Design and Implementation of a Digital Guitar Effects Pedal Using a STM32H753ZI Microcontroller

Anas El Fali
dept. of Electrical and Computer Engineering (ECE)
Concordia University
Montreal, Canada
A.Elfali@live.concordia.ca

Nikolas Darlas
dept. of Electrical and Computer Engineering (ECE)
Concordia University
Montreal, Canada
Nikolas.Darlas@live.concordia.ca

***Abstract*—The design and implementation of a digital guitar effects pedal utilizing the STM32H753ZI microcontroller offers new opportunities for enhancing sound modulation in live performances. This system processes an analog input signal from an electric guitar, applying various digital effects to produce a refined output, played through an amplifier. Key features include distortion, echo, flanger, and delay effects, along with a looper function for real-time recording and playback. The system can also modulate other sounds that will be loaded through memory, such as human or animal sounds. The use of a high-speed DSP processor of the H7 series from ST allows for a powerful and efficient system as a result of its specialized DSP peripherals. The system is designed for ease of use and affordability, offering a versatile and powerful tool for musicians. A comprehensive evaluation of the pedal's performance, including comparisons between products from published papers, in terms of hardware and feature differences, is performed in order to demonstrate the reliability effectiveness of the product in a live music setting. The resulting waveforms are recorded and compared with a set of simulated waves to validate the implemented design. Finally, the system is also assessed to determine future work that can be done and the ability to branch the product to other related fields such as voicecoding.**

***Index Terms*—Digital Signal Processing DSP, Guitar Effects Pedal, Microcontroller, Real-time Audio Processing**

I. Introduction

From jazz and pop to rock and blues, the electric guitar has been a mainstay in many musical genres. Guitar effects pedals are essential in creating the distinctive sounds that are associated with each of these genres. Musicians can improve their performances with the help of traditional guitar effects pedals, which are mostly based on analog circuitry and provide sound modulation choices including distortion, echo, and vibrato [1], [2]. However, the development of digital guitar effects pedals has been fueled by the limits of analog designs in terms of versatility, compactness, and multi-functionality. Digital effects processing has emerged as a high-quality, adaptable, and portable substitute because to developments in microcontroller technology, enabling the integration of intricate effects into a single device and real-time audio signal modification [3].

In order to accomplish high-fidelity audio effects processing, this project intends to build and implement a digital guitar effects pedal using the STM32H753ZI microcontroller, utilizing its robust DSP capabilities, integrated Floating Point Unit (FPU), and wide peripheral support [3]. In addition to a looper feature that allows players to record and mix sounds in real time, the pedal incorporates a number of well-known effects, such as distortion, echo, and vibrato. It is appropriate for live performances because of extra capabilities like tap tempo synchronization, which provide musicians exact control over effect timing.

The STM32H753ZI allows for an effective, small design without sacrificing performance or sound quality, in contrast to earlier digital pedals that sometimes relied on many parts or additional processors [4]. In order to increase the pedal's usefulness, we also look at possible additions and other features. With this project, we provide a high-performance, multipurpose, and reasonably priced tool to the guitar effects market that can be used by both enthusiasts and professional musicians.

In the following sections, this paper details the Related Work in digital guitar effects processing to provide context for this project. Next, the System Architecture and Design section explains the hardware and software components that form the core of our digital guitar effects pedal. The Implementation section describes the digital signal processing algorithms and additional functionality integrated into the system. Experimental Setup and Testing outlines the methodology for assessing the pedal's performance, while Results and Discussion presents an analysis of the findings and compares them with similar solutions. Finally, the Conclusion and Future Work section summarizes the project's contributions and suggests potential directions for future development.

II. System Architecture/Design

A. Hardware overview

1) MCU: The STM32 series of MCUs were chosen, due to their ease of programming using their native development environment, Keil Muvision and their implementation

on NUCLEO boards [4]. In summary, STM32H753ZI is a high-performance DSP MCU. This is characterized by a high maximum core clock and DSP specific peripherals, such as a high resolution ADC with configurable 1024x oversampler, internal comparators, digital filters for external sigma delta modulators (DFSDM) and internal OPamps [4]. Thus, within the STM family of MCUs, this was the clear choice. All the DSP peripherals to the exception of the internal comparators and the DFSDM were used, although these can have a major part in future works of this product [4]. Thus, the ability to expand on future work was a key consideration for the MCU selection. Table I below gives an overview of the peripheral device instances in use. The system block diagram in Figure 1 illustrates the signal flow for digital guitar effects processing, starting from the guitar's pickup, passing through analog conditioning, digitization via ADC, digital signal processing in the MCU, and conversion back to analog form via DAC for output to an external speaker.

TABLE I: *PERIPHERALS OVERVIEW*

Peripheral	Peripheral Instance
ADC	ADC1
GPIO	GPIOA, GPIOB, GPIOC, GPIOE
DAC	DAC1
TIM	TIM6
OPAMP	OPAMP2
DMA	DMA1

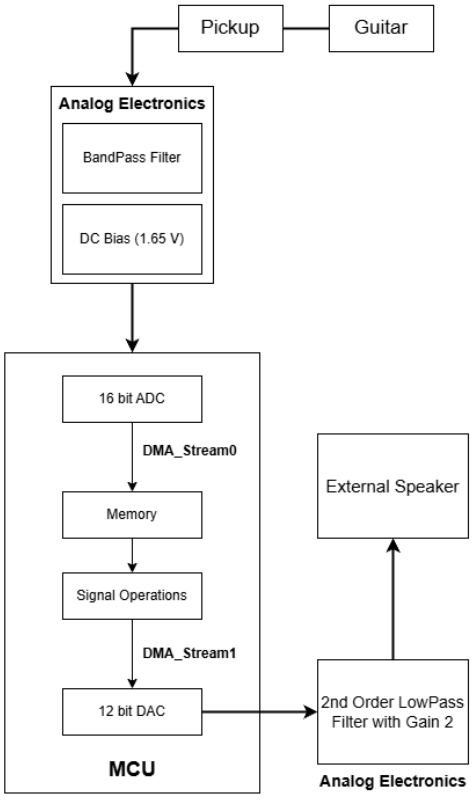


Fig. 1: System Block Diagram for Digital Guitar Effects Processing

Another popular option for embedded DSP projects are the ESP line of microcontrollers. Table II presents a detailed comparison between the STM32H753ZI and the ESP32 microcontrollers, highlighting their key features relevant to this project. The STM32H753ZI, with its 480 MHz clock speed and integrated Floating Point Unit (FPU), is optimized for high-performance digital signal processing (DSP), making it an ideal choice for real-time audio applications [4]. Its 16-bit ADC offers superior resolution compared to the 12-bit ADC in the ESP32, ensuring high-fidelity signal sampling. Additionally, the STM32H753ZI features 2 MB of internal flash memory, providing faster access and reducing latency compared to the ESP32's reliance on external flash memory [4]. Furthermore, the STM32H753ZI exhibits lower power consumption, making it more efficient for standalone applications like a digital guitar effects pedal. This comparison underscores why the STM32H753ZI is the preferred microcontroller for this project [4].

TABLE II: COMPARISON OF STM32H753ZI AND ESP32 MICROCONTROLLERS

Feature	STM32H753ZI	ESP32
Clock Speed	480 MHz	240 MHz
Floating Point Unit	Yes	No
ADC Resolution	16-bit	12-bit
Internal Flash Memory	2 MB	4 MB (external flash)
Power Consumption	Low	Moderate to High

B. Analog Electronics overview

1) Operational amplifier (OP-amp): The NE5532 OP-amp, which is a dual internally compensated OP-amp was selected. The internal compensation is crucial as these OP-amps will be used in the context of unity gain, in which non-internally compensated OP-amps require external compensation for gains less usually than 3, which is extra work. In addition, since they are dual, is means every IC has two amps, which means a band-pass Sallen-key filter can be created with just 1 of these, reducing the cost. These kinds of filters are commonly used in audio processing applications. Figure 2 shows the analog electronics of the designed system.

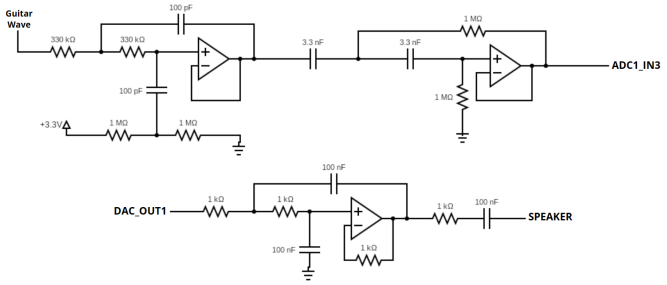


Fig. 2: Circuit Design Schematic for Analog Signal Conditioning and Output Amplification

2) High-Pass Filter: The high-pass filter design ensures that low-frequency noise and interference are effectively attenuated while preserving the desired frequency range of the guitar signal. The Bode plot Figure 3, illustrates the frequency response of the high-pass filter, showing a significant gain

increase beyond the cutoff frequency of 48.22 Hz calculated using equation 1, where $R1, R2 = 1M\Omega$ and $C1, C2 = 3.3nF$. This characteristic is critical for eliminating unwanted low-frequency hums while allowing the higher frequencies to pass through unaltered. This is because the lowest note that can be played on the guitar is 80 Hz.

$$f_c = \frac{1}{2\pi\sqrt{R_1 C_1 R_2 C_2}} \tag{1}$$

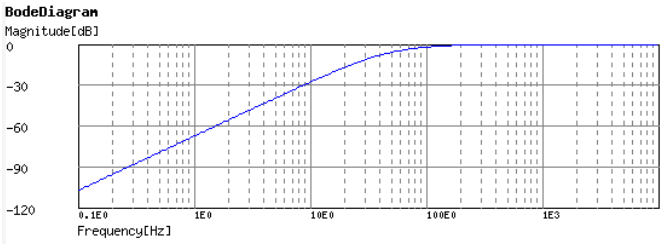


Fig. 3: Bode Diagram of the Frequency Response of the High-Pass Filter

3) Low-Pass Filter: The low-pass filter is designed to attenuate high-frequency noise introduced during signal processing, particularly after the DAC stage. The Bode plot, Figure 4, demonstrates the decreasing gain beyond the cutoff frequency of 4822.87 Hz calculated using equation 2, where $R1, R2 = 330k\Omega$ and $C1, C2 = 100pF$, ensuring that the output signal is smooth and devoid of high-frequency artifacts. This filtering is essential to maintain the integrity of the audio signal before amplification and playback.

$$f_c = \frac{1}{2\pi\sqrt{R_1 C_1 R_2 C_2}} \tag{2}$$

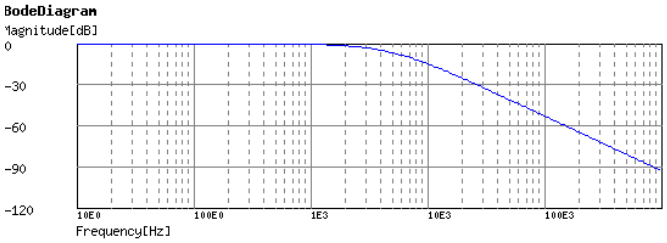


Fig. 4: Frequency Response of the Low-Pass Filter

C. Analog to Digital Converter (ADC)

Table III gives an overview of the functionality of the ADC peripheral. It operates directly from a dedicated 40 MHz clock line. However, it is internally prescaled by a factor of 2, leaving the effective ADC clock at 20 MHz. The ADC has a sampling time of 9 clock cycles for 16 bit resolution, which would imply a sample rate of 0.432 MHz, which is well within the ADC capabilities. However, we will see that using oversampling, we will increase the sample rate to well beyond the ADC capabilities. Figure 5 represents the system architecture of ADC-DAC processing for the pedal, where an analog input signal is digitized using an oversampling ADC, processed via DMA to RAM, and subsequently converted back to analog form using a DAC, with synchronization provided by a timer operating at 48 kHz.

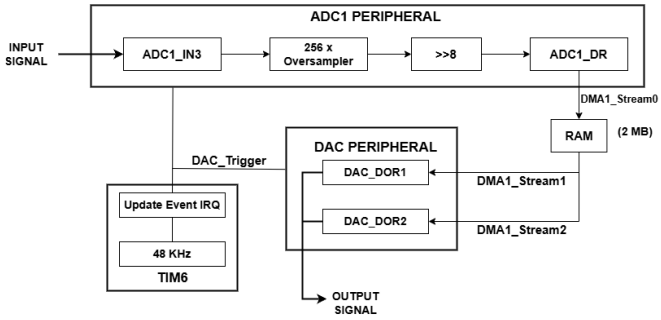


Fig. 5: System Architecture for ADC-DAC Processing

TABLE III: *ADC1 OVERVIEW*

ADC1	Overview
Clock	20 MHz
Trigger	Timer6
Trigger Detection	Rising Edge
Oversampling	256x
Oversampling Right Bit Shift	8 bits
DMA	DMA1 Stream 0
Boost Mode	Yes
Pin	GPIOA6

1) Anti-aliasing band pass filter: To satisfy the Nyquist sampling criterion, frequencies greater than half the sampling frequency need to be filtered out, or else they can wrongfully act as harmonics. Figure 5 shows the circuit that was used to generate the band pass filter that also functions as the anti-aliasing filter.

2) DC biasing: Due to the sinusoidal nature of the waves, if the ADC were to read negative values, it would wrongfully record a zero. Thus, we need to move the operating point where the lowest trough of the guitar signal would still be positive. The chosen biasing point is 1.65 V, which is half of the ADC supply voltage (3.3V). This biasing needs to be taken into account for signal operations. In figure 5, the bias is inserted in addition to the input wave as the non-inverting input to the first OP-amp. The bias is created from a voltage divider using two 1 MΩ resistors.

3) Power: The application is exclusively powered by the NUCLEO-H753ZI board, through USB. The 5V pin is used to power all the analog components, and the 3.3V pin is used to create the reference for the voltage divider for the DC bias. In practice, the supplied voltage is around 4.8V and 3.15V, which is going to reduce the accuracy of the system by a bit.

4) Oversampling: The chosen MCU contains a configurable oversampler within its ADC peripheral. The chosen oversampling ratio is 256x. Within the hardware, the oversampling data register will contain a 24 bit result. To comply with the 16 bit ADC requirement, the result is right shifted by 8 bits.

We note that a 256x oversampling will increase the clock cycles per ADC sample to $9 * 256 = 2304cc$. In theory,

this would require a ADC with $ADC_f > 100MHz$ for a 50000 kHz sampling speed. This is not feasible, as the $ADC_{f_{max}} = 50MHz$ and the effective sampling rate of the system will only be 8.6 kHz, which is well below the expected audio sample rate of 48 kHz. However, due to the effects of oversampling, it will have no perceived difference on the quality of the output. As each full sample of the ADC contains much more quality then the non-oversampled signal. If we tried to sample at 8.6 kHz without the oversampling, the audio quality would be very poor.

D. Digital to Analog Converter (DAC)

Table IV gives an overview of the DAC. As with the ADC, it is triggered by the Timer6 at 50 kHz. Although the ADC samples at a 16 bit resolution, the DAC can only output at 12 bits. In figure 5, the bottom circuit follows the from the output of the first DAC channels, which contains the modulated guitar signal. It is passed through a 2nd order low pass filter which also acts as a decimation filter. Oversampling is a technique which pushes the noise to higher frequencies, hence the need for a post-processing low pass filter to remove the noise. The output is then passed through a coupling capacitor to remove any unwanted DC bias to prevent speaker damage. We note that the second DAC channel, which contains the non-guitar effects, is not passing through any circuit, but directly connects to the speaker.

TABLE IV: *DAC1 AND DAC2 USAGE*

DAC	Usage
DAC1_channel1	Timer 6 trigger Connected to GPIOA pin 5
DAC1_channel2	Timer 6 trigger Connected to GPIOA pin 4 and OPamp2

E. Direct memory access (DMA)

DMA is very important for high speed embedded applications, at it allows us to bypass the CPU for memory operations. A summary of this is shown in table V. On the STM32H7 series, the DMAMUX controller needs to be configured to select channels for streams, which is shown the last column. The since all the DMA streams are tied to the ADC/DAC, which are triggered by a 50 kHz timer, we can also say that the DMA streams are updated at a rate of 50 kHz. The first DMA stream loads the current ADC data from the *DR* register and stores it in RAM. The second and third DMA streams stores the desired output data into the DACs, from a variable in RAM. The first DAC, which contains the guitar output, will store the data into *DH12R1*, (12 bit right aligned channel 1 register), which explain the need for a 16 bit DMA transfer. The second DAC, which contains the non-guitar audio output, will store the data into *DH8R2* (8 bit right aligned channel 2 register), hence the DMA transfer only needs 8 bits.

F. Operational Amplifier (OPamp)

To save on hardware resources, we decided to use one of internal OP-amps for a specific function, namely amplifying the non-guitar audio segments. This is done to avoid any interference between the two DACs, as they do not pass through a common OPamp. It is worth noting, that for unknown reasons, the OPamps would saturate at about 2.2V. This is problematic as we lose about 1/3 of data resolution, hence we did not use the internal OPamp for guitar amplification. Table VI summarizes the use of the internal OP-amp. We note that OPamp2 has to be used because it is can be internally connected to DAC2's output.

TABLE V: *DMA1 USAGE*

DMA1 Streams	From	To	Data size	DMAMUX Channel
Stream0	ADC1	RAM	16 bits	9
Stream1	RAM	DAC_1	16 bits	67
Stream2	RAM	DAC_2	8 bits	68

TABLE VI: *OPamp2 USAGE*

OPamp2	Usage
Configuration	Non-inverting amplifier
Gain	2
Input	dac_out1
Output	PE7

G. General purpose input output (GPIO)

Table VII shows an overview of the GPIO usage. GPIOA is mainly used for analog peripherals, where as the other GPIO pins are for LED's and user switches. The pins are chosen for their ease of connections through the associate pin headers on the NUCLEO-H753ZI board. On this board, a wire is connected from PA4 to the post-DAC processing circuit, out to the speaker. PA5 or PE7 are exclusively connected to the speaker directly (without any processing) for the non-guitar audio segments. This is to avoid interfering with the circuit that was intended for the guitar waves.

TABLE VII: *GPIO USAGE*

GPIO Pins	Connection	Mode
PA4	DAC_OUT1	Analog
PA5	DAC_OUT2	Analog
PA6	ADC1_INP3	Analog
PB0	Green User LED	Output
PB14	Red User LED	Output
PC13	Nucleo Push Button	Input
PE0	User Push Button	Input
PE1	Yellow User LED	Output
PE7	OPAMP2_OUT	Analog

H. Basic timers (TIM)

On the STM32H753ZIT6, Timer6 and the DAC share an interrupt line. Table VIII presents an overview of the timer peripherals. Timer6 and Timer7, listed as 'basic timers', contain very little functionality such as PWM wave generation, and are only used to generating update events to be used with an ADC or DAC peripheral. Hence, Timer6 is used to trigger both DAC channels, at 50 kHz. Equation 3 shows the calculation

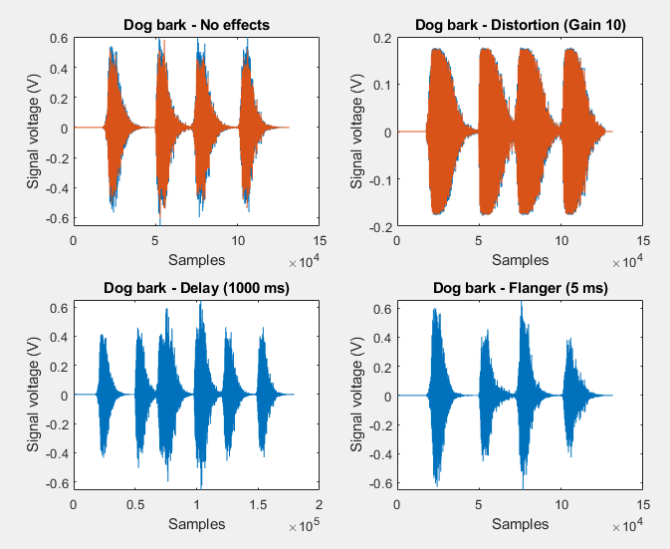


Fig. 6: Music effects on a dog bark

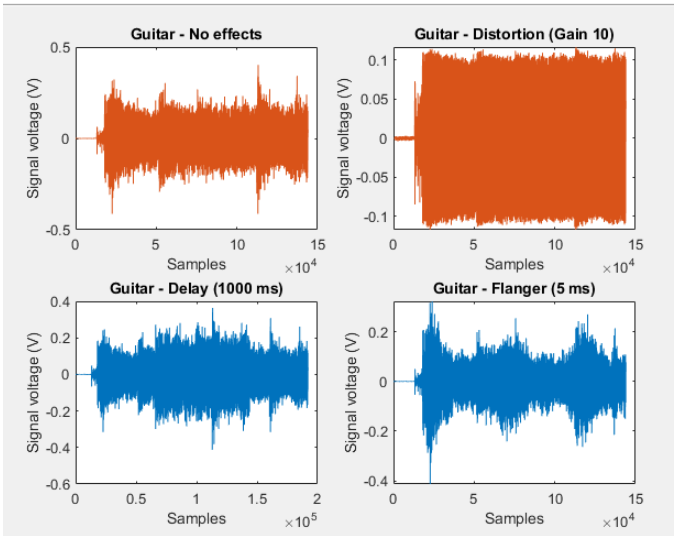


Fig. 7: Music effects on a guitar track

for obtaining the values for the period, using a CPU rate of 64 MHz and a prescaler of 64.

$$Period = \frac{64 * 10^6}{Prescaler - 1} * (\frac{1}{Rate}) - 1 = 19 \qquad (3)$$

TABLE VIII: *TIM6 USAGE*

TIM6	Usage
Prescaler	63
Period	19
Counter Mode	Up
Trigger Event	Update Event

I. Nested Vector Interrupt Controller (NVIC)

Table IX presents the interrupts that are used in the application. With the TIM6 IRQ, the sample of the currently loaded .wav file is incremented to play the next sample. Within the DMA IRQ, a flag is set to signal that a new audio sample is ready to be processed from the guitar. The ADC IRQ is run every time a new conversion is finished, and allow us to synchronize and keep track of the DMA transfers.

TABLE IX: *NVIC USAGE*

Interrupt	Description
TIM6_DAC_IRQn	Update event on timer (CNT=ARR)
ADC_IRQn	On ADC sequence conversion
DMA1_Stream0_IRQn	On fully complete DMA transfer

III. Results And Discussion

A. Inventory

Table X shows the cost needed to build this product. This is a crucial point, as a main goal goal for the product was to create a guitar pedal system that is cheaper than anything on the market. The total comes to **$35.07**. In the comparisons that will be made later on, we note that some products used the cost of producing the PCB instead, where as we just used a breadboard. In the end, if we did a PCB, it would come around to a higher cost of around $60, as although the NUCLEO board is no longer needed, the price of ordering the PCB is still high.

TABLE X: *INVENTORY COST*

Item	Amount	Cost (Total)	Procurer
NUCLEO-H753ZI	1	$26.46	estore.st.com
NE5532 (OPamp)	2	$0.82	digikey.com
Resistors	10	$1	digikey.com
Capacitors	7	$0.77	digikey.com
Breadboard	1	$3	digikey.com
Jumpers/Buttons/LEDs	1	$3	digikey.com

B. Comparison of hardware use

TABLE XI: *HARDWARE COMPARISON OF OUR PRODUCT AND PRODUCT FROM [1] AND [2]*

Feature	Our Project	[1] - Lang	[2] - Malko
MCU	STM32H753ZI	STM32F446RC	CS48560
External ICs	No	Yes	No
Cost	$35.07	$80	N/A
ADC Sampling rate	50 kHz	48 kHz	48 kHz
DAC Sampling rate	8.3 kHz	48 kHz	48 kHz
Clock Speed	64 MHz	180 MHz	150 MHz

Table XI above presents an overview of the differences within the hardware implementation of our project, and papers from Lang[1] and from M.Malko[2]. The main observation is that our product requires fewer resources, mainly due to the choice of the MCU. In Lang[1], the use of the STM32F4, which is a non-specific DSP chip forced them to select an external 16 bit ADC and DAC chip, which drove up their costs. Additionally, due to the limited memory of that chip, an external flash ROM was used to hold their audio recordings. Our DAC sampling rate is 8.3 kHz because of the previously mentioned oversampling situation, in which because we used a 50 kHz ADC sample rate, due to the 256x oversampling, the output sample rate got scaled down. This means that our

buffers get full at a much lower rate than the other projects, which increases our efficiency as we can store data for longer before having to refresh the buffers.

In Malko[2], they used a DSP specific chip, the CS48560, with internal 12 bit ADC and DAC, while not using external ICs. However, this chip is expensive and obsolete. They did not implement a full circuit, hence their costs cannot be determined. Our clock speed lower at 64 MHz as we did not get obtain better results from higher clock speed. Thus, our product is cheaper and has a slower CPU rate, decreasing power consumption. For more complex operations requiring a faster clock speed, we have ample ressources to do so.

C. Comparison of features

Table XII presents an overview of the differences within the implemented features. Our product does contain most of the features from Lang's[1] product implements, with the exception of a few trivial effects such as tremolo and chorus, which are simply slight variations of the already implemented effects. They did however contain an real-time audio tuner, which is a non-trivial process that was not implemented, as it was outside the scope of this project. However, our product does contain the ability to modulate non-guitar audio clips, which is not observed in any of the papers. Our product was far superior to product from Malko[2], in terms of the amount of implemented effects. In their product, they did not implement any analog circuitry to handle the noisy signals that distortion effects create, hence their lack of distortion, which is one of the most important effects for guitar players. Due to the quantity and quality of implemented effects, our product is very successful in this aspect. Our looper was also impressive, achieved a maximum recording duration of 144 seconds, which was done by using 10 individual 128KB SRAM segments. Unlike Lang[1], we did not use an external flash module, they had a much longer looper duration.

TABLE XII: *FEATURE COMPARISON OF OUR PRODUCT AND PRODUCTS FROM [1] AND [2]*

Feature	Our Project	[1] - Lang	[2] - Malko
Guitar Effects	Yes	Yes	Yes
Non-Guitar Effects	Yes	No	No
Looper	144 s	24 min	No
Distortion	Yes	Yes	No
Delay	Yes	Yes	Yes
Flanger	Yes	Yes	Yes
Echo	Yes	Yes	No
Chorus	No	Yes	Yes
Tremolo	No	Yes	No
Tuner	No	Yes	No

D. Recorded signals

Figure 8 shows the obtained signal data for the implemented effects. This was done by logging all the values through the debugger where they were written to a file. In summary, all the recorded signals effects match the corresponding effects that were simulated in figure 7, demonstrating the correctness of the results.

1) Top left - ADC: We see the original signal through the ADC, which was a chord played three times, hence three spikes in the wave.

2) Top right - DAC - Clean: We see a loopback test, where the output should match the output.

3) Middle left - DAC - Delay: In the middle left, a delay is shown, where we can see the superposition of the chord with itself, creating multiple peaks.

4) Middle right - DAC - Echo: In the middle right, the echo effect is shown, where there is a superposition with a changing gain, which explains the many small peaks that we observe.

5) Middle right - DAC - Flanger: In the bottom left, we see the flanger effect. Within each chord, there is a sinusoidal pattern in the amplitude, which shows the modulation with the sinusoidal wave.

6) Bottom right - DAC - Distortion: In the bottom right, we see the distortion, which heavily clips the signal beyond the capabilities of the DAC. This projects that the output sound will be extremely noisy and loud.

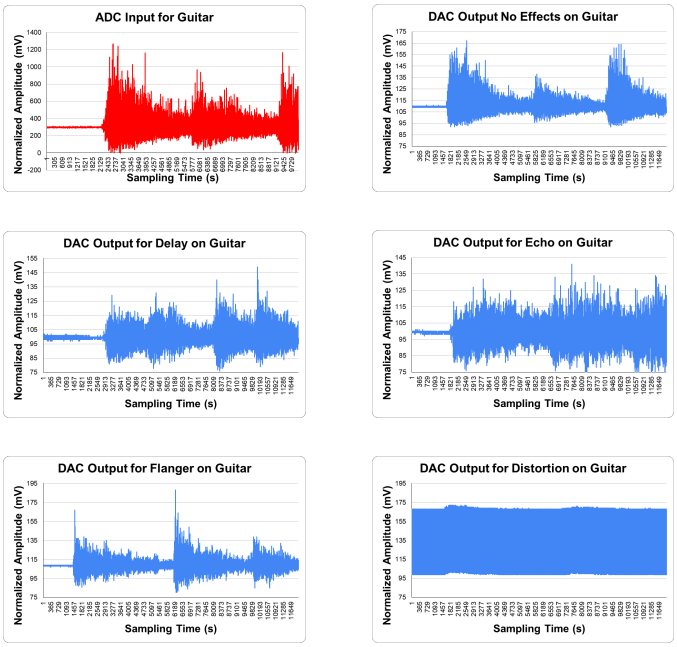


Fig. 8: Captured signals of 3 strummed 'E' Chords

E. Listening tests

1) Live guitar playing: Due to the nature of this application being auditory, our ears provide the best verification of results. Without any effects, the guitar sound is clean with minimal noise. However, the higher frequencies are not as present in the sound as what would be wanted. This is due to the choice of the filtering.

The distortion effect, true to its name, heavily distorts the signal. Naturally, there is alot of noise in this effect, which can be hard to listen to for prolonged periods of time.

The modulation effects were very successful, flanger, delay and echo. There is no noise added from the superposition of

the time-shifted signals on top of each other. Up to 10 signals were superimposed for the echo effect, and the sound was very clear, with a noticeable exponentially increasing attenuation of the time-shifted samples.

The looper was recording and playback the guitar, for a long enough to play a meaning melody over the track. The audio level of the playback is half the amplitude of the recorded guitar wave, and this is done to ensure that the playback music does not overpower the guitar player. In the demonstrations, we were able to play a melody over an 'E' chord that was recorded and played back, proving the viability of the looper.

2) Non-guitar audio segments: A secondary objective of this project was to show common every day sounds such as a dog barking, or a baby crying could be modulated and played through the speaker. This was done successfully, although we emphasize that the audio segments need to be pre-loaded into RAM before flashing the device, as we did not have time to support live microphone recordings, unlike the guitar. The signal quality is clear and presents minimal noise. It is also shown that the modulations effects can work on this, as was shown with the delayed baby-crying in the provided video demonstration.

IV. Conclusion And Future Work

To conclude, we are very happy with the implementation of the guitar pedal. The product successfully implements all the classic guitar effects that are crucial to live music performance, such as echo, delay, flanger and distortion. In addition, a looper function allows us to record and playback our music in real-time. All of this is implemented with fewer resources than anything available on the market, which demonstrates it potential for commercialization. Lastly, we showed that we can also apply effects non-guitar sounds, such as animal sounds or infant sounds, which demonstrate the correctness of the theoretical models, and potential for product development in more areas than just guitar, such as voice coding.

Future works can involve upgrading the DAC, without using external DAC devices. There are promising techniques to increase SNR to obtain a better signal such as dithering [17], which involve generating a noise wave. On the STM32 DACs, they are capable of noise generation, which could be an promising future implementation. In addition, a notable feature that would be developed is the use of a sigma-delta modulator (SDM). This, in combination with oversampling, has been shown to massive increase signal SNR, and is considered the gold standard for ADC sampling techniques [18]. The STM32H753ZI contains a digital digital filter for sigma delta modulation peripheral, which can be used with an external SDM circuit. We would like to improve efficiency by using the specialized master direct memory controller (MDMA), which allows for high speed memory-to-memory transfers.

For a commercially viable product, a PCB will have to be designed and mounted within a chassis, which will require much more work. In the scenario of a PCB design, we will keep the same MCU, and simply upgrade the ICs to the latest iterations on the market. Furthermore, a power conversion circuit will need to be designed, as our product is powered by +5V USB, and the analog circuits are powered by the MCU's dedicated +5V and +3.3V.

Future works related to the MCU are shown in table XIII. In short, each peripheral has some improvement that can be made to increase efficiency, and reduce power consumption. Future related to the product features are shown in table XIV.

TABLE XIII: *FUTURE WORK ON MCU FEATURES*

Feature	*Improvement*
ADC	Increase clock frequency to maximum
DAC	Implement dithering
Comparator	Implement wake-up function
OPamp	Resolve saturation issues
DMA	Implement RAM to RAM DMA transfers
DFSDM	Implement filters for external SDM

TABLE XIV: *FUTURE WORK ON PRODUCT FEATURES*

Feature	*Improvement*
Modulation effects	Implement chorus,vibrato
Distortion effects	Implement fuzz,overdrive
Tuner	Implement real-time tuner
PCB	Custom PCB instead of NUCLEO board
Power	Power conversion circuit

References

[1] I. Lang, "Digital guitar effects pedal," Senior Project Report, Dept. of Electrical Engineering, California Polytechnic State University, San Luis Obispo, CA, USA, June 2018. [Online]. Available: https://digitalcommons.calpoly.edu/eesp/434.

[2] M. Malko, J. Kovacevic, R. Peckai-Kovac, M. Gajic, and M. Jovanovic, "Implementation of digital audio effects for electric guitar on DSP platform," 2011 19th Telecommunications Forum (TELFOR) Proceedings of Papers, Belgrade, Serbia, Nov. 2011, pp. 723-726. IEEE, 2012. doi: 10.1109/TELFOR.2011.6143741.

[3] J. Pakarinen and D. T. Yeh, "A Review of Digital Techniques for Modeling Vacuum-Tube Guitar Amplifiers," Computer Music Journal, vol. 33, no. 2, pp. 85–100, 2009, Available: https://www.jstor.org/stable/40301029.

[4] STMicroelectronics, STM32H753xI - ARM Cortex-M7 Microcontroller with 2MB Flash and 1MB RAM, Datasheet, 2023. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32h753zi.pdf. [Accessed: Nov. 6, 2024].

[5] A. McCabe, "The evolution of analog guitar pedals," Journal of Audio Engineering, vol. 34, no. 3, pp. 112-120, 2018.

[6] V. Verfaille, U. Zölzer, and D. Arfib, "Adaptive digital audio effects (A-DAFx): A new class of sound transformations," IEEE Trans. Audio, Speech, and Language Processing, vol. 14, no. 5, pp. 1817–1831, Sep. 2006.

[7] G. Stevens, "Digital guitar effects: Cost-effective alternatives," Guitar World, Jul. 2021.

[8] J. Pickup, Guitar Pickup Design and Applications, 3rd ed., New York, NY: McGraw-Hill, 2017.

[9] A. Oppenheim, R. Schafer, and J. Buck, Discrete-Time Signal Processing, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

[10] S. W. Smith, The Scientist "and Engineer's Guide to Digital Signal Processing, 2nd ed., San Diego, CA: California Technical Publishing, 1999.

[11] M. U. Torry, "Signal Processing Techniques for Distortion Effects," in Journal of Audio Engineering Society, vol. 46, no. 8, pp. 617-630, 1998.

[12] D. E. Johnson, Digital Delay and Effects Systems, 2nd ed., Boston, MA: Springer, 2003.

[13] T. Yoshida, "Echo Effect Implementation Using FIR Filters," in IEEE Transactions on Signal Processing, vol. 45, no. 7, pp. 1347-1355, Jul. 1997.

[14] B. Thompson, "Flanger Effects and Frequency Modulation Techniques," in Journal of Audio Signal Processing, vol. 23, no. 6, pp. 529-540, 2005.
[15] R. Baker, "Audio Looping Systems: Techniques and Applications," in IEEE International Conference on Audio Processing, 2010.
[16] P. Green, Practical Tap Tempo Synchronization, New York, NY: Academic Press, 2015.
[17] Stuart, J. Robert; Wilson, Rhonda J.; 1992; A Search for Efficient Dither for DSP Applications [PDF]; Boothroyd Stuart Ltd., Huntingdon, Cambridgeshire, United Kingdom; Paper 3334; Available from: https://aes2.org/publications/elibrary-page/?id=6799
[18] R. Gray, "Oversampled Sigma-Delta Modulation," in IEEE Transactions on Communications, vol. 35, no. 5, pp. 481-489, May 1987, doi: 10.1109/TCOM.1987.1096814. keywords: Delta-sigma modulation;Mathematical model;Quantization;Analog-digital conversion;Noise robustness;Additive noise;Additive white noise;Circuit noise;Demodulation;Nonlinear systems,