Polyphonic Analog to MIDI Converter for Musical Applications

Andrew Obeso-Silva, Noah Watts, and Colin Smith

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — While examples of real-time analog to MIDI converters can be found on the market, all of them are monophonic. Polyphonic analog to MIDI converters tend to be computer programs that do not work in real-time. This project is a proof-of-concept for a real-time polyphonic analog to MIDI conversion device suitable for use in a live music or studio setting. This Polyphonic Analog to MIDI Converter is programmed with algorithms to detect up to six notes at a time, generate a MIDI stream, and output that stream via MIDI or USB port. It serves as a starting point for the development of a novel device in the music technology industry.**

***Index Terms* — DAW, FFT, MIDI, Monophonic, Polyphonic**

I. Introduction

Many different technologies, both ancient and modern, are used in the creation of music. The instruments that musicians use can be either acoustic or electric, and the electric instruments can be either analog or digital. In the production studio, the sounds from all these different instruments can be captured electronically for manipulation in a software application called a digital audio workstation (DAW). In music production, there are many ways to capture and incorporate analog and acoustic instruments into a DAW. Most of these solutions involve USB microphones or an audio interface that digitizes your analog signal for DAW use. Very few options exist that allow you to use your analog or acoustic instruments in the same way that you might use a digital instrument. A digital instrument using MIDI protocol will have certain parameters of the notes encoded into a signal rather than a digital replication of an analog signal. The sounds of acoustic instruments are captured with a microphone that converts the sounds into an analog electrical signal. Analog electrical instruments also output an analog electrical signal. This signal is then digitized and recorded in a computer. Digital instruments send digital signals. They can send a digital representation of an analog signal, or they can send messages that can tell another system what sounds to produce. For example, a keyboard can send a message that tells the computer that a note was played. The computer can then output the corresponding sound. Unfortunately, only these digital instruments are capable of sending messages. Other instruments, like an electric guitar or a flute, cannot do this. What this project aims to achieve is to create a device that gives acoustic and analog electrical instruments the ability to send messages just like a digital instrument in full six-voice polyphony.

The Polyphonic Analog-to-MIDI Converter for Musical Applications is a device that will accept an analog signal and convert it to a MIDI signal of up to six voices. It is designed to be used in both a music studio and live performance settings. MIDI technology is most often purely digital, with some digital MIDI controllers interfacing with a program or instrument that operates on the MIDI standard. While some products exist which convert an analog music signal into a digital MIDI stream, they are usually monophonic (single-voiced). We aim to develop an analog-to-MIDI device that is capable of translating each string of a standard-tuning guitar simultaneously. While the main goal is for the device to work with a guitar, it will be designed to work with any instrument capable of producing a consistent tone in the 12-tone equal-temperament tuning system.

II. Goals

The primary purpose of this device is to detect more than one note being played by an instrument at the same time in an analog audio signal and transmit that information through a MIDI stream. The device accepts analog input from a ¼" tip-sleeve mono audio jack. Most electric instruments typically transmit analog audio signals through these connections. For example, electric guitars, basses, and synthesizers use ¼" cables to carry their signal. XLR was considered for this device, as having an XLR input would allow for a microphone to be used as an input transducer for the device. There are other interfaces that could be implemented into the device, like USB, coaxial, and optical. However, the interfaces we have selected are the most common in music production applications, and these analog interfaces are simpler to implement than those three digital interfaces. It has a switch to select the source of the input. The device can output a MIDI stream through a MIDI port. This is the most useful interface through which MIDI streams can be transmitted, as it allows direct interfacing with MIDI-controllable instruments and connection to a DAW

on a computer via MIDI to USB cabling or a MIDI-compatible audio interface. The device will also have another ¼-inch tip-sleeve jack port to output an exact copy of the input. The purpose of this is to allow the signal to be passed through the device to other systems. This passthrough makes integration of this device into a system of other audio devices more simply and with less conflicts. We will need to buy ports for these interfaces and implement electrical circuitry that transmits the signals to and from these ports.

The device must be capable of digitizing audio signals within the range of frequencies that MIDI establishes as notes for processing. To prepare the analog signal for the analog-to-digital converter (ADC), we will have a preamp stage that amplifies the line-in signal to a range that is accepted by the ADC. The user can adjust the gain for this preamp using a potentiometer on the device in order to improve the accuracy of the device for whatever input source they are using and to prevent clipping. This preamp stage will also include a bandpass anti-aliasing filter to make the signal easier to work with once it is digitized. An ADC is then used to collect samples of the signal, but its sample rate must be high enough to digitize the signal accurately at the highest frequencies. The device must analyze a chunk of the audio signal and determine which notes are being played, if any. Musical notes are related to different sound frequencies, so the device must calculate the frequency content of the chunk of signal. A discrete Fourier transform (DFT) is capable of converting a chunk of a digital audio signal into a spectrum of frequencies that make up the sound. The frequency spectrum lists the magnitudes of each multiple of a fundamental frequency, thus also giving information on the loudness of each note the device hears. However, the issue with this is that the spectrum has a linear frequency scale, while musical notes have a logarithmic frequency scale. The device needs enough frequency density in the spectrum to be able to accurately relate frequencies to notes. We will use a processor that is powerful enough to perform the DFT in real time on chunks of the audio signal. Also, there may be several notes being played at the same time, so the device must be capable of detecting more than one note. This is called polyphony. Other analog to MIDI products exist, but they are exclusively monophonic, they can only convert one note at a time. The novel feature of our device will be its ability to convert multiple notes at once.

III. Research

In our market research, we found some products currently that exist that are similar to the PAMC. There are two current products made by Sonuus that deal with audio to MIDI conversion, the G2M V3 and the i2M. The G2M V3 is a great tool for translating guitar and bass sounds to MIDI with a MIDI output port, ¼" passthrough port, and a built-in tuner. The main limitation of the G2M V3 is that it is monophonic only. The i2M musicport is a similar product created by Sonuus that has many of the same features as the G2M but with some changes in implementation and hardware. Like the G2M, the i2M musicport is a plug-and-play device that takes musical audio from guitar, bass, voice or wind instrument and converts it to MIDI. The main difference of the i2M musicport from the G2M is that it uses a USB interface for power and MIDI output. Again, the i2M is only capable of monophonic translation.

We also researched various technologies to begin working on our design, including filter design [1]. Comparing the gentle cutoff and relatively linear response of a Bessel filter to the sharper cutoff and flatter response of the Butterworth Filter led us to determine that the Butterworth Active Band-Pass filter was the correct choice for our device. We want as much attenuation as we can on rejected frequencies with as little impact on the passed frequencies as possible.

A preamp section is also needed to ensure that the input signal is at an appropriate level for the ADC to capture. We had to research different op-amp circuit types to use in our design. With several op-amp circuits, we can do filtering, amplification, buffering, and signal splitting. Op-amp buffers allow us to isolate the input and output so the load doesn't affect the input and also gives us more ideal impedances to our circuit. This should be very useful for splitting and mixing signals. The implementation of this signal splitting is called a distribution amplifier. This means it recreates multiples of the same signal [2].

We knew we would have to use an ADC in our design, so we had to determine what parameters of our signal would affect the performance of the ADC. There are two main factors that contribute to the accuracy of the analog to digital conversion which are bit rate and sampling rate. Increasing the bit rate of an ADC improves the precision of the approximations in its digital output [3]. The sample rate is important due to the Nyquist sampling theorem which states that the sampling rate of the ADC must be at least twice as fast as the highest frequency component of the waveform being sampled. If the sampling rate is not at least equal to twice the highest frequency component, there can be inaccuracies in the sampling due to aliases. The minimum sampling rate that we will need to use for the analog to digital conversion for our device will depend on the frequency range of possible musical instruments that will utilize our device and the maximum musical frequency. The maximum note frequency used in MIDI is

12543.854 Hz. We want our device to be able to be used with multiple musical instruments including guitar, piano, voice and more. These instruments have varying frequency ranges. Table I shows the frequency ranges of some musical instruments and MIDI notes.

TABLE I
Frequency Ranges of Various Instruments

Musical Instrument	Minimum Frequency (Hz)	Maximum Frequency (Hz)
Guitar	82.41	1318.51
Piano	27.5	4186
Bass	41	262
Human voice	87	1047
Flute	262	1976
Trumpet	165	988
Clarinet	165	1568
MIDI	8.1758	12543.854

All of the common musical instruments listed in the chart have frequencies that are below the maximum frequency of the MIDI note range. This means that the ADC part of our device will need to be able to sample at least 25087.708 Hz which is double the max MIDI note frequency. If there are musical instruments that play notes at a higher frequency than the highest MIDI note frequency it would be irrelevant for the uses of our device since MIDI would not have a note available to play in that frequency.

IV. Technologies and Standards Used

One of the key ideas kept in mind during the design of the PAMC device was compatibility. The PAMC should be compatible with as many instruments and devices as possible, so it is important for it to make use of common technologies and standards to achieve this.

First, in order to be compatible with the greatest number of electric instruments, we decided to implement a ¼" audio input jack. This allows practically any electric instrument to be used as an input, with compatible electric instruments including the electric guitar, analog and digital synthesizers, and even acoustic instruments that are outfitted with electric pickups like the acoustic guitar, mandolin, and even violin. The ¼" jack is a simple tip-sleeve connector with signal being carried in the tip and the sleeve being used for ground connection. The only commonly-used instruments that are not compatible with this input scheme are non-pitched percussion instruments like drums or shakers. This is not an issue, since products already exist that can turn acoustic drums into digital instruments, such as trigger clips and trigger pads.

Next, for ease of use and compatibility with existing technologies, MIDI output has been implemented. The MIDI output can be sent out to an audio interface or MIDI to USB cable and work with a DAW through that or interface directly with MIDI-compatible synthesizer instruments.

Finally, as a way of being useful in the greatest number of situations a passthrough was implemented which buffers the input signal and sends it straight out of the device. This way the user will not lose their original sound and can use the PAMC device in combination with, rather than instead of, any other devices they may be sending their signal to such as a mixer or amplifier. Additionally, to be more useful to performers in a live setting, a center-negative barrel jack is used to deliver 9V power to the device. This is the most common power standard in music technology for live performance, so most guitarists or other performers with electric instruments are likely to already have compatible power supplies.

Fig. 1 The Critter and Guitari Septavox is an example of an instrument with MIDI in and out ports that can control other MIDI instruments or be controlled by a device such as the PAMC.

V. Hardware Overview

In the design of the hardware of the PAMC, the MSP430FR5992 was chosen to be the MCU due to its specialized fast fourier transform (FFT) hardware that would allow the device to process the input signal fast enough to meet our latency standards and its easily programmable nature. To make the signal easier to process, a preamplifier section was placed before the MSP430's on-board analog to digital converter to filter out unwanted data, bring the signal to line level. The output ports are the ¼" passthrough port and the MIDI port. The USB port requires a USB controller IC to be used. All of this is powered by 5 volt and 3.3 volt regulators designed on TI Webench.

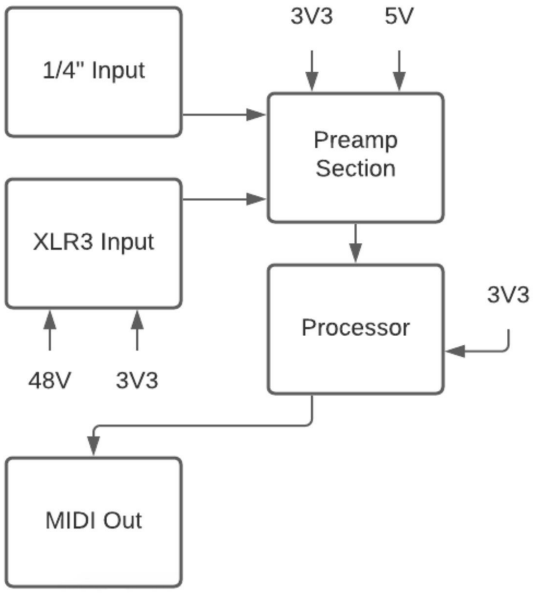

Fig. 2 Simplified Hardware Block Diagram

A. Preamp Section

The preamp section serves the dual purpose of preparing the input signal for the ADC and buffering the signal for the passthrough. Four dual op-amp packages are used, with seven of the op-amps used in the preamp design and one op-amp left unused. The first stage of the preamp is used to bring the signal up to line level with a non-inverting amplifier. The user can adjust the gain of this amplifier with a potentiometer which serves as the feedback resistor of this amplifier. The last stage of the preamp section is an active band-pass filter with a low frequency of 20Hz, a high frequency of 16kHz to capture the lowest note of the bass guitar (40Hz) and the highest note of the MIDI protocol (15kHz).

B. Power Section

TI Webench was used to create three different power conversion circuits based on the TPS563231DRLR and TPS62825DMQR chips for 9V to 5V conversion and 5V to 3V3 conversion respectively. Unfortunately, due to component failure and time constraints these power circuits were not implemented on the final demo board of the PAMC device. As an emergency solution, an external power solution is being used to provide the necessary 5V and 3V3 power to the board.

C. MCU

The MSP430FR5992 has 80 pins, only 23 of which are used. Two of these are used for the ADC, with one pin to bias the signal to 1.2V and the other to read the output of the preamp section. Another two pins are used for USB communication, three for SPI communication and one for the MIDI output sent to the MIDI output circuit. The rest of the connections are used mostly for power pins.

D. MIDI Output

The MIDI output circuit is a simple NPN transistor circuit where the transistor is used to convert the logic level of the signals from the MSP430 from 3.3V to 5V to meet the MIDI protocol standards.

VI. Software overview

The software of the PAMC device consists of four main components: the input section, the fourier transform, the note detection algorithm, and the MIDI output stream.

A. Input Section

The input section involves processing the input from the analog to digital converter and storing the input as samples in memory. Since we are using the ADC built into the MSP430FR5992 chip, we did not need to write any drivers for the ADC. A driver was needed for saving the ADC data as a useful data type for the fourier transform to be applied.

B. Fourier Transform

The goal of the Fourier transform section is to convert the digital signal from the time domain to the frequency domain. This is so that we can process the frequencies and determine the notes being played. This is achieved by implementing an FFT similar to the Cooley-Turkey FFT, using bitwise operations rather than arithmetic operations

wherever possible to reduce the amount of time required for the Fourier transform. This is one of the most important software blocks to optimize in this way because we expect that the majority of the time spent converting the analog notes to MIDI will be spent performing this Fourier Transform.

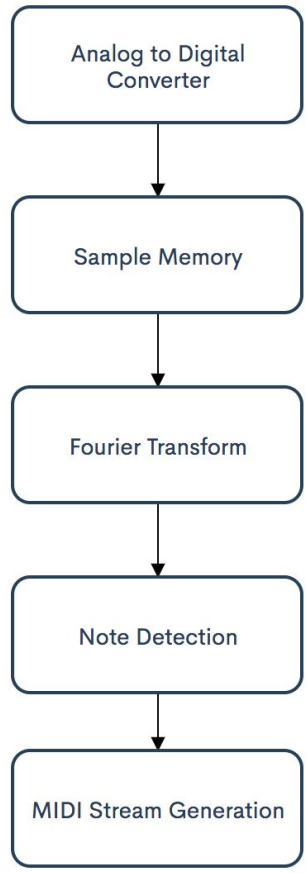

Fig. 3 Simplified Software Flow Chart

C. Note Detection

The note detection algorithm has multiple stages including spectra aggregation and magnitude clamping, a unique harmonic product spectrum generation, and final note selection and output logic.

First the spectra aggregation section simply takes the 3 frequency spectra that we obtain from our series of FFT's and aggregates them into a common, ordered frequency spectrum. This spectrum is checked to see if the magnitudes are high enough to indicate a note being played, and then all of the magnitudes are multiplied to clamp the highest value to a set magnitude. This ensures that whenever a note is being played the peak frequency magnitudes are the same every time, making the note detection process more consistent.

Next, the primary part of the algorithm is the generation of a harmonic product spectrum. Traditionally a harmonic product spectrum would divide an entire frequency spectrum by an integer multiple and multiply it by the base spectrum for the first few integer multiples such as 2, 3, 4 and 5. This makes it so that the harmonics of a fundamental frequency get multiplied into the fundamental frequency bin giving the fundamental a greatly exaggerated magnitude as compared to harmonics and noise.

In our case we go bin by bin and divide and multiply individually. While this is more difficult, it allows us greater accuracy as we can split bins that do not divide evenly into a single bin. For instance, say we have 200 Hz, 400 Hz, and 700 Hz bins. If we are dividing the 700 Hz by two and multiplying that magnitude into the respective bin, that would result in 350 Hz being multiplied into nothing since we do not have a 350 Hz bin. Instead we multiply it into both 200 Hz and 400 Hz with the 400 Hz bin being heavily favored. This works similar to how the magnitudes on an FFT would show up if your bins don't line up with the input frequencies.

Another benefit of doing every bin individually is that we can account for different harmonic differences between notes on a guitar. The harmonic content of notes on a guitar vary based on the pitch of the note and where you play the note. This means the magnitudes of the harmonic product spectrum will vary based on the pitch being played if we do not account for that. To fix that we analyzed the harmonic content of the notes on a guitar and scaled the harmonic response accordingly when dividing and multiplying harmonics in the harmonic product spectrum. While this response is specifically tailored to suit guitar, a finalized product could generate a response for every instrument and input response by having the user play every note and analyzing the harmonic content.

Finally, after the harmonic product spectrum is generated there must be determined what notes are being played. To do this we first check for frequencies above a magnitude threshold and map those frequencies to their closest MIDI note values. When two consecutive bins are above the note threshold, we select the largest. This happens when a note is being played in which the frequency is between two bins. We then take these detected notes and compare them to the current output array. We then remove the notes that are no longer being detected and add the unique notes from the most recent loop. This keeps an up to date array of up to 6 notes being played which is used as the input for our output drivers.

D. MIDI Stream Generation

The MIDI stream generator collects the data gathered by the note detection algorithm and encodes it into MIDI protocol. MIDI is a very simple interface to implement with just a digital output pin and some passive and active components. MIDI uses asynchronous data transmission and transmits as bytes at 31.25 kilobits per second. The interface uses a start bit, 8 data bits and a stop bit. This means for each serial byte there are a total of 10 bits that are sent for a period of 320 microseconds. MIDI has this low data transfer rate because it usually only needs to do basic instructions of which MIDI notes to play and changing its timing, velocity and etc. The primary MIDI parameter that we are concerned with is note pitch, which is encoded as a single byte. Our pitch bytes are between the values of 0x28 for our lowest note, E2, and 0x58 for our highest note, E8. Sending this pitch byte along with a byte to set the velocity of the note will be enough information for a MIDI device to play a note. Velocity is a measure of the force and presence with which a note is played. We have set the velocity of all of our notes to be 0x7F.

A UART channel on the MSP430 is used to set the baud rate of the MIDI stream. The system stores the state of finite state machines for each note. Each state machine has 2 states, where 0 is off and 1 is on. The state machine for a note is set to 0 by default or if the note is not in the current output frame and is set to 1 if the note is in the output of the current frame. This provides a three frame buffer between when a note is no longer detected and when the state machine is set to 0, ending the transmission of that note. This buffer is important to reduce stuttering where a note will cut in and out due to the note mistakenly not being detected in some frame. On a transition from off to on, the system will append a Note On message for that note with the velocity parameter = 0x7F. On a transition from on to off, the system will append a Note On message for that note with the velocity parameter = 0x00

VII. Results

The three criteria that we have decided to judge the PAMC on are the accuracy of its note detection, the latency or delay from the time a note enters the PAMC system to when it leaves as a MIDI signal, and the overall functionality of the device including its versatility and ability to stand on its own.

A. Note Detection Accuracy

In order to test the PAMC device's note accuracy, we recorded its response when a single note is played, a triad is played, or an open position chord. The target for this prototype is a note accuracy of 80%. Ideally, we would look for accuracy closer to 100%, but as proof of concept 80% will suffice.

Four different tests were performed. The first test is a test of the PAMC device's ability to convert single notes by playing open strings on the guitar without pressing down on the fingerboard. The second is another single note test, but this time the notes tested are fretted notes. The third test is a polyphonic note detection test where three different triads (chords composed of three notes) are played. The final note detection test is also a polyphonic note detection test in which open position chords of four to six notes are played. In these tests, accuracy of 85%, 82%, 48%, and 53% was observed, respectively.

For the purpose of this test, accuracy is determined by whether or not the played note or notes appeared in the MIDI stream. Some of the notes received have a stuttering effect where the note starts and stops several times despite being played on the guitar only once. This stuttering is most common with notes of lower pitch on the A string or low E string. These notes are still counted because the correct note was detected. There are also cases where the right note and a wrong note are present at once. It is not uncommon for the correct note to be sent along with a note an octave above (twice the frequency) or of some other harmonic. The octave is the most common wrong note because it is the first harmonic in the harmonic series and on guitar it is often the harmonic with the greatest magnitude. In cases where the correct note and some wrong note are both played, it is counted as a correct note being played since the correct note was detected.

TABLE II
Results of Single Note, Open String Testing

Note	Fraction Correct	Percent Correct
E2	19/20	95%
A2	18/20	90%
D3	16/20	80%
G3	12/20	60%
B3	17/20	85%
E4	20/20	100%
Total	102/120	85%

When a single note is played, the PAMC will detect the correct note 83% of the time. When a chord is played, the PAMC will detect 51% of the correct notes. This data highlights the difficulty of detecting multiple notes at once, where even a reasonably good algorithm for detecting a single note may have difficulty in detecting multiple. In addition to improving the ability of the device to detect the notes being played, improvements can also be made in reducing stuttering and reducing the detection of octaves and other harmonics.

TABLE III
Results of Multiple Notes, Triad Testing

Notes	Fraction Correct	Percent Correct
C, E, G	15/30	50%
F, A, C	16/30	53%
A, C, E	12/30	40%
Total	43/90	48%

B. Latency

The human brain has great difficulty in discerning any delay of less than 10 milliseconds. Musicians refer to any audible delay of approximately 30 milliseconds or less as a "slapback" delay, where it sounds as if two instruments are being played at once. Any delay greater than 30 milliseconds is very noticeable and can impede performance if the sound of an instrument is delayed by this much. For this prototype of the PAMC device, the latency should be under 100 milliseconds. While this seemingly generous amount of time can allow for long sounding delays, it is a small enough delay that it proves a low-latency version of the PAMC could exist with sufficient optimization.

Initial testing of the PAMC device indicated that delay was likely in the target range of less than 100 milliseconds. When played alongside a 30 millisecond slapback delay, the PAMC output seemed to be twice as delayed as the slapback, meaning it would be delayed by approximately 60 milliseconds.

In order to actually test the latency, we used the DIGILENT Analog Discovery 2 kit and Waveforms software to view the analog input of our device on one channel and the MIDI output on another. We can measure the length of time between the start of the analog input signal and the beginning of the MIDI signal. This time turned out to be 56 milliseconds which is just barely below our estimation. This delay makes the current version of the device unsuitable for most uses in a live setting, but it does suggest that it is possible to reduce the latency down to a point where it would be suitable for live performance. With more optimized software, a more powerful processor, and a faster clock rate the latency could very likely be reduced below 30 milliseconds and perhaps could even reach the 10 millisecond mark where the delay would become imperceptible.

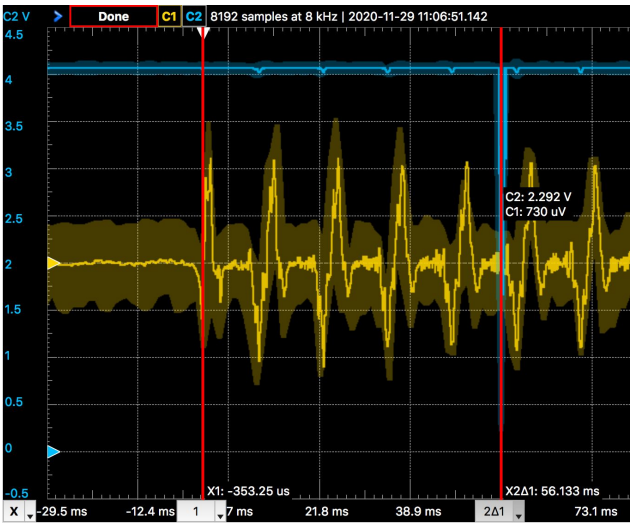

Fig. 4 Oscilloscope output for latency test. Analog input (yellow) is followed by digital MIDI output (blue) after a 56 millisecond delay.

C. Overall Functionality

This is a subjective criteria where the PAMC device is judged based on the completeness of its feature set and the variety of its potential applications.

This current version of the PAMC device has a very limited feature set, with only base functionality supported by the preamp section and MIDI output. Additional features were discussed, such as USB connectivity, XLR input to support microphones, XLR passthrough output to send the passthrough signal to line-in inputs, and 48V phantom power delivery to condenser microphones. All of these features were put on hold to focus on getting the core functions of the PAMC device running as smoothly as possible. If possible, future iterations of the PAMC device should include these features in order to make it more versatile and appealing. As a result of the reduced feature set of the current version, there are fewer uses for it. In fact, due to a malfunction of the power section, the current version does not even include a 5V or 3.3V power source. 5V and 3.3V power are being delivered from an

external source. In order to make a more useful product with a more robust feature set, XLR input with optional 48V phantom power and XLR passthrough output should be implemented to increase the pool of possible input devices and USB should be implemented to eliminate the need for peripheral devices to connect to a DAW. If we should choose to pursue further development of this device, these are features that we would be interested in including in a later revision.

VIII. Conclusion

Though the latency of our current version of the PAMC may be too great to use effectively in live performance settings, it is a great starting point for further development of a more usable version of this device. With greater than 20 milliseconds between the note being played and the MIDI stream being produced, there will be a noticeable delay but exploring other options in spectrum analysis of the input signal and streamlining the filtering and note detection algorithms could lead to a PAMC with a much less noticeable delay. The accuracy of the note detection algorithm is a strong point that is not to be overlooked, as note detection is likely the main reason there are so few existing polyphonic analog to MIDI products on the market currently.

Many planned features of the PAMC device had to be cut in order to deliver the finished prototype on time, including USB functionality and XLR cable I/O jacks to support microphone input. In the end we determined what features were essential to our design and to meet the requirements we laid out and cut the features that we could not afford to spend the time implementing.

Across the last two semesters of our time at the University of Central Florida, the three of us worked together and supported each other as a team to solve each problem presented in our ambitious design. Though we may not have created a product that is ready for market in its current state, we have gained valuable knowledge and experience in research, design, and collaboration in engineering.

Acknowledgement

The authors wish to acknowledge Sean Dillon for his contributions to this project.

References

[1] IEEE GlobalSpec, “Active Band Pass Filters Information”, 2020. URL: https://www.globalspec.com/learnmore/semiconductors/analog_mixed_signals/amplifier_linear_devices/active_bandpass_filters

[2] Larry Davis, “Distribution Amplifier”, 7 Mar 2012. URL: http://www.interfacebus.com/distribution-amplifier.html

Biographies

Andrew Obeso-Silva is a graduating Computer Engineering student at the University of Central Florida. He plans to work in digital design and computer architecture after graduating.

Noah Watts is a graduating Computer Engineering student at the University of Central Florida. He plans to pursue a career in hardware design and computer architecture.

Colin Smith is a graduating Electrical Engineering student at the University of Central Florida. He spent a year working as a test engineering intern and is beginning a career in test engineering.