Saketh Aripirala

Effects of Bass Guitar Pickups on Pitch Detection and Pitch Shifting

Metropolia University of Applied Sciences

Bachelor of Engineering

Electronics

Bachelor’s Thesis

1 March 2021

Abstract

Author: Saketh Aripirala

Title: Effects of Bass Pickups on Pitch Detection and Shifting using Digital Signal Processing

Number of Pages: xx pages + x appendices

Date: 1 March 2021

Degree: Bachelor of Engineering

Degree Program: Electronics

Professional Major: Bachelor’s Thesis

Supervisors: Heikki Valmu, Principal Lecturer

Juha Kivekäs, Technical Supervisor

Keywords: Digital Signal Processing, Python, Bass Guitars

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Contents

List of Abbreviations

[1 Introduction 1](#_Toc68862901)

[2 Chapter Heading 1](#_Toc68862902)

[2.1 Subheading 1](#_Toc68862903)

[2.2 Subheading 3](#_Toc68862904)

[3 Document Structure 3](#_Toc68862905)

[3.1 Tables 3](#_Toc68862906)

[3.1.1 How to Create Tables in Word 4](#_Toc68862907)

[3.1.2 Secondary Subheading 5](#_Toc68862908)

[3.2 Quotes 5](#_Toc68862909)

[3.3 Lists 5](#_Toc68862910)

[3.4 Listings 6](#_Toc68862911)

[3.5 Formulas 6](#_Toc68862912)

[4 Alternative Text 7](#_Toc68862913)

[4.1 How to Add Alternative Text to Figures 7](#_Toc68862914)

[4.2 How to Add Alternative Text to Tables 8](#_Toc68862915)

[5 Document Accessibility 8](#_Toc68862916)

[5.1 Finish the Document Properties 8](#_Toc68862917)

[5.2 Check the Accessibility of Your Thesis 9](#_Toc68862918)

[5.3 Save Word Document as Accessible PDF 10](#_Toc68862919)

[References 12](#_Toc68862920)

Appendices

Appendix 1: Appendix name

Appendix 2: Appendix name

List of Abbreviations

DSP: Digital Signal Processing

FFT: Fast Fourier Transform

DFT: Discrete Fourier Transform

F0: Fundamental Frequency

ACF: Autocorrelation Function

CMNDF: Cumulative Mean Normalized Difference Function

Op Amp: Operational Amplifier

# Introduction

In the world of digital audio processing, pitch manipulation effects and sound synthesis are commonly researched subjects and are widely used by musicians to alter and produce new sounds. The origins of sound synthesizers traces back to the early 20th century, where analog oscillators are utilized to produce pure tone sounds such as sine, square, and sawtooth waves. In more modern and robust applications, synthesis uses digital signal processing and hybrid systems to produce more complex musical tones. Similarly, pitch manipulation is a very popularly used tool to modify the perceived pitch of an instrument or speech. Most common styles of perceived pitch manipulation are often used to shift the signal to different musical intervals or alter the formants. An octaver is widely used on instruments to shift the signal down an interval of an octave, essentially halving the frequency of the signal.

With more emerging audio technologies, the signal of a stringed instrument can be used to synthesize pure or complex tones by tracking the pitch of the note played. Although, it may seem trivial to track the pitch or fundamental frequency of an instrument; in reality, there are complexities stemmed from the timbre (tonal quality of a sound [1].) and the nature of the instrument that cause the tracking errors or inconsistencies. Comparable issues occur when the pitch is shifted and worsened with certain cases where an error cause perceivable auditory discrepancy.

By understanding the fundamentals of guitar pickup technology, a much wider comprehension of the role pickups play in the harmonic contents of the signal can be achieved. Moreover, methods to mitigate errors in these algorithms can also be investigated.

To test the role of pickup types in these errors and the overall functionality of the algorithms, a test bass guitar containing two specific types of pickups was utilized: a generic humbucker pickup in a split-coil configuration and an Ernie Ball piezo bridge pickup. To test these pickups in individual and mix configurations, a debugging PCB was designed using Altium Designer, an ECAD software. The primary test points include various heights, positions, and configurations of the pickups. Using the data in python, a programming language widely used for data analytics, correlation functions are implemented to study the changes in the fundamental frequency tracking stability, errors, and phase changes. Lastly, the analysis of the harmonic contents in the signal and the testing method is validated using Sonic Visualizer.

The findings of the research aid Darkglass Electronics, a Finnish bass guitar accessory manufacturer, in pursuing technology and methods to implement bass guitar effects embedded into an instrument. The algorithms used to acquire the test data are effects made in-house by Darkglass Electronics, which include a faithful modelling of an analog octaver, a digital hybrid octaver, and a bass guitar synthesizer.

# Fundamental Theories and Concepts

To understand the errors conditions and research goals, it is quite essential to have a solid comprehension of the fundamentals of the implementation of the algorithms, guitar pickup technology, and digital signal processing and spectral analysis. The subsequent section covers the necessary prerequisites.

## Digital Signal Processing and Waves

Digital signal processing is a commonly used technique to analyze and alter real world signals such as sounds, measurements, and data. Analog signals are discretized digitally using Analog-to-Digital converts and using fundamental mathematical functions, the data is manipulated [2.] To discretize analog signals, the signal is sampled frequent instances. The rate at which these instances are captured is known as the sampling frequency (*Fs*)[3*.*] According to the Nyquist-Shannon sampling theorem, an analog signal can be accurately reconstructed only if the sampling frequency is more than twice the maximum frequency of the sample [4]. Equation (1) represents the mathematical form of the Nyquist-Shannon sampling theorem:

|  |  |
| --- | --- |
|  | (1) |

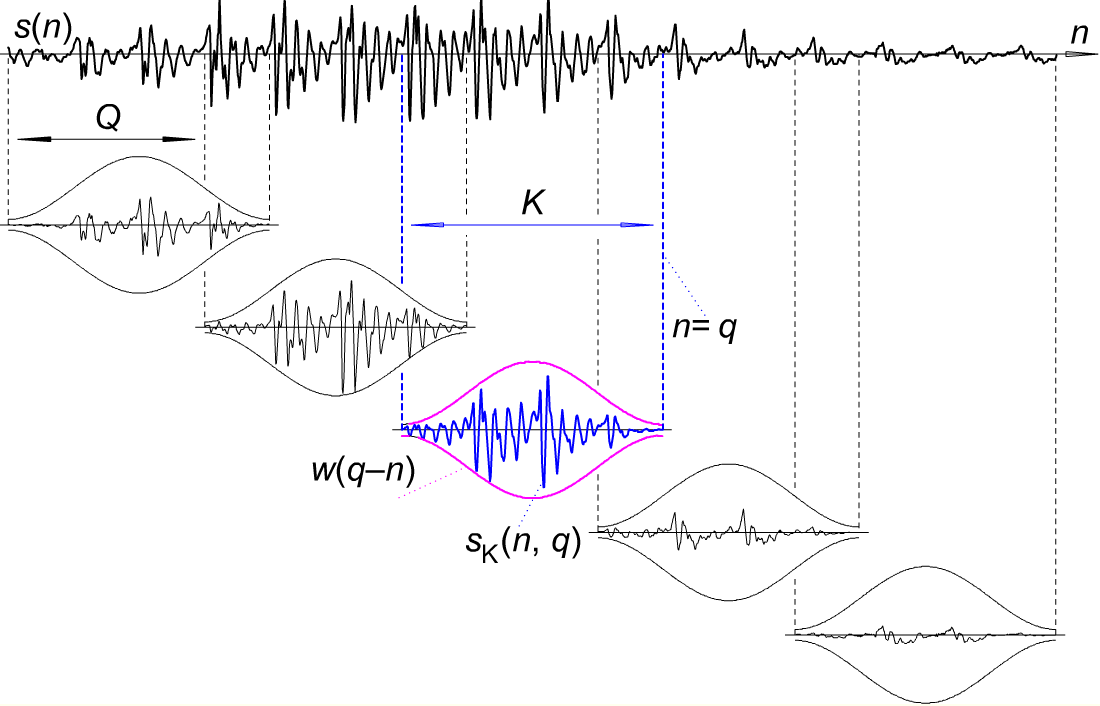
An important concept in Digital Signal Processing is windowing and hop size. Windowing divides a signal into smaller intervals of signal for which the processing is performed. Typically, windows are overlapped after each other; the number of samples in non-overlapping regions of the window is called the hop size [5.] Figure 1 depicts windowing and hop size for an audio sample.  
  


Figure 1. Windowing and Hop size. Q denotes the hop size and K represents the window length [6].

In spectral analysis, the Fourier Transform of a signal is performed to calculate the magnitude of each frequency component present in a signal. The Fourier Transform is translated into DSP via the discretized and sample based Discrete Fourier Transform (DFT). The mathematical implementation of the Fourier Transform and DFT is shown in Equation (2) and (3):

Let be a random function. Then the Fourier Transform of the function is given as follows:

|  |  |
| --- | --- |
|  | (2) |

Where is the complex imaginary unit and is the frequency.

Then the DFT for number of samples is given by:

|  |  |
| --- | --- |
|  | (3) |

Each value of denotes a frequency bin. The magnitude and phase of the frequency bin is calculated by using Equations (4) and (5):

Let be a complex number where is the imaginary unit.

|  |  |
| --- | --- |
|  | (4) |
|  | (5) |

An algorithm that improves the implementation of the DFT is the Fast Fourier Transform (FFT) algorithm. It requires fewer computational steps to calculate the DFT.

The spectral information calculated using DFT can be represented by graphing the magnitude for each frequency bin or spectrograms. The graphing method provides the harmonic contents of the signal as a function of its magnitude, whereas a spectrogram provides the magnitude of the harmonic contents, or frequency bins, as a function of time. Figure 2 and Figure 3 contain the graphing and spectrogram representations.

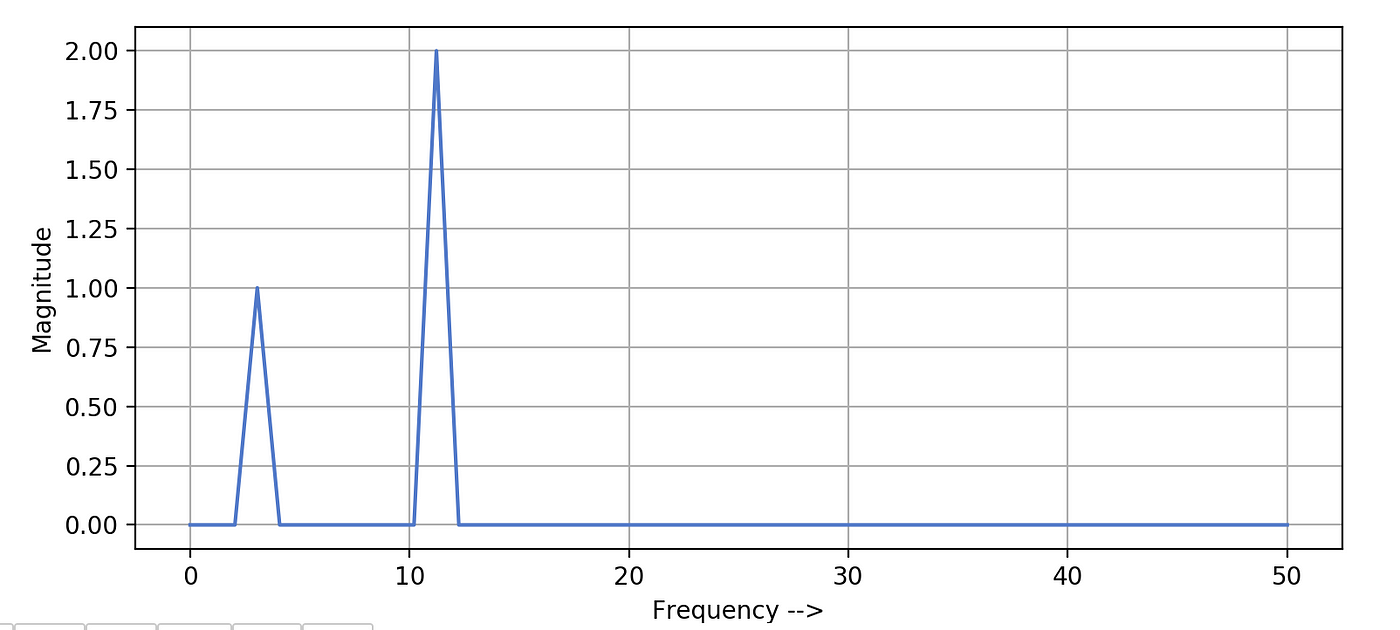


Figure 2. Graphing Frequency as a Function of Magnitude [7].

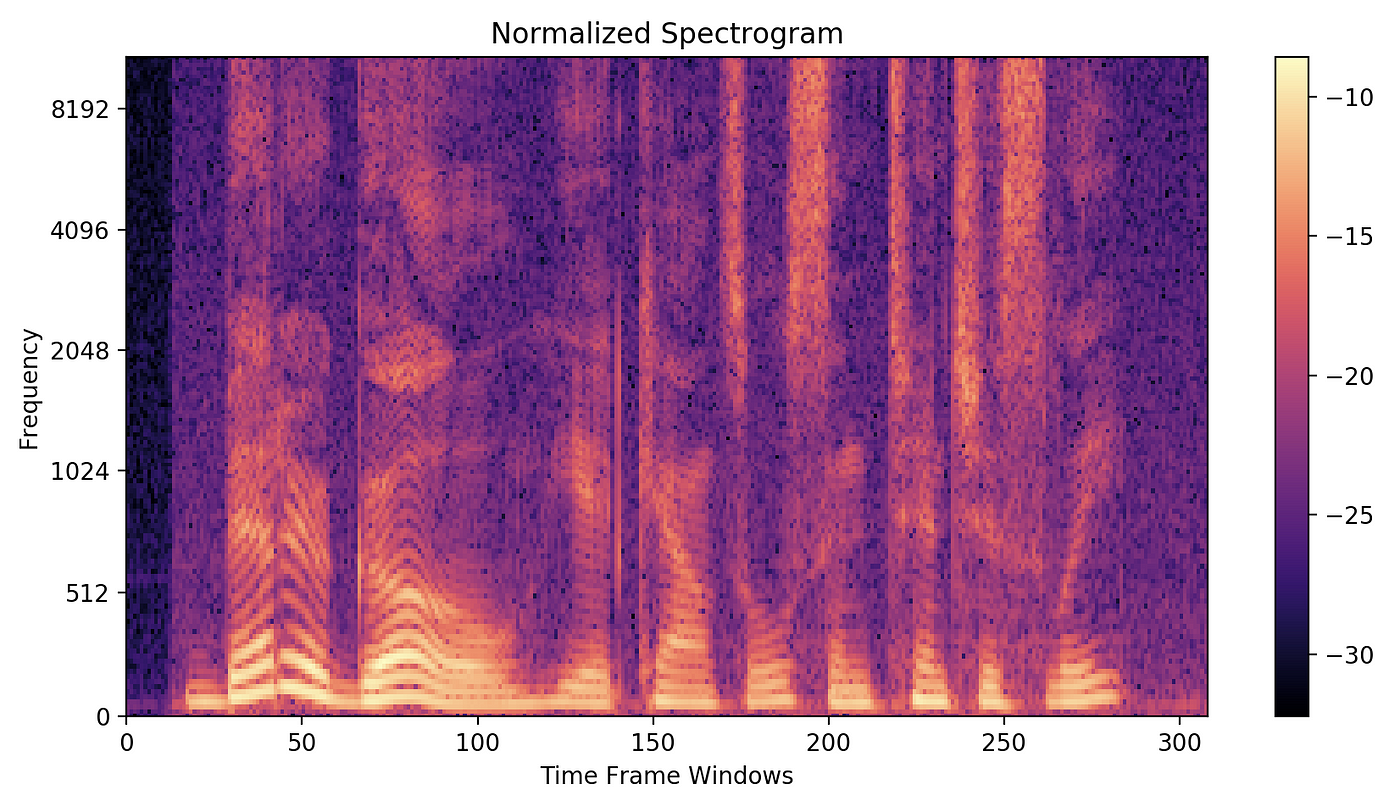


Figure 3. Spectrogram [7].

The lowest resonant frequency component present in the signal is known as the fundamental frequency. In a periodic signal, multiples of the fundamental frequency are known as the harmonics or overtones [8.] The relationship between the fundamental frequency and subsequent harmonics is shown by Equation (6).

|  |  |
| --- | --- |
|  | (6) |

Where is the 1st overtone (or 2nd harmonic) and is the nth harmonic. The timbre of the sound is unique for different sounds due to the varying magnitudes of the harmonics. Figure 4 describes the relationship between the fundamental frequency and the subsequent harmonics. The period of the wave doubles for each harmonic i.e., frequency is twice. In music, it is generally accepted that the perceived pitch is the fundamental frequency of the signal.

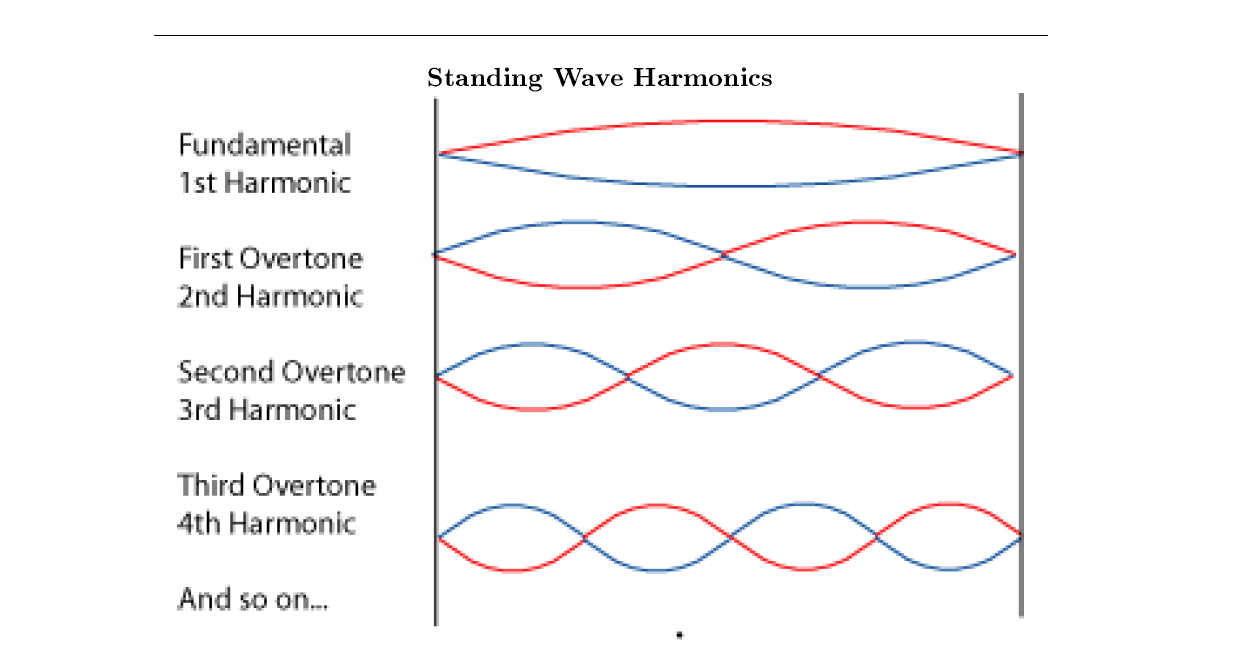


Figure 4. Relationship between Fundamental Frequency and Subsequent Harmonics [9].

During spectral or harmonic analysis, it is quite valuable to apply the windowing when calculating the DFT of data to optimize accuracy or performance. The windows are often truncated by applying various window functions such that the samples taper to zero at the start and the end of the window. It is beneficial to apply windowing functions to the window when performing spectral analysis to avoid the effect of spectral leakage. The phenomenon of spectral leakage causes the magnitude information of a frequency bin affects the other frequency bin [10]. This is often caused due to the overlapping windows causing discontinuities in the signal, as shown in Figure 5.



Figure 5. Discontinuities Produced due to No Window Function [10].

## YIN Algorithm

As initially established, estimating the fundamental frequency is a non-trivial subject due to the varying harmonic contents of a signal. The harmonic contents play a large role in the transient changes and time domain contents of the signal. Most fundamental frequency estimations fail to account for these changes. The YIN algorithm by Alain de Cheveigné and Hideki Kawhara [13] is a robust method that improves existing implementations for fundamental frequency estimations. The three stages of the YIN algorithm are as follows:

1. Autocorrelation
2. Difference Function
3. Cumulative Mean Normalized Difference Function

The YIN algorithm is used in Darkglass’ in-house bass guitar synthesizer, which detects the frequency of the bass guitar signal and produces a synthesized sound. Moreover, it is also implemented as a plugin in Sonic Visualizer to help validate the accuracy of the pitch correlation tool created for the research.

### Autocorrelation Function

The Autocorrelation Function (ACF) is commonly used in statistics and signal processing for measuring the correlation between the signal and its time delayed variant [11]. The ACF of a periodic signal always returns a perfect correlation and the smallest time delay denotes the period of the signal [12]. The inverse of the time delay is an estimate of the frequency of the signal. Mathematical representation of the ACF function is presented in Equation (7):

Let be a periodic signal, then its ACF with delay is:

|  |  |
| --- | --- |
|  | (7) |

The equation can be applied for a discrete signal with a window width of and time delay , hence Equation (7) can be modified; as shown in below in Equation (8):

|  |  |
| --- | --- |
|  | (8) |

The ACF holds for estimating the fundamental frequency of pure tones; but for varying periods in a signal, the ACF fails and produces errors. In Figure 6 the autocorrelation of a sample signal can be observed.

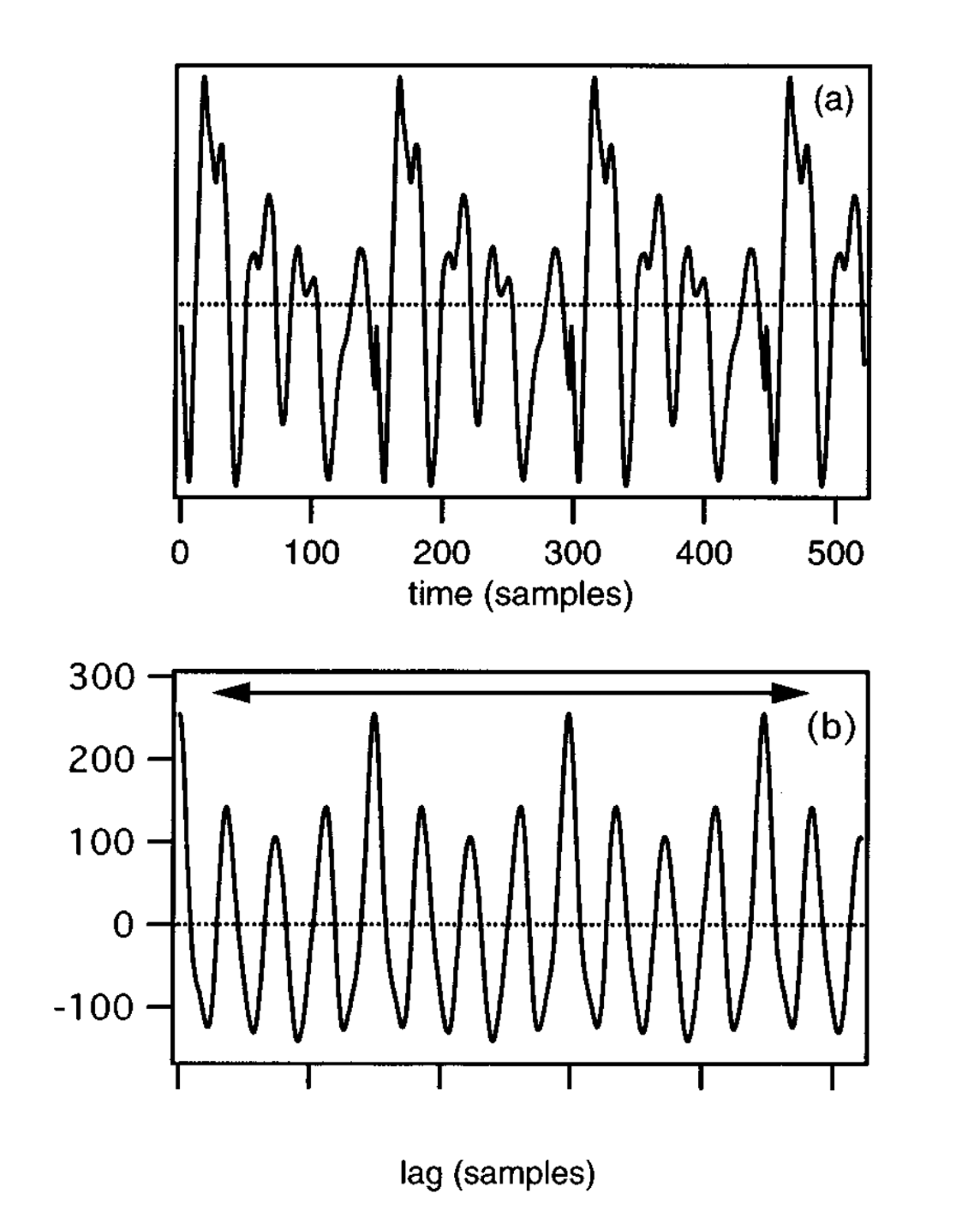


Figure 6. (a) Sample Signal. (b) Autocorrelation of the Sample Signal [13].

Listing 1 contains the python script to find the frequency of a sine wave using the ACF. The script utilizes a window width of 200 samples and generates a 5-second-long sine wave with a sampling frequency of 500.

def f(x):

f\_0 = 1

return np.sin(x \* np.pi \* 2 \* f\_0)

#Generates a sine wave with frequency of 1 Hz

def ACF(f, W, t, lag):

return np.sum(f[t : t + W] \* f[lag + t : lag + t + W])

def returnACF(f, W, t, fs, bounds):

ACFv = [ACF(f, W, t, i) for i in range(\*bounds)]

sample = np.argmax(ACFv) + bounds[0]

return fs / sample

Listing 1. Python script to create a sine wave and find its frequency using the ACF.

As expected, the ACF holds and returns a result of 0.9823 Hz, which can be rounded up to 1 Hz. Applying an exponentially decaying envelope to the sine signal as shown in Listing 2 proves that the ACF does not hold for decaying signals with varying amplitude or fluctuating periods. Using the same parameters, the ACF returns a frequency estimate of 25 Hz.

def f(x):

f\_0 = 1

envelope = lambda x: np.exp(-x)

return np.sin(x \* np.pi \* 2 \* f\_0) \* envelope(x)

Listing 2. Modified Sine Wave with a Decaying Envelope.

The sine wave with the envelope is shown in Figure 7 below.

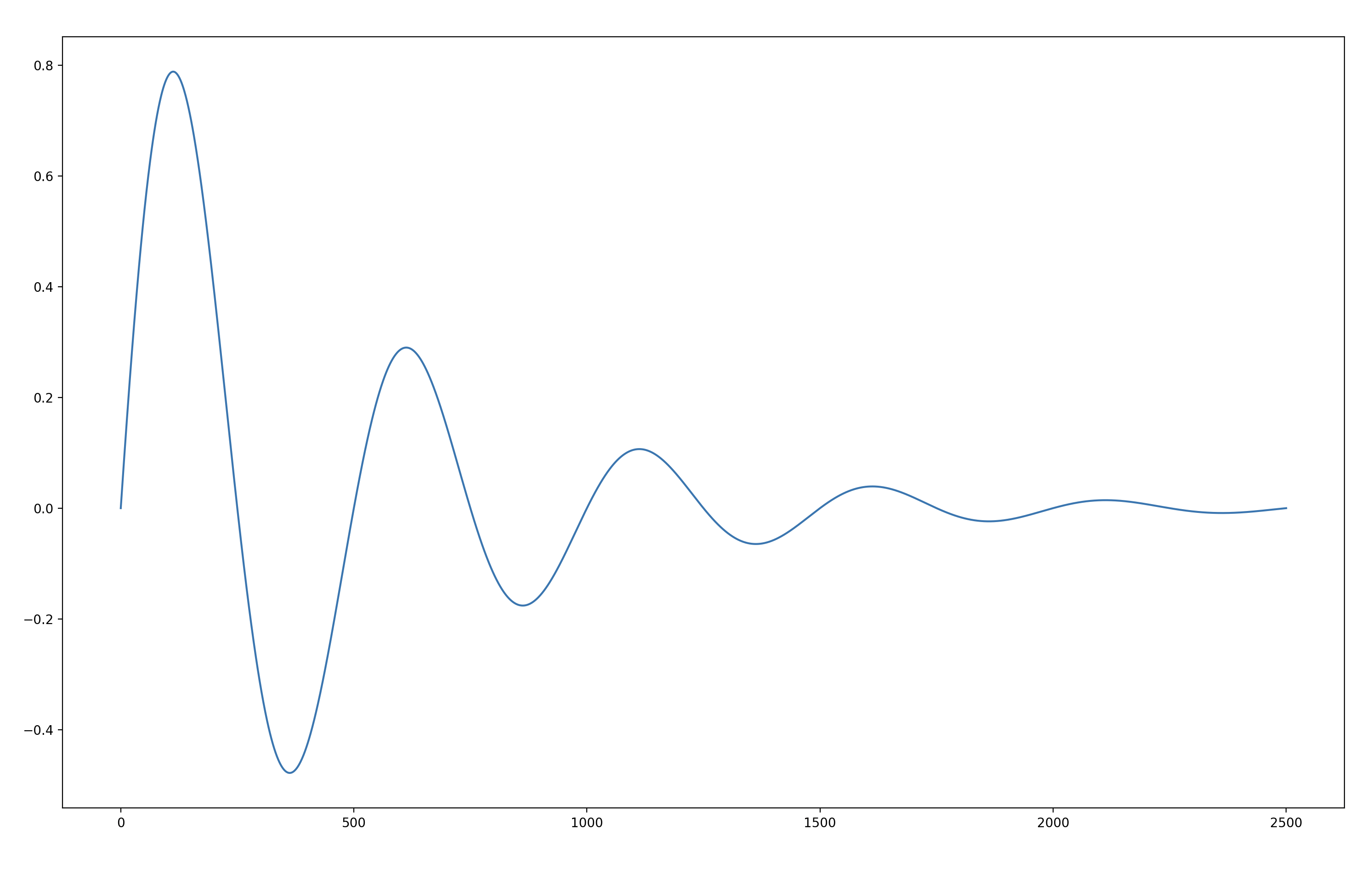


Figure 7. Sine Wave with an Exponentially Decaying Envelope.

### Difference Function

For a shift invariant system, it is true that when a discrete signal with period is shifted by a time constant the output is equally shifted. Using this property, the following Equation (9) also holds true [13.]

|  |  |
| --- | --- |
|  | (9) |

Similarly, by squaring the sums of Equation, the smallest time shift where the difference is zero gives the period of the signal. Equation (10) below describes the difference function for a discrete signal using a sampling window of and time shift [13.]

|  |  |
| --- | --- |
|  | (10) |

The difference function does not improve over the ACF in terms of estimating the fundamental frequency, but rather reduces the overall errors produced by the ACF. An elucidation for this is that the ACF is much more sensitive to amplitude changes therefore causing the ACF values to increase with growing amplitudes, as observed in Figure 6. Moreover, the difference function can be described in terms of the ACF as shown by Equation (11) [13.]

|  |  |
| --- | --- |
|  | (11) |

The same is implemented in a python function as shown in Listing 3.

def DF(f, W, t, lag):

return ACF(f, W, t, 0) + ACF(f, W, t + lag, 0) - (2 \* ACF(f, W, t, lag))

Listing 3. The Difference Function Implemented in terms of the ACF using Python.

Due to the difference function being an intermediate step of the YIN algorithm, the values of the difference function being used for estimating the fundamental frequency bears no value despite producing lower errors than the ACF. Hence, it was not tested using python.

### Cumulative Mean Normalized Difference Function

The cumulative mean normalized difference function (CMNDF) is the final stage of the YIN algorithm. The presence of the 2nd harmonic causes the difference function to produce zero lag regions, hence inducing errors [13]. The CMNDF avoids these zero lag regions and improves upon the difference function. Equation (12) below shows the CMNDF:

|  |  |
| --- | --- |
|  | (12) |

The function accomplishes lower errors by dividing the preceding difference function value over its average of shorter lag values. To further reduce errors, using an absolute minimum threshold for the lag values is useful since if no lag values are found, the function can default to the absolute threshold [13.]

Using the parameters from Listing 1 and the decaying sine wave, the CMNDF implementation and F0 detection implementation in python is presented in Listing 4 below.

def CMNDF(f, W, t, lag):

if lag == 0:

return 1

return DF(f, W, t, lag) / np.sum([DF(f, W, t, j+1) for j in range(lag)]) \* lag

def detect\_pitch(f, W, t, fs, bounds, thresh = 0.1):

CMNDF\_vals = [CMNDF(f, W, t, i) for i in range(\*bounds)]

sample = None

for i, val in enumerate(CMNDF\_vals):

if val < thresh:

sample = i + bounds[0]

break

if sample is None:

sample = np.argmin(CMNDF\_vals) + bounds[0]

return fs / sample

Listing 4. CMNDF and Fundamental Frequency Estimation.

Applying the detect pitch function on the decaying sine wave returns a F0 estimation of 1.002 Hz, which is 0.2% above the exact F0 of 1 Hz. Furthermore, the recommended threshold is 0.1 [13]; increasing the minimum threshold to 1 returns an estimate of 1.36 Hz, consequently increasing the error. Introducing bounds to the search range of the lag value is also beneficial to the processing time and improves the accuracy of the algorithm [13].

## Octaver Algorithm and Model

The octaver is a signal processing effect that shifts a signal down a musical interval of an octave, which leads to the signal’s frequency to be halved. Octavers are commonly used with guitars and basses to produce sub-bass frequencies. Furthermore, the octave down signal is also typically mixed with other types of effects such as distortion. Octavers are popularly used in the pedal format, but digital versions are also available and largely utilized. The digital formats provide the advantage of polyphonic processing; allowing a signal containing multiple musical intervals to be shifted. Whereas, for analog octavers, the processing is strictly monophonic.

The working principle of an octaver is to mute the signal at every second cycle of the period signal. To achieve the cyclic muting, the peak of the signal needs to be detected and every other peak of the signal is considered as a candidate for the mute control circuit. As previously established, audio signals do not have perfect periodicity and can have varying amplitudes. Due to this, the mute control can have varied periods and cause discrepancies in the resulting audio signal. The block diagram of an analog octaver is shown in Figure below.

## Pickup Fundamentals and Technology

# Testing Prerequisites and Methods

The ideal behavior of the pickup is to produce the least number of errors with the octaver models and the YIN algorithm-based bass guitar synthesizer. It is necessary to understand the harmonic contents of the signals produced by the two pickups and compare the types of errors produced. The subsequent section covers the testing methods for the pickups and requirements.

## Debugging PCB and Bass Modifications

To test the piezo and humbucker pickups, a generic bass guitar was modified to house both pickups. The bass ideally is setup such the following criteria are met:

1. Produces clean signal from both pickups.
2. Maintains original gain of both pickups and has an active circuitry.
3. Does not get affected by power supply noise.
4. Readily available debugging data.
5. No deviation in intonation for each string.

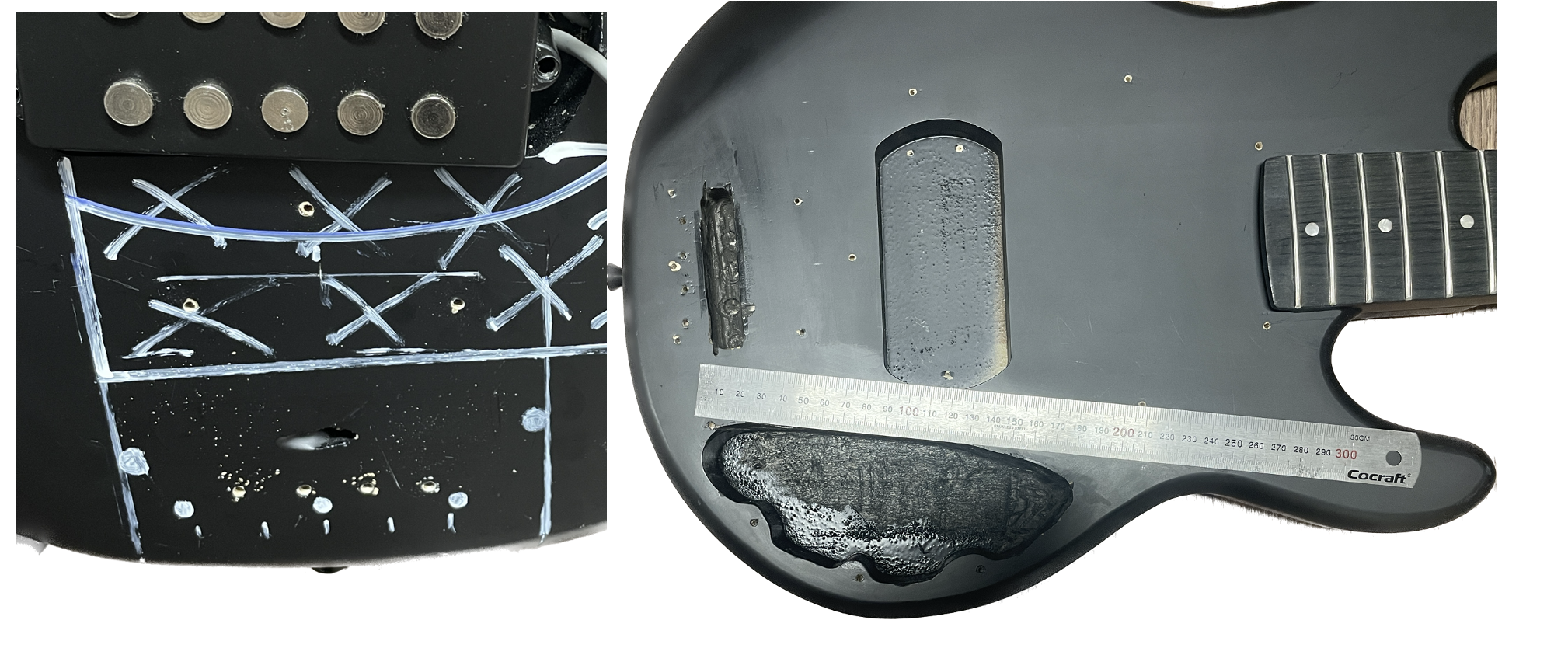
In [fig] below the generic bass guitar is presented.

A white electric guitar

Description automatically generated with medium confidence

Fig

The piezo pickup is housed in the bridge of the bass guitar, therefore requiring the original bridge to be replaced. It is essential for the original string tension to be maintained since the neck of the bass guitar could arc and cause playability issues, or worse, tuning instability. The position of the saddles on the bridge were marked, and the piezo bridge was placed in a similar position. Furthermore, the original cavity for the electronics was required to be enlarged for the debugging PCB to be accommodated. Similarly, an additional cavity was required for routing the cables for the piezo pickup. [Fig] below contains the cavities and bridge position markings.



Fig

Due to spatial restrictions in the PCB cavity, the DC jack for the power supply and audio output jack could not be accommodated. To overcome this, two mounting holes were made. In [Fig] below the mounting holes are shown. The output jack’s position was crucial due to the location of the potentiometer body, which is connected to ground.

## Python Testing Script and Sonic Visualizer

## Test Data

## Considerations

# Results

## Data Correlation

## Error Cases and Types

## Pickup Types and Effects

# Discussion

# Conclusion

References

* 1. <https://en.wikipedia.org/wiki/Timbre#:~:text=In%20music%2C%20timbre%20(%2F%CB%88,choir%20voices%20and%20musical%20instruments>.
  2. <https://www.analog.com/en/design-center/landing-pages/001/beginners-guide-to-dsp.html>
  3. <https://www.analog.com/en/design-center/glossary/sampling-rate.html>
  4. <http://musicweb.ucsd.edu/~trsmyth/digitalAudio171/Nyquist_Sampling_Theorem.html>
  5. <https://towardsdatascience.com/all-you-need-to-know-to-start-speech-processing-with-deep-learning-102c916edf62#:~:text=Window%20length%20is%20the%20length,portion%20of%20the%20window%20length>. Harsh Maheshwari
  6. <https://dsp.stackexchange.com/questions/40784/what-is-the-relation-between-windowing-and-hopping-in-audio-dsp> Laurent Duval
  7. <https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520> Karthik Chaudary
  8. <http://hyperphysics.phy-astr.gsu.edu/hbase/Waves/funhar.html>
  9. <https://learncigarboxguitar.com/content/standing-waves-and-harmonic-series>
  10. <https://www.physik.uzh.ch/local/teaching/SPI301/LV-2015-Help/lvanlsconcepts.chm/Spectral_Leakage.html#:~:text=In%20spectral%20leakage%2C%20the%20energy,exactly%20repeats%20throughout%20all%20time>.
  11. <https://www.tutorialspoint.com/autocorrelation-function-and-its-properties> Manish Kumar Saini
  12. <https://pages.mtu.edu/~suits/autocorrelation.html>
  13. YIN Alain de Cheveigné and Hideki Kawahara

Use one of the referencing systems below. Remove the one that you do not use.

Harvard (author-date) system:

The reference list entries need to be in alphabetical order according to the last name of the author mentioned first in the list of authors.

Davies, Barbara; Jameson, Peter & Smith, John. 2013. Advanced economics. Oxford: Oxford University Press.

Mitchell, John Arnold & Thomson, Magdalena. 2017. A guide to citation. London: London Publishings.

Vancouver (numbering) system:

1. Mitchell, John Arnold & Thomson, Magdalena. 2017. A guide to citation. London: London Publishings.
2. Davies, Barbara; Jameson, Peter & Smith, John. 2013. Advanced economics. Oxford: Oxford University Press.

Title of the Appendix

The appendices are not inserted into the table of contents automatically. Instead, they must be mentioned separately just below the auto-generated part of the table of contents.

Should you insert figures or tables into an appendix, Word numbers them automatically as if they were in the thesis main section. Fix the numbering of figures and tables in the appendixes manually so that the numbering starts from one in each appendix.

Below are instructions for adding and removing attachments so that the headers remain correct.

Instructions for adding a new attachment:

1. Move the cursor to the end of the last existing attachment page.
2. Choose the “Page Layout” tab. From the ribbon select “Page Break” / “Next Page” under “Section Breaks”. This completes the printing of the new attachment, but the number in its header is not correct.
3. Double tap the header of the new attachment page with the wrong attachment number. If the “Link to previous” option is selected in the ribbon, press that button so that the option is no longer selected.
4. Please correct the attachment number.

Instructions for removing an unnecessary attachment:

1. First select the entire attached page and press “Delete” to delete its contents.
2. When you are at the beginning of the attachment page you have emptied (see figure), double tap the header of the blank attachment page and press the “Link to Previous” button on the ribbon. The dialogue box appears (figure 1). Answer “Yes”.

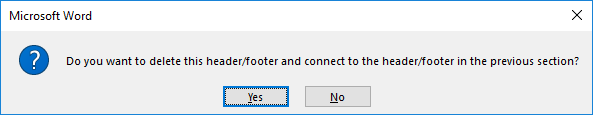


Figure 1. Confirmation of deleting a header.

1. From the “Home” tab, toggle hidden characters if they are not visible:Piilomerkki painikkeen kuvake
2. Remove the section break before the unnecessary attachment (figure 2).

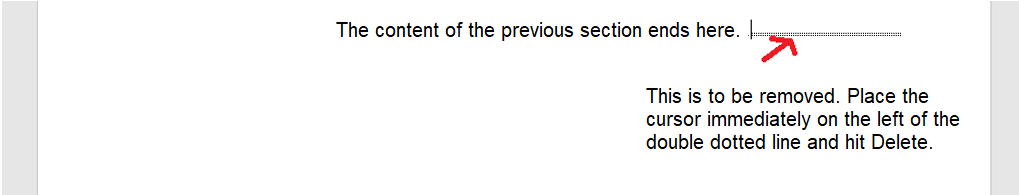


Figure 2. Removal of a section break.

.

Title of the Appendix

Content of the appendix is placed here