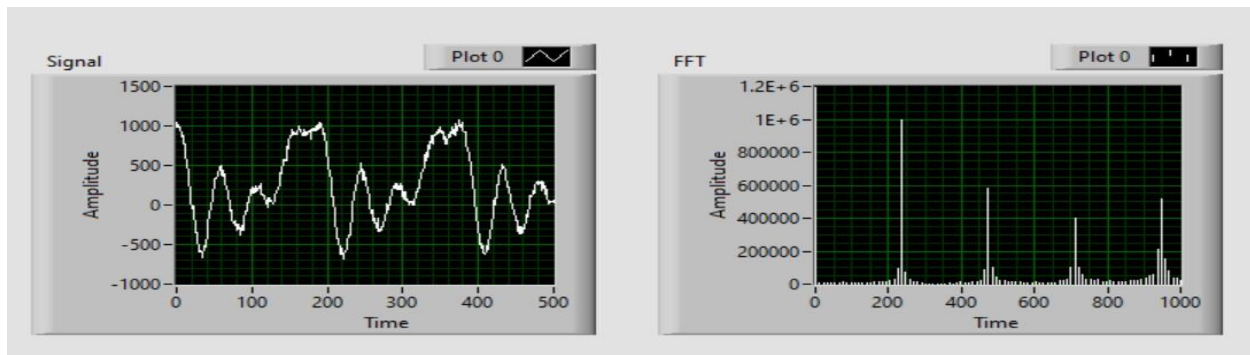


Figure 44: Mic to I2S, “Ah”

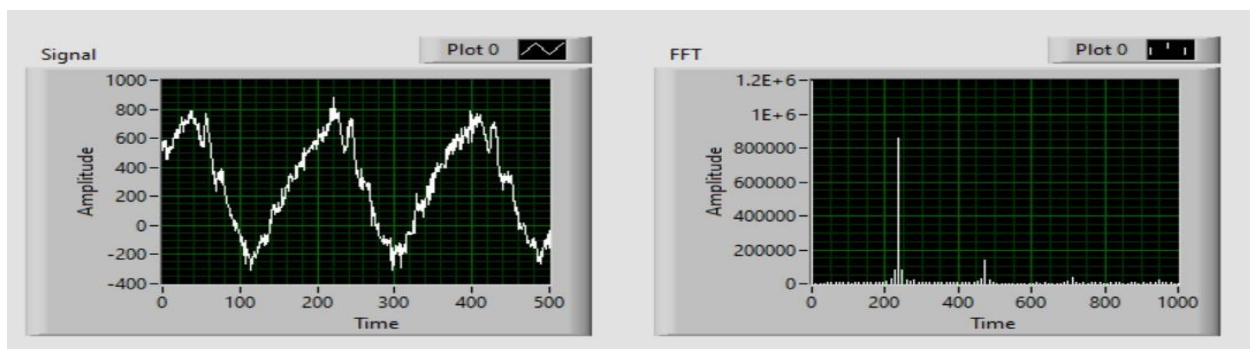


Figure 45: Mic to I2S, “Ee”

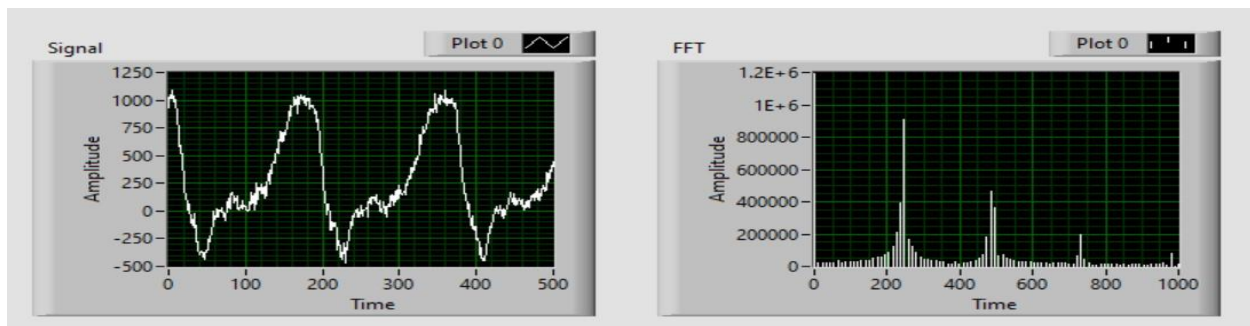


Figure 46: Mic to I2S, “Eh”

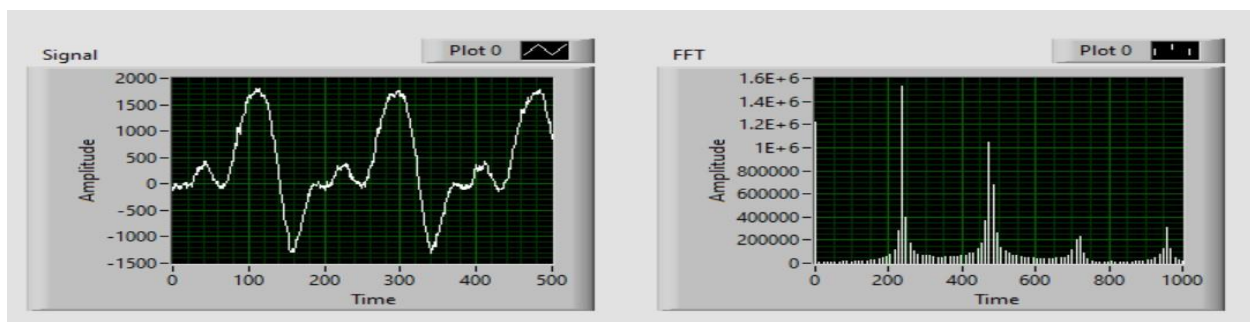


Figure 47: Mic to I2S, “Oh”

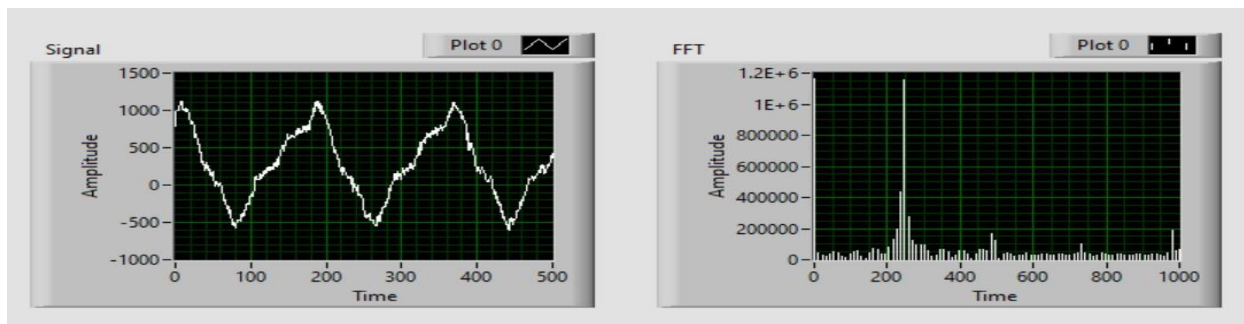


Figure 48: Mic to I2S, “Oo”

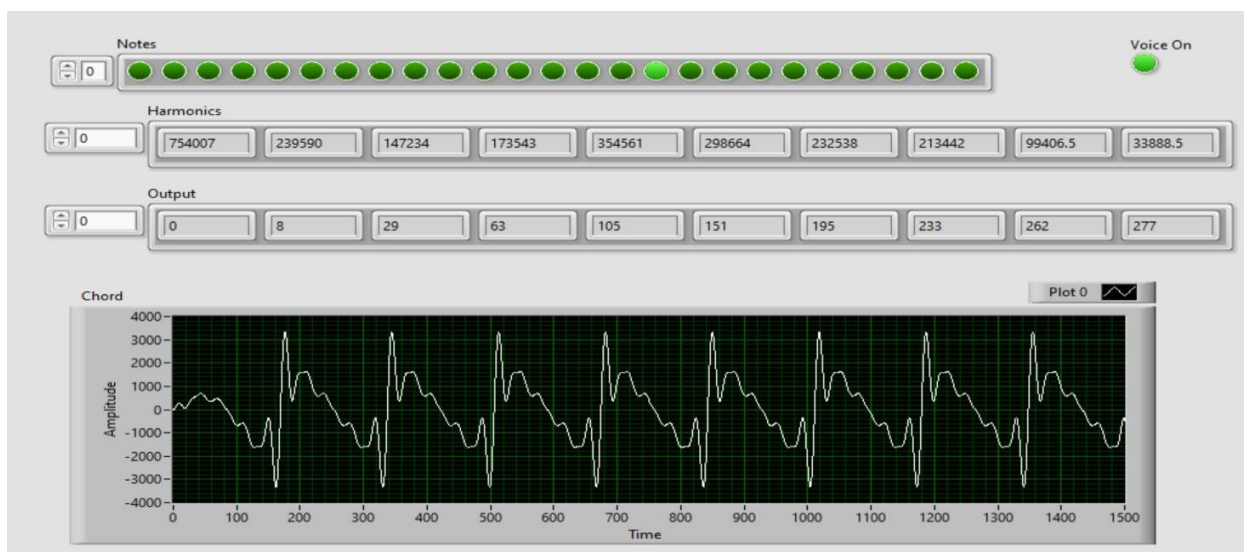


Figure 49: Mic to Chord Output, “Ah”

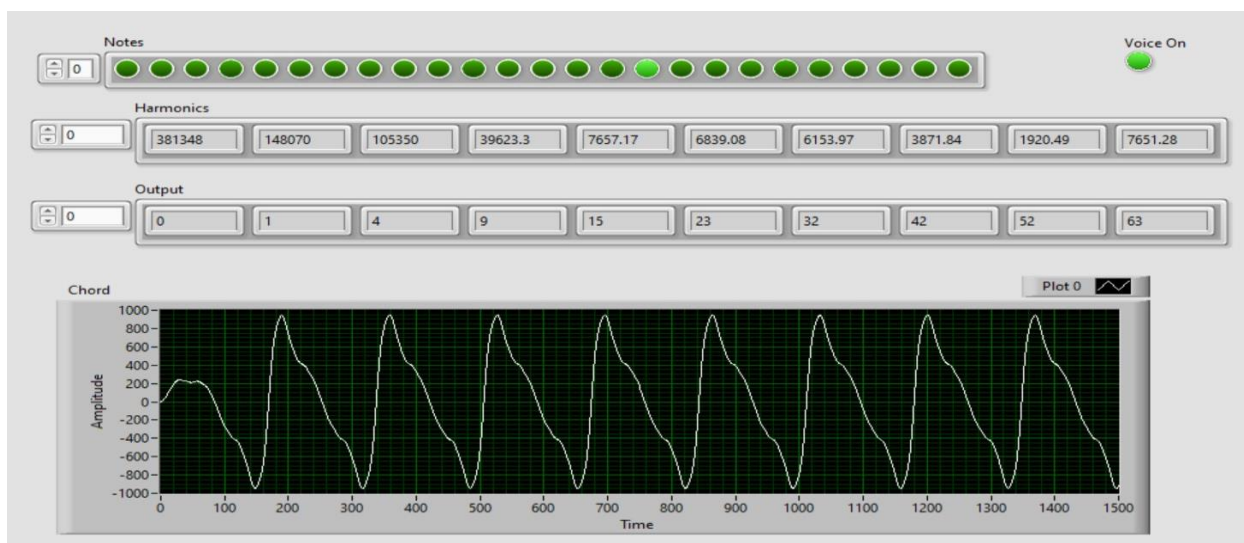


Figure 50: Mic to Chord Output, “Ee”

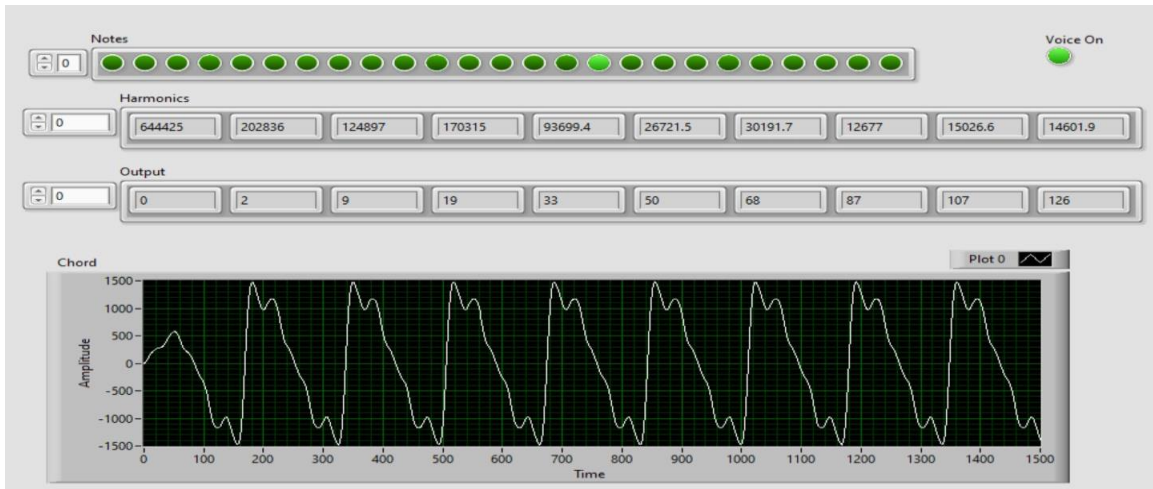


Figure 51: Mic to Chord Output, "Eh"

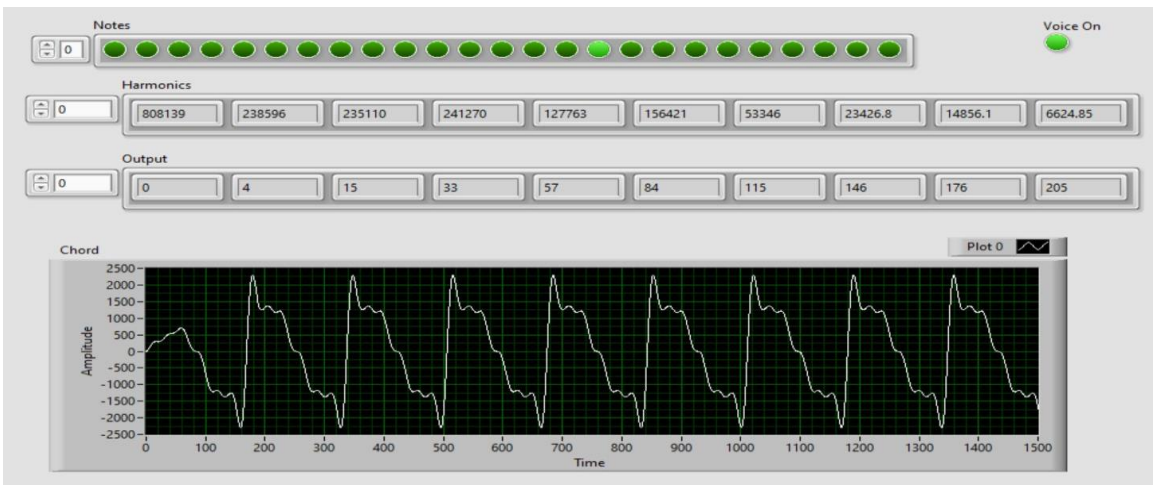


Figure 52: Mic to Chord Output, "Oh"

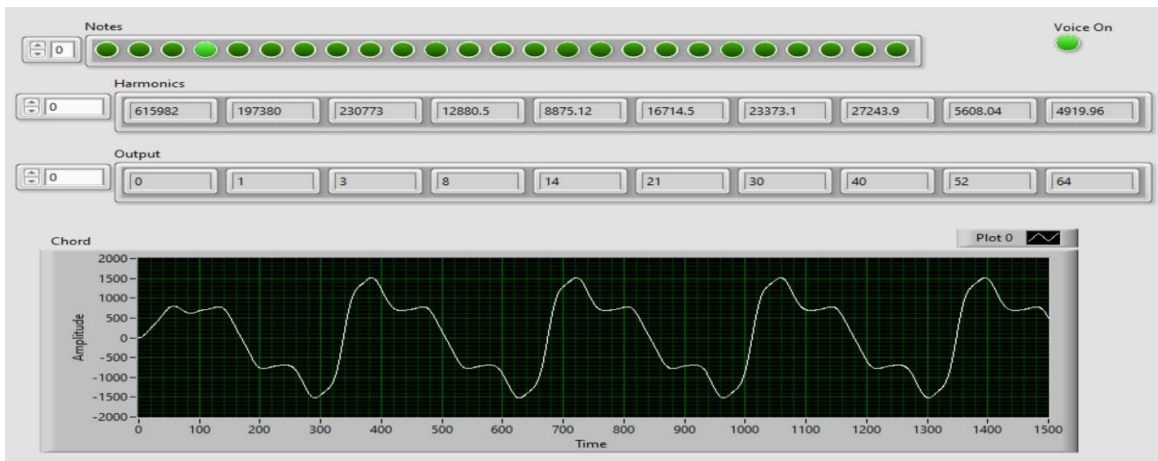


Figure 53: Mic to Chord Output, "Oo"

Figures 44-48 show the microphone signal for five vowels as it is read to the processor via I2S. Figures 49-53 show the output signal produced by the DSP algorithm given microphone inputs for the same five vowels. While there are differences, it is notable that in both cases, “ah” (perhaps the loudest vowel) clearly has the strongest higher-harmonic activity, whereas “ee” and “oo” have the weakest higher-harmonic activity.

Appendix F: Budget

Date	Quantity	Description	Cost	Budget Remaining (From \$500)
10/9/2020	1	PCB Order	\$33	\$467
10/12/2020	1	CAPACITOR, 0.47 μ F	\$0.41	\$466.59
10/12/2020	3	3.5 mm Audiojacks	\$1.19	\$463.02
10/12/2020	1	myRIO_MXP Connector	\$5.28	\$457.74
10/12/2020	1	Codec	\$8.26	\$449.48
10/12/2020	1	Capacitor, 1 μ F	\$1.99	\$447.49
10/12/2020	1	12 MHz Crystal	\$1.40	\$446.09
10/17/2020	1	1/4 In Female to 1/8 In Male Stereo Headphone Adapter	\$6.86	\$439.23
10/17/2020	2	1/4 in TS Mono to 1/8 in TRS Stereo Interconnect Cable	\$10.99	\$417.25
10/17/2020	1	Wired Handheld MIC Dynamic Singing Microphone	\$54.99	\$362.26
10/26/2020	2	13k Resistor	\$0.10	\$362.06
10/26/2020	2	LMC6482IN/NOPB	\$4.67	\$352.72
11/3/2020	1	WWW Electronics Stuff Codec	\$5.40	\$347.32
11/6/2020	1	PCB Order	\$33	\$314.32
11/9/2020	1	Codec	\$8.26	\$306.06
11/9/2020	2	LMC6482IN/NOPB	\$5.05	\$295.96
11/9/2020	3	3.5 mm Audiojacks	\$1.19	\$292.39
11/9/2020	1	12 MHz Crystal	\$1.40	\$290.99
11/9/2020	5	Capacitor 1 μ F	\$1.04	\$285.79
11/9/2020	2	180k Resistor	\$0.14	\$285.51
11/9/2020	2	Capacitor, 0.1 μ F	\$0.38	\$284.75
11/9/2020	1	Capacitor, 0.47 μ F	\$0.49	\$284.26
11/9/2020	2	Capacitor, 470 pF	\$0.52	\$283.22
11/9/2020	2	Capacitor, 1000 pF	\$0.36	\$282.50
11/9/2020	2	8.2k Resistor	\$0.19	\$282.12
11/9/2020	2	13k Resistor	\$0.26	\$281.60
11/9/2020	2	33k Resistor	\$0.10	\$281.40
11/16/2020	1	WWW Electronics Stuff Codec	\$5.40	\$276.00
11/20/2020	1	3D Printing	\$67.18	\$208.82

Figure 54: Total Project Budget Breakdown