ECE 445
Spring 2025
Design Document:

Digital Pitch Shifter for Guitar

Team #60:
William Chang (wqchang2)
Eric Moreno (emoren40)
Zhengjie Fan (zfan11)

TA: Shengyan Liu
Professor: Michael Oelze

Table Of Contents

1 Introduction

1.1 Problem

There are new guitar players every year that learn how to use the guitar and continue to advance their skills. Players become more advanced and eventually learn more technical skills. Unfortunately, there are different types of guitars that can perform in different ways. Guitarists without access to a tremolo system face significant limitations in their ability to create expressive vibrato and pitch-bending effects, which are essential for adding emotional depth and dynamic variation to their playing. Without these techniques, the guitar's sound can feel static or restrained, especially in genres like rock, blues, and jazz, where pitch manipulation is crucial. Traditional tremolo systems, though effective in addressing this issue, require invasive modifications to the guitar body, such as routing or altering the bridge. These changes not only compromise the guitar's original design but can also affect its sound and value. Additionally, such systems may not be suitable for all playing styles, or for guitarists who prefer a more minimalist approach. As a result, players seeking greater versatility in their instrument face the difficult choice between sacrificing their guitar's aesthetics or settling for limited expressive capabilities. This is the gap the proposed project aims to fill.

1.2 Solution

The solution to the aforementioned issue is a compact, attachable digital pitch-shifting device that uses a sonic sensor to detect the proximity of the guitarist's hand to the bridge of the guitar. This will be so that the user can attach this component to any electric guitar with a jack cable that would normally be attached to an amplifier. The user would connect their guitar in the

system and this will begin the modulation of any signal put through by having the sensor and microcontroller work together in an algorithm. This software will pitch the guitar up and down according to what the user decides.

As the player moves their hand closer or farther from the sensor, the pitch of the guitar signal is dynamically adjusted, allowing for real-time pitch shifts up or down. The distance is calculated and adjusted for any small movements made. This enables the guitarist to perform expressive techniques like vibrato and pitch bending, similar to those provided by traditional tremolo systems, but without the need for invasive body modifications. Additionally, the device includes a switch or button that lets the player toggle between upward or downward pitch shifts, offering greater flexibility in controlling the pitch. As seen in Figure 1, the core of our project resides in the microcontroller. The amplifications will be the output of what the microcontroller assessed the sensor to measure and the setting it is on, switched from buttons on the system. This lightweight solution enhances the player's creativity while preserving the guitar’s natural design and playability. Furthermore, the additional buttons or switches can enable further effects such as reverb, chorus, or delay, giving the player more creative control over their sound. These augmentations enhance the guitarist’s ability to experiment with a wider range of tones and textures without needing to modify the guitar's body or permanently alter its design.

1.2 Visual Aid

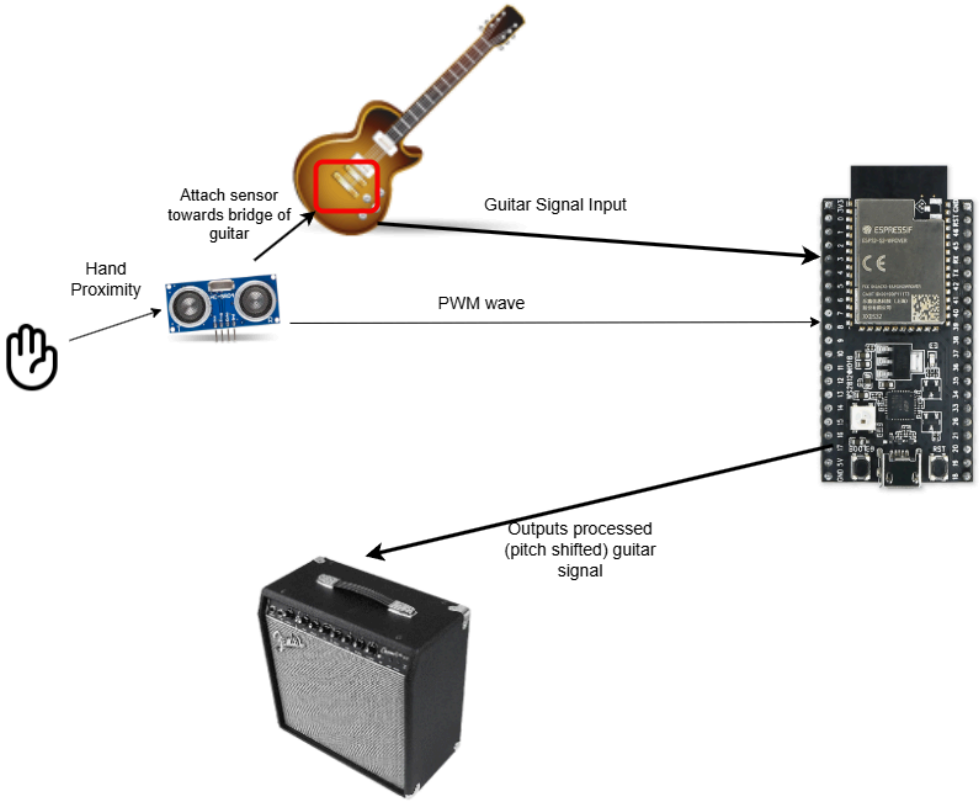


Figure: Design Overview

1.3 High Level Requirements

The project may only be successful under the following conditions:

Real-Time Pitch Shifting with Low Latency

- The system must process incoming guitar signals and shift their pitch without perceptible delay (<10ms total latency) to ensure natural playability.

High-Fidelity Audio Processing

- The pitch-shifted output must maintain at least 48 kHz sampling rate and 12-bit depth for ADC and 8-bit depth for DAC. (ESP32 microcontroller)

User-Controlled Pitch Adjustment

- The device must allow adjustable pitch shifting from -2 to +2 semitones.

2 Design

2.1 Block Diagram

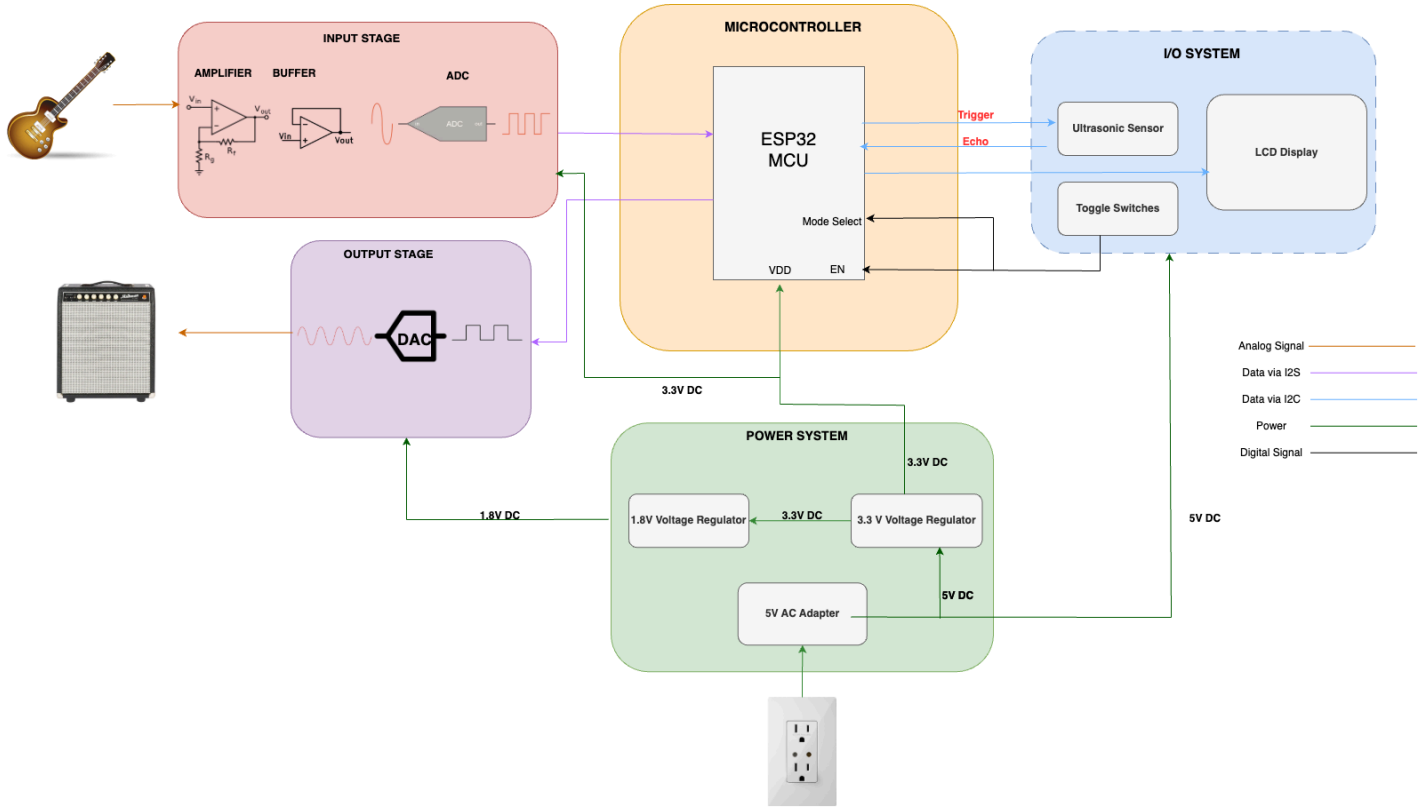


Figure: Block Diagram

Our design consists of 5 different subsystems that include the ADC & preamplification of a signal to be controlled by the microcontroller. A sensor that sends back data to the microcontroller and then finally to our guitar's amplifier through a DAC. All being powered by our LDO step down subsystem to ensure correct rated voltages.

2.2 Functional Overview & Block Diagram Requirements

2.2.1 Input Stage Subsystem

The preamp subsystem is designed to condition the guitar's signal before it is processed by the ESP32 microcontroller. A guitar preamp pedal is used to boost the signal, ensuring it reaches a level suitable for the ESP32-S3. In this subsystem, the guitar's signal is first connected to a 1kΩ resistor, while a 3.3V voltage source is fed through another resistor to create a DC offset. These signals are then processed by two op-amps, which both amplify and buffer the voltage. This ensures the signal is stabilized, eliminating any negative voltage and reducing unwanted noise. The conditioned signal is then fed into an ADC, which converts the analog signal into a digital format that the ESP32 microcontroller can interpret. The entire preamp subsystem is powered by a 3.3V power supply from the power subsystem, ensuring stable operation. This setup guarantees that the microcontroller receives a clean, noise-free signal within its limited voltage range for accurate processing.

Requirements	**Verifications**

• The preamp must condition the guitar signal before processing by the ESP32-S3.
• A guitar preamp pedal must boost the signal to a suitable level for the ESP32-S3.
• A 1kΩ resistor must be used to connect the guitar signal, and a 3.3V voltage source must be applied through another resistor to create a DC offset.
	• ADC Testing:
	  ○ Use an oscilloscope to measure the ADC input and verify that it accurately digitizes the analog signal.
	• Op-Amp Functionality Check:
	  ○ Measure input and output voltages of the op-amps to confirm amplification, buffering, and noise reduction.
	• Noise and Distortion Analysis:
	  ○ Perform frequency and

• The preamp must include two op-amps to amplify and buffer the signal, ensuring stability and noise reduction. • The circuit must eliminate any negative voltage to protect the ADC input. • The ADC must convert the conditioned analog signal into a digital format readable by the ESP32. • The output signal must remain within the ESP32's allowable voltage range for accurate processing.	spectrum analysis to check for unwanted noise or signal distortion.

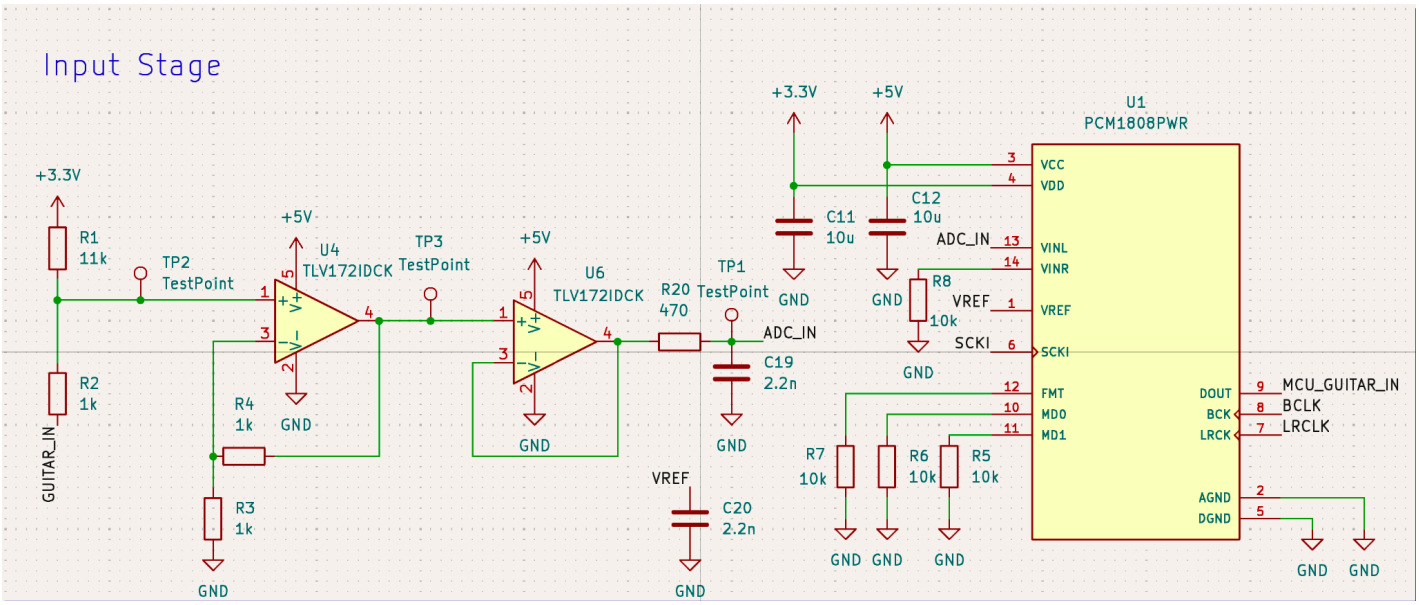
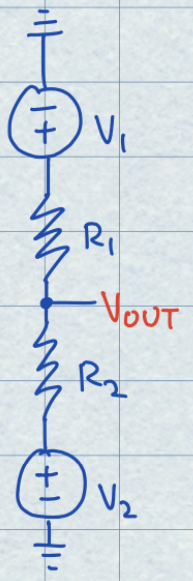


Figure: Input Stage Subsystem Schematic

DC offset



$$\frac{V_1 - V_{OUT}}{R_1} = \frac{V_2 - V_{OUT}}{R_2}$$

$$\frac{V_1}{R_1} - \frac{V_2}{R_2} = \frac{V_{OUT}}{R_1} - \frac{V_{OUT}}{R_2}$$

$$\frac{R_2V_1 - R_1V_2}{R_1R_2} = \frac{R_2V_{OUT} - R_1V_{OUT}}{R_1R_2}$$

$$R_2V_1 - R_1V_2 = (R_2 - R_1)V_{OUT}$$

$$V_{OUT} = \frac{R_2V_1 - R_1V_2}{R_2 - R_1}$$

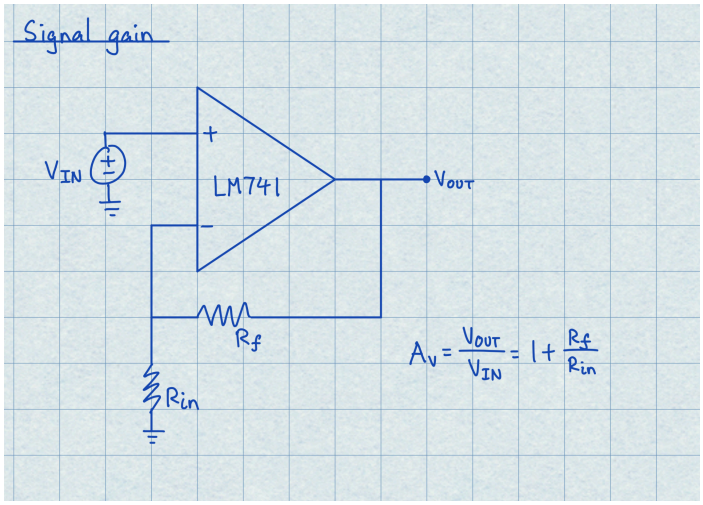
- $V_1 = 3.3\,V$ (from power supply)
- $V_2 \approx 0.3\sin(\omega t)\,V$ (guitar signal) test to see voltage range

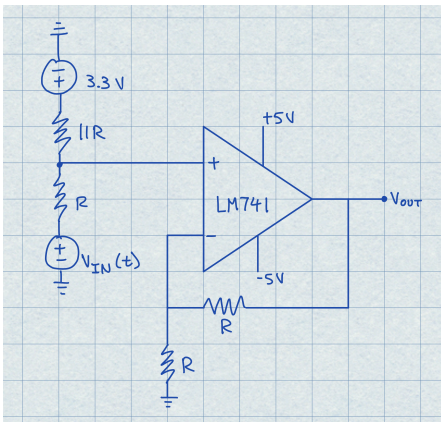
Choose R_1 and R_2 $\longrightarrow$ $0 \le V_{OUT} \le 1$ - min value = 0 V

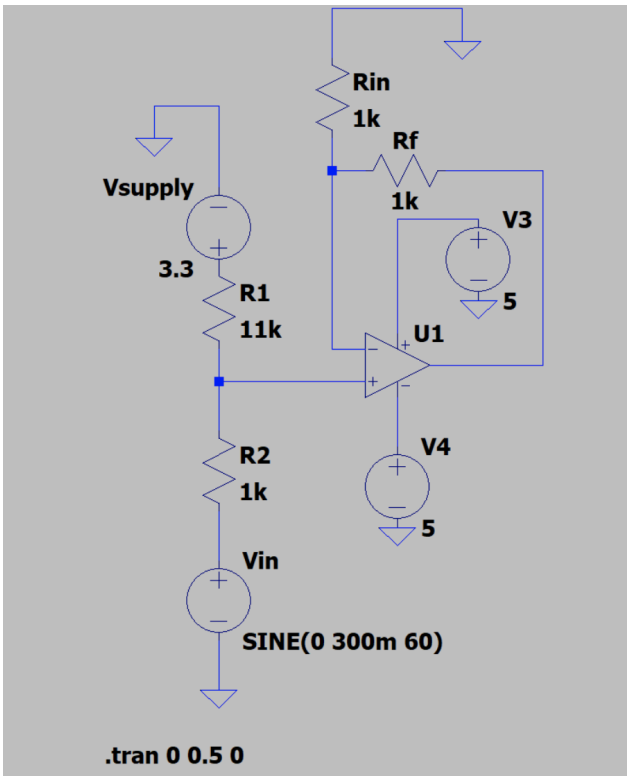
$$V_{OUT} = \frac{R_2V_1 - R_1V_2}{R_2 - R_1}$$

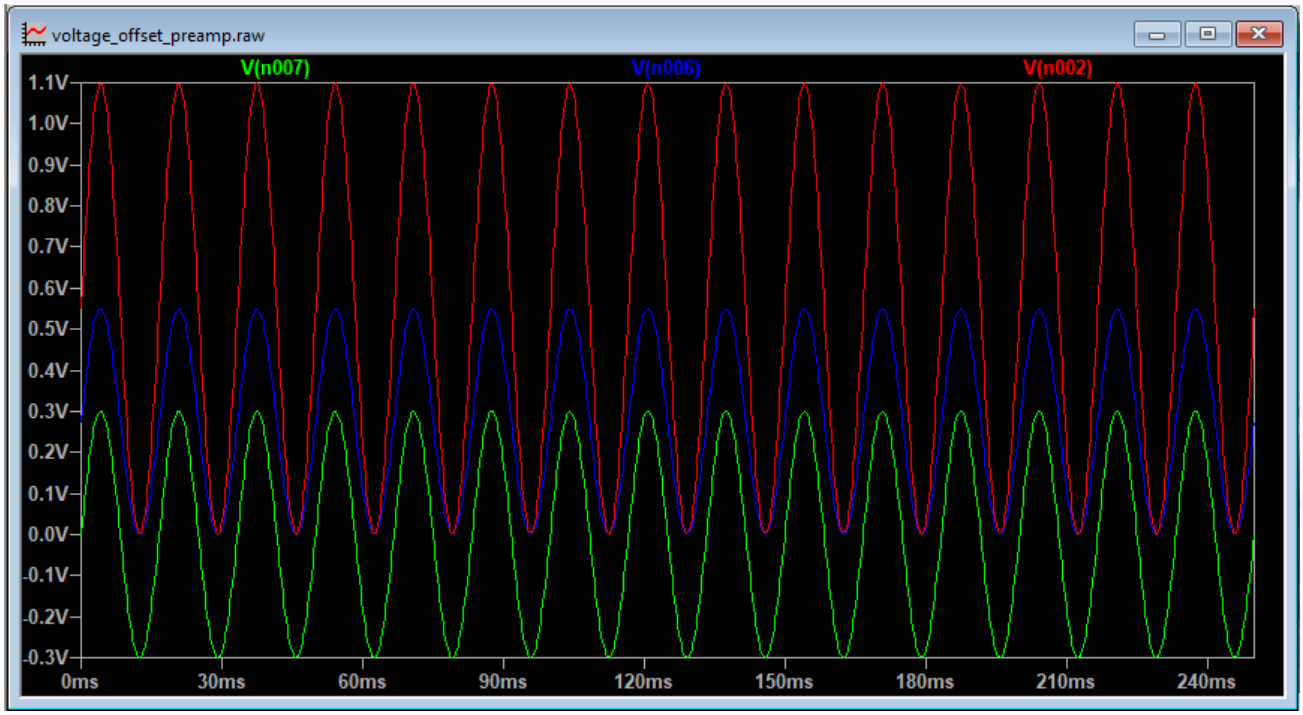
$$0 = \frac{R_2(3.3) - R_1(-0.3)}{R_2 - R_1}$$ → lowest sine wave voltage

$0 = 3.3R_2 + 0.3R_1 \longrightarrow$ $R_1 \ge 11R_2$ make sure $V_{OUT} \ge 0$


Signal gain
VIN
LM741
VOUT
Rf
Rin
$A_V = \frac{V_{OUT}}{V_{IN}} = 1 + \frac{R_f}{R_{in}}$


3.3 V
11R
R
VIN (t)
+5V
LM741
VOUT
-5V
R
R


Rin
1k
Rf
1k
Vsupply
3.3
R1
11k
V3
5
U1
R2
1k
V4
5
Vin
SINE(0 300m 60)
.tran 0 0.5 0



Graph: Preamp Circuit Simulation

Green: Input Signal, Blue: DC offset, Red: Preamp Output

2.2.2 Microcontroller Subsystem

The ESP32 microcontroller acts as the core processing unit, making both the signal input/output operations and performing real-time signal processing critical to the system's functionality. A key responsibility of the ESP32 is managing audio data, utilizing its Analog-to-Digital Converter (ADC) and Digital-to-Analog Converter (DAC) interfaces. The analog guitar signal is first digitized by the ADC, then processed using pitch-shifting algorithms. Following processing, the signal is reconverted into an analog form via the DAC, allowing it to be output to a guitar amplifier for playback. Beyond audio signal processing, the ESP32 interfaces with an HC-SR04 ultrasonic sensor to capture real-time hand proximity data. It generates a high trigger pulse on a

GPIO pin to initiate the sensor's distance measurement cycle. The pulse remains active to ensure continuous data readings, which are then returned to the microcontroller. The ECHO pin output is subsequently read by the ESP32 to calculate the distance between the sensor and the player's hand. This distance is used as a dynamic input, modulating the pitch-shifting parameters in real time. For precise and stable operation, the microcontroller relies on both internal and external clock sources to synchronize its processes and improve the accuracy of real-time pitch adjustments. The system is powered on and configured via a button in the IO Subsystem, which also allows the user to select between different operating modes or settings. Additionally, the microcontroller integrates strapping data from the software to ensure proper initialization during startup. The ESP32 also manages user interactions, such as effect toggling or parameter adjustments, through GPIO pins connected to external buttons or switches. This enables flexible user control over the system's functionality in a seamless manner.

Requirements	**Verifications**
● The ESP32 must act as the core processing unit, managing both signal input/output operations and real-time signal processing. ● The ADC must accurately digitize the analog guitar signal for processing. ● The DAC must correctly reconstruct the processed digital signal into an analog format for playback through a guitar amplifier. ● The ESP32 must run pitch-shifting algorithms in real time with minimal latency. ● The microcontroller must interface with an HC-SR04 ultrasonic sensor to capture real-time hand proximity data.	● Real-Time Processing Validation: ○ Measure processing latency using a logic analyzer to ensure pitch-shifting algorithms execute in real time without audible delay. ● Ultrasonic Sensor Integration Testing: ○ Use an oscilloscope to verify that the ESP32 generates the correct trigger pulse for the HC-SR04 sensor. ○ Capture ECHO pin responses and confirm accurate distance calculations based on expected hand positions. ○ Check that detected distance values properly modulate

● The ESP32 must generate a high trigger pulse to initiate distance measurement and continuously receive ECHO pin data for distance calculation. ● Hand proximity data must dynamically modulate pitch-shifting parameters in real time. ● Internal and external clock sources must ensure precise and stable operation, improving pitch adjustment accuracy. ● The system must be powered on and configured via a button in the IO Subsystem. ● The ESP32 must support multiple operating modes and settings, selectable via external buttons or switches. ● Strapping data from software must be correctly interpreted during startup to ensure proper initialization. ● User interactions, such as effect toggling or parameter adjustments, must be handled via GPIO pins for seamless control.	pitch-shifting parameters in software. ● Clock Synchronization Testing: ○ Verify the stability of internal and external clocks using an oscilloscope to ensure timing consistency. ● User Input Response Testing: ○ Press external buttons and switches to confirm proper operation of effect toggling and parameter adjustments. ○ Log GPIO inputs to ensure accurate command interpretation by the ESP32.

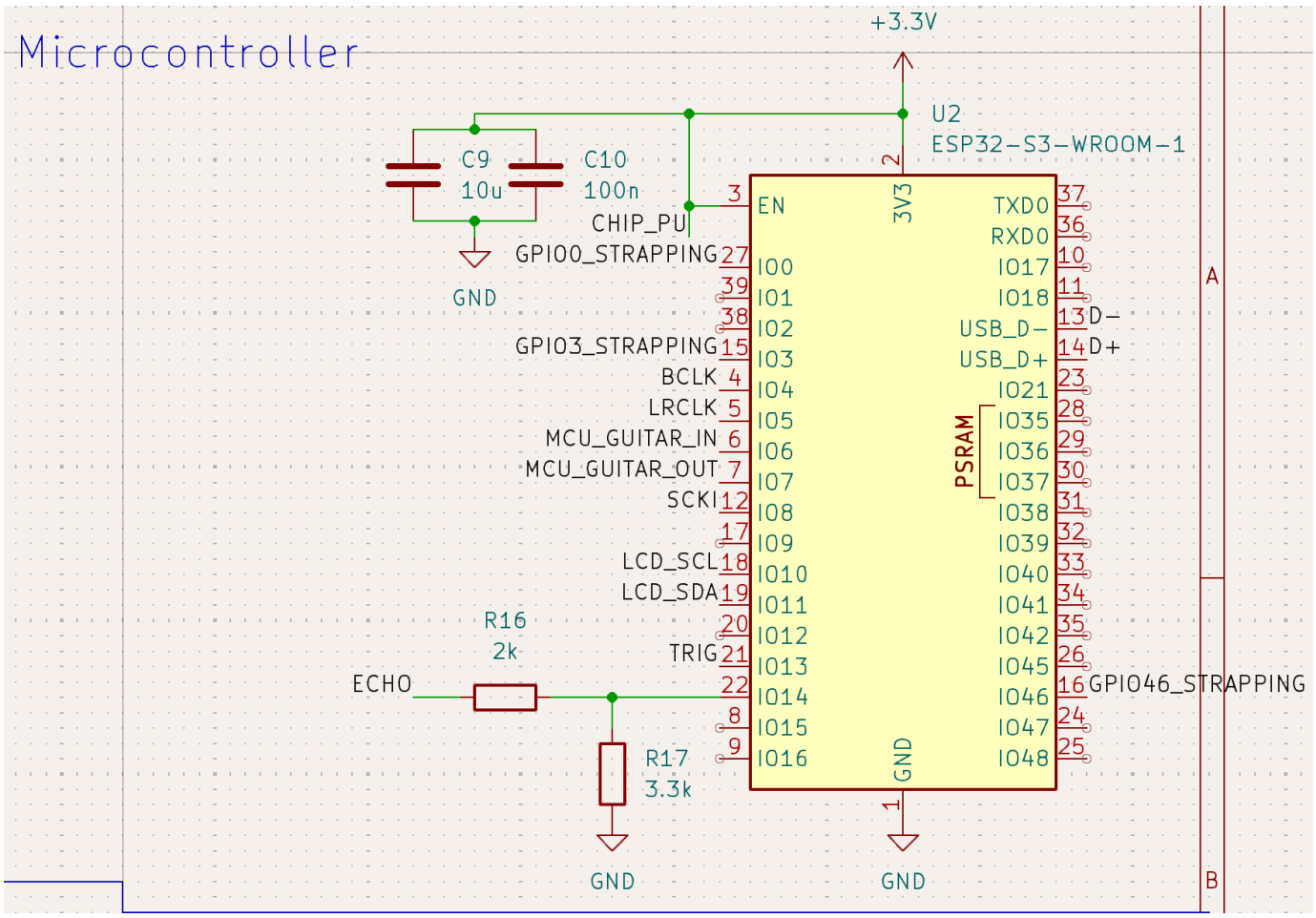


Figure: Microcontroller Subsystem Schematic

2.3.3 I/O Subsystem

The I/O subsystem integrates multiple components to facilitate user interaction and system functionality. The ultrasonic sensor is a key input device, providing real-time distance measurements for processing. Pull-down resistors are implemented to ensure that push buttons remain in a consistent low state when not pressed, preventing unintended signals. A Vbus connection is included to enable USB connectivity, allowing for programming and power delivery to the system. Additionally, an LCD screen is incorporated to display essential information, including component statuses and labeled outputs, ensuring clear user feedback. To enhance system configuration, GPIO strapping is implemented, allowing certain pins to define system behavior during startup. These strapping pins help configure the boot mode, clock settings, and other essential parameters, ensuring that the system initializes correctly and operates as intended without requiring manual intervention each time the device is powered on.

Requirements	**Verifications**
● The I/O subsystem must integrate multiple components to facilitate user interaction and system functionality. ● The ultrasonic sensor must provide real-time distance measurements as an input for processing. ● Pull-down resistors must be implemented on push buttons to maintain a consistent low state when not pressed, preventing unintended signals. ● A Vbus connection must be included to enable USB connectivity for programming and power delivery. ● An LCD screen must display essential information, including component	● Ultrasonic Sensor Input Testing: ○ Use an oscilloscope to verify that the sensor's signal is correctly received and interpreted by the ESP32. ○ Check real-time data output to confirm accurate distance measurements. ● Push Button Stability Testing: ○ Measure voltage levels at the button inputs with a multimeter to verify that pull-down resistors prevent floating signals. ○ Observe button response in software to confirm proper

statuses and labeled outputs, for clear user feedback. ● GPIO strapping must be implemented to define system behavior during startup. ● Strapping pins must correctly configure boot mode, clock settings, and other essential parameters to ensure proper system initialization without requiring manual intervention.	state changes. ● USB Connectivity Testing: ○ Test programming functionality through the USB interface to ensure reliable data transfer. ○ Verify that the system powers on correctly via USB power delivery. ● LCD Display Validation: ○ Check that the LCD screen correctly initializes and displays expected information. ○ Simulate different system states to ensure real-time updates on the display. ● GPIO Strapping Verification: ○ Measure GPIO strapping pin states at startup to confirm correct boot mode and clock configuration. ○ Power cycle the system multiple times to verify consistent initialization behavior.

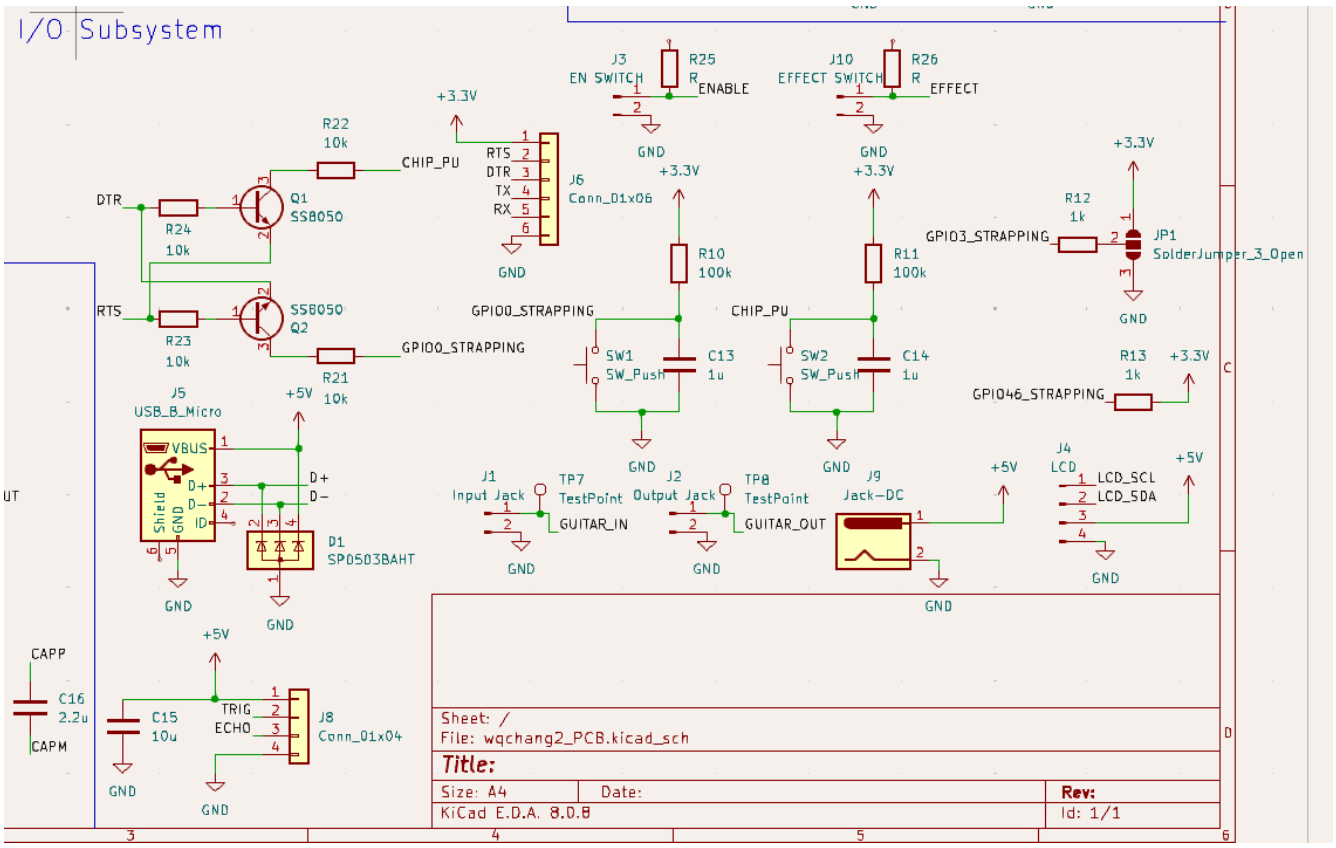


Figure: I/O Subsystem Schematic

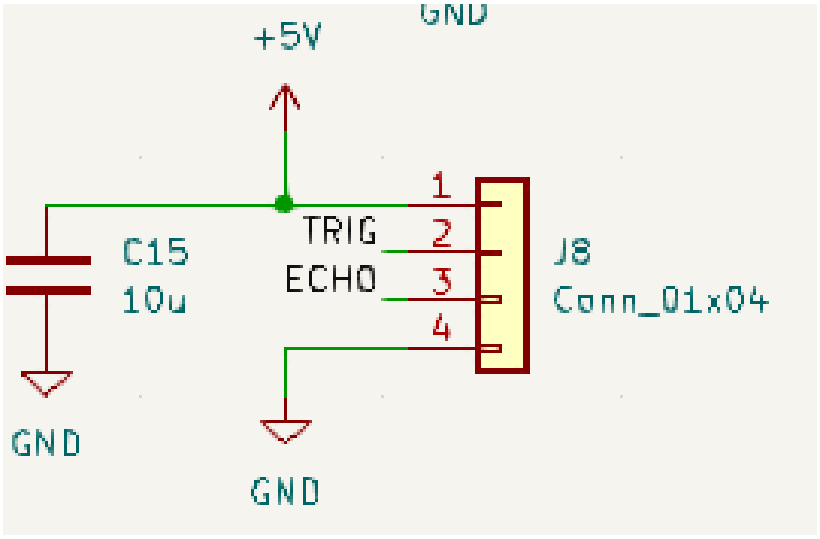


Figure: Sensor Schematic

2.3.4 Output Stage Subsystem

The output stage of our project is responsible for converting the modified digital audio signal from the ESP32 microcontroller back into an analog signal that the guitar amplifier can process. This is achieved using a Digital-to-Analog Converter (DAC), which reconstructs the waveform from the .wav file generated based on sensor measurements. To ensure accurate signal reproduction, the DAC operates using the same internal and external clocks as the microcontroller and ADC, maintaining synchronization and preventing artifacts in the audio output. Since the ESP32 DAC outputs a signal between 0V and 3.3V, it must be shifted and conditioned to match the input requirements of the guitar amplifier. The amplifier's instrument input expects a signal level of approximately 100mV to 1V peak-to-peak. To achieve proper signal conditioning, the DAC output undergoes DC offset removal using a capacitor, ensuring that the signal is centered around 0V rather than the ESP32's 1.65V bias. Additionally, a voltage divider or an op-amp circuit is used to scale the signal to the appropriate voltage range.This stage is essential for preserving the integrity of the digitally modified audio while ensuring that the output signal is properly formatted and voltage-matched for seamless amplification.

Requirements	**Verifications**
● The system must convert the modified digital audio signal from the ESP32 into an analog signal using a DAC. ● The DAC must reconstruct the waveform accurately based on the .wav file generated from sensor measurements. ● The DAC must operate using the same internal and external clocks as the ESP32 and ADC to maintain	● Signal Conversion Testing: ○ Use an oscilloscope to verify that the DAC outputs a properly reconstructed waveform corresponding to the digital input signal. ○ Clock Synchronization Verification: ○ Measure timing signals using an oscilloscope or logic analyzer to ensure the DAC,

synchronization and prevent audio artifacts. ● The ESP32 DAC output, which ranges from 0V to 3.3V, must be conditioned to match the guitar amplifier's instrument input requirements (approximately 100mV to 1V peak-to-peak). ● A DC blocking capacitor must be used to remove the ESP32's 1.65V DC offset and ensure the signal is centered around 0V. ● A voltage divider or an op-amp circuit must scale the signal to the appropriate voltage range for the amplifier input.	ESP32, and ADC remain synchronized. ● Signal Level Adjustment: ○ Measure the DAC output voltage range and confirm that it is properly conditioned to the amplifier's required input range. ○ Check the effect of the voltage divider or op-amp circuit in scaling the signal appropriately. ● DC Offset Removal Testing: ○ Use an oscilloscope to verify that the capacitor removes the DC offset and centers the signal around 0V. ○ Audio Quality Assessment: ○ Connect the output to the guitar amplifier and test for distortion, noise, or loss of clarity.

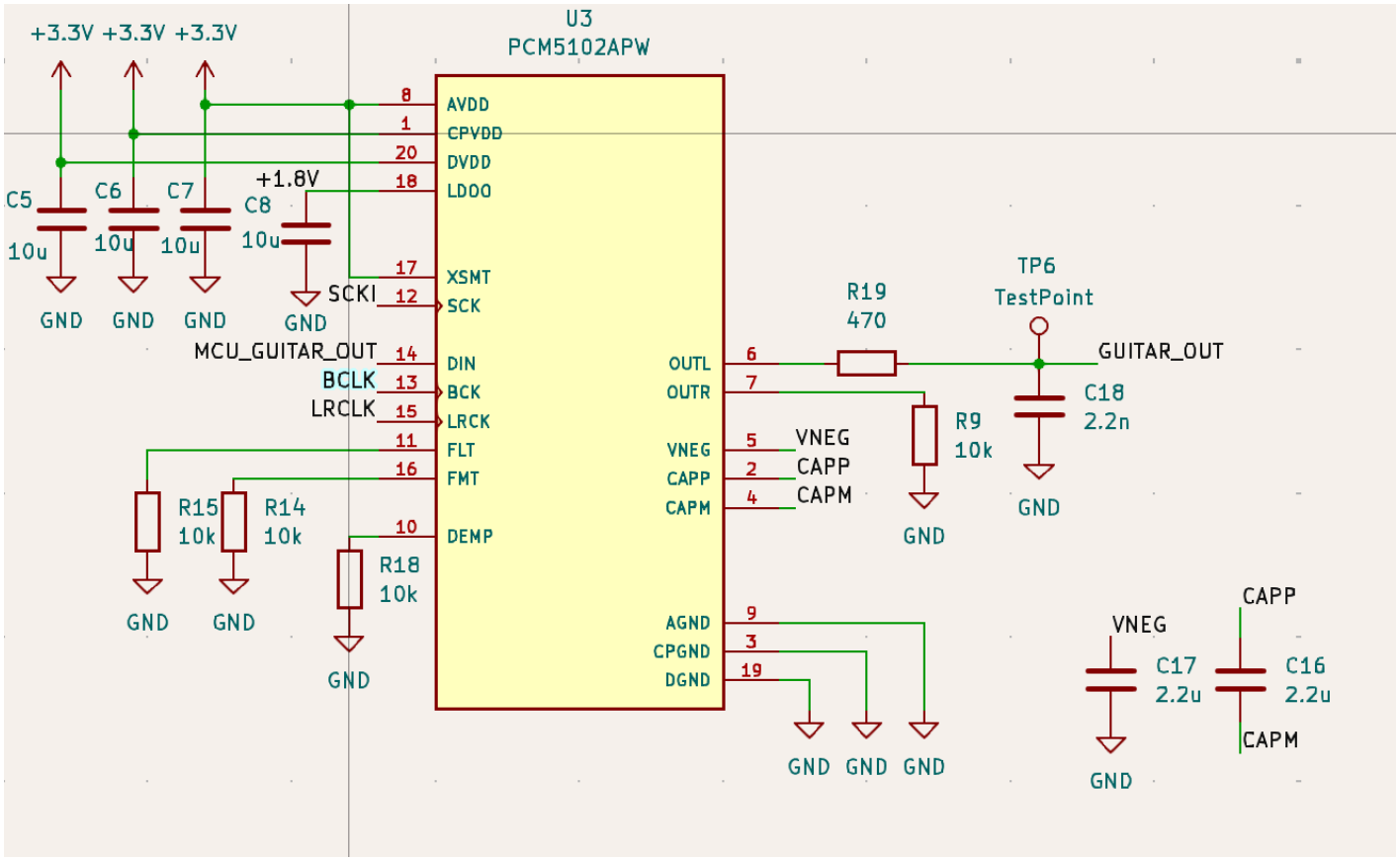


Figure: Output Stage Subsystem Schematic

2.3.5 Power Subsystem

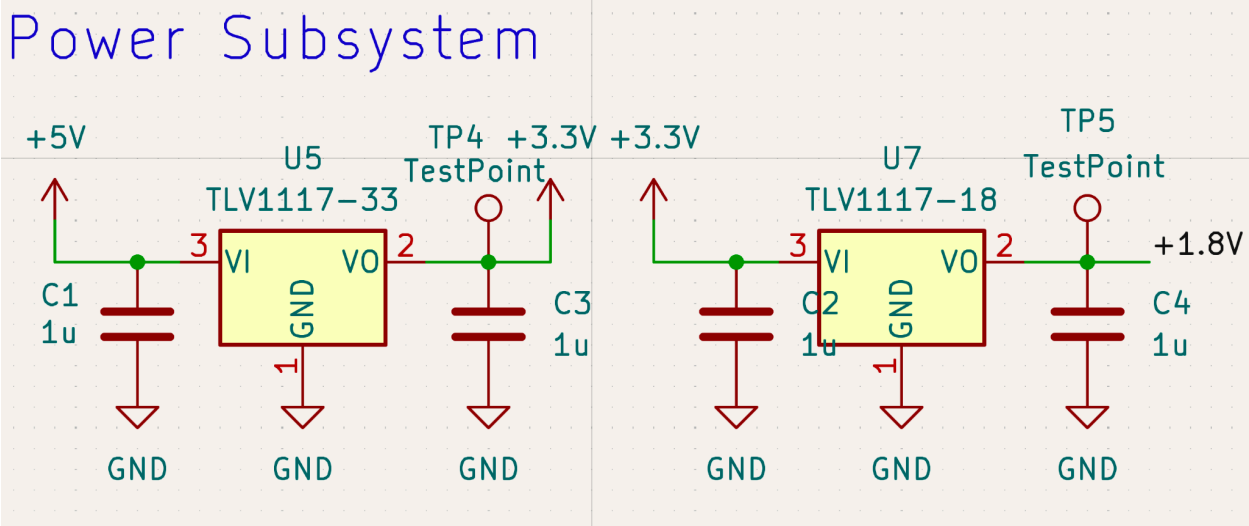


Figure: Power Subsystem Schematic

The power management system ensures stable operation of both the ESP32 microcontroller and the HC-SR04 ultrasonic sensor using a 5V power supply. Since the ESP32 operates at 3.3V, a Low Dropout Regulator (LDO) is used to step down the 5V supply, providing a stable 3.3V output. The LDO regulator efficiently converts the voltage while minimizing power loss and ensuring a steady supply to the microcontroller. The HC-SR04 sensor, which requires 5V, is powered directly from the same 5V source to maintain proper functionality. A common ground is shared across all components to ensure reliable communication. Additionally, since the HC-SR04's Echo pin outputs a 5V signal, a voltage divider is used to safely step down the signal to 3.3V, preventing damage to the ESP32's GPIO while allowing accurate signal detection.

Requirements	**Verifications**
• The system must provide a stable 3.3V supply to the ESP32 microcontroller using an LDO regulator from a 5V	• Voltage Measurement: ○ Use a multimeter or oscilloscope to verify the LDO

input source. ● The HC-SR04 ultrasonic sensor must be powered directly from the 5V supply to ensure proper operation. ● The LDO regulator must efficiently convert 5V to 3.3V with minimal power loss. ● A common ground must be shared across all components to maintain reliable operation and signal integrity. ● The HC-SR04 Echo pin's 5V output must be safely stepped down to 3.3V using a voltage divider, preventing damage to the ESP32's GPIO. ● The system must maintain stable voltage levels under expected load conditions.	output is 3.3V ± 5% under different load conditions. ○ Measure the 5V supply to ensure it remains stable when powering the ESP32 and HC-SR04. ● Current Load Testing: ○ Measure the current draw of the ESP32 and HC-SR04 to ensure the LDO regulator can handle the expected load. ● Signal Integrity Testing: ○ Use an oscilloscope to check the Echo pin voltage before and after the voltage divider to confirm it steps down from 5V to 3.3V. ○ Verify that the stepped-down signal is correctly detected by the ESP32 GPIO.

2.3 System Software Logic & Requirements

2.3.1 Overview

This system implements real-time pitch-shifting of audio signals based on the ESP32-S3 chip. The system captures and outputs audio signals through the integrated ADC (Analog-to-Digital Converter) and DAC (Digital-to-Analog Converter) of the ESP32-S3. It performs pitch-shifting of the audio using a phase vocoder algorithm. The performer can dynamically adjust the pitch-shifting parameters by moving their hand to trigger the ultrasonic sensor installed on the guitar. The system also supports interaction and information display through a serial port or an LCD screen.

2.3.2 Interface

Audio Input Interface: Captures the analog audio signal of the electric guitar using the ADC interface of the ESP32-S3.
Audio Output Interface: Converts the processed audio signal to an analog signal through the DAC interface of the ESP32-S3 and outputs it to the guitar amplifier.
User Interaction Interface: Connects an LCD screen via UART or I2C to display information such as the current pitch shift value and device status.
Sensor Interface: Connects an ultrasonic sensor to capture gesture or distance data via GPIO, which is used to dynamically control pitch-shifting.

2.3.3 Design Decisions

Pitch-Shifting Algorithm Selection: The phase vocoder algorithm is used for pitch-shifting, which allows pitch adjustment while maintaining the original audio duration, making it suitable for real-time audio processing scenarios.
Audio Buffer Design: A ring buffer (RingBuffer) is used to manage audio data, ensuring the continuity of the audio stream and preventing data loss or overflow.
FFT Implementation: Custom FFT and IFFT functions are implemented for spectral analysis and synthesis to ensure the flexibility and scalability of the algorithm.
User Interaction Design: Supports parameter display and debugging through a serial port or an LCD screen.

2.3.4 Requirements & Verification

Requirements	**Verifications**
● Real-time pitch-shifting of audio input, supporting non-contact pitch control via gesture data captured by the ultrasonic sensor. ● Display of device status, including the current pitch shift value and system status, through an LCD screen or serial port.	● Functional Testing: Test the integrity and quality of audio input and output to ensure accurate pitch-shifting. Verify the function of adjusting pitch-shifting parameters through hand gestures. ● Performance Testing: Test the latency of audio processing to ensure the system meets real-time audio processing requirements.

	• User Testing: Validate the system's usability and functionality through user feedback to ensure it meets practical usage requirements.

2.4 Tolerance Analysis

The sonic sensor plays a crucial role in controlling the pitch shift by detecting the distance between the guitarist's hand and the sensor. Accurate and timely readings are essential for real-time pitch-shifting control. The sonic sensor's performance is highly dependent on distance measurement accuracy, which is usually ±1 cm under ideal conditions. A deviation larger than this could lead to inconsistent pitch shifts or delays in detecting changes in the guitar's position, making it difficult to achieve smooth pitch modulation. Furthermore, the response time of the sensor is critical for real-time interaction with the system. A delay in sensor updates could introduce latency in the pitch-shifting feedback loop, causing noticeable lag between the guitar player's action and the system's response.

To quantify the impact of measurement accuracy and timing drift, we'll use the sensor's distance measurement model. The sensor typically uses an ultrasonic wave to measure distance, where the time is given by:

$$d = \frac{v^{*}t}{2}$$

Where:

- d is the distance to the object,
- v is the speed of sound (approximately 343 m/s),
- t is the time of travel.

If the sensor has an accuracy of ±1 cm, this means that the error in distance measurement is approximately ±0.01 m.

$$\Delta t = \frac{2^{*}\Delta d}{v}$$

$$\Delta t = \frac{2^{*}(0.01)}{(343)} \approx 58.3\ \mu s$$

This time error could affect the precision of pitch-shifting calculations if it accumulates over time, especially in events of constant pitch shifting sequences where real-time updates are necessary. Even

small errors in time measurement could result in audio discrepancies between sensor updates and the microcontroller. To ensure minimal impact, the system should handle this error through calibration of the sensor or microcontroller's sensitivity. By adjusting the threshold of distance and pitch shifting, we can increase the tolerance of the sensor readings so even small changes made in distance will be ignored and avoid triggering any unwanted changes.

3 Cost & Schedule

3.1 Cost Analysis

3.1.1 Parts & Materials

The process of picking out parts was obtained through the My.ECE portal, ECE service shop, and online retailers.

3.1.2 Estimated Hours of Compensation

The members in the project are all Grainge College of Engineering students who are studying Electrical Engineering. The post-graduate average pay according to the Grainger College of Engineering website, the average starting salary for an Electrical Engineering graduate is $87,769 per year [5] which equates to $42.20 per hour.

Category **Estimated Hours**

	Eric (100)	**William (120)**	**Fan (95)**
Schematic Design	**0**	**30**	**0**
Software Design	**0**	**0**	**30**
Layout Design	**15**	**15**	**0**

Assembly & Troubleshooting	**55**	**55**	**50**
Documentation	**30**	**20**	**15**

Table: Estimated Hours

Using the hours in table above and the rated pay found through research, we can compute the estimated cost for overall labor as:

$42.20(Hourly Rate) * 315(Total Hours) = $4,960

Note that this is an averaged amount taking taxes into consideration.

3.1.3 Resources

Description	Part Number	Unit Price	Quantity	Total Cost	Vendor
ESP32-S3 Microcontroller	ESP32-S3-WROOM-1-N8	$4.95	2	$9.90	Link
ESP32-S3 Development Board	ESP32-S3-DEVKITC-1-N32R8V	$15.00	1	$15.00	Link
24 bit ADC	PCM1808PW	$1.91	2	$3.82	Link
32 bit DAC	PCM5102APW	$4.47	2	$8.94	Link
Voltage Regulator (3.3V)	TLV1117-33IDCYG3	$0.71	2	$1.42	Link
TVS Diodes	SP0503BAHTG	$0.63	2	$1.26	Link
Op-Amps	TLV172IDCKT	$1.34	5	$6.70	Link
Ultrasonic Sensor	HC-SR04	$3.40	2	$6.80	Link
AC Adapter (5V)		$7.99	1	$7.99	Link
DC Barrel Jack		$6.99	1	$6.99	Link

LED (Variety)		$8.99	1	$8.99	Link
Resistor (10k)		$0.30	10	$3.00	ECEB
Resistor (1k)		$0.30	10	$3.00	ECEB
Capacitor (10uF)		$0.20	10	$2.00	ECEB
Capacitor (1uF)		$0.20	5	$1.00	ECEB
Total Cost				$86.81	

3.1.4 Total Cost

The total cost of expected labor and for parts will be $5046.81.

3.2 Schedule

Date	Task	Group Member
Week 2/24	• Finalize PCB schematic and begin drafting layout of design • Order parts for breadboard prototype • Research algorithms for software implementation	William Zhengjie
Week 3/03	• Construct breadboard prototype • Finalize software interface and backend • Begin PCB layout design	William Zhengjie Eric
Week 3/10	• Finalize PCB layout design • Submit audit for PCB order	Eric Zhengjie

Week 3/17	Spring Break	N/A
Week 3/24	• Assemble PCB	William Eric
Week 3/31	• Assemble PCB	Eric Zhengjie
Week 4/7	• Assemble PCB	William Zhengjie
Week 4/14	• Unit test power system • Unit test analog to digital input system • Unit test digital to analog output system	William Zhengjie Eric
Week 4/21	• Program Board • Conduct test with guitar	William Zhengjie Eric
Week 4/28	• Troubleshoot/Refine • Add additional features (if time allows)	William Zhengjie Eric

4 Ethics & Safety

4.1 Ethics

Ethical Considerations

Our digital pitch-shifting guitar project must follow ethical guidelines set by the IEEE and ACM Codes of Ethics to ensure fairness, responsibility, and safety in both its development and use.

1. Intellectual Property and Fair Use
    - Since our project may involve existing signal processing techniques or open-source software, we must ensure that we properly cite and follow all licensing agreements.
    - The IEEE Code of Ethics emphasizes honesty in authorship and respect for intellectual property. To follow this, we will credit all sources, avoid plagiarism, and ensure that any external components we use are legally allowed.
2. User Safety and Responsible Design
    - High audio output levels could damage hearing if not properly managed. To prevent this, we will limit volume levels, implement safety features, and test the device under different conditions.
    - The ACM Code of Ethics encourages designing systems that improve quality of life while minimizing harm. Our design will ensure that users are protected from sudden loud noises or unintended hardware malfunctions.
3. Accessibility and Inclusivity

- Our device should be usable by all musicians, regardless of experience level. We will provide clear documentation, an easy-to-use interface, and setup guides to help users understand how to operate it safely and effectively.
- Ensuring inclusivity aligns with the ACM Code of Ethics, which states that computing professionals should make technology accessible to a wide audience.

4.2 Safety

Our project includes electronic and software components, each of which presents potential safety risks. To ensure safe operation, we will follow industry standards and regulations while implementing features to minimize hazards.

4.2.1 Circuit Protection Safety

Our device operates on low-voltage DC power, but protection against short circuits, overvoltage, and overheating is still essential to prevent damage or injury. To mitigate these risks, fuses, voltage regulators, and a physical containment box are used to safeguard both the device and the user. Additionally, capacitors and resistors are strategically placed in the circuit to help filter voltage spikes, stabilize power delivery, and limit excessive current flow, further reducing potential electrical hazards.

4.2.2 Personal Health Safety

Prolonged exposure to loud audio can lead to permanent hearing damage, with OSHA Noise Exposure Standards indicating that sounds above 85 dB can be hazardous over extended periods. To mitigate this risk, our design will incorporate built-in volume limits and gain control features to prevent excessive sound levels from being output. Additionally, warning indicators or visual

feedback may be integrated to alert users when sound levels approach unsafe thresholds. Beyond device features, personal safety measures will be emphasized in the user manual, educating musicians on the risks of prolonged exposure to high sound levels. Recommendations may include taking regular breaks, using hearing protection (e.g., earplugs), and maintaining safe listening distances. These precautions will help ensure that users can enjoy enhanced audio experiences without compromising their long-term hearing health.

5 References

1. SparkFun Electronics, "Ultrasonic Distance Sensor - HC-SR04," [Online]. Available: https://www.sparkfun.com/ultrasonic-distance-sensor-hc-sr04.html. [Accessed: 13-Feb-2025].
2. Espressif Systems, *ESP32-S3 Datasheet*, [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf. [Accessed: 13-Feb-2025].
3. Espressif Systems, *ESP-IDF I2S API Reference (ESP32-S3)*, [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-reference/peripherals/i2s.html. [Accessed: 13-Feb-2025].
4. Digi-Key Electronics, *What Is the I2S Communication Protocol?*, [Online]. Available: https://www.digikey.com/en/maker/tutorials/2023/what-is-the-i2s-communication-protocol. [Accessed: 13-Feb-2025].
5. "Salary Averages." Electrical & Computer Engineering at Illinois, University of Illinois Urbana-Champaign, https://ece.illinois.edu/admissions/why-ece/salary-averages. [Accessed 4-Mar-2025].
6. Ellis, D. (2010, October 26). *A phase vocoder in Matlab*. Columbia University. Retrieved February 20, 2025, from https://www.ee.columbia.edu/~dpwe/resources/matlab/pvoc/
7. Johnston, P. (2022, December 22). *Creating a circular buffer in C and C++*. Embedded Artistry. Retrieved March 2, 2025, from https://embeddedartistry.com/blog/2017/05/17/creating-a-circular-buffer-in-c-and-c/