Critical Design Review

5/20/2020

MAE 157 Final Project

SeanLiam McCarthy

Project Overview

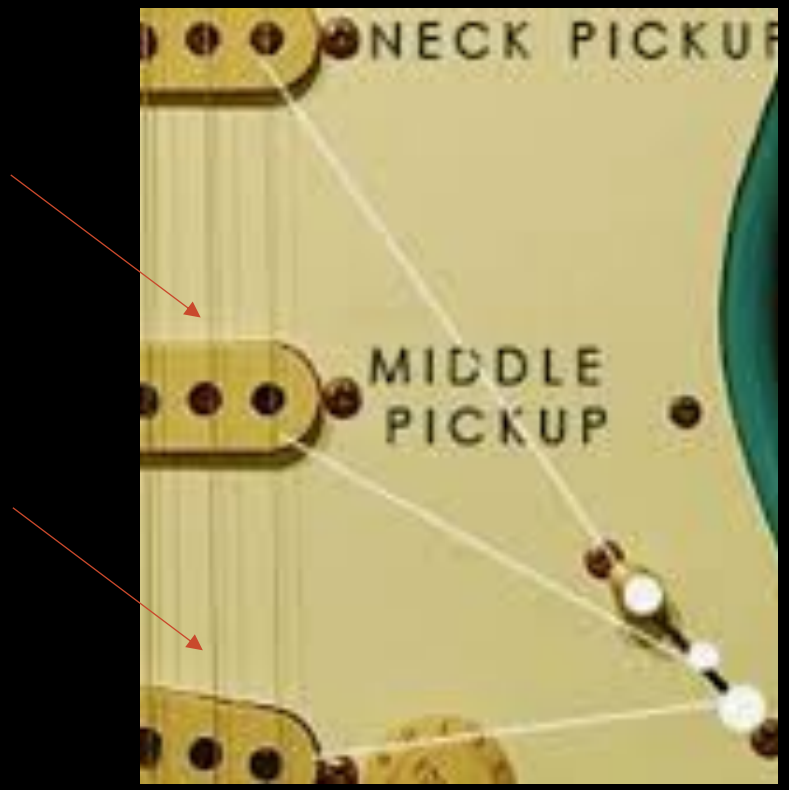
Theory and Concepts

Requirements

Concept of Operations

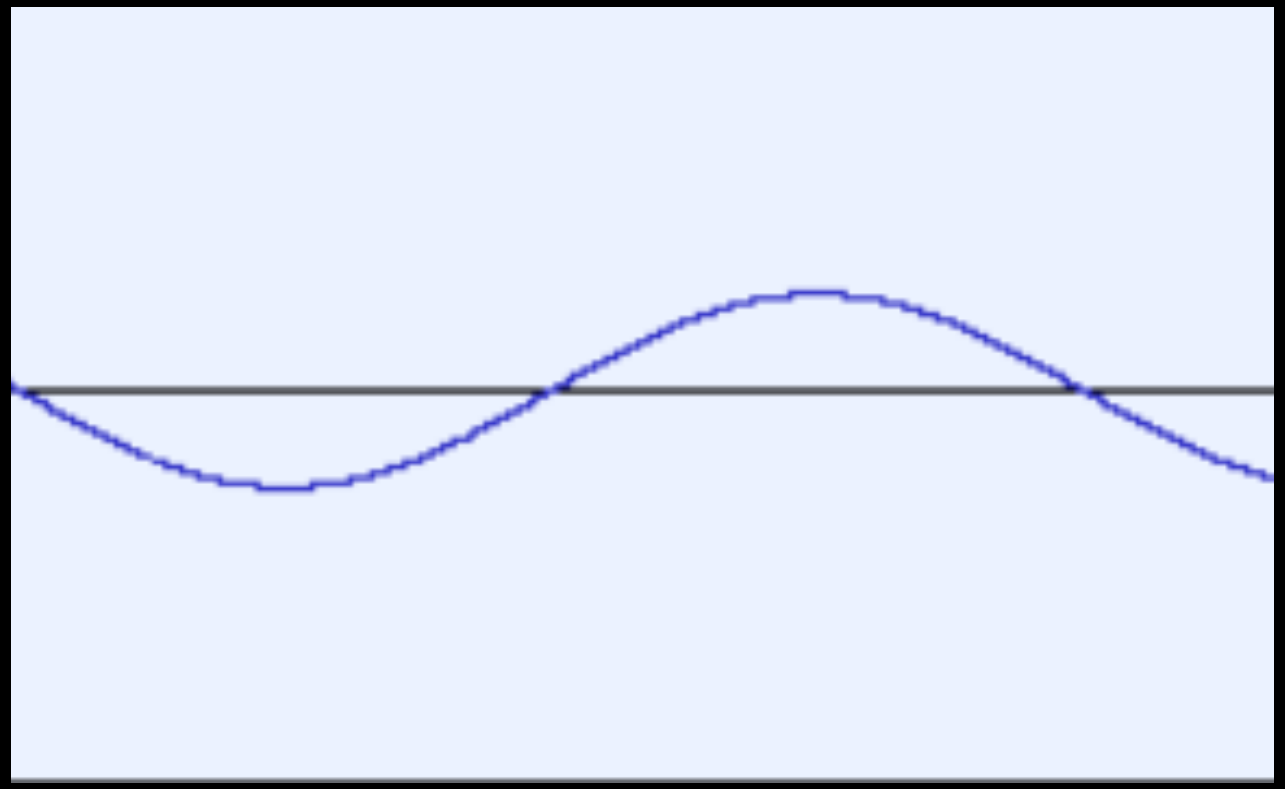
Expected Results

Sound to Electricity



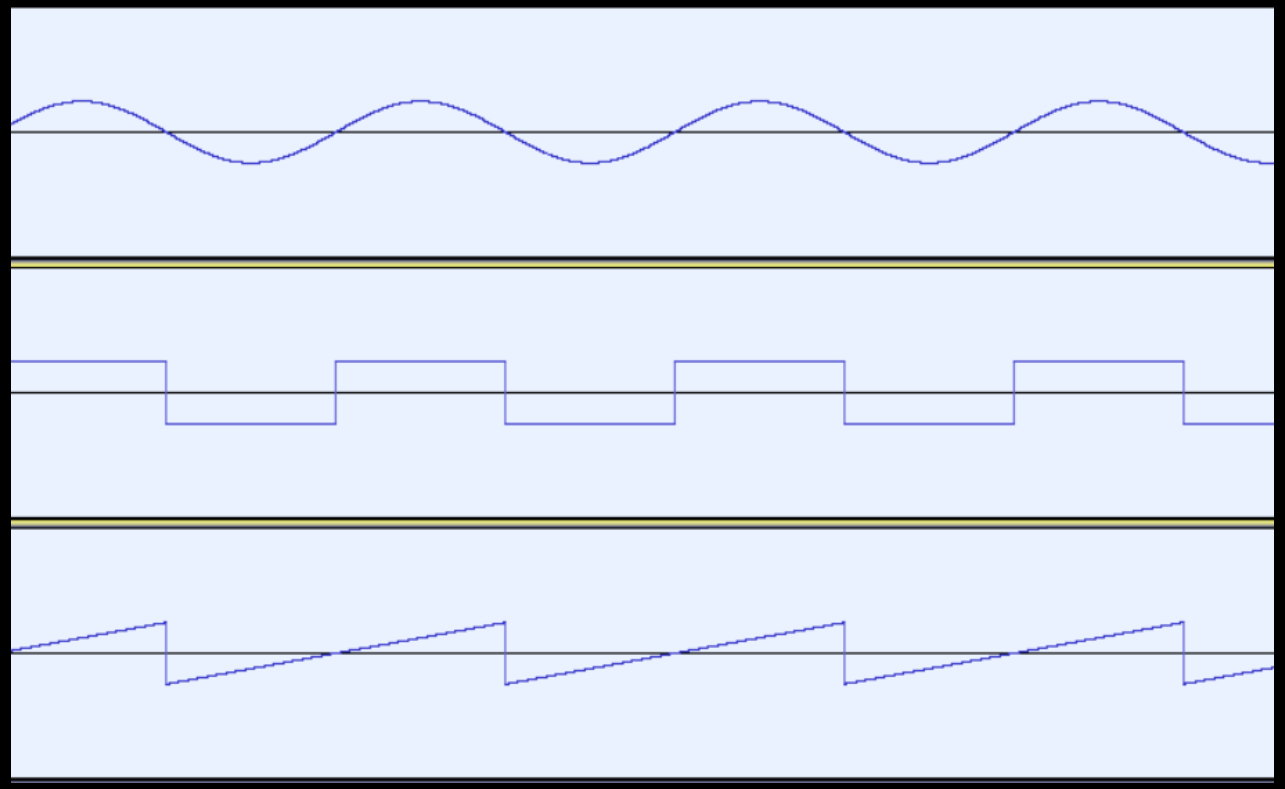
- In an electric guitar, there are pickups
- These pickups convert the sound wave to an alternating current
- Pickups are inductors
- Vibrations in strings disturb magnetic field, inductor tries to restore field
- Alternating Current is created
- Leaves guitar through ¼” mono-audio jack

Sound to Electricity – Simple Signals



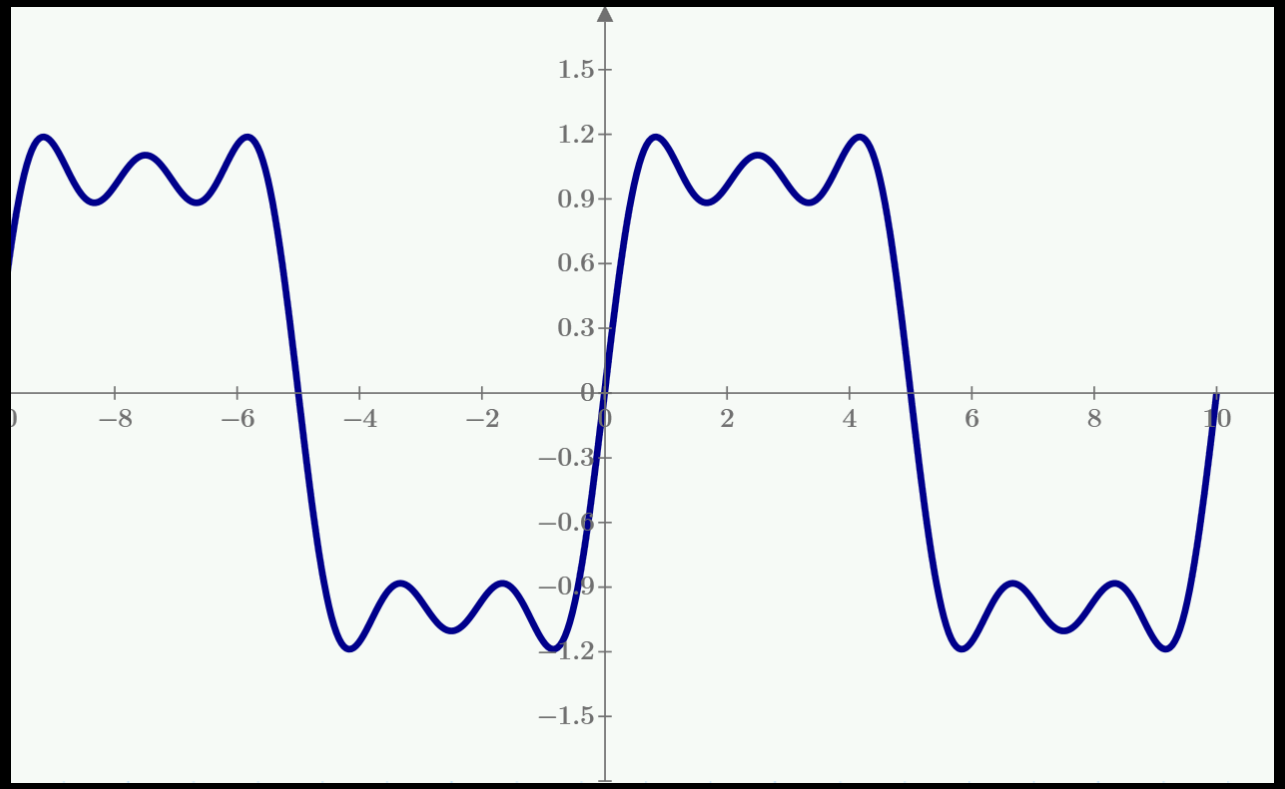
- The period of a sin function corresponds to the frequency emitted
- The wave can be modified while retaining the same frequency

Sound to Electricity – Simple Signal Examples



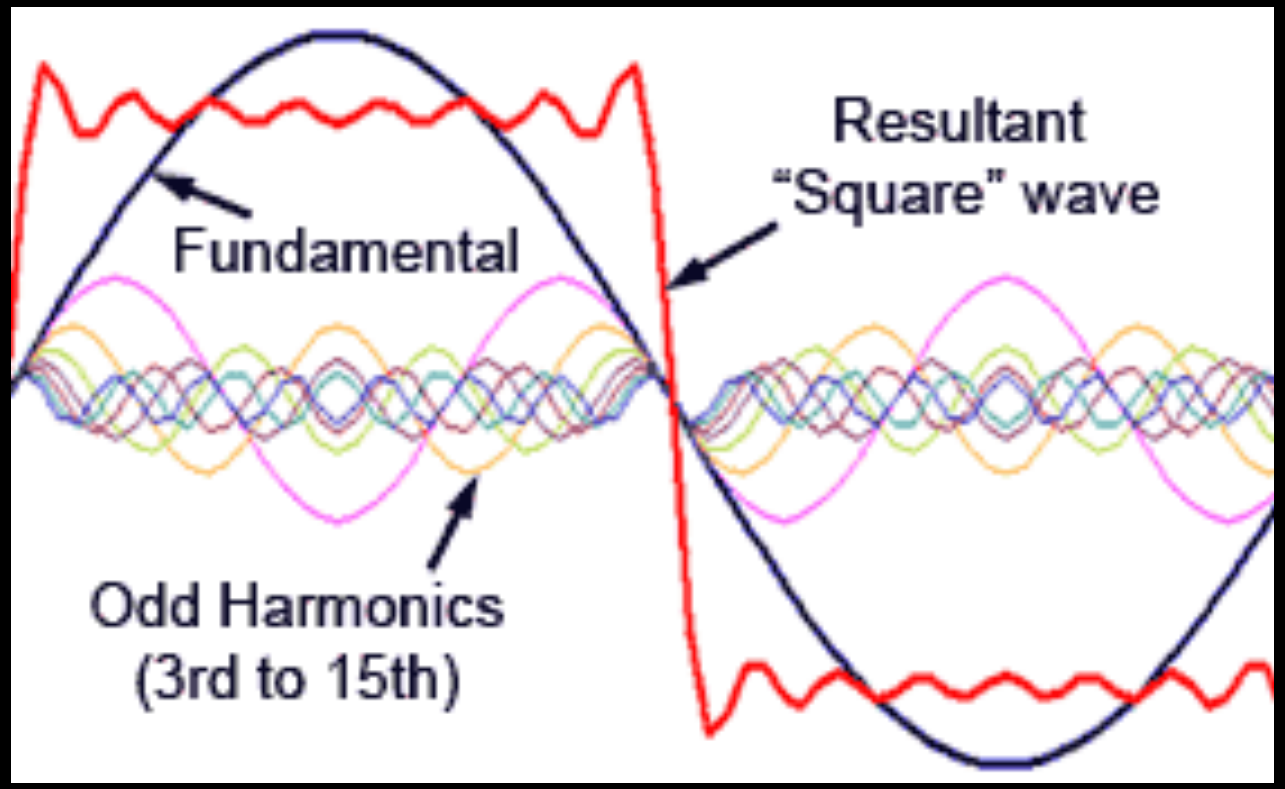
- Sound is Converted to Alternating Current
- Sine Wave
- Square Wave
- Sawtooth Wave

Distortion Signals



- Distortion is just a term describing how a signal has been modified.
- Square and sawtooth waves can be approximated by adding harmonics to the fundamental frequency
- Playing harmonics on top of the fundamental frequency leads to superposition

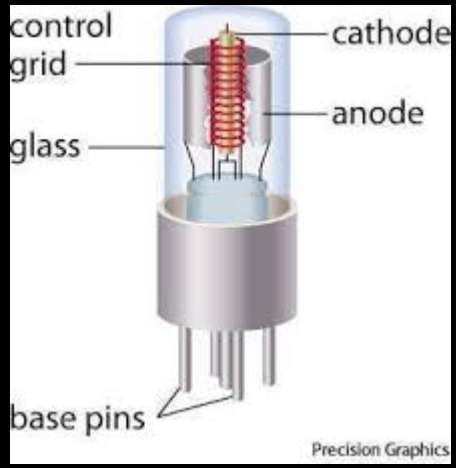
Waves



- The Fundamental Frequency is the first Harmonic
- The first overtone is the second harmonic
- Both are integer multiples of the fundamental frequency
- By playing odd numbered overtones on top of the fundamental frequency, we can start to generate something like the 'square' wave on the previous slide

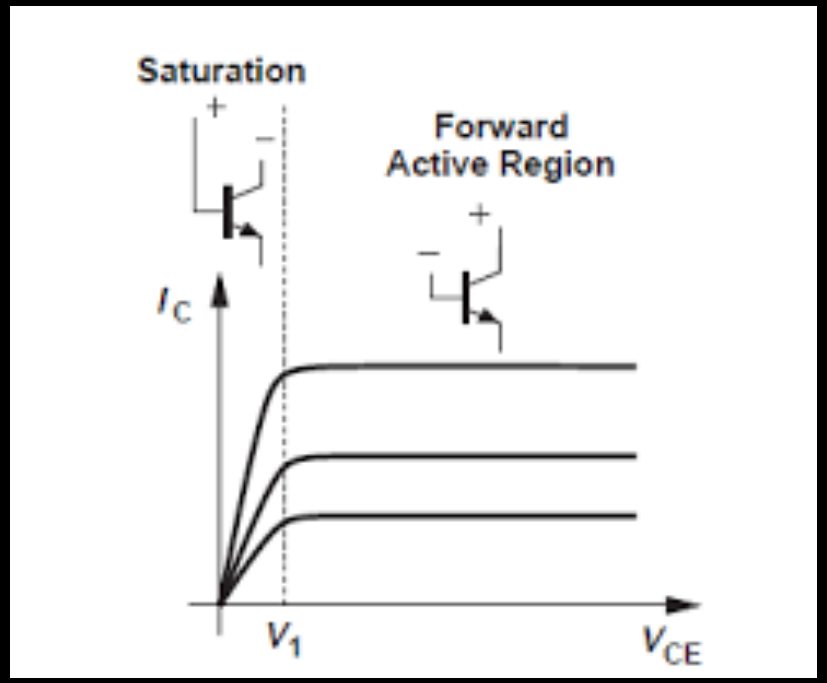
Tube Amplification & Distortion

- Until the invention of the transistor, electronic amplification was achieved with Vacuum Tubes
- Triode – Three charged regions
- Charge decomposes at fundamental frequency, but also at higher harmonics
- Allows for amplification and distortion described previously
- Still used today
- Warmer Timbre

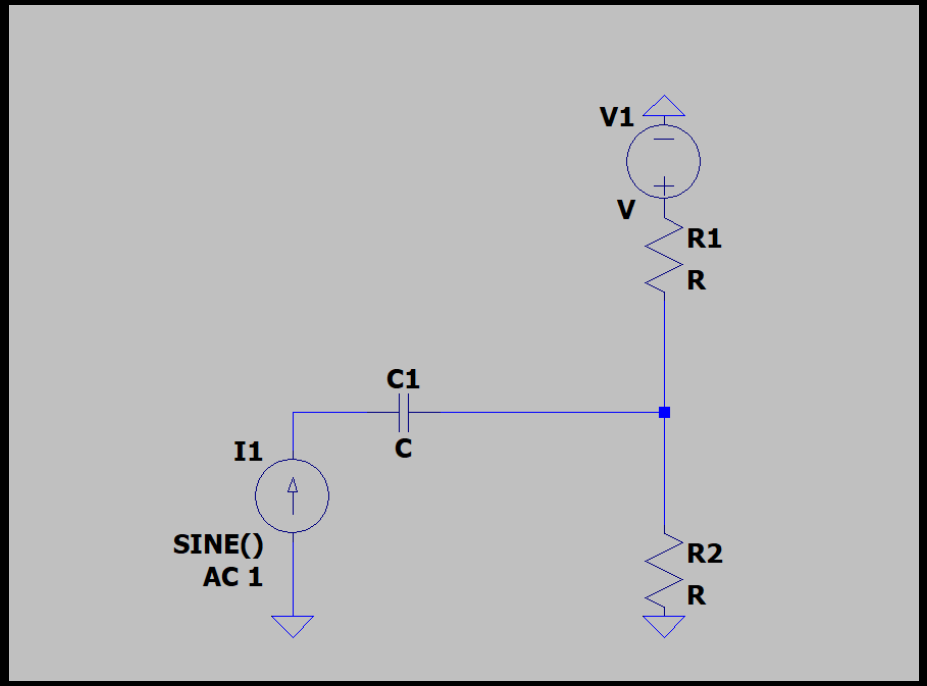


Solid State Amplification & Distortion

- Alternative to Triodes: Transistors
- Allow for amplification with much smaller and cheaper components
- Artificially recreate the sounds of vacuum tube amplification and distortion
- Not as “high performance” due to sound quality and timbre
- Because it is cheaper, this will be focused on distortion through the use of a transistor and diodes

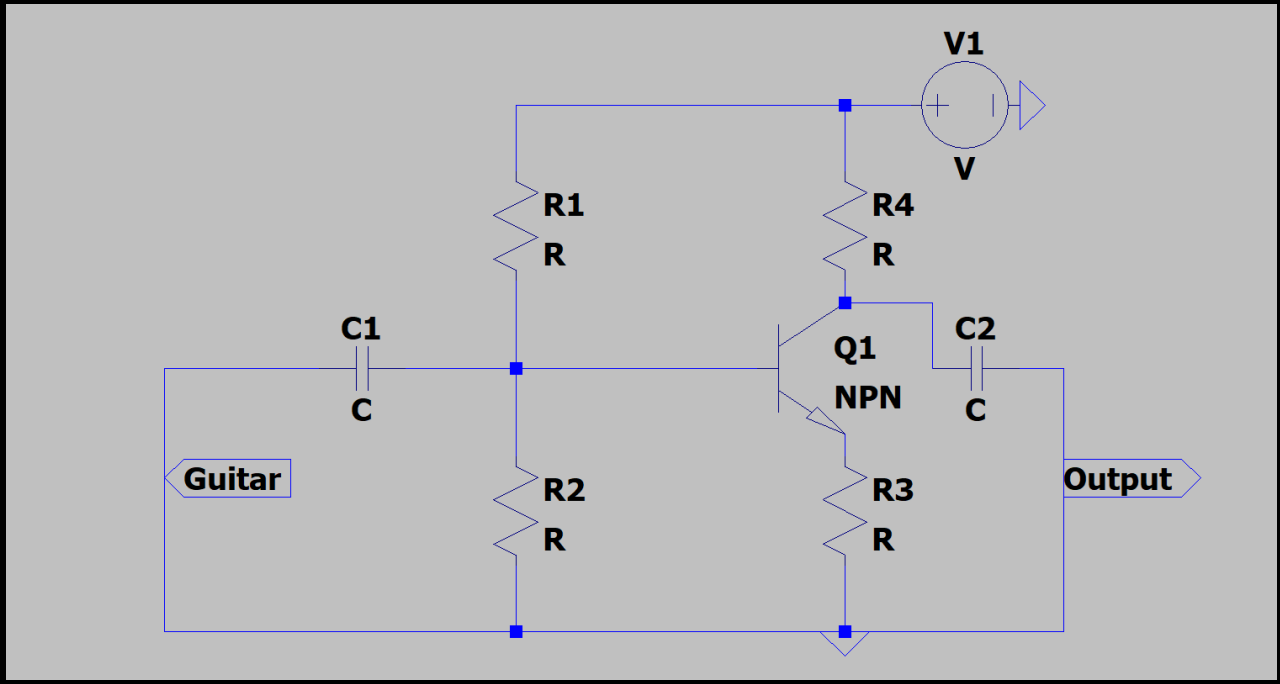


DC Bias



- Adds Bias to AC Signal
- Allows signal to be used by Arduino
- Basically adds a constant voltage offset

Common Emitter



- Adds Bias to AC Signal
- Inverts and Amplifies Signal
- Commonly used in speakers
- Relevant here

Objectives

- The objective of this project is to add harmonics to the observed wave form so that distortion may be achieved
- To accomplish this, the following must be achieved:
  - Obtain the initial guitar waveform (we will simplify this project initially and only conduct this data acquisition at a single frequency, which I've chosen to be 82 Hz).
  - Bias, Amplify
  - Use Audacity to create and output a similar waveform to the Arduino
  - Construct a circuit capable of superimposing odd overtones onto the signal, creating distortion
  - Apply this distortion to higher frequencies

Design Requirements

1. Construct a common emitter amplifier that is capable of biasing and amplifying a signal ranging from 83.23 Hz to 329.6 Hz.
2. Record data for these frequencies and write a program capable of displaying the amplitude and bias caused by this circuit.
3. Construct distortion circuit, which can apply distortion across this desired range of frequency.
4. Again recreate the distorted waveform in python

Additional Requirements

1. Conduct error analysis on the components used to create the common emitter
2. List the average error for all the data points using the largest amplifier value
3. Record audio effect of pedal!

Common Emitter Requirements

This circuit must:

- be able to take input from either a guitar or from the audio jack of a computer
- bias the incoming signal so that all oscillations take place within the range of 0-5V when read in through the Arduino
- amplify the incoming signal between 1.5-3x of the initial waveform
- invert the input signal (negative gain)

Distortion Pedal Requirements

In addition to the amplification and biasing from the common emitter, the pedal must:

- distort the incoming signal through the use of clipping diodes
- create a recognizable, audible effect (distortion)
- be able to take input from either a guitar pedal or an Arduino

Arduino Requirements

The Arduino must:

- be able to read in the waveform at a rate that is reasonable enough to recreate the waveform without discontinuities
- NOT allow for a changing sampling rate, as this will make it incompatible with the python software being used to create CSV files from the Arduino.

Python Requirements

SerialPlotter.py be used as a Blackbox.

Additionally, a separate python program must:

- Recreate the audio waveform
- Remove any bias
- Use Fourier Transform to recognize harmonics
- Attempt recreation of waveform with Superposition of overtones

Concept of Operations

To recreate this Guitar Pedal:

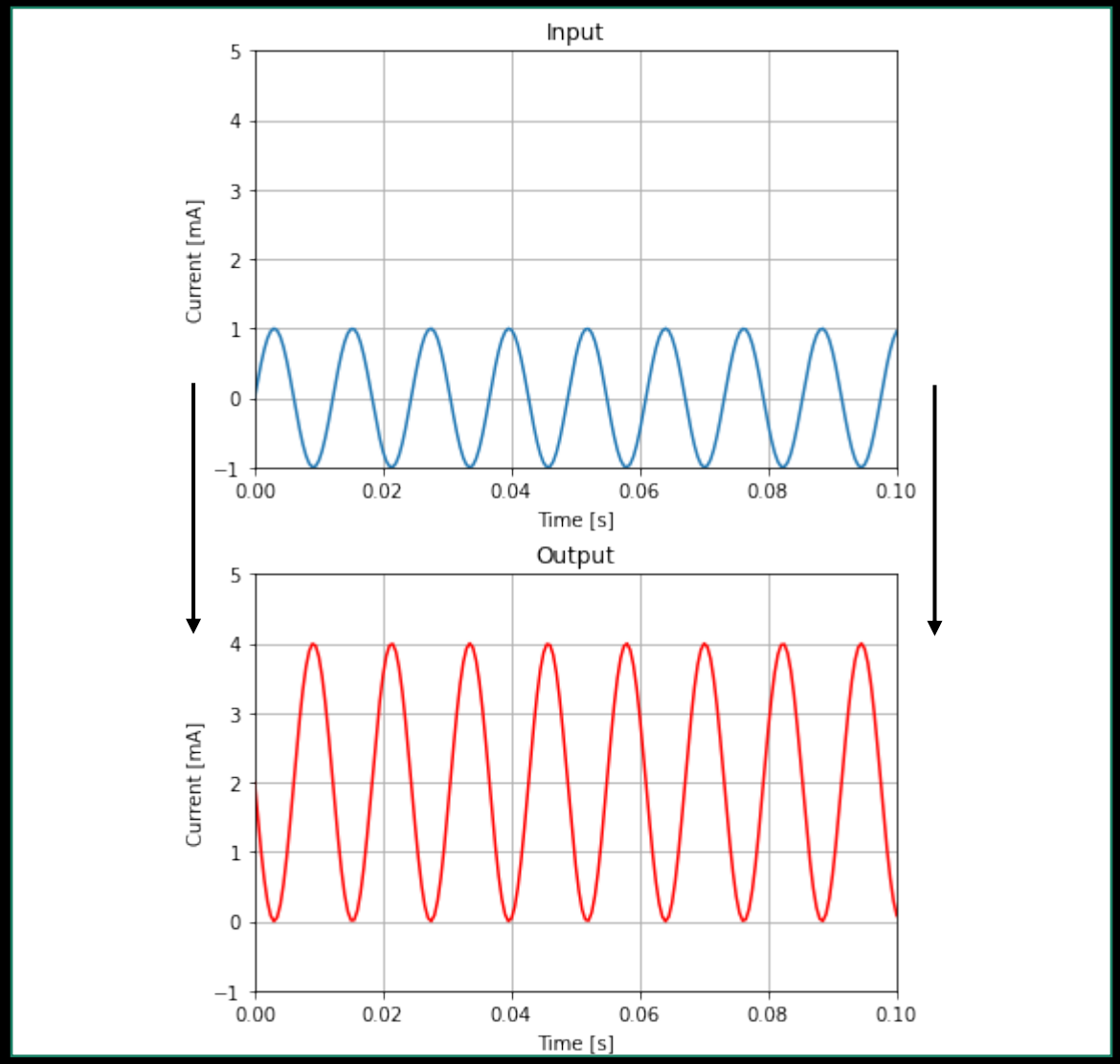
- Assemble the common emitter amplifier described in the following subsections
- Calibrate SerialPlotter.py conversion factor
- Measure the voltage across each component for the common emitter
- Perform Error Analysis on this circuit
- Record wave forms for low and high E, recreate them in python
- Create the distortion circuit with LED clipping described in the scematic
- Record and recreate waveforms generated by playing guitar and enjoy the effects of some altered audio!

Concept of Operations

To use the Guitar Pedal:

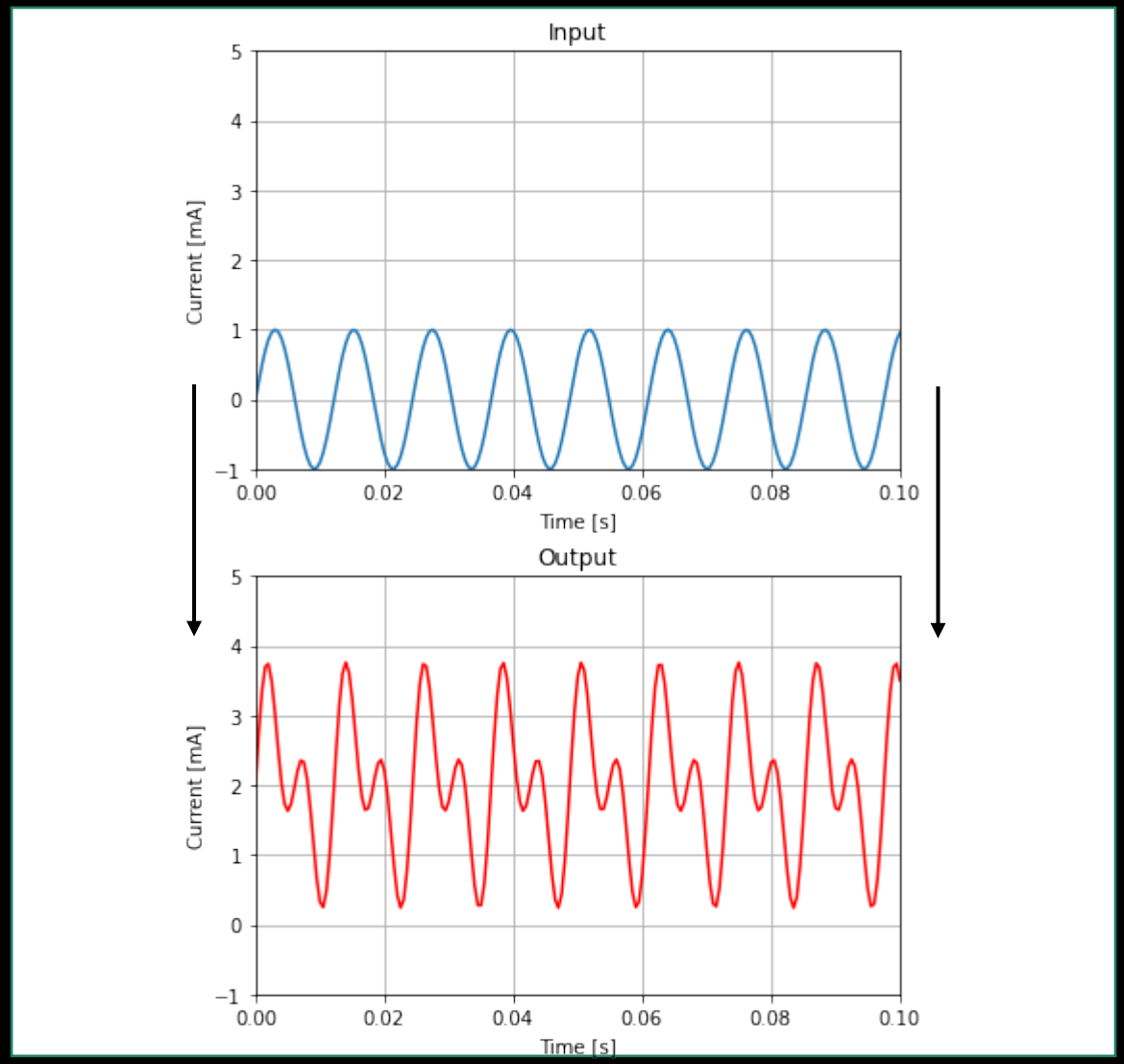
- Plug their electric guitar into one of the two mono-audio jacks
- Plug their amp into the other mono-audio jack
- Attach a 9V video
- Play guitar with the provided distortion

Expected Results – Common Emitter



- The signal will be converted from AC to DC with a bias
- Signal will be inverted
- Signal will be amplified
- This will apply for the initial guitar and audacity waveforms

Expected Results – Final Pedal



- Through superposition, the output waveform should be altered to include higher frequencies
- This will create desirable distortion in the output
- Additionally, the signal will be amplified and biased

System Overview

Signal Amplifier & Bias

Breadboard Circuit – Bass Pedal, Guitar Pedal

Arduino Code

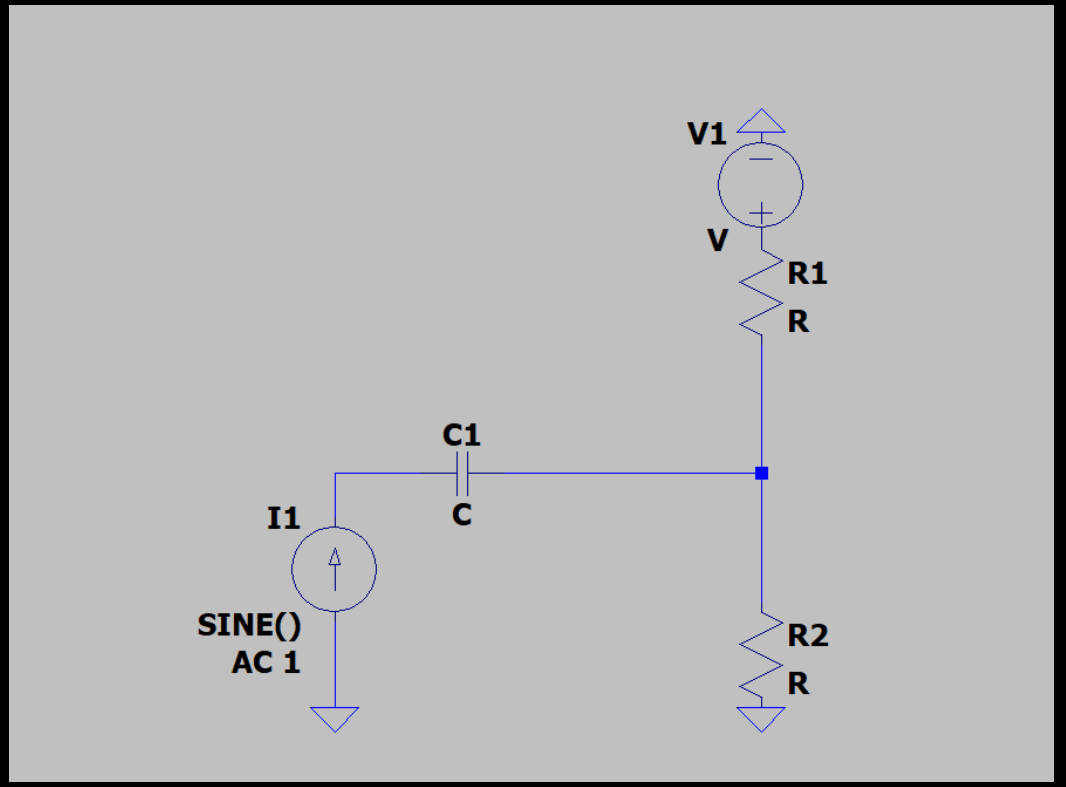
Python Analysis

Breadboard Circuit – Signal & Bias Issue

- Guitar Emits in AC
- Arduino reads 0-5V
- Must change the signal from AC to DC Voltage
- Signal is small
- Must be amplified
- Solution is Common Emitter Amplifier

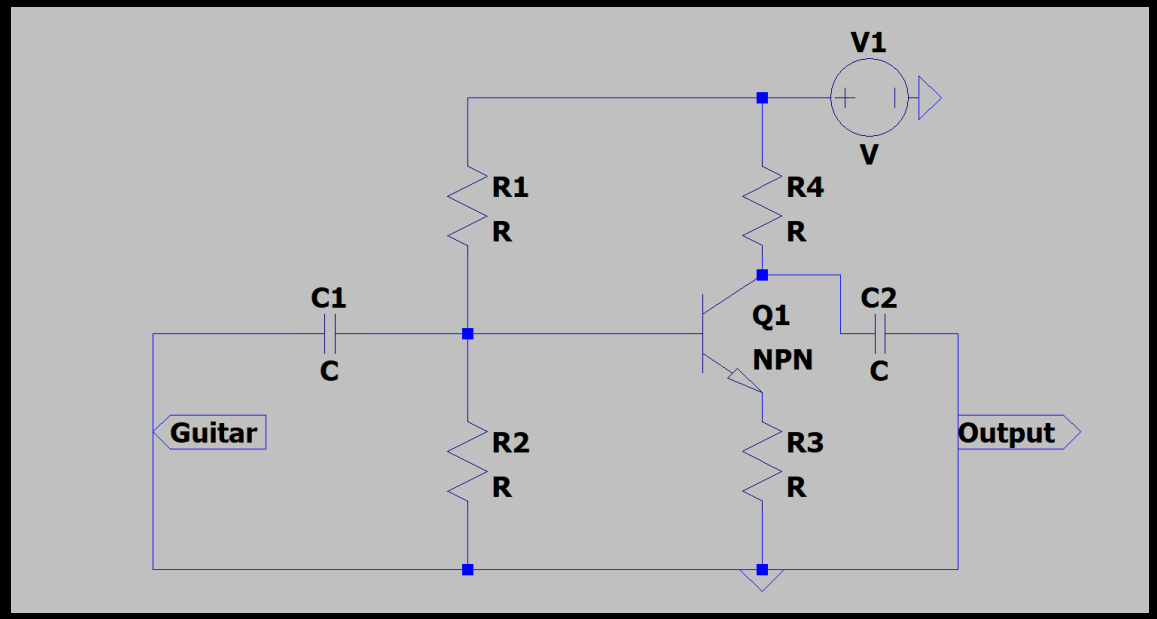
Breadboard Circuit – Bias Issue

- To increase the signal, the following basic circuit will be used
- Common Emitter Circuit - Both Input and Output are connected to ground
- Provides a Bias on left half
- Amplifies Signal in right half
- V+ will be supplied by 9V Battery
- Using a stereo jack, input can also be Audacity wave file

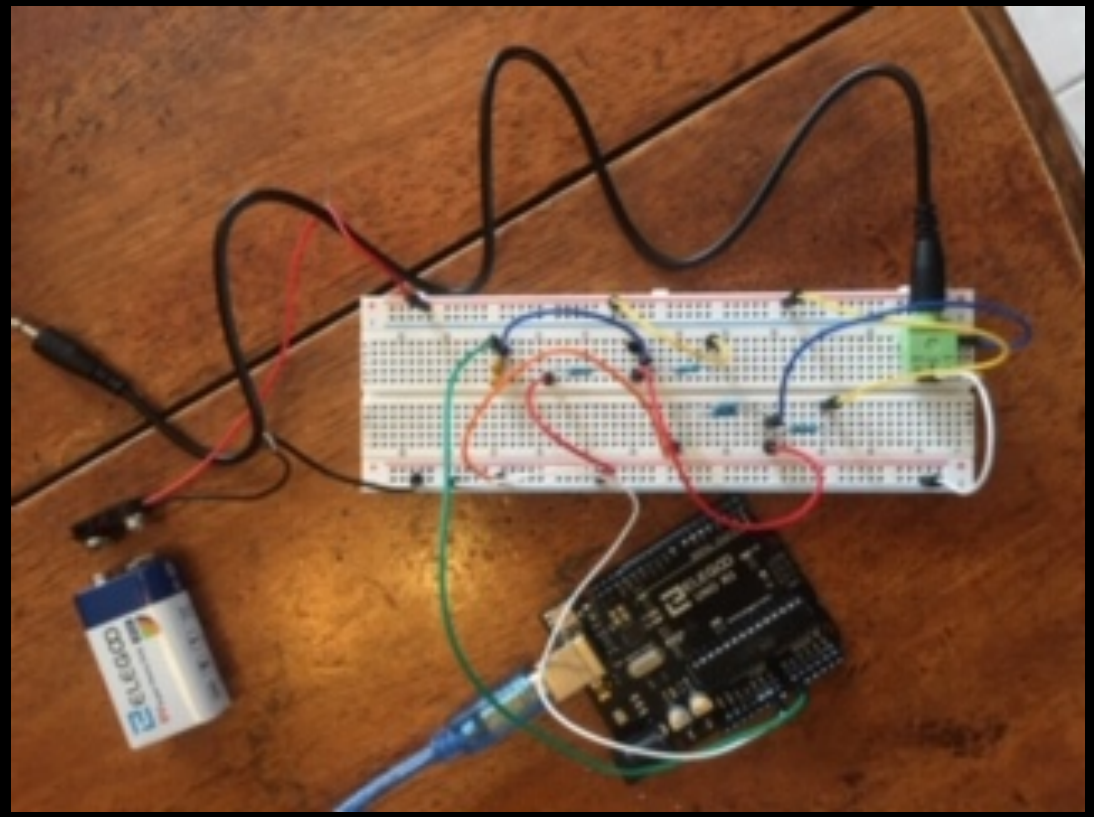


Breadboard Circuit – Amplifier

- To increase the signal, the following basic circuit will be used
- Common Emitter Circuit - Both Input and Output are connected to ground
- Provides a Bias on left half
- Amplifies Signal in right half
- V+ will be supplied by Arduino
- Using a stereo jack, input can also be Audacity wave file



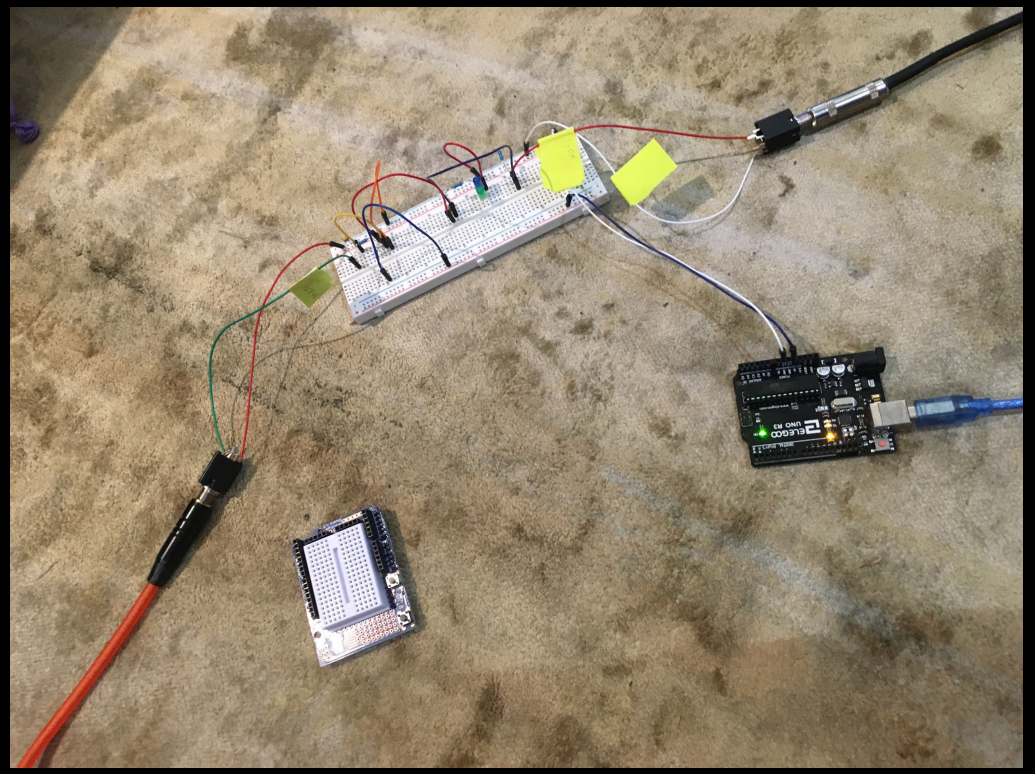
Amplifier for Guitar Signal - Specifics

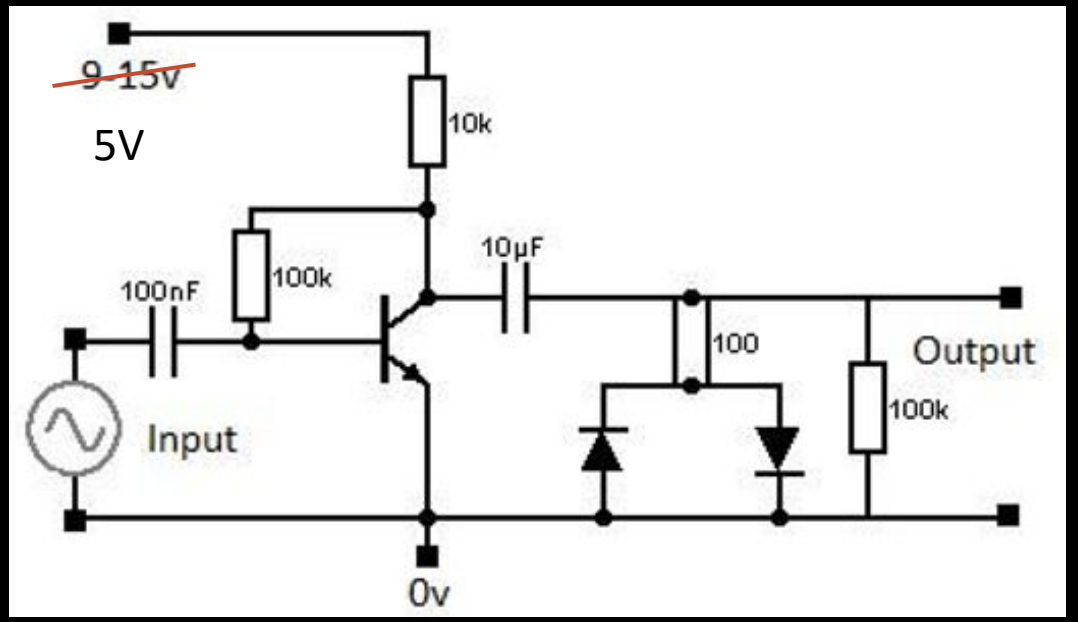


- C1 = 0.1 µF
- C2 = 20 µF
- R1 = 100 KΩ
- R2 = 50 KΩ
- R3 = 500 Ω
- R4 = 10 KΩ
- NPN = PN2222
- V1 = 9V

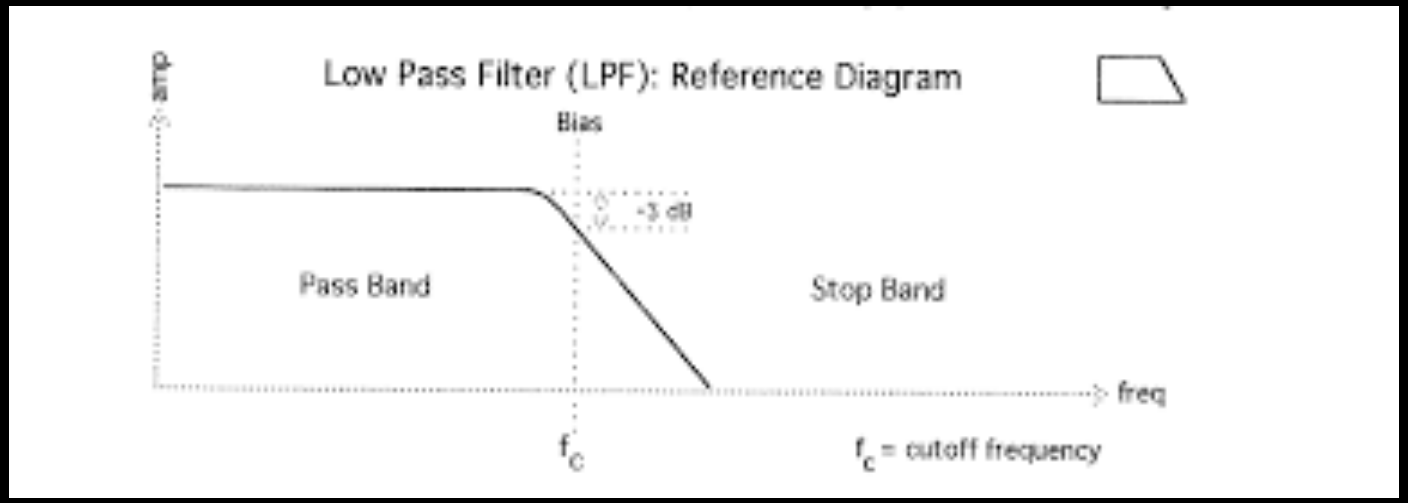
Amplifier & Distortion Pedal - Bass

- Constructed Bass Pedal





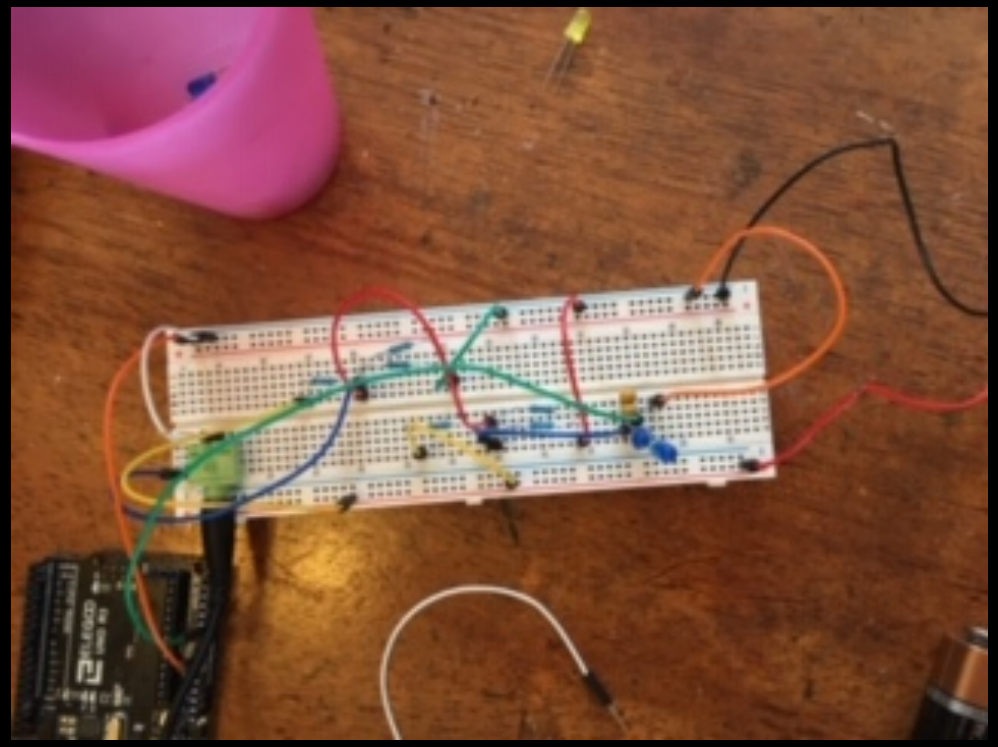
Bass Pedal Frequency Response

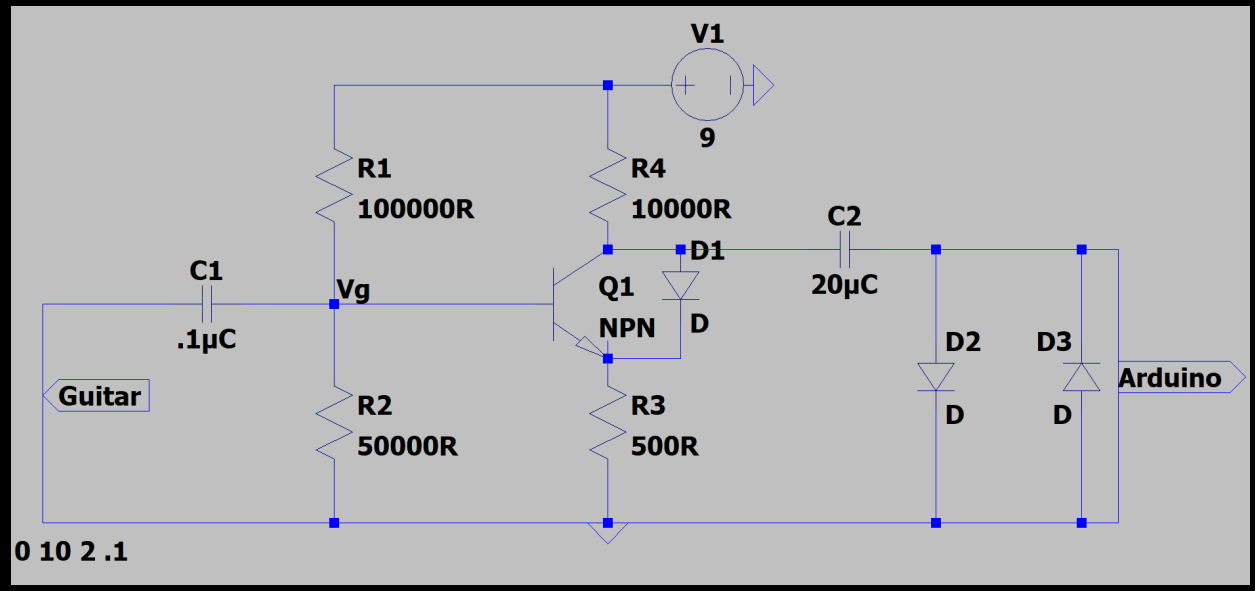


- Distortion cuts off at roughly 120Hz
- NOT A TRUE LOW PASS FILTER
- Still allows higher frequencies through

Amplifier & Distortion Pedal – Final Product

- Constructed Guitar Pedal





Arduino Code

- Arduino will need to read in values as quickly as possible
- If the sampling rate is too low, then higher frequencies will not be able to be recognizable
- To do this, removing analog conversion is the most effective way
- There will be two sampling rates, one at 4453Hz, and again at over 20000Hz
- This can be achieved through two Arduino scripts

Python Analysis

- Application will be used to recreate the waveforms of the Guitar
- Original Form:
  - Remove Bias
  - Invert to correct phase
  - Plot Waveform
- Distorted Form:
  - Plot Wave form
  - Use Fourier Transform to recognize harmonics
  - Attempt recreation of waveform with Superposition of overtones

Subsystem Design

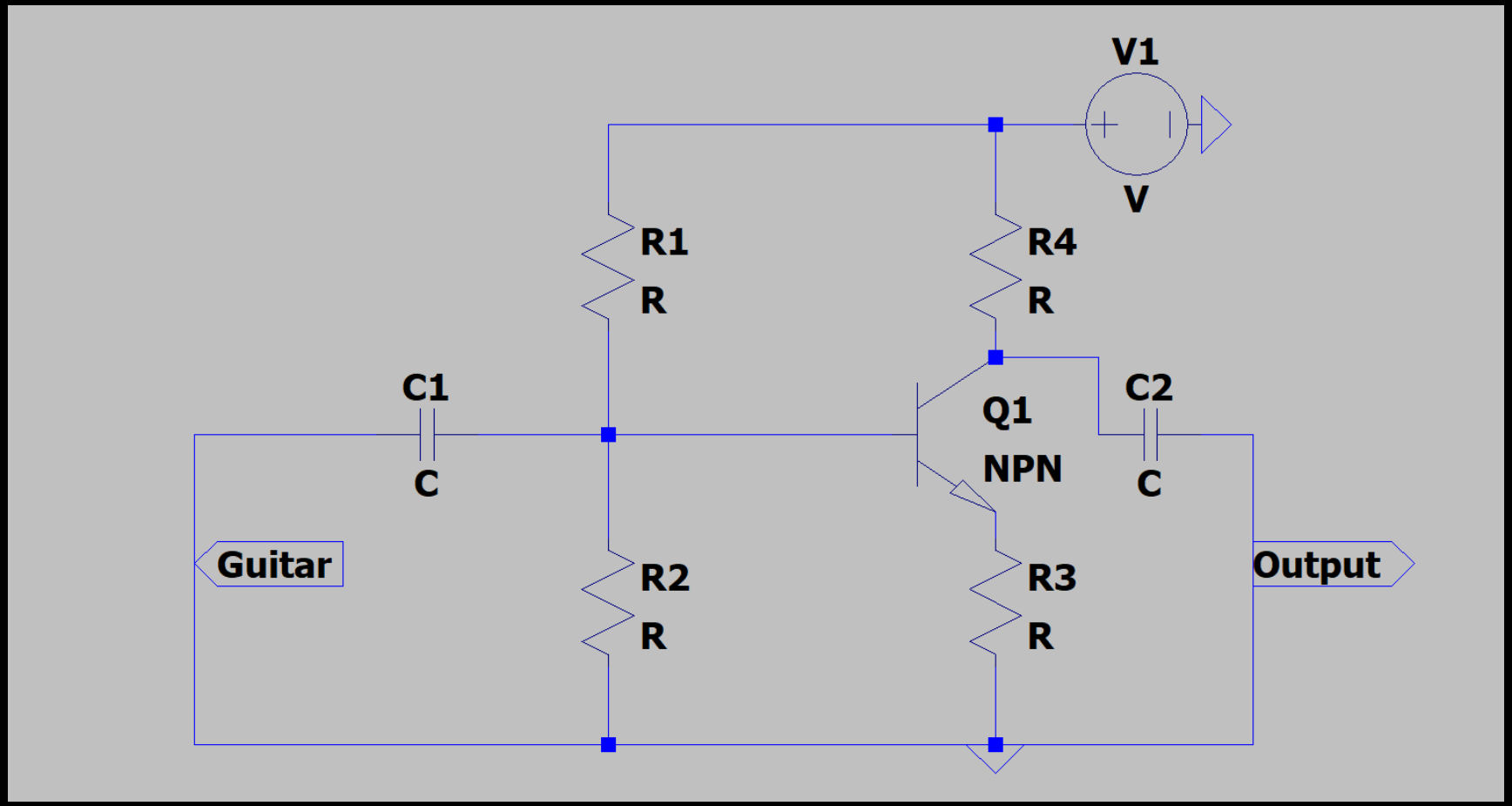
Common Emitter

Bass Pedal – Offset

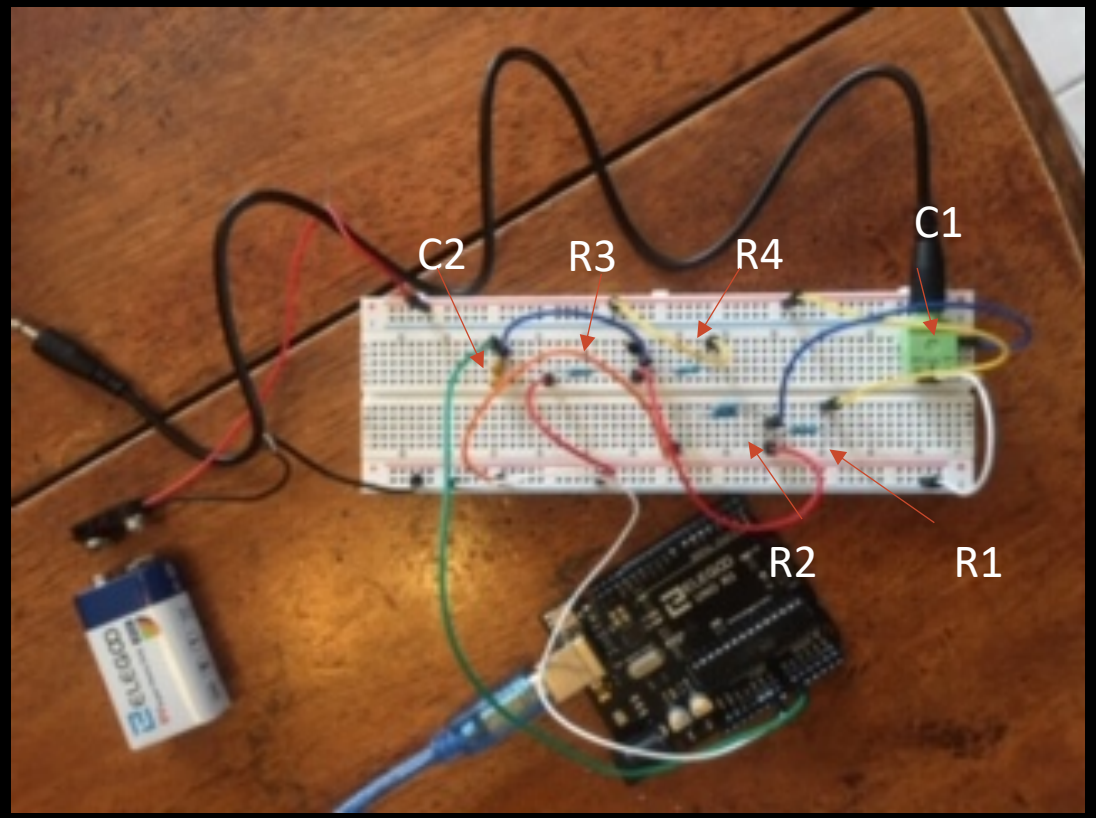
Bass Pedal - Amplifier

Bass Pedal - Overtones

Common Emitter Amplifiers

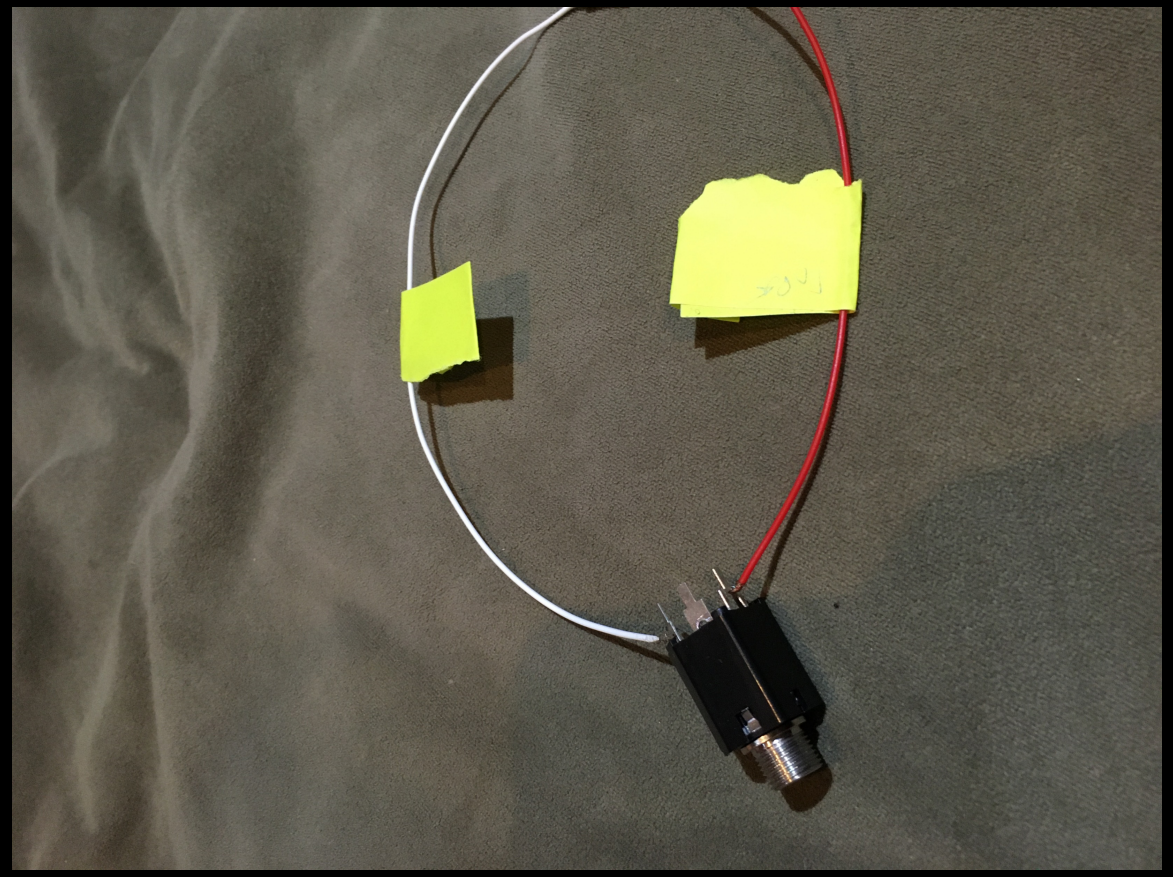


Common Emitter Amplifier



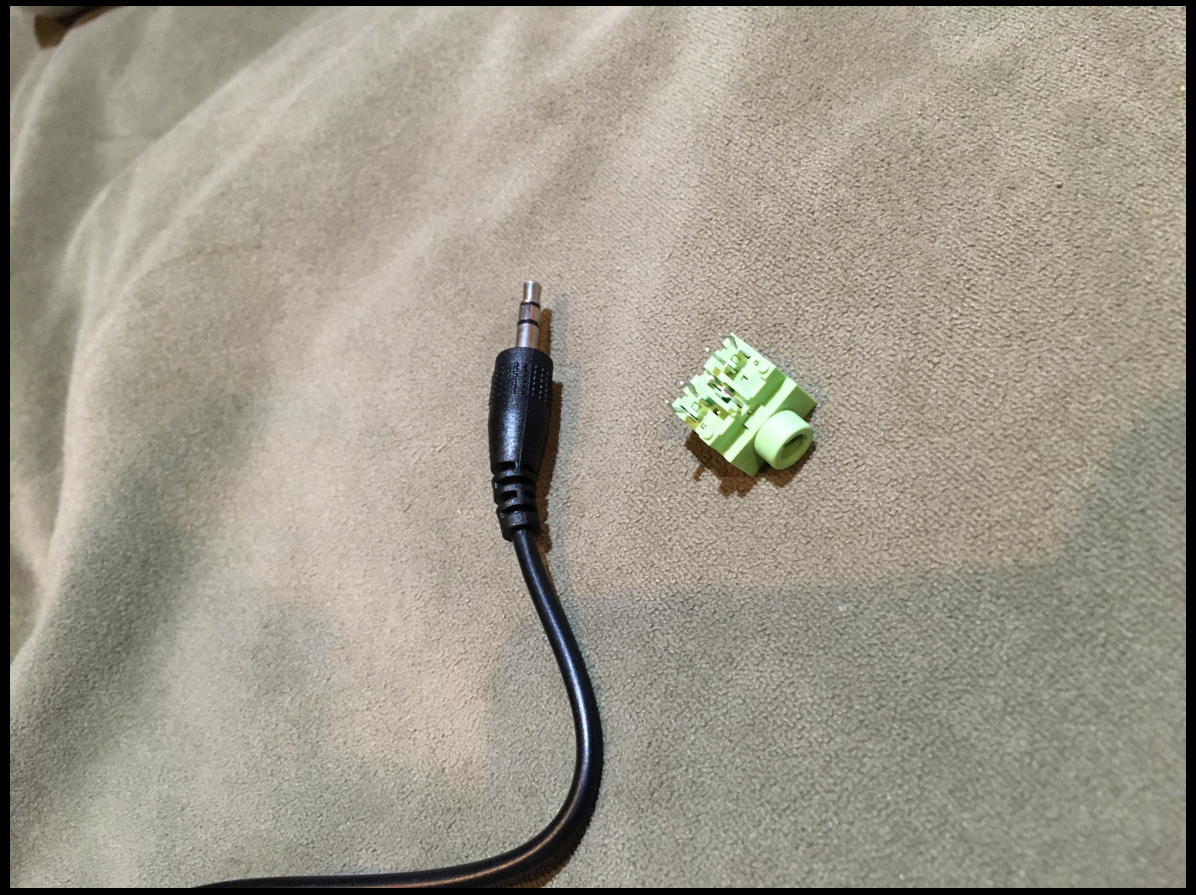
- C1 = 0.1 µF
- C2 = 20 µF
- R1 = 100 KΩ
- R2 = 50 KΩ
- R3 = 500 Ω
- R4 = 10 KΩ
- NPN = PN2222
- V1 = 9V

Common Emitter Amplifier – Guitar Input



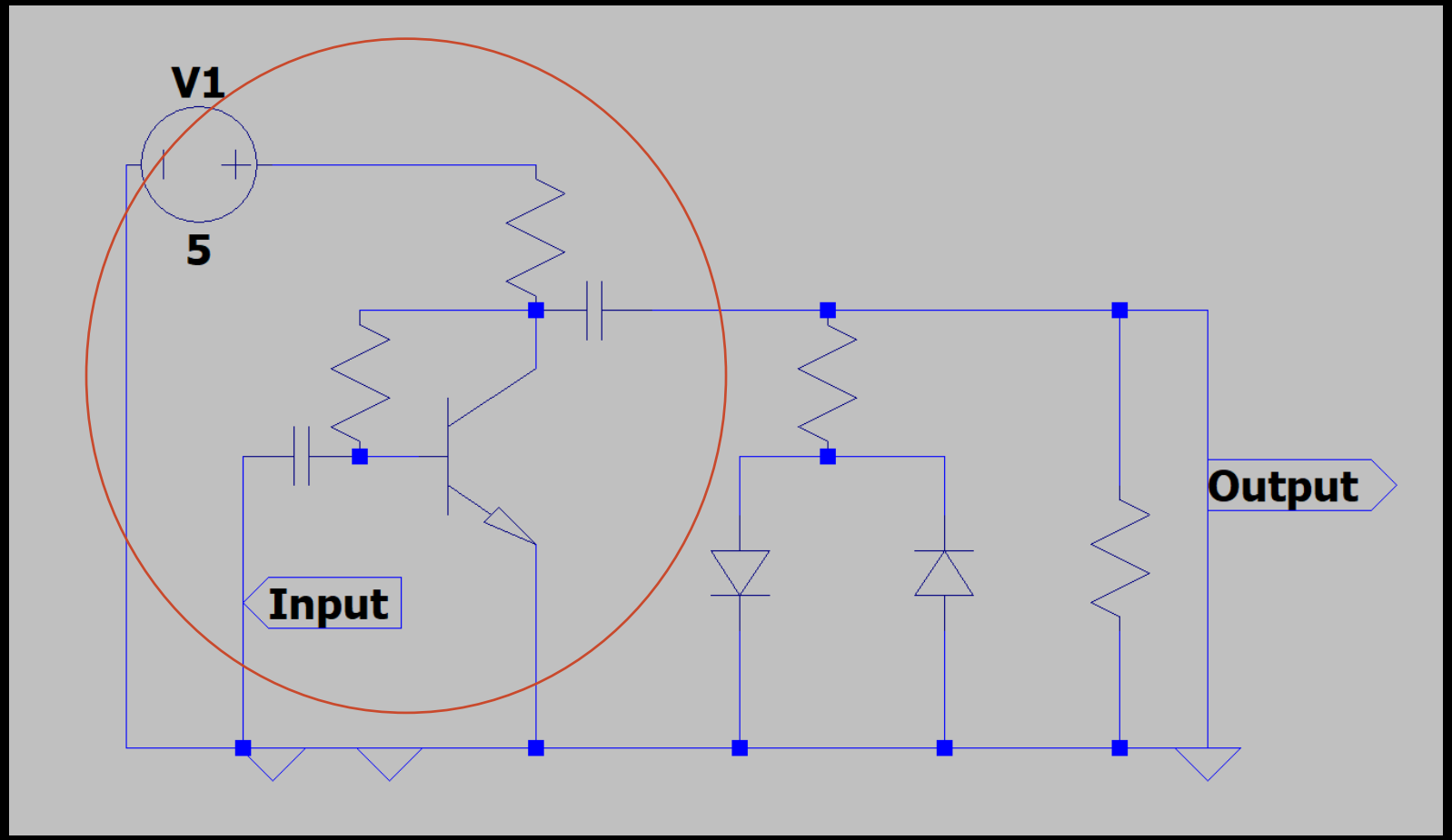
- ¼” Mono-Audio Jack
- White cable goes to ground
- Jumper Cables were cut and stripped, tied to pins of audio-jack
- Receives input from audio jack

Common Emitter Amplifier – Computer Input

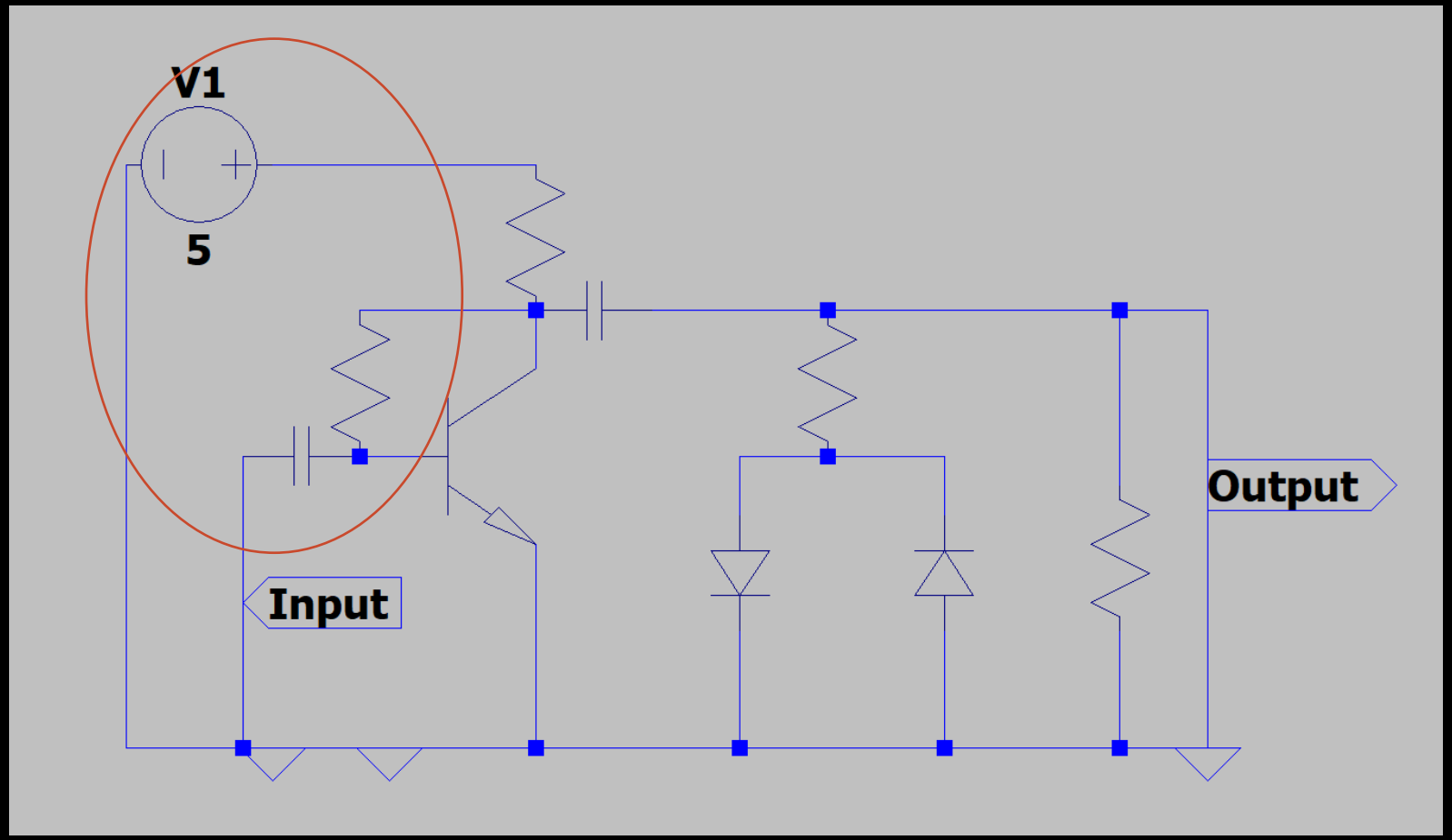


- Stereo-Audio Jack
- One end to Computer, other to Circuit
- 5 Pins – Central is ground
- 2 Right (Stereo)
- 2 Left (Stereo)

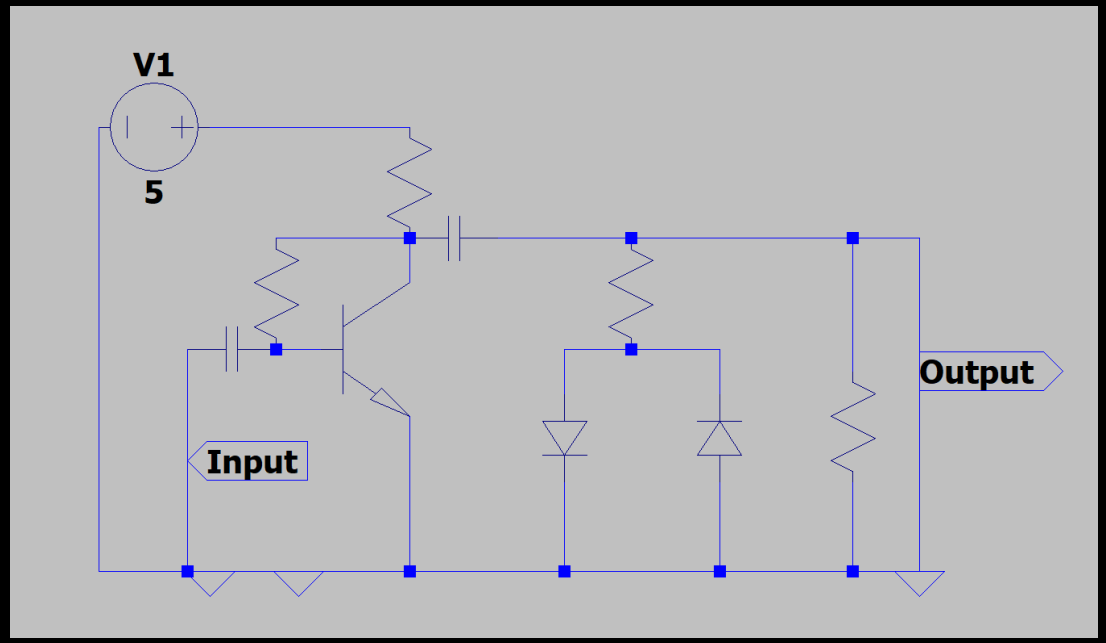
Bass Pedal – Simplified Common Emitter



Simplified Common Emitter - Bias

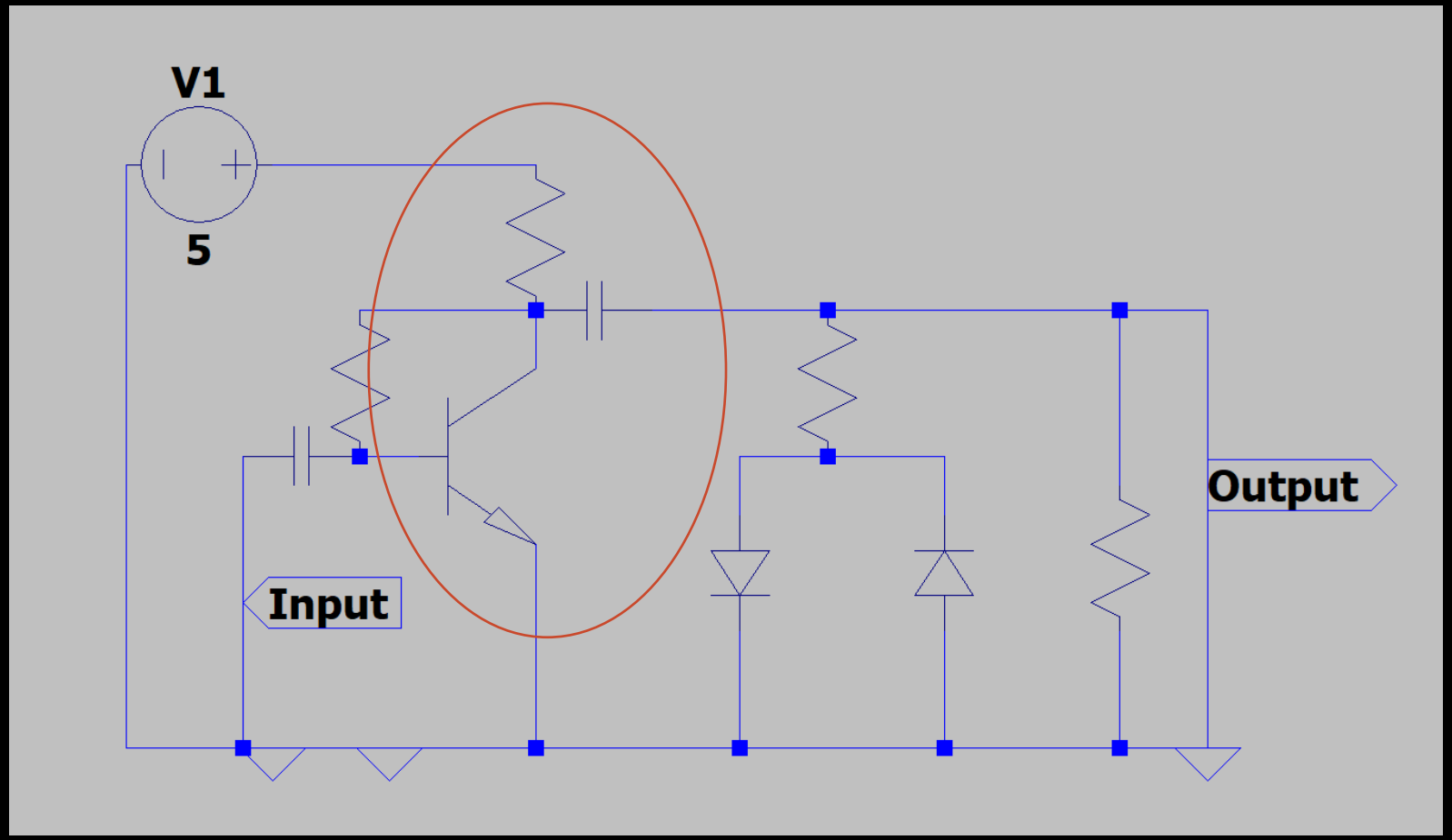


Simplified Common Emitter - Bias

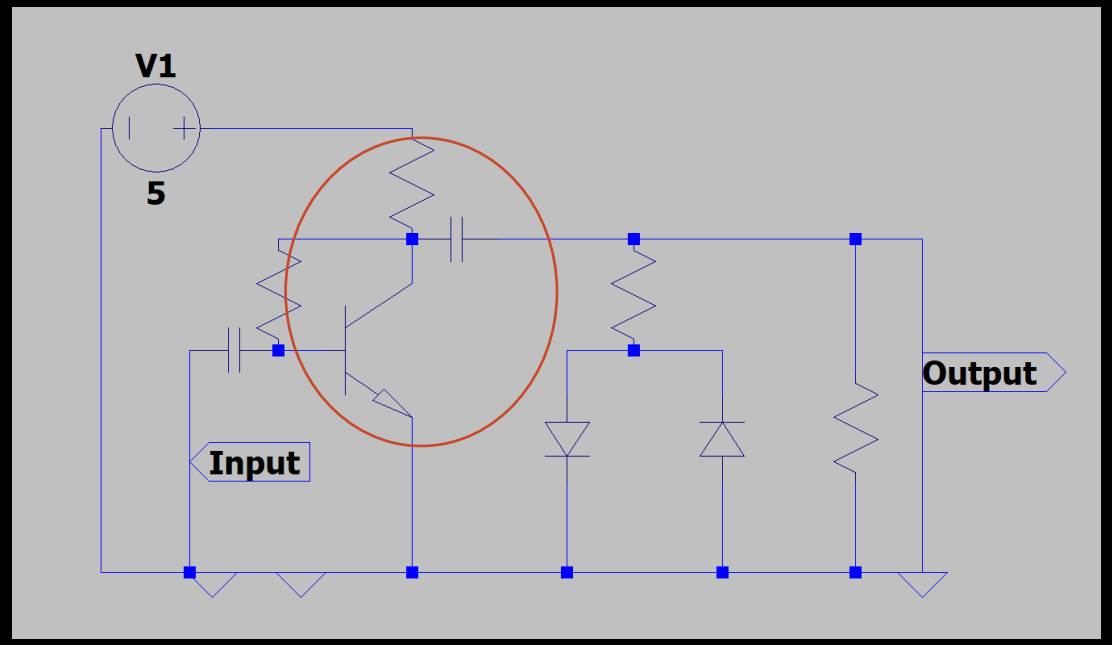


- This is a simplified version of the Bias seen earlier
- Uses Capacitor to smooth signal and protect input
- Resistors in series add current/voltage

Simplified Common Emitter - Amplifier

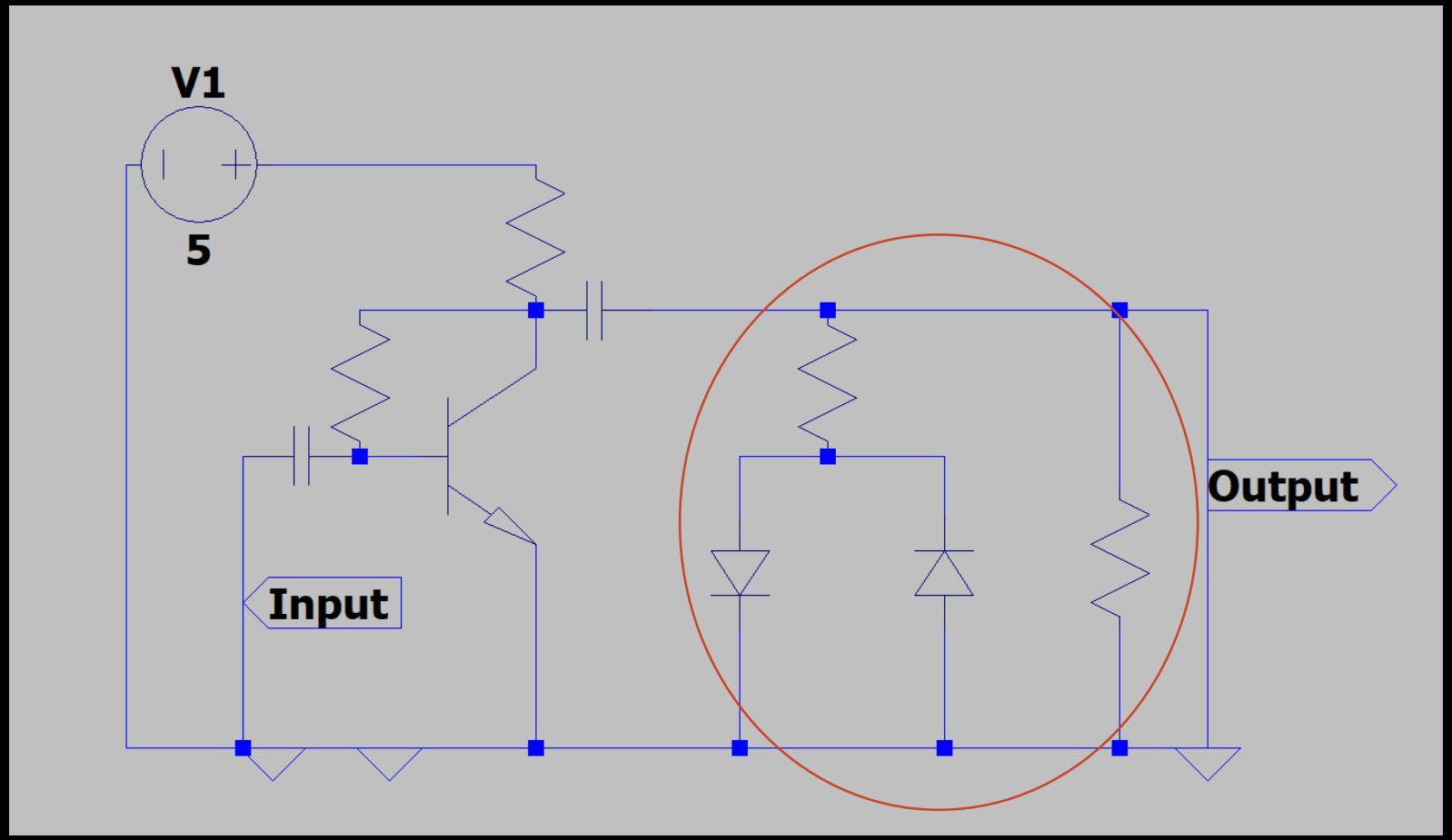


Simplified Common Emitter - Amplifier

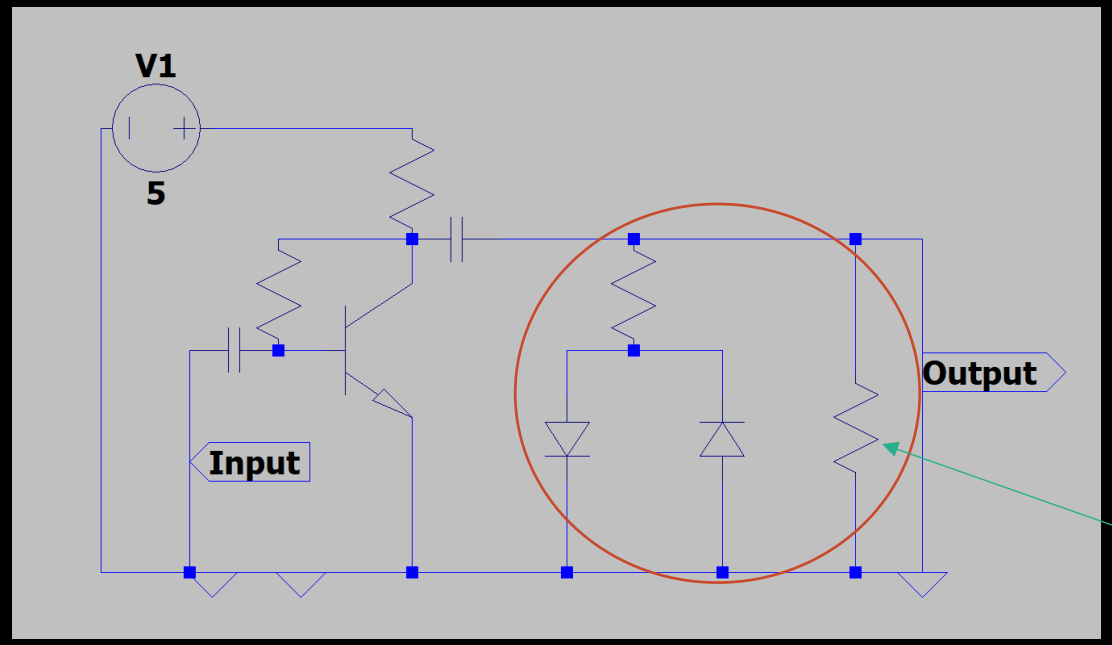


- Remainder of the Common Emitter Circuit
- Uses Capacitor to smooth output/protect microcontroller
- Voltage altered on gate allows for variable current on emitter

Bass Pedal - Distortion

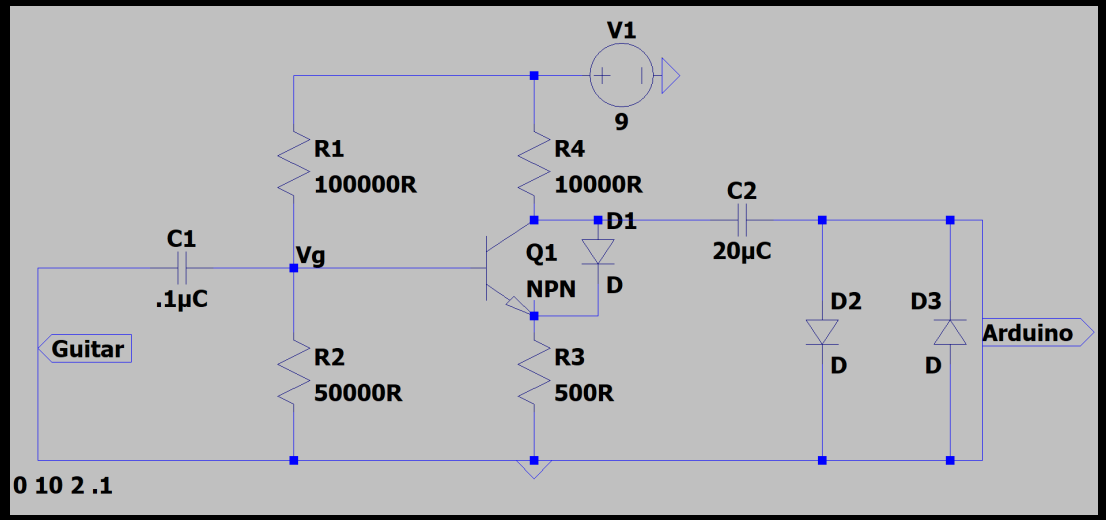


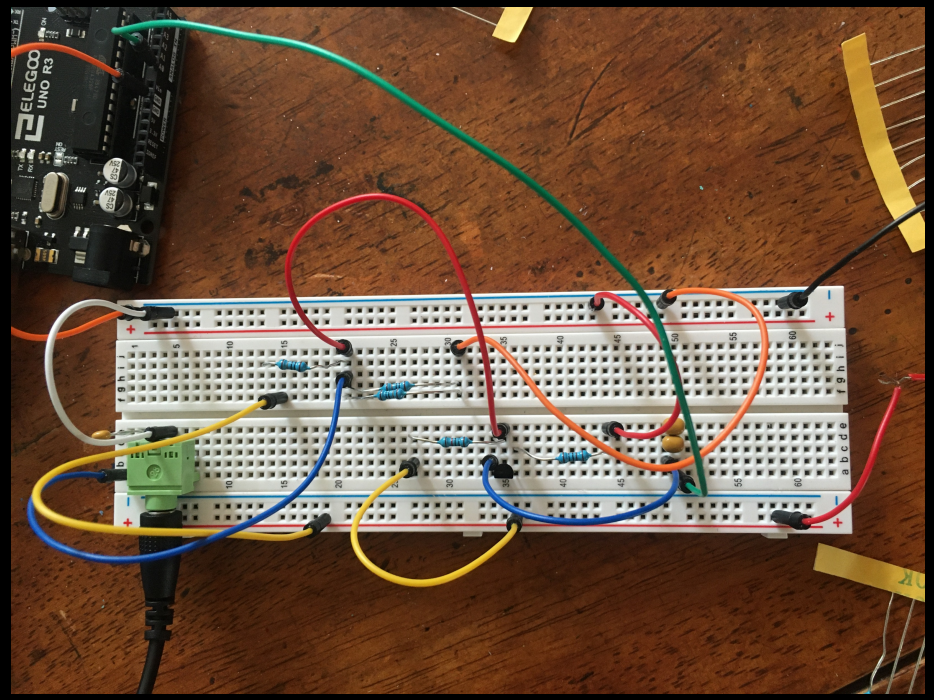
Bass Pedal - Distortion



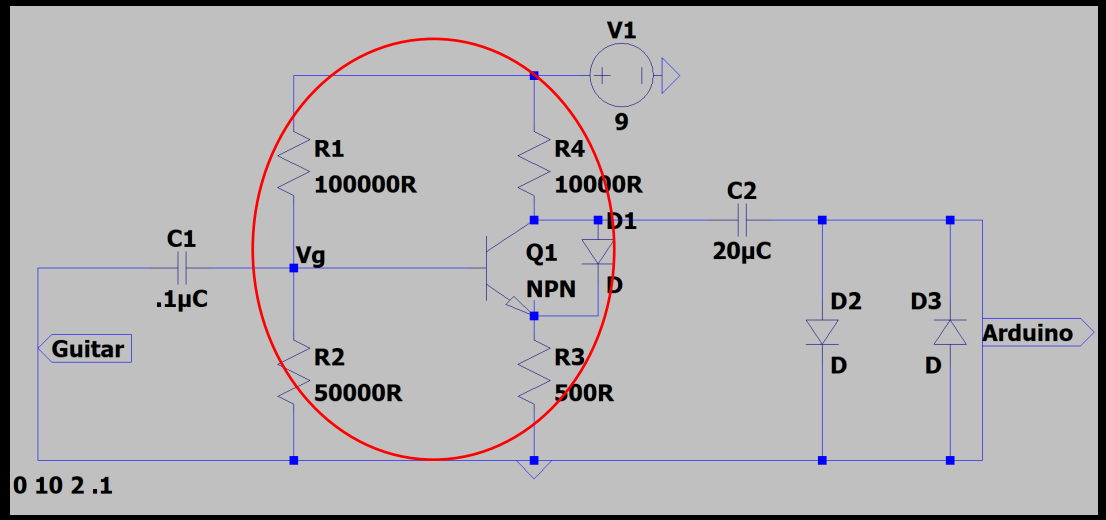
- Provides actual distortion for the system
- Overlays overtones
- Alters wave form
- Output is equal to voltage across this resistor (in parallel)

Guitar Pedal



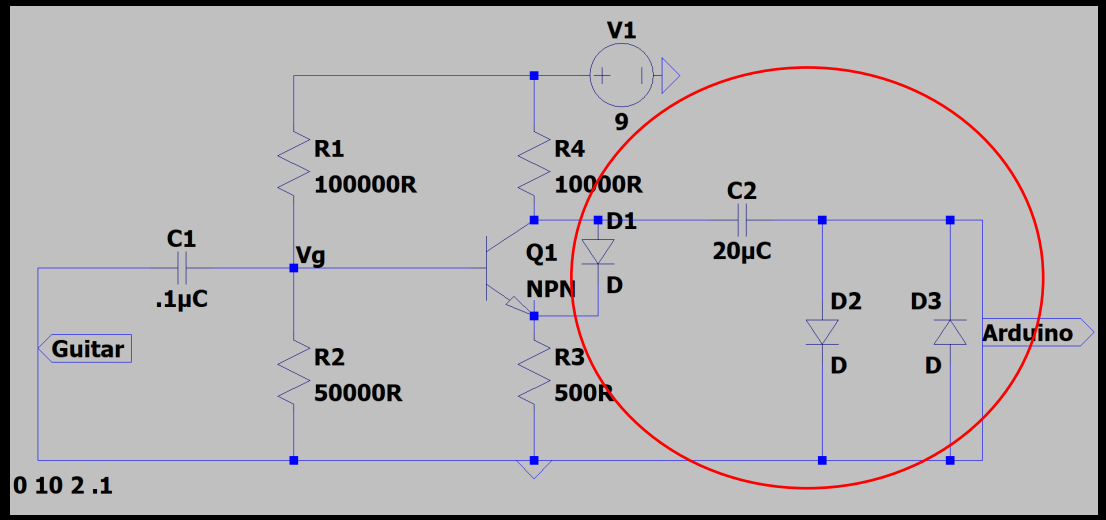


Guitar Pedal – Common Emitter



- Identical to Common Emitter Described Previously
- Same resistance and capacitance values (displayed)

Guitar Pedal – Distortion (Clipping)



- Solid State Distortion
- Clip the original signal
- Diodes strip peak voltages from alternating sides of the wave form
- Diode across transistor assists with distortion

Arduino Code

- The first code block simply reads from the Arduino's analog port – simple

The second:

- Converts incoming data into hexadecimal instead of ascii
  - ADCSRA = (ADCSRA & 0xf8) | 0x04
- By preventing this calculation, we remove 13-16 ADC's of internal clock
- Additionally, creates a buffer
- Reads a continuous sequence quickly
- Writes this sequence
- Creates incredibly quick sampling rate

Python Code

Separated by parts:

- SerialPlotter.Py
  - Reads in, coverts hexadecimal by bitshifting
  - Prints output to screen to create a pseudo oscilloscope
- Recreate_Waveforms.py
  - Calibrates conversion for serialplotter.py from 16bit to voltage
  - Reads in data from files
  - Plots waveforms
  - Takes FFT
  - Analyzes frequency

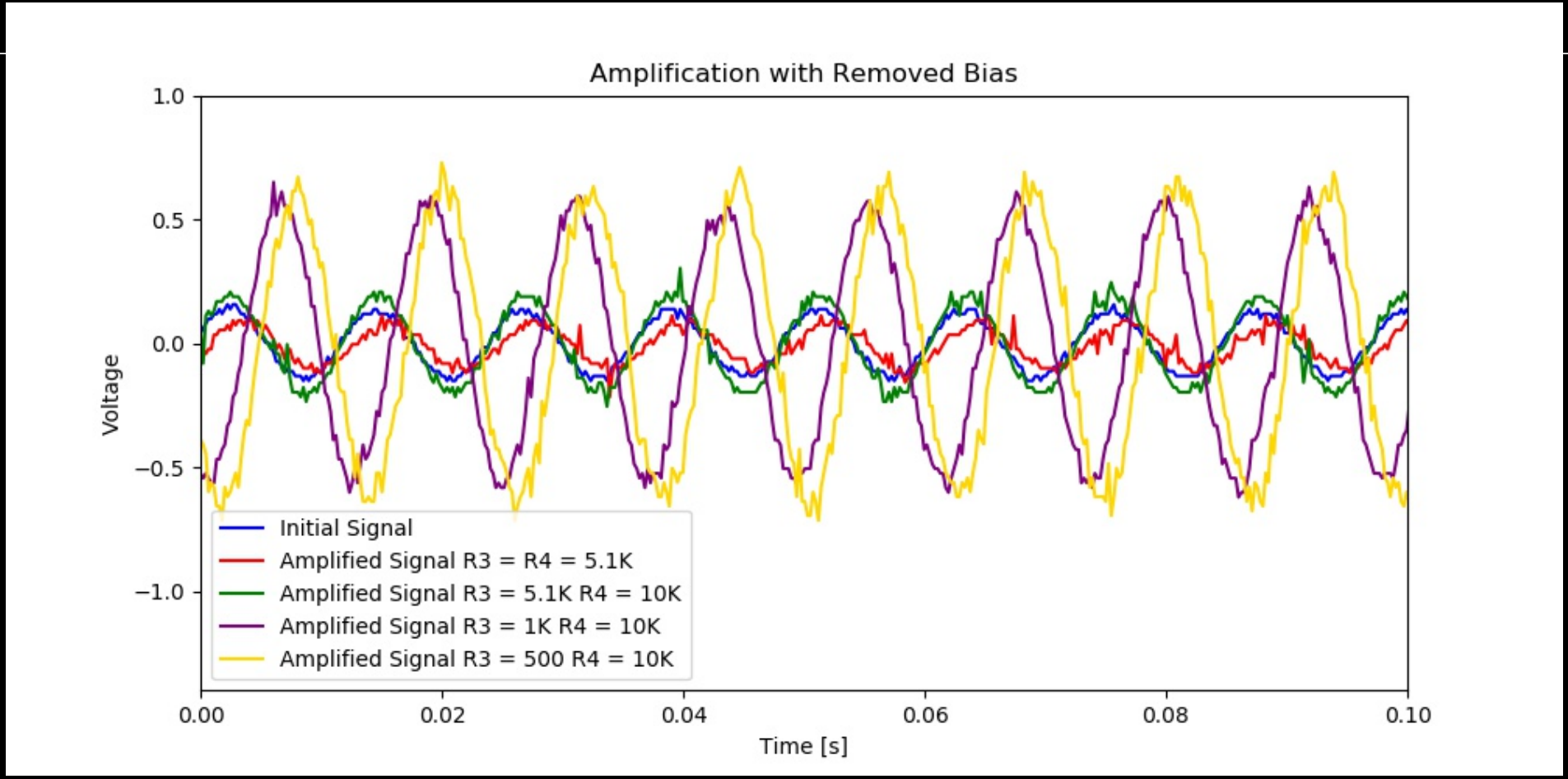
Prototyping & Analysis

Amplification

Final Pedal Effect – Low Frequencies, High Frequencies

Common Emitter Error Analysis

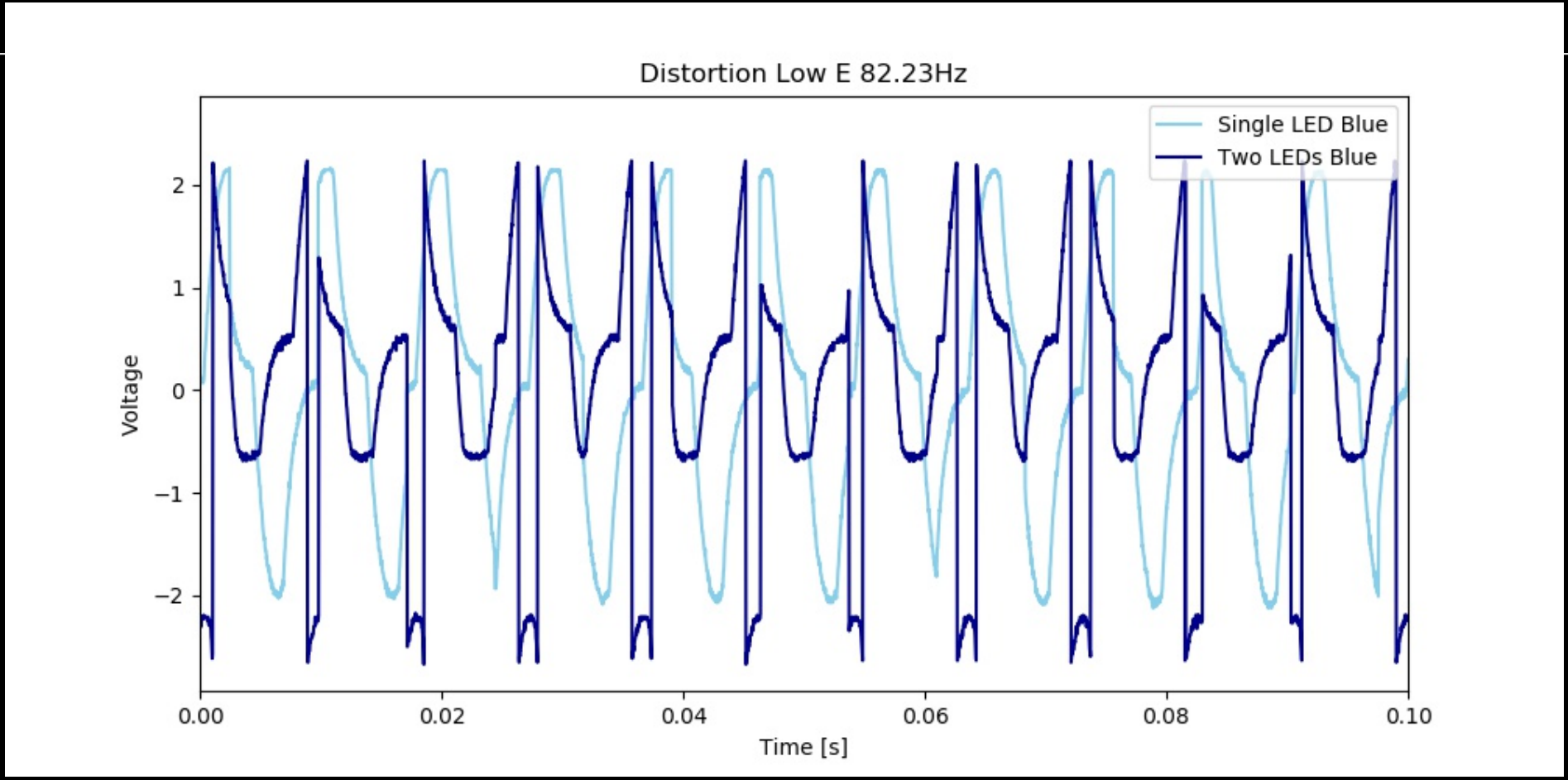
Amplification



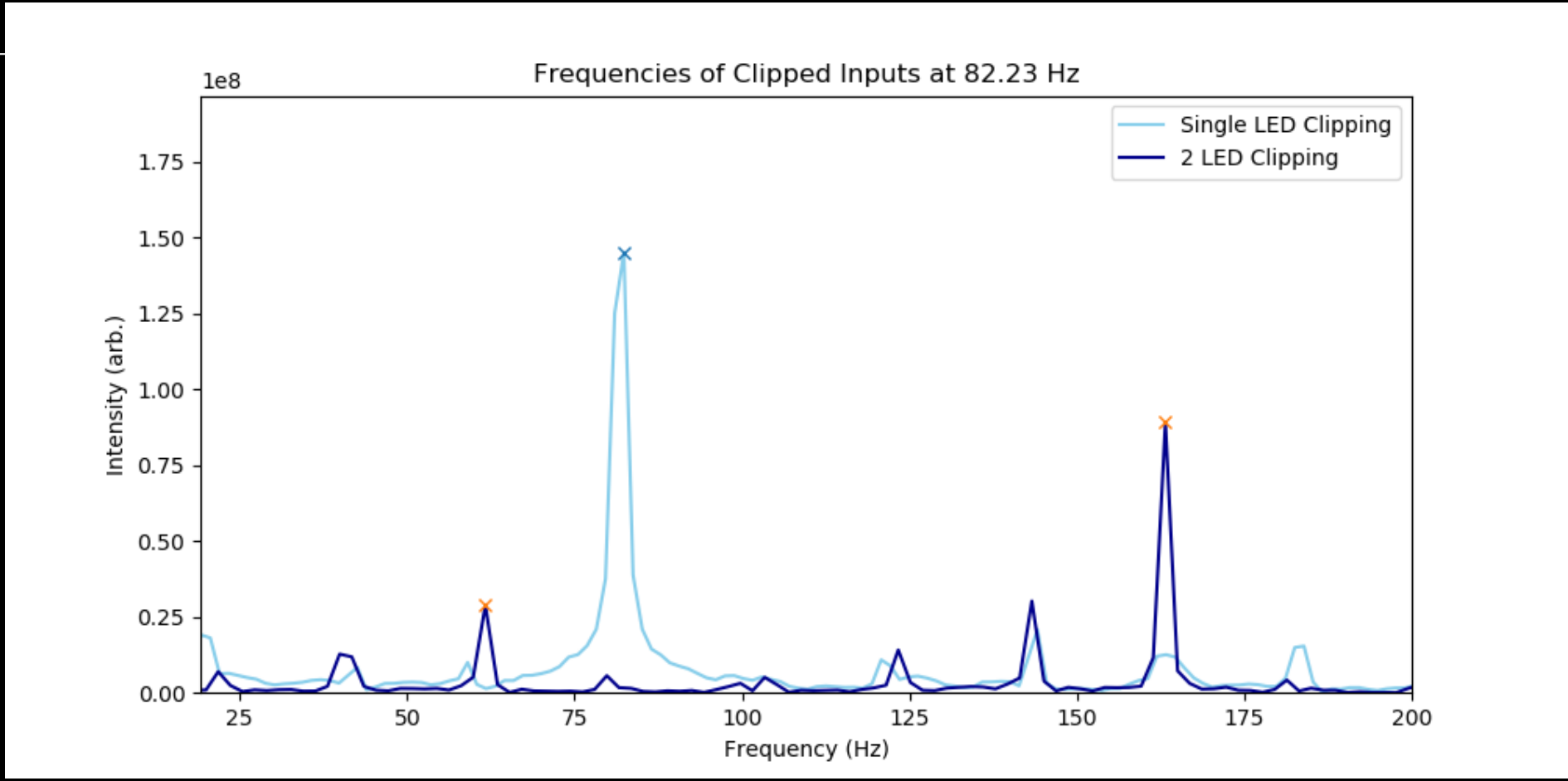
Amplification

Component Name	Voltage Across Each Component [V] ± Least Count
R1	7.54±.01
R2	.873±.001
R3	1.54±.01
R4	7.44±.01
C1	.011±.001
C2	1.13±.01
Vce	1.58±.01
Vbe	.601±.001

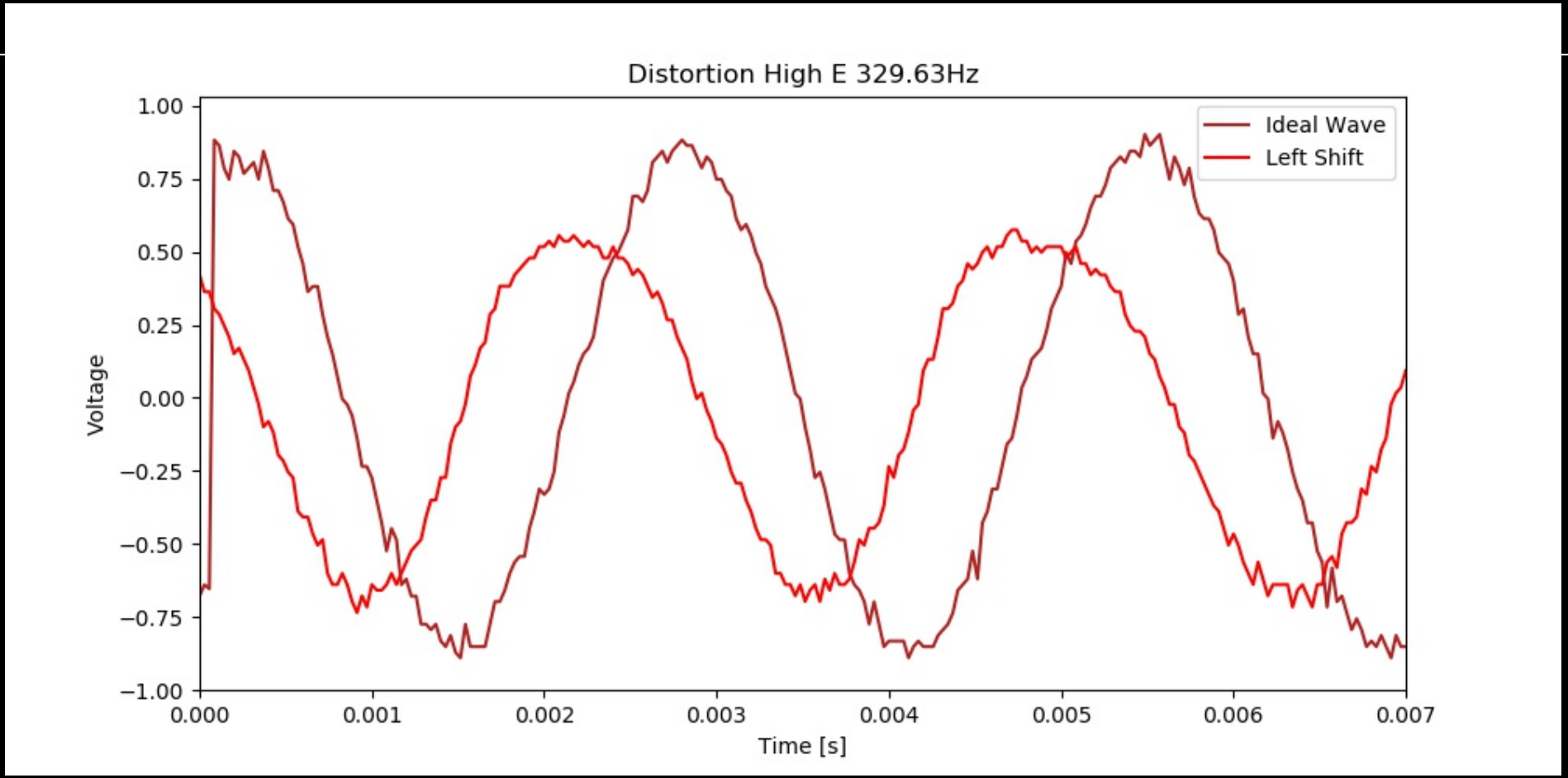
Solid State Distortion



Frequency of Harmonics

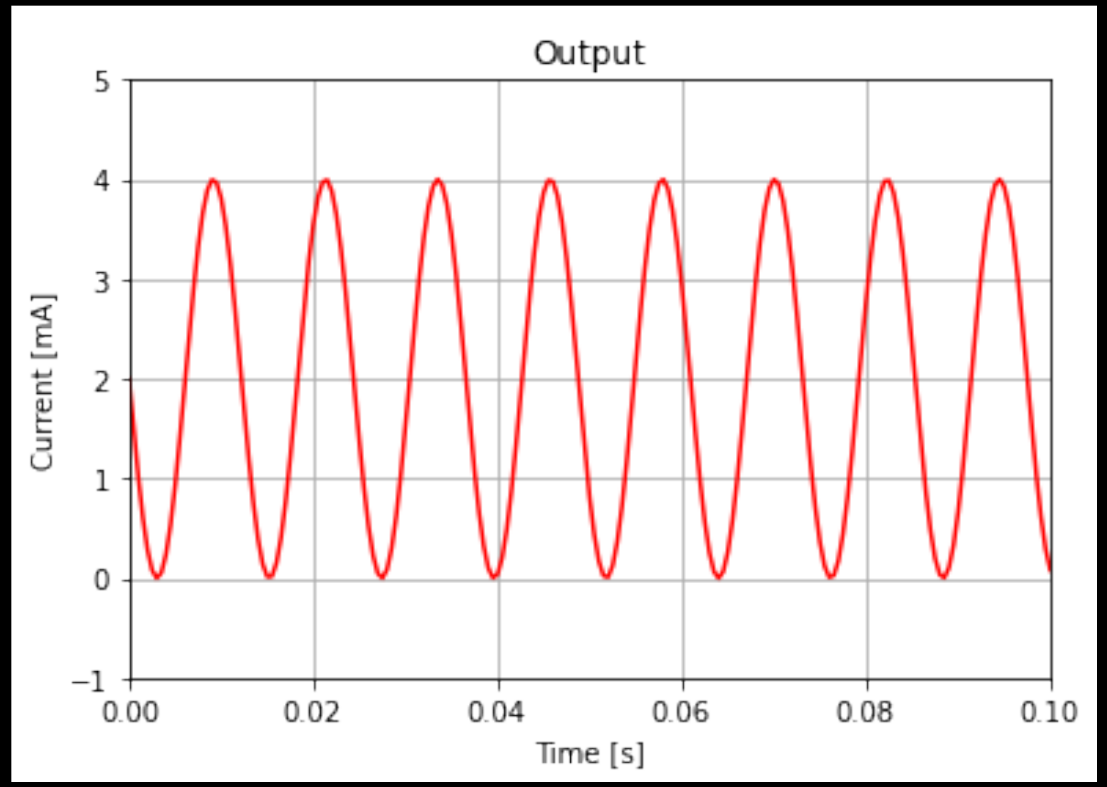


Solid State Distortion

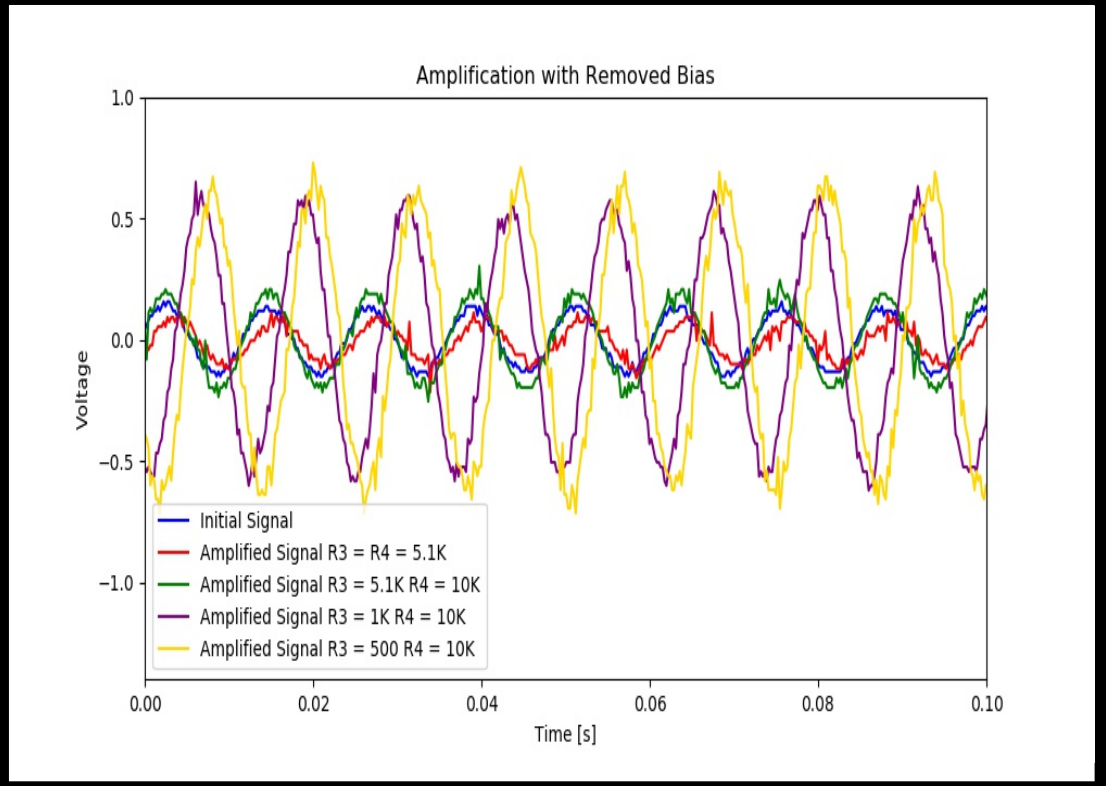


Comparison with Expected Results

Expected Amplification

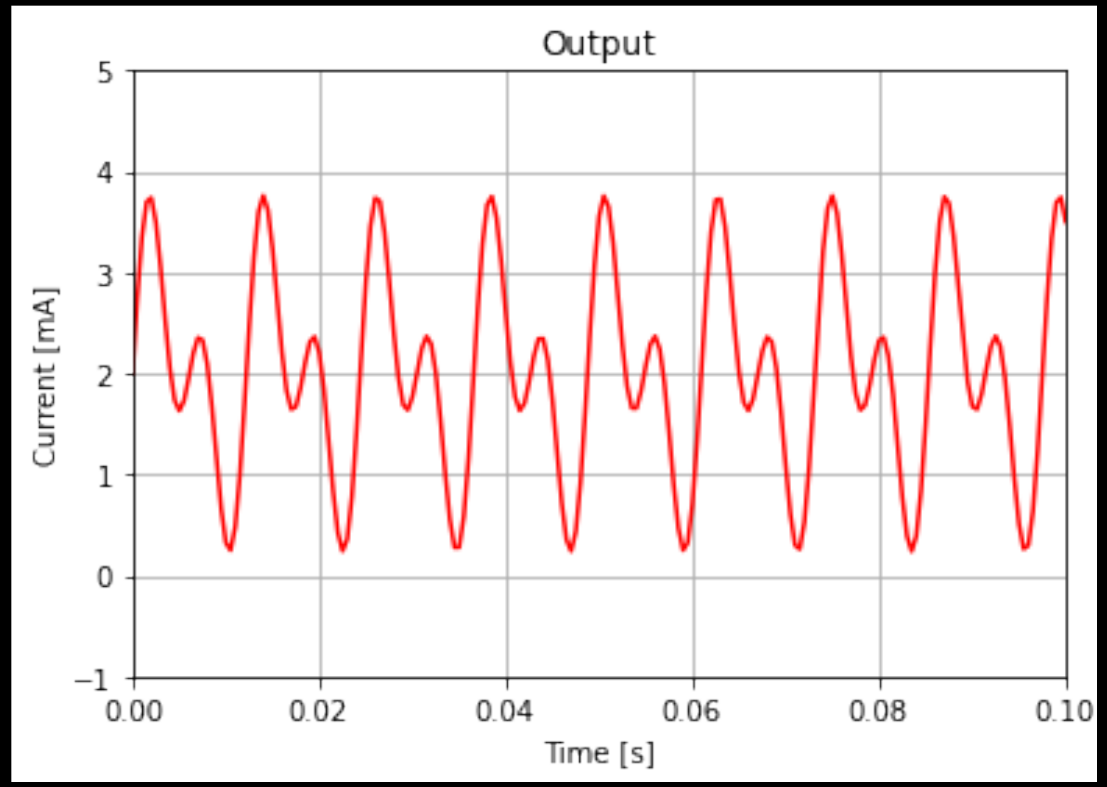


Experimental Amplification

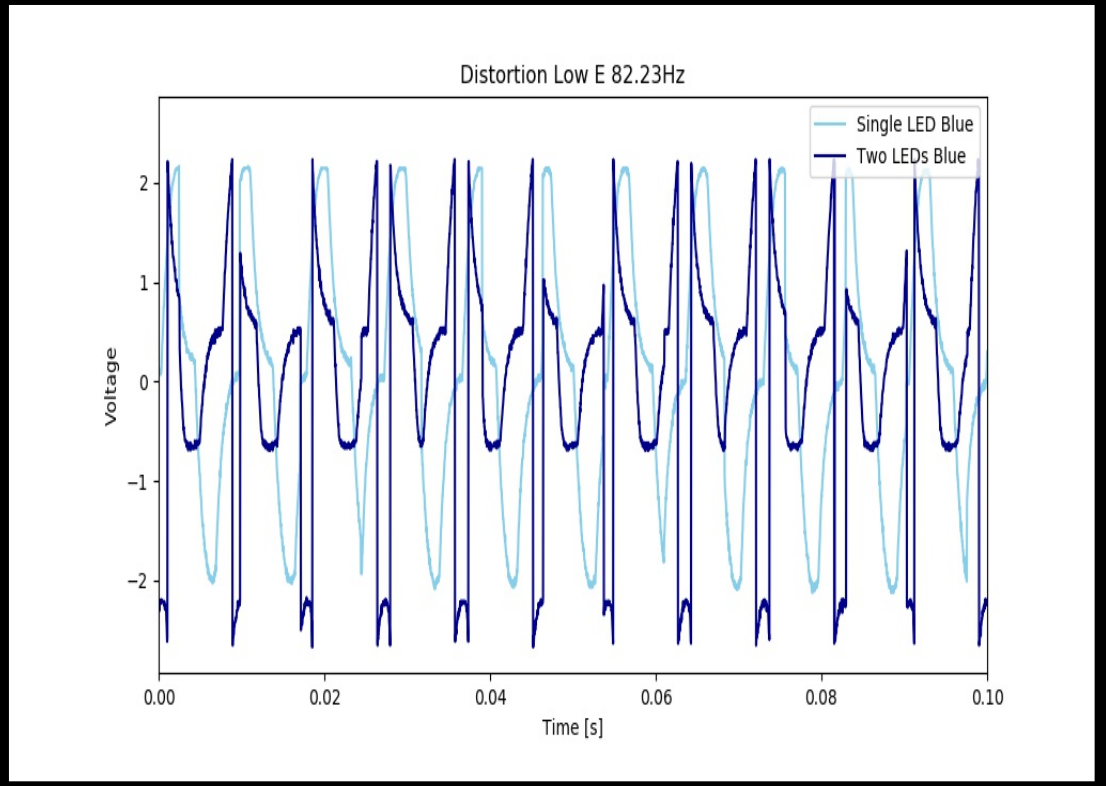


Comparison with Expected Results

Expected Distortion



Experimental Distortion



Common Emitter Error Analysis

- To compute the error in this experiment (and any recreated hereafter) the voltage across each component should be recorded. Resistors and Capacitors have a 5% value guarantee.
- $V = IR \rightarrow \frac{\delta V}{\delta R} = I$, where I can be calculated using V/R
- $V = \int iCdt \rightarrow \frac{\delta V}{\delta C} = \int idt$, where I is the alternating current
- Voltage across a transistor is dependent on the saturation current, not an internal resistance. We will assume an additional uncertainty of 5% for these calculations.

Common Emitter Error Analysis

Component Name	Voltage Across Each Component [V]	δV/δx
R1	6.54	6.54×10^{-6}
R2	.873	1.746×10^{-5}
R3	1.54	3.08×10^{-3}
R4	7.44	7.44×10^{-4}
C1	.011	sin(2*pi*t) -> .006283
C2	1.13	sin(2*pi*t) -> .006283
Vce	1.587	1
Vbe	.601	1

Common Emitter Error Analysis

- The reproducibility of the voltage measured for the common emitter circuit can be calculated by the following:

$$P = [\sum_{i=0}^{n} \frac{\delta V}{\delta x_i}]^{\frac{1}{2}} = [\sum_{i=0}^{7} (\frac{\delta V}{\delta x_i} P_i)^2]^{\frac{1}{2}}$$

$$= \sqrt{(.05 * 6.54\text{x}10{-}6\ )^2 + (.05 * 1.746\text{x}10{-}5\ )^2 + (.05 * 3.08\text{x}10{-}3\ )^2 \ldots}$$

$$\sqrt{\ldots + (.05 * 7.44\text{x}10{-}4\ )^2 + (.05 * .006283)^2 + (.05 * .006283)^2 + (.05 * 1)^2}$$

=.084V

Audio

Switch over to audio: Westcoast Collective, Dominic Fike

Conclusions and Future Testing Plans

Conclusions

System Level Testing

Mechanical Elements

Electrical Elements

Software Elements

Conclusions from Testing

- Solid State Distortion is possible with the main components of an Arduino kit
- Common Emitter Circuits can be used to specifically bias a voltage to the optimal level of the Arduino's input window
- Analog read speeds can be shaved by neglecting Conversion to ascii
- The error of the signal recorded by the common emitter is .08V
- However, this pedal leaves much to be desired
  - Additional distortion should be added over a higher range of frequencies
  - This could be achieved with the following additions

Mechanical Elements – Future Testing

Parts!

- Soldering Iron would help with further analysis
  - Connections for 9V Battery and ¼” Mono-Audio Jack are not solid
  - Twisted Wires lead to losses
  - Preventing the most accurate transmission of signal

Electrical Elements

The Electrical Hardware in this system met all of the desired requirements

- As inherent properties of the common emitter amplifier with the transistor in saturation, the signal was biased, inverted, and amplified
- Both circuits were compatible with guitar and computer inputs, and amplifier and Arduino outputs
- An audible effect was created by this pedal, thanks to the use of clipping diodes for a variation of solid-state distortion
- However, this distortion is not nearly noticeable at frequencies exceeding 300Hz, and successive iterations would be required to successfully integrate this range

Electrical Elements – Future Testing

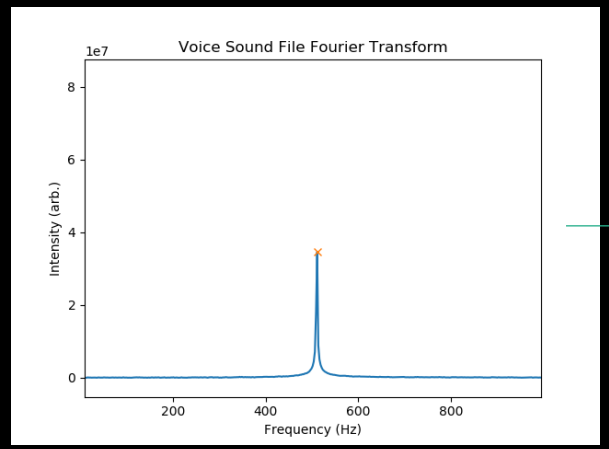
This design could be improved in the future by:

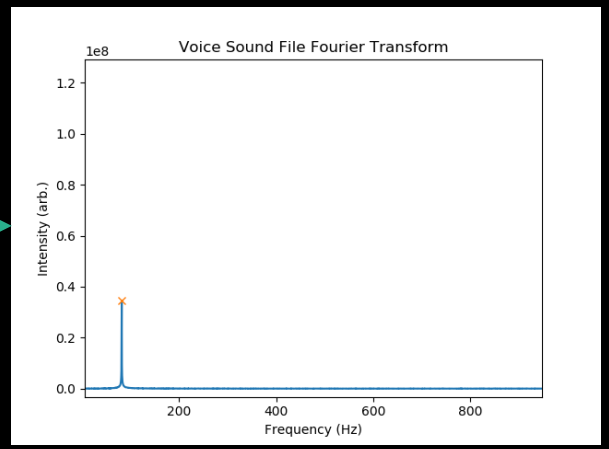
- Including other transistors
  - Transistors in series can provide an additional amount of distortion and amplification
  - Higher beta values will also allow for increased amplification
- Op-Amps
  - Having components specifically designed to amplify signals is an obvious benefit when it comes to this kind of analysis
  - Allows for additional amplification and other kinds of distortion (more fuzz, tone control, etc.)
- Real Diodes
  - LEDs are diodes but have a fairly low threshold voltage
  - Including other kinds of diodes in future experiments can allow for more pronounced clipping

Software Elements

Software requirements were met in this experiment.

- The Arduino was used to read in wave form data
- The sampling rate was determined by taking the Fourier Transform of a file recorded with an 82.23 Hz input, and manipulating the sampling rate until the frequency obtained matched this value [4453 Hz]





Software Elements

- Additionally, python was used to recreate the waveforms
- Bias was removed, and amplitude was recorded
- Additionally, in the expected results section, the second harmonic frequency was superimposed to imitate the clipped waveform
- Fourier Transforms were taken to represent the frequency recorded by these waveforms

Software Elements – Future Testing

- This software could be improved by a more thorough understanding of serialplotter.py
- This code was created by T.A. Javier Carmoa, and was used with his consent for this project
- It involves bit-shifting the incoming analog signal, and converts the recorded voltage into an 8-bit character
- To calibrate the conversion factor, the Arduino's 3.3V pin (which really yielded a 3.36V potential) was read in through this program, and then reduced to its correct value in 3 trials. The average was then taken.

References

- https://www.edn.com/for-signal-distortion-phase-matters/
- https://www.dummies.com/programming/electronics/components/electronics-components-amplify-with-a-transistor/
- https://www.howtogeek.com/64096/htg-explains-how-do-guitar-distortion-and-overdrive-work/
- https://www.coda-effects.com/2015/05/what-is-guitar-signal.html
- https://www.electronics-tutorials.ws/amplifier/emitter-resistance.html
- https://media.digikey.com/pdf/Data%20Sheets/ON%20Semiconductor%20PDFs/PN2222.pdf
- https://soundbridge.io/tube-distortion/

References

- https://www.premierguitar.com/articles/21291-build-your-own-stompbox?page=6
- https://circuitdigest.com/electronic-circuits/guitar-distortion-pedal-circuit-diagram
- https://forum.arduino.cc/index.php?topic=598823.0
- https://cs50.stackexchange.com/questions/4756/recover-buffer-as-uint8-t-instead-of-int
- https://www.guitarworld.com/gear/how-does-a-guitar-pickup-really-work
- https://electronics.stackexchange.com/questions/156301/how-does-a-vacuum-tube-amplifier-work
- https://www.musiciansfriend.com/thehub/difference-between-tube-and-solid-state-guitar-amps