Assignment 2

Digital Voice Recorder

EGB240

Adam Telfer
N12068373

Executive Summary

This report details the design, implementation, and evaluation of input signal conditioning circuitry for a digital voice recorder, built to interface with a QUT-designed development board for a Teensy 2.0 microcontroller. The circuitry was developed to meet the Teensy 2.0's specifications, including its 5V power supply, 15.625 kHz sampling rate, and 8-bit sampling depth.

The core objective was to accurately record human speech, which was done in the narrowband frequency range (300 Hz to 3.4 kHz) with the provided microphone. This necessitated an anti-aliasing filter to attenuate signals above 7.8 kHz by at least 48 dB to prevent recording errors. A Chebyshev filter response was chosen for its good attenuation, thus minimising component count. The circuit utilised a TL974 operational amplifier to implement Voltage-Controlled-Voltage-Source (VCVS) active low-pass filters in series.

The circuit design included the microphone interfacing; applying a 2.5 V DC offset to the microphones signal using a LM358P operational amplifier; calculating the details of the anti-aliasing filter; and using an inverting amplifier with a gain of 100 to amplify the signal for the Teensy's Analog-to-Digital-Converter (ADC).

The circuit design was simulated in LTSpice to verify it performance with ideal and non-ideal components and prototyped on a breadboard to experimentally verify its performance. The circuit was demonstrated to have approximately 50 dB of attenuation at 7.8 kHz, thus meeting specifications dictated by the ADC.

While the primary aim of designing, implementing, and evaluating the input conditioning circuitry was achieved for narrowband speech recording, future improvements could include more accurate microphone testing and exploring alternative filter responses to accommodate the wideband specification.

Table of Contents

1. Introduction

The task is to develop the input signal conditioning circuitry for a digital voice recorder, with the process to design, implement and evaluate the circuit being documented. The circuit must interface with QUT-designed development board for a Teensy 2.0 microcontroller board, with the output of the conditioning circuit going to one of the Teensy's analog to digital converters (ADC).

The circuit should be designed according to sound engineering principles and optimised within the limitations imposed by the ADC and provided components. The input signal conditioning is made up of the provided microphone and the circuitry to interface with it, and an analog input conditioning circuit to make the signal output from the microphone suitable for the ADC. A TL974 quad operational amplifier (op-amp) [1] and Electret condenser microphone [2] are provided and thus will be used in the design. The Teensy 2.0 microcontroller is powered from USB-C which provides it with 5 V, so the analog conditioning circuit must operate at 5 V. The Teensy's ADC has a sampling frequency of 15.625 kHz and a sampling depth of 8 bits, so the conditioning circuit must adhere to the requirements that this creates.

The project is to develop a prototype of the analog conditioning circuitry, with the other parts of the DVR being provided. The prototype will be implemented on a breadboard to evaluate its performance and will not be followed up with a PCB design. The DVR will capture human speech at an acceptable quality, i.e. it is not going to record high fidelity audio. It is assumed that minimal componentry should be used to achieve this acceptable quality.

The report follows the steps of the process, being the background and literature review, circuit design, simulation in LTSpice, experimental results and conclusions. The background Information and Literature Review covers the specifications, constraints, theory and general hardware implementation of the circuit. The circuit design involves calculations with and without MATLAB to calculate the specific hardware details such filter order and component values. The filter topology and component values are then simulated in LTSpice to verify its function before being implemented on a breadboard. The simulation takes the desired filter topology and components values and outputs the expected performance of the whole circuit. The experimental results are where performance of a prototype is measured, evaluated, and compared to the simulations. Conclusions and further improvements will discuss how the process of designing, implementing and evaluating the analog conditioning circuitry went and what improvements to the process can be made.

2. Background and Literature Review

The analog conditioning circuit needs to conform to the specifications of the ADC, while adhering to sound engineering design principles. This can be done by researching human speech and audio recording standards, the effect the ADC's requirements have on the recording of speech, and the hardware implementation to accommodate the ADC, TL974 and Electret Condenser Microphone. The circuit design fed back into research as a 6th order filter was needed requiring an additional op-amp, with the LM358P [3] selected due to availability, low cost [4] and working with a 5 V supply voltage.

2.1. Speech Frequency and Pressure

The digital voice recorder is intended to record the human voice, so it is important to understand which sound frequencies are needed to record human speech. Human speech ranges from approximately 100 Hz to 7 kHz and hearing 20 Hz to 20 kHz, with normal conversation having a sound pressure level of 60-70 dB SPL [5, pp. 7-10, Fig. 4]. The telephony industry has standards for the speech frequencies recorded and transmitted, being narrowband at 300 Hz to 3.4 kHz, wideband at 50 Hz to 7 kHz, super-wideband at 50 Hz to 14 kHz, and fullband at 20 Hz to 20 kHz [6, pp. 14, 21, 33, 39].

2.2. Aliasing and Anti-Aliasing Filter Requirements

When converting an analog signal to digital an ADC is used [7, pp. 900-902], whose characteristics must be considered. These are the sampling depth and sampling rate and will dictate some of the requirements of the lowpass filter section.

The sampling depth is how many discrete levels the analog signals amplitude can be divided into, being 2^n values where n is the number of bits of the ADC, and is called the dynamic range, approximately equal to 6n dB. Since 2^8 equals 256, any unwanted signals must be less than 1/256 of the peak-to-peak voltage (V_{PP}) range of the ADC.

The unwanted signals are given by the sampling rate of the ADC. If a sine wave (such as the signal from the microphone) is sampled at less twice its frequency, the points recorded will trace out a false signal with an incorrect frequency, which is called aliasing [7, pp. 901, Fig. 13.23]. To avoid aliasing the maximum frequency input to the ADC must be half the sampling frequency, known as the Nyquist frequency. To remove any signals above the Nyquist frequency an anti-aliasing lowpass filter is used [7, p. 395]. The Nyquist frequency is half the sampling frequency, shown in equation (1). Any signals above 7.8 kHz must be attenuated to 1/256 the voltage range of the ADC, which can be converted to gain in equation (2).

$$Nyquist\ Frequency = \frac{f_{samp}}{2} = \frac{15.625\ kHz}{2} \approx 7.8\ kHz \tag{1}$$

$$Gain_{min} = 20\log_{10}(1/2^n) = 20\log_{10}(1/2^8) \approx -48\ dB \tag{2}$$

2.3. Filter Response

When choosing the type of filter response to use, its gain verses frequency is an important aspect, with different regions and points defined when identifying its characteristics. The passband is the region in which frequencies are relatively unattenuated, and the passband (or cutoff) frequency

is the beginning of the transition region, where attenuation increases significantly. The end of the transition region is the stopband frequency and stopband where the frequencies are heavily attenuated [7, p. 399]. By specifying the gain allowed in the passband, and needed in the stopband, along with the passband and stopband frequencies, the different filter responses can be compared.

Butterworth, Chebyshev and Bessel filter responses provide different advantages and disadvantages and can be implemented using a VCVS lowpass filter [7, p. 407]. The Butterworth and Bessel both offer a flat response in the passband but come at the cost of a less step transition between the passband and stopband compared to the Chebyshev [7, pp. 401-402, Fig. 6.21]. Chebyshev has a steeper transition at the cost of ripple, where the gain oscillates up and down in the passband around a point with a larger allowance for ripple having a steeper transition [7, pp. 408-409, Fig. 6.30]. However, "If this ripple is considered simply in terms of the human ear's sensitivity to amplitude changes, then a passband ripple specification of, say, ±0.5dB might be just acceptable" [8].

There is also the Elliptical (or Cauer) filter which requires a lower filter order for a given attenuation but are more complex [9, p. 78]. Table 3.26 in Analog and Digital Filter Design [9, p. 118] shows the order of Elliptical filter needed for a given ratio of stopband frequency to passband frequency. With the ratio of 7.8 kHz to 7 kHz (to achieve the wideband specification) being 1.1, requires an 8th order filter to achieve a 49 dB attenuation.

The effects of component tolerance will need to be accounted for when designing the filter [7, pp. 402, Fig. 6.22], as it will change the frequency response and potentially take a filter designed for 48 dB of attenuation with ideal components below the requirements.

The transfer functions for the Butterworth and Chebyshev responses and how to calculate them can be found in Active Filter Design [10, pp. 13-25]. An nth order filter will have a nth order polynomial that can be factored into it roots [10, pp. 17, Table 2.1, 2.2, 2.3]. The complex conjugate roots can be multiplied together to form quadratics shown in Example 2.4 [10, p. 24] to get the equation.

$$H(s) = \frac{1}{s^2 + a_1 s + a_2} \tag{3}$$

Where the coefficients a_1 and a_2 can be matched to the coefficients for the standard form of a 2nd order lowpass filter in equation (5). There are also useful functions in the MATLAB Signal Processing Toolbox [11] and Control System Toolbox [12] to help with the design of analog filters.

2.4. Filter Topology

The VCVS filter is a "simple active lowpass design" that "can be used for Bessel, Butterworth and Chebyshev responses" [9, p. 133]. The Texas Instruments Application Report SLOA024B provides the ideal lowpass VCVS transfer function [13, pp. 4, Eq. 9] that has been rewritten in equation (4) so that s^2 has a coefficient of 1. It also provides "Simplification 2" [13, p. 5] for calculating component values but has been modified since ω_n and Q are known and C and m are set. The coefficients of equations (4) can be equated with the coefficients of the standard form for a 2nd order lowpass filter in equation (5) [14, p. 37]. Equations (6) to (10) show how the equated coefficients can then have their values substituted and rearranged to solve for n and R in terms of ω_n, Q, C and m, thus allowing R_1, R_2, C_1, C_2 to be calculated.

By changing m, R_1, R_2 and C_2 can be tuned so that they closely match E12 series values, which will be referred to as their rounded values. Taking these rounded values and putting them into equations (6) and (8) to calculate the actual ω_n and Q for those component will produce, the error between the ideal and calculated values of ω_n and Q can be found for a given value of m. By iterating through values of m a set of components with a low error can be found.

$$H(s) = \frac{\frac{1}{C_1 C_2 R_1 R_2} K}{s^2 + s\left(\frac{1}{C_2 R_1} + \frac{1}{C_2 R_2} + \frac{1-K}{C_1 R_2}\right) + \frac{1}{C_1 C_2 R_1 R_2}} \tag{4}$$

$$H(s) = \frac{\omega_n^2}{s^2 + s\frac{\omega_n}{Q} + \omega_n^2} \tag{5}$$

Letting $\mathrm{R}_1 = \mathrm{mR}, \mathrm{R}_2 = \mathrm{R}, \mathrm{C}_1 = \mathrm{C}, \mathrm{C}_2 = \mathrm{nC}, \text{and } \mathrm{K} = 1$

$$\omega_n = \frac{1}{\sqrt{C_1 C_2 R_1 R_2}} = \frac{1}{\sqrt{CnCmRR}} = \frac{1}{\sqrt{mnR^2C^2}} = \frac{1}{RC\sqrt{mn}} \tag{6}$$

$$\frac{\omega_n}{Q} = \frac{1}{C_2 R_1} + \frac{1}{C_2 R_2} + \frac{1-K}{C_1 R_2} = \frac{1}{Q}\frac{1}{\sqrt{C_1 C_2 R_1 R_2}} \tag{7}$$

$$Q = \frac{\sqrt{C_1 C_2 R_1 R_2}}{C_1 R_2 + C_1 R_1} = \frac{RC\sqrt{mn}}{RC(1+m)} = \frac{\sqrt{mn}}{m+1} \tag{8}$$

$$n = \left(\frac{Q(1+m)}{\sqrt{m}}\right)^2 \tag{9}$$

$$R = \frac{1}{\omega_n C\sqrt{mn}} \tag{10}$$

Matching coefficients between equation (3) and (5) to get the Q and ω_n

$$a_2 = \omega_n^2 \Rightarrow \omega_n = \sqrt{a_2} \tag{11}$$

$$a_1 = \frac{\omega_n}{Q} \Rightarrow Q = \frac{\omega_n}{a_1} = \frac{\sqrt{a_2}}{a_1} \tag{12}$$

2.5. Operational Amplifier Specifications

The inputs of an op-amp have a certain voltage range at which it will operate correctly and will be less than the supply voltage [7, p. 245]. The supply voltage to the op-amps will be 0 V to 5 V, and at V_{CC} = ±2.5 V the TL974 has a "Common-mode input voltage" of ±1.35 V [1, p. 5] which translates to 1.35 V to 3.85 V, and the LM358P a "Common-mode voltage range" of V_{CC-} to V_{CC+} - 1.5 which translates to 0 V to 3.5 V [3, p. 10].

An op-amp typically "cannot swing its output closer than a volt or two from either supply rail" [7, p. 246], so this limitation will need to be considered when amplifying a signal from 0 V to 5 V, otherwise the signal will be clipped at the top and bottom. The TL974 has a "Rail-to-Rail Output Voltage Swing" of ±2.4 V at V_{CC} = ±2.5 V [1, p. 1], whereas the LM358P has a "Voltage output swing from rail" of 5 mV of the negative rail and 1.5 V for the positive rail [3, p. 10] making it unsuitable to amplify the signal.

Due to the non-ideal nature of op-amps having current flow in or out of the inputs, called the "input bias current", and cause the output to have an unwanted DC offset if the resistances "seen" by both inputs are not matched. This will create an issue with when amplifying the signal as a DC offset not at 2.5 V may cause the signal to be clipped. This can be solved by using a compensating resistor [7, pp. 252, Fig. 4.55].

The maximum gain of an op-amp is dependent on the frequency it is operating at, with the frequency at which it has unity gain called the gain bandwidth product. The maximum gain an op-amp can have with a given bandwidth (the range of frequencies it operates at) is given by $Gain = Gain\ Bandwidth\ Product * Bandwidth$ [7, pp. 247, Fig. 4.47]. The TL974 has a "Gain bandwidth product" of 12 MHz [1, p. 5], so for a bandwidth of 7.8 kHz the maximum gain is approximately 1500.

2.6. Electret Condenser Microphone

The Electret microphone has a close to flat frequency response between 100 Hz to 3 kHz and only rising by a few dB at 3.4 kHz [2, p. 2], so it's response can be assumed as equal at all frequencies of interest. The data sheet indicates that the "current consumption" at V_S = 1.5 V_{DC}, R_L = 1.5 kΩ will be 0.5 mA, meaning there will be a voltage drop across R_L, so the voltage at the microphones positive pin will be determined by the value of R_L. Due to the common-mode input voltage of the op-amps and the signal needing to be centred at 2.5 V when amplified there is a need for circuitry to set the DC offset of the signal at 2.5 V.

2.7. Conclusions

With the literature review and research done important conclusions that inform the circuit design can be made.

The Nyquist frequency of 7.8 kHz means that only narrowband and wideband speech frequencies can be achieved, with the wideband requiring a very steep transition from the passband to the stopband.

A Chebyshev filter response would likely be the most suitable as it will provide a steeper transition while still having an acceptable amount of ripple in the passband.

The design of the anti-aliasing filter informed that an additional op-amp would be required, with a review of the data sheets for candidate op-amps indicating that the TL974 would be best for the amplifier and the lower performance LM358P used for the voltage reference section.

Testing of the microphone must be done to determine it's expected peak-to-peak voltage so that the required gain of the amplification stage can be determined.

3. Circuit Design

3.1 Microphone Interfacing

The microphone interfacing is based on the application circuit in the microphones data sheet [2, p. 2]. On the application circuit R_L is 1.5 kΩ for a V_{dc} of 1.5 V, but the supply voltage for this circuit is 5 V, so experimentation of different values R_L will be done to get the voltage at the microphones positive pin close to 2.5 V. The DC blocking capacitor will behave as a passive highpass RC filter. If the cutoff frequency is set at 300 Hz in line with narrowband specifications it will have -3 dB at that point which is not wanted. Instead we can design for -0.5 dB at 300 Hz, by finding the cutoff frequency using equations (14) and (15) at 105 Hz, and then the capacitor value with equations (16) and (17) at 152 nF, which is rounded down to 150 nF.

3.2 Voltage Reference

The signal that travels through the anti-aliasing filter and is then amplified needs to be centred at 2.5 V. This can be done by passing the signal through a unity gain inverting amplifier [7, pp. 249, Fig. 4.50] with a reference voltage of 2.5 V applied at the non-inverting input. The LM358P is used here since its output voltage cannot get as close the supply rail voltages as the TL974. The need for output voltages closer to the supply rail voltages is discussed in the amplification section analysis.

3.3 Antialiasing Filter

MATLAB was used to decide between using a Butterworth or Chebyshev response, and the filter order needed to meet the specifications. This was done with the MATLAB script in Appendix F.1, which takes the stopband frequency at 7.8 kHz, maximum passband ripple of 0.1 dB (explained soon), and minimum stopband attenuation of 48 dB. It increases the passband frequency from 100 Hz to 7.8 kHz in increments of 10 Hz, checking the required order of Butterworth and Chebyshev filter needed. It takes those results and finds the maximum frequency for a given order of filter and creates a plot, seen in Figure 3.1 for the first 20 orders. See Figure B.2 for the full plot, as the order grows exponentially as the passband frequency approaches the stopband frequency. A bode plot for the frequency response of the first 20 Chebyshev filters can be seen in Figure B.3, showing the diminishing returns of increasing the filter order.

This allows the feasibility of achieving narrowband and wideband to be assessed. A 9th and 69th Butterworth filter would need to achieve narrowband and wideband respectively. A Chebyshev filter would be a 6th and 18th order to achieve the same attenuation as the Butterworth. Thus, the narrowband speech frequency will be used with a Chebyshev filter, as even an 18th order filter is excessive and wideband speech not strictly necessary for this design. Now that the passband frequency is set at 3.4 kHz, the stopband attenuation for a given order can be calculated and plotted in Figure 3.2 which shows for a 6th order Chebyshev filter will have 54 dB of attenuation at 7.8 kHz, a suitable margin from the minimum of 48 dB. This explains the decision for 0.1 dB ripple, as a 6^{th} order filter has enough margin in stopband attenuation that the passband ripple can be decreased to give some margin to help keep it under 0.5 dB when using real components.

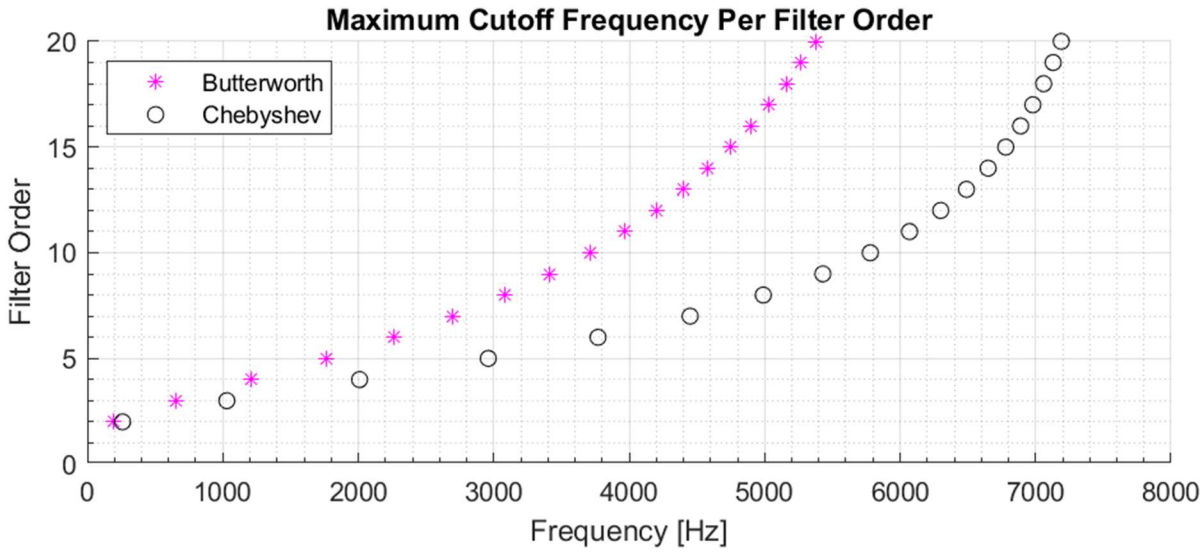

Figure 3.1: The maximum frequency that Butterworth and Chebyshev filters of order 1 to 20 can achieve.

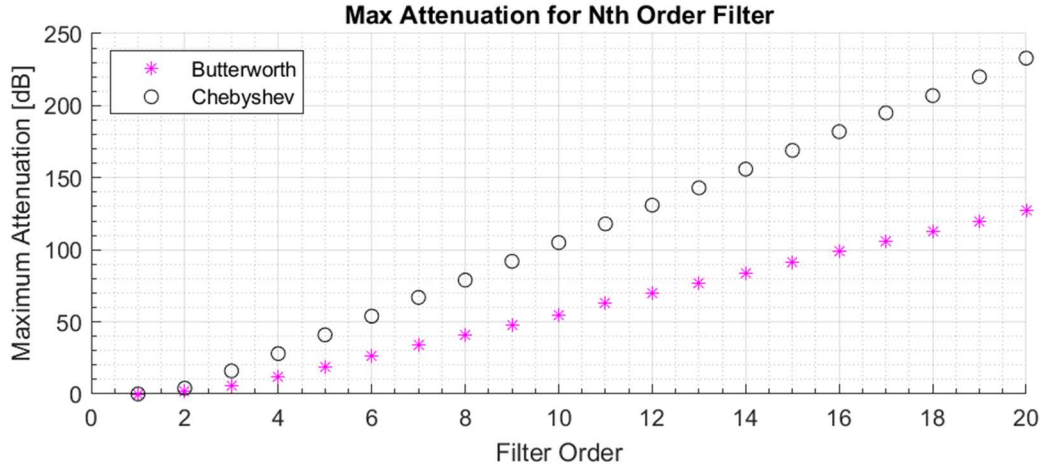

Figure 3.2: The maximum attenuation that a Butterworth and Chebyshev filter of orders 1 to 20 can achieve.

Now with the passband frequency set at 3.4 kHz, the MATLAB Script in Appendix F.2 can be used. Using functions from the MATLAB Signal Processing Toolbox and Control System Toolbox, the transfer functions poles can be found.

```
[n_cheb, wn_cheb] = cheb1ord (wp, ws, Amax, Amin, 's'); % Order and natural freq
[b, a] = cheby1(n_cheb, Amax, wn_cheb,'low','s'); % Transfer func coefficients
H = tf(b,a); % Transfer Function
[z, p, k] = tf2zpk (b, a); % Get the poles
```

This allows calculation of each stage's natural frequency and Q-factor based off equations (11) and (12). The natural frequency and Q-factor are input as arguments for the custom MATLAB function (component_calculator_vcvs) in Appendix F.3 that applies equations (9) and (10), and outputs a table of tuned component values for each stage, with all three tables being Table D.2, Table D.3 and Table D.4.

```
a1A = -p(1)-p(2);
a2A = p(1)*p(2);
wnA = sqrt(a2A);
QA = wnA/a1A;
fprintf(1,'Stage A: wn = %g rad/s, Q = %g\n',wnA,QA);
stageAresults = component_calculator_vcvs(QA, wnA, C_base, true, false);
```

Table 3.1 is an excerpt from Table D.1 showing the lines selected for each stage. K is the tuning iteration, wn_ack and Q_ack the natural frequency and Q-factor for the stage calculated from the rounded component values, and wn_error and Q_error the percentage error between the ideal and calculated values. R1 to C1 are the ideal component values and R1_rnd to C2_rnd are the closest E12 values. Not seen are additional columns for the percentage error between the ideal and rounded component values. Full tables for component tuning are Table D.2, Table D.3 and Table D.4.

Table 3.1: Excerpt from Table D.1 generated in MATLAB showing component values for stage A, B and C

k	wn_ack	Q_ack	wn_err	Q_err	Avg_wn_Q_err	R1	R2	C1	C2	R1_rnd	R2_rnd	C1_rnd	C2_rnd
12	22587	4.6801	0.51224	1.0197	0.76596	1038.3	3283.3	2.2e-09	2.587e-07	1000	3300	2.2e-09	2.7e-07
23	17277	1.3288	3.0859	0.21146	1.6487	1898.7	17250	2.2e-09	4.367e-08	1800	18000	2.2e-09	4.7e-08
16	10990	0.60822	0.24788	1.4608	0.85433	12260	56905	2.2e-09	5.421e-09	12000	56000	2.2e-09	5.6e-09

Each line was chosen to minimise the natural frequency and Q-factor error, with a greater weight put on the Q-factor to better maintain the shape of the frequency response of the stage. The rounded component values are used to calculate the transfer function of the filter, which is plotted against the theoretical transfer function calculated, shown in Figure 3.3, with Figure B.1 zoomed in at the passband frequency. The same plot but with a large error in the natural frequency and Q-factor for each stage can be seen in Figure B.4, showing the effect of poor component tuning.

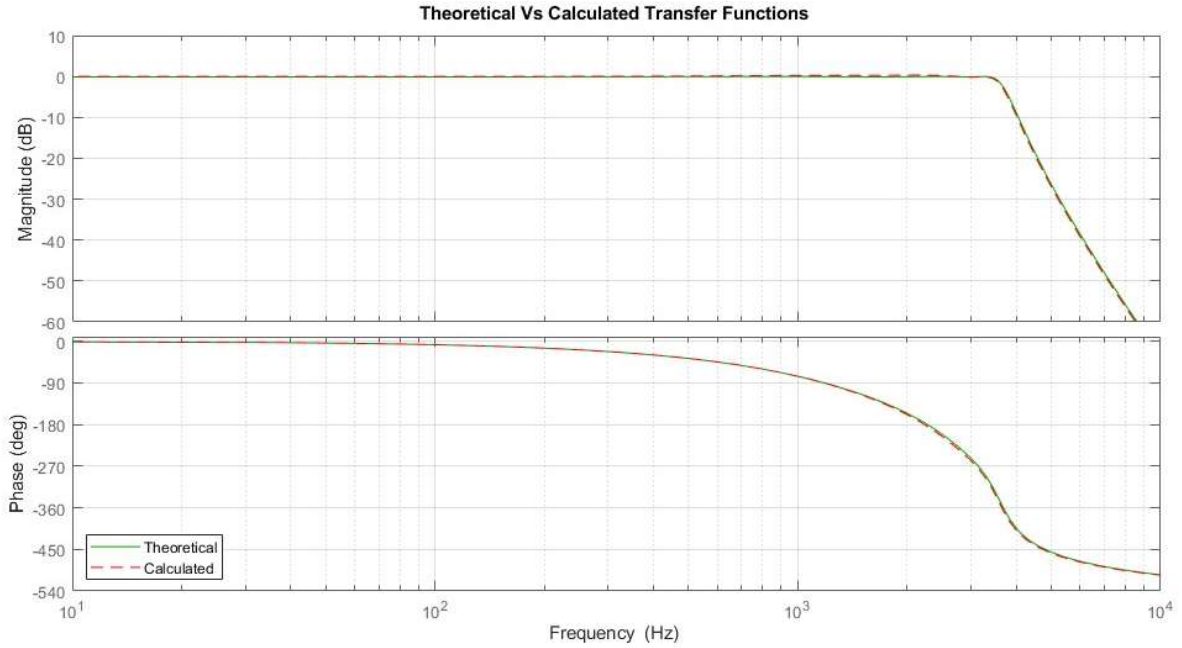

Figure 3.3: MATLAB frequency response showing the theoretical and calculated transfer functions of the lowpass filter section from 10 Hz and 10 kHz.

This shows the components selected should give the frequency response desired, with the next step being to validate the filter design through simulation in LTSpice.

3.4 Amplification

To make use of the ADC's 5 V range the expected signal will need to be amplified. One of the characteristics of an op-amp is how close the output voltage can get to the high and low supply voltages. The TL974 can get within 100 mV of the high and low supply voltages at 5 V [1, pp. 5,

Table 7.5], which is why it was used for the amplifier and not the voltage reference section, as lower performance op-amps typically cannot get very close to their supply voltages.

To understand the level of amplification needed the microphone's output V_{PP} needed to be measured. The method used to do this was by using a sound source above the microphone and measuring the AC signal at the microphones positive pin. The TDK PS1720P02 piezoelectric buzzer was used which produced tones at 2 kHz, 2.8 kHz and 5 kHz with a 3 $V_{0\text{-}P}$ square wave drive [15, p. 8]. These tests were done at 1 cm, 5 cm and 10 cm, with Figure B.5 showing the buzzer and microphone 1 cm, 5 cm and 10 cm apart, and Table D.5 the results. A V_{PP} from the microphone was chosen at 40 mV as it was a middle ground for the 2.8 kHz results for 5 cm and 10 cm, as 1 cm would be extremely close for a person to use the microphone. An issue with this method is that a piezoelectric buzzer does not accurately represent the human voice, but it offered a repeatable way of producing a measurable waveform. This could be improved using a better audio source that is more representative of the human voice and a recording environment with a low ambient noise level.

With an expected signal have a peak-to-peak voltage of 40 mV, and a range of 4.8 V output from the TL974, the amplifier can have a maximum gain of 120. The simulation results in Figure 4.3 show that non-ideal components will cause the circuit to not have a flat frequency response with the passband, having a peak gain of 43 dB (equal to a gain of 140), so a margin in the amplification should be used to avoid clipping the signal with non-ideal components. A gain of 100 (equal to 40 dB) is thus chosen as it provides a margin and is easy to calculate for E12 component values. An inverting amplifier can be used, with its gain given by equation (13). By setting R_1 to 10 kΩ, R_2 will be 1 MΩ.

$$Gain = -R_2/R_1 \tag{13}$$

During testing the issue of input bias current made it difficult to centre the amplified signal at 2.5 V, with calculations and simulation not giving the same results as seen in testing. This was fixed by the addition of a trim-pot between the voltage divider and non-inverting input, allowing for the DC voltage offset of the amplified signal to be tuned.

4. Simulation in LTSpice

To validate the calculations and component values the analog conditioning circuit was simulated in LTSpice, with the circuit diagram in Figure 4.1.

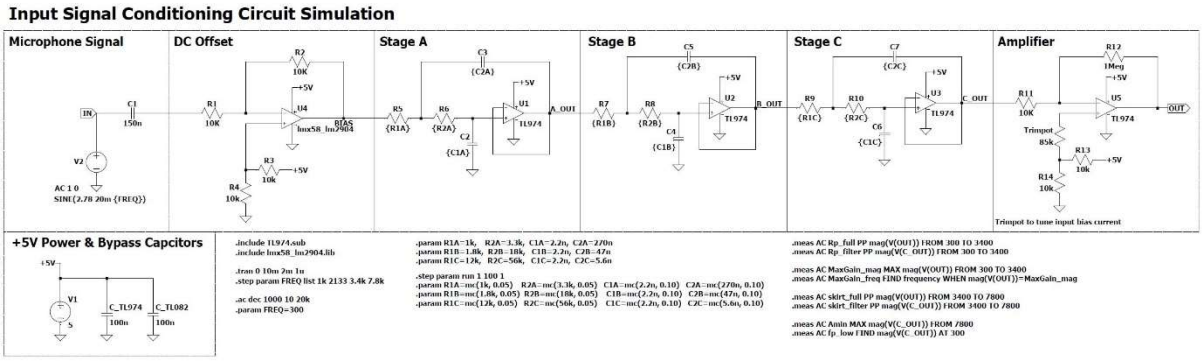

Figure 4.1: Input Singal Conditioning Circuit Simulation in LTSpice.

The frequency response for exact component values is shown Figure 4.2, where the attenuation at 7.8 kHz is 54 dB, thus meeting 48 dB requirement. The lowest point in the passband is at 300 Hz, thus the passband frequency for the filter is when it pass 38.7 dB at 3.4 kHz. Using the measure functions the maximum gain in the passband is 39.4 dB, giving a ripple of 0.7 dB, more than the design ripple of 0.1 dB and the intended ripple of 0.5 dB.

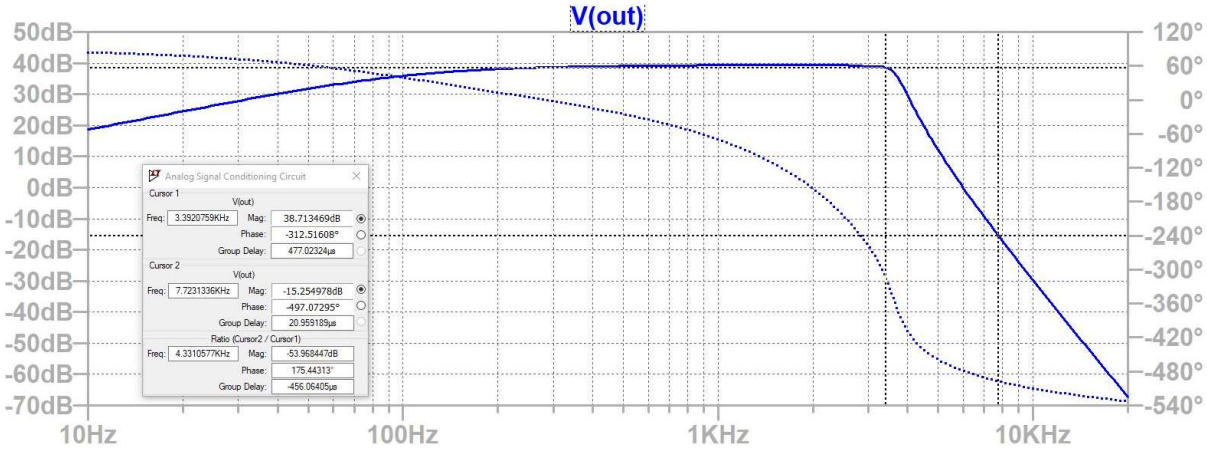

Figure 4.2: Simulated frequency response of the circuit from 10 Hz to 20 kHz.

To verify that the analog input conditioning circuit will perform to specification even when accounting for non-ideal components, a Monte Carlo simulation was performed in Figure 4.3, with the resistors having an error of 5% and the capacitors 10% and was run 100 times. The plot shows the range of frequency responses, with measure functions used to accurately quantify their results. The minimum attenuation achieved was 49 dB, with most being above 50 dB, confirming that the circuit should perform to specification.

Transient responses were simulated using an input signal of 40 mV_{PP} centred on 2.78 V based on testing of the microphone. Figure 4.4 show the lower passband frequency and Figure 4.7 the upper passband frequency. Figure 4.6 for where the maximum passband gain is. Figure 4.8 for the stopband frequency. Figure 4.5 shows the transient response at a frequency that is not likely to be affected by unwanted gain in the passband seen in Figure 4.3.

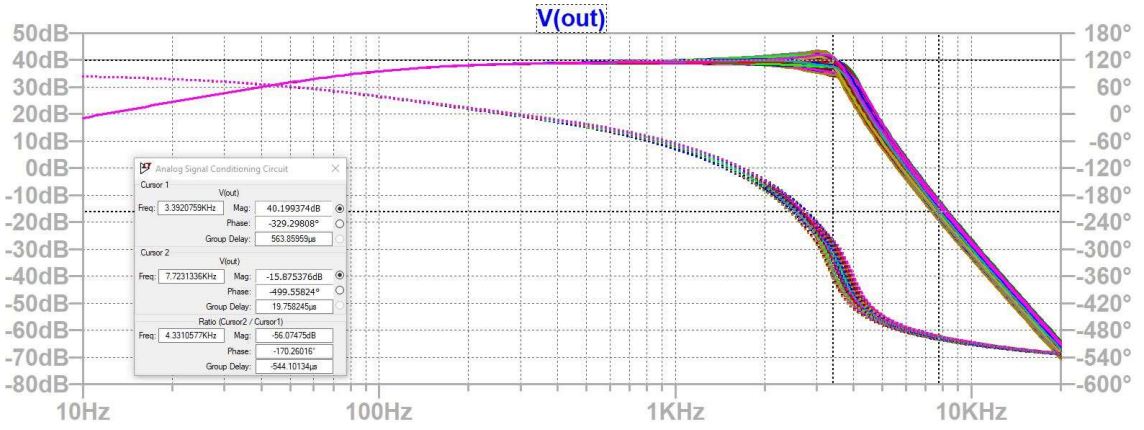

Figure 4.3: Simulated frequency response using a Monte Carlo simulation for component values.

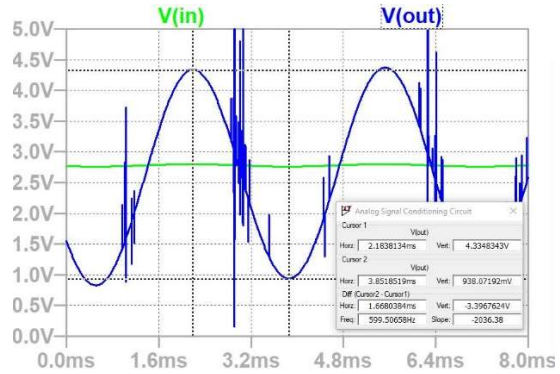

Figure 4.4: Simulated transient response at 300 Hz with an V_{PP} of 3.44 V. Gain is 38.7 dB.

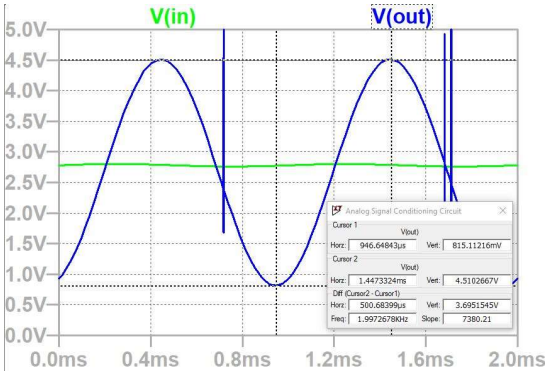

Figure 4.5: Simulated transient response at 1 kHz with an output V_{PP} of 3.70 V. Gain is 39.3 dB.

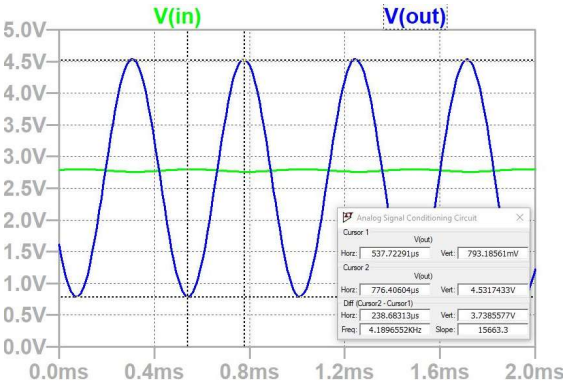

Figure 4.6: Simulated transient response at 2.133 kHz with an output V_{PP} of 3.75 V. Gain is 39.4 dB.

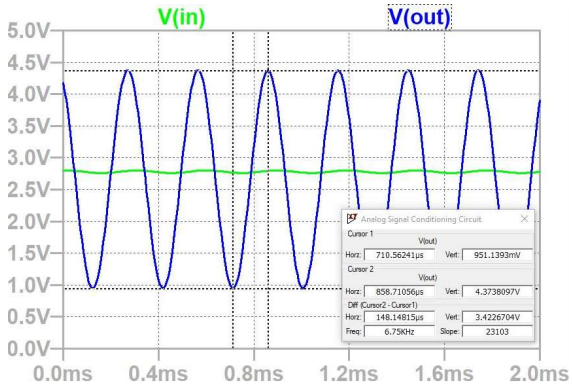

Figure 4.7: Simulated transient response at 3.4 kHz with an output V_{PP} of 3.44 V. Gain is 38.7 dB.

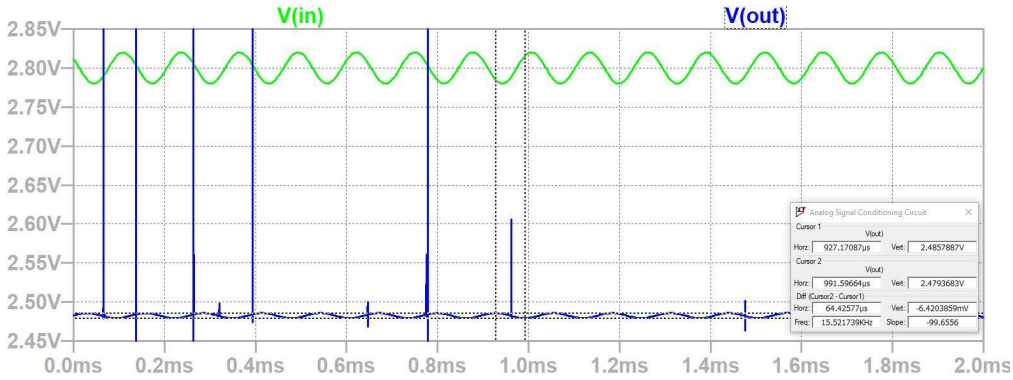

Figure 4.8: Simulated transient response at 7.8 kHz with an output VPP of 6.42 mV. Gain is -15.9 dB.

5. Experimental Results

Experimentally verification of calculations and simulations are important in designing a circuit. This was done by assembling the input conditioning circuit on a bread board and taking measurements at different points in the circuit. The frequency response of the circuit demonstrates its behaviour across different frequencies, with the primary output being the gain of the input signal. The transient response shows how the circuit responds to a change over time, with the primary output being the change in phase and amplitude of an input signal. By characterising the performance of the circuit, it can be determined if it meets the specifications.

5.1. Breadboard Prototype

The analog signal conditioning circuit was built on a breadboard (Figure 5.1) to experimentally verify that it functions as intended. It was set up with an R_L of 10 kΩ, and a 250 kΩ trim-pot which after tuning was measured at 105 kΩ. The rest of the circuit was built as seen in the simulation diagram in Figure 4.1. The yellow jumper near the microphone was where the signal was input, and the other the output from the amplifier stage. The orange jumpers made it easy to isolate each stage of the circuit, with the orange jumper at the microphone's positive pin moved to the yellow jumper when testing the circuit. Looking from the top of the breadboard in Figure 5.1 (left), the left IC is the LM358P with the voltage reference stage. The right IC is the TL974, with the bottom left op-amp being stage A, bottom right stage B, top left stage C and top right the amplifier.

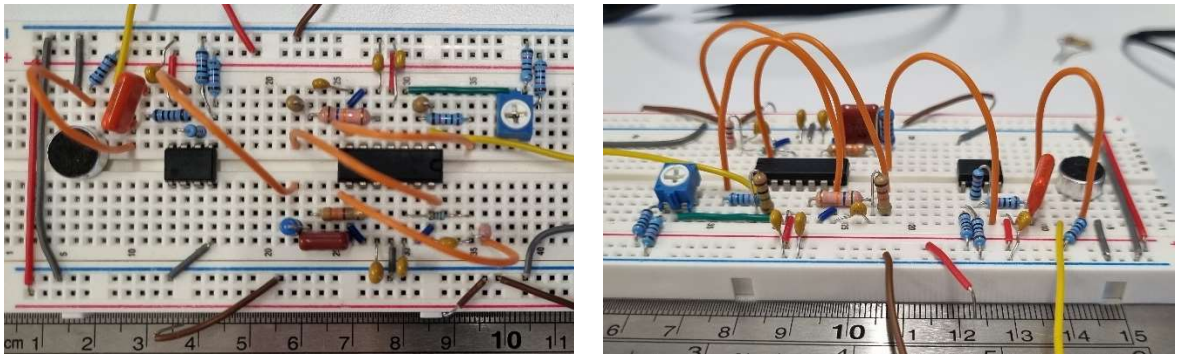

Figure 5.1: Photographs of the breadboard prototype from the top (left) and side (right). Scale seen is in cm.

5.2. Microphone Captures

The oscilloscope captures for buzzer 10 cm above the microphone can be seen in Figure 5.2, Figure 5.3 and Figure 5.4, and Table D.5 shows measured V_{PP} for an R_L of 4.7 kΩ and 10 kΩ at the different heights and frequencies tested.

The voltage reference stage can be seen working in Figure 5.5 where C1 is the voltage at the microphones positive pin at 3.02 V, and C2 the voltage at the output of the voltage reference stage at 2.5 V. A buzzer was held 1 cm above the microphone to generate a signal to show that C1 and C2 are the same signal but with different DC offsets.

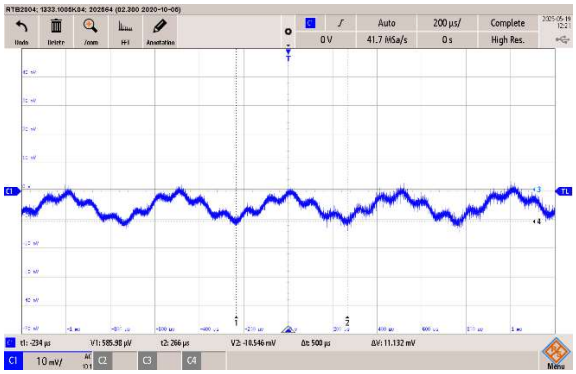

Figure 5.2: Microphone testing oscilloscope capture. Buzzer was 10 cm away with a 2 kHz tone. V_{PP} is 11 mV.

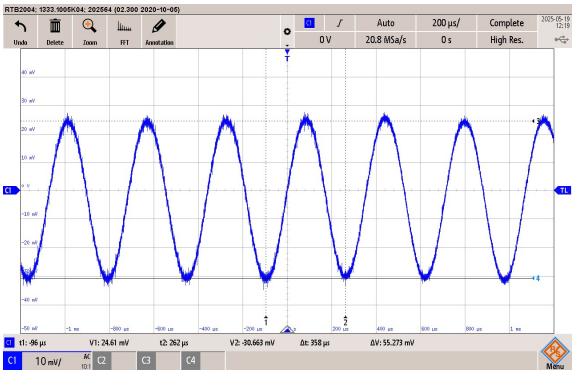

Figure 5.3: Microphone testing oscilloscope capture. Buzzer was 10 cm away with a 2.8 kHz tone. V_{PP} is 55 mV.

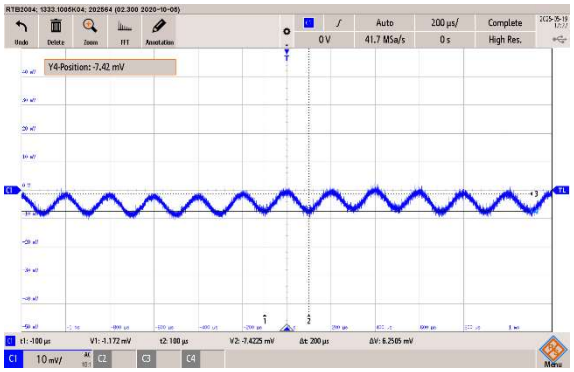

Figure 5.4: Microphone testing oscilloscope capture. Buzzer was 10 cm away with a 5 kHz tone. V_{PP} is 6 mV.

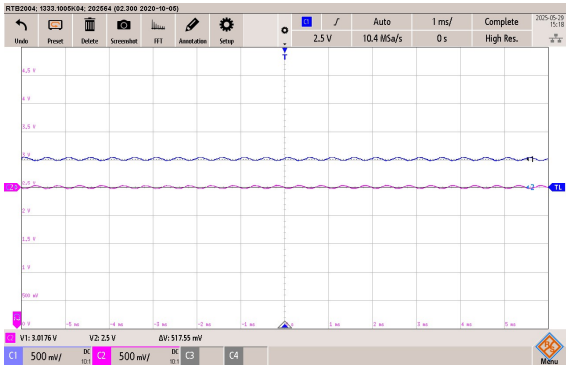

Figure 5.5: Oscilloscope capture showing the voltage reference stage working.

5.3. Frequency and Transient Response Captures and Analysis

The frequency response of the whole analog conditioning circuit is shown in Figure 5.6 (see Figure A.1 for screenshot), which has a relatively flat response within the passband. After about 8 kHz the oscilloscope struggled to measure the output, and the data began to scatter. The frequency response measured at the output of the filter the amplifier is included in Figure 5.7 (see Figure A.2 for screenshot) as it has a notably different response towards the upper end of the passband. This discrepancy between both frequency responses and the transient response are discussed later in this section. The csv data from the oscilloscopes bode plot function was imported to MATLAB to created cleaner graphics, using the script in Appendix F.4.

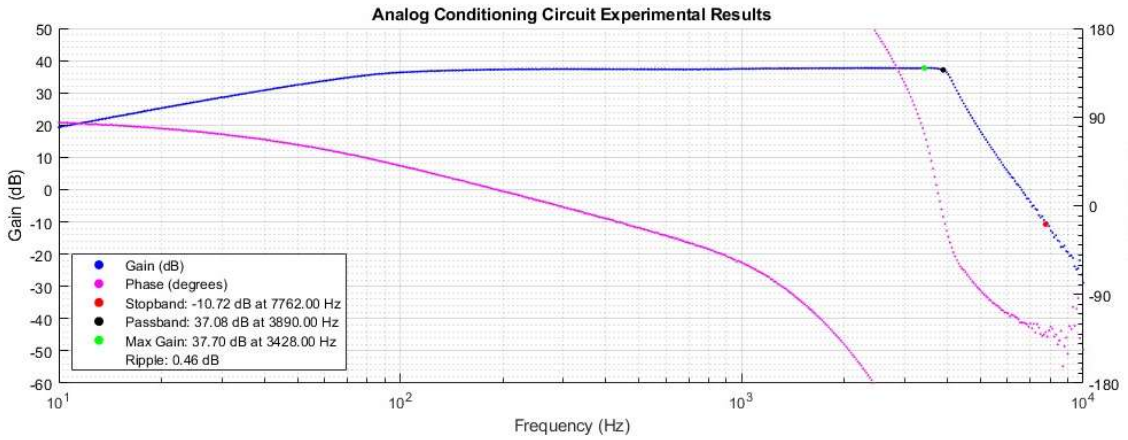

Figure 5.6: Oscilloscope capture of the frequency response for the full analog conditioning circuit.

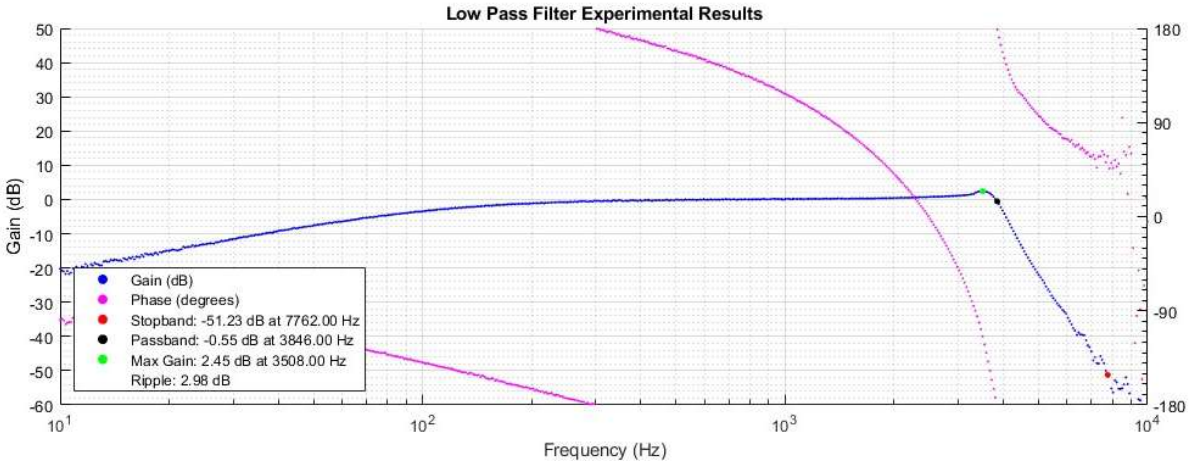

Figure 5.7: Oscilloscope capture of the frequency response captured at the output of the low pass filter.

The transient response measured at the output of the lowpass filter in Figure 5.8 shows the gain at 105 Hz is -3.22 dB, which along with Figure B.7 showing -0.63 dB at 300 Hz validates the high pass filter response of the microphone capacitor. Figure 5.9 is the transient response at 3.508 kHz showing that at the maximum gain of the circuit (based on Figure 5.7) the signal is not clipped. The transient response in Figure 5.10 shows the gain at 7.8 kHz is -11.5 dB, 0.7 dB lower than the frequency response. At 1 kHz, Figure 5.7 shows the filter has approximately zero gain, so by measuring the output from the amplifier at that frequency it is seen that the amplifier has a gain of 40 dB, exactly what is expected.

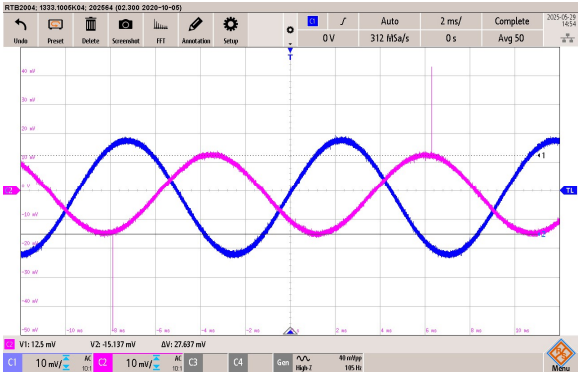

Figure 5.8: Oscilloscope capture of the high pass response of the microphone capacitor at 105 Hz. Gain is -3.2 dB.

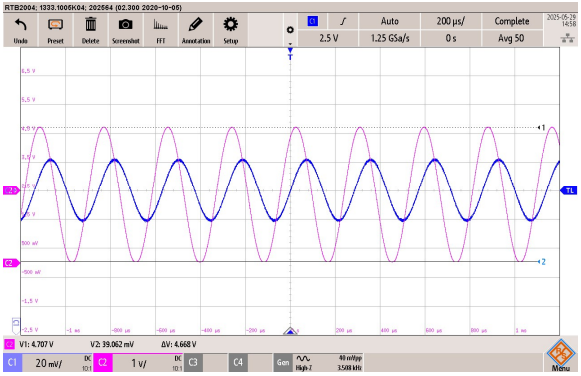

Figure 5.9: Oscilloscope capture of analog conditioning circuit at 3.501 kHz. Gain is 41.3 dB.

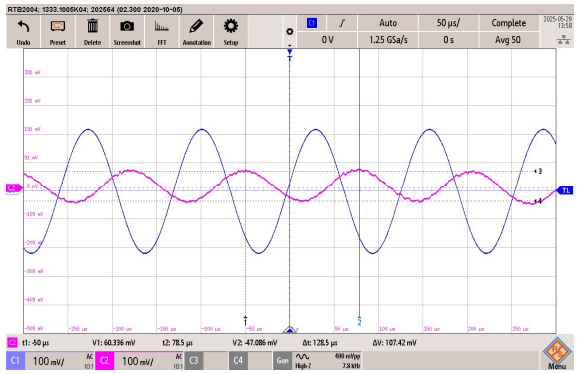

Figure 5.10: Oscilloscope capture of analog conditioning circuit at 7.8 kHz. Gain is -11.5 dB.

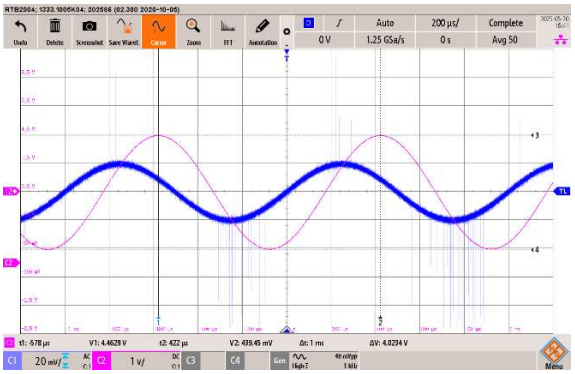

Figure 5.11: Oscilloscope capture of analog conditioning circuit at 1 kHz. Gain is 40.0 dB

There is a notable discrepancy between the frequency response as measured at the output of the low pass filter verses the amplifier. The expectation is that the amplifier will only amplify the signal and not affect the frequency response. A possible explanation is the signal is amplified at the upper end of the passband too much and is clipped, creating a ceiling on the gain the oscilloscope measures. The frequency response was measured with an input signal of 30 mV_{PP} (Figure A.4), where no peak at the end of the passband is observed. When looking at the transient response and sweeping the input signal frequency, the was a noticeable increase in gain at the higher passband frequencies but no clipping occurred (with the maximum shown in Figure 5.9), so the peak in Figure 5.7 should be present in Figure 5.6. As the signal was not clipped when amplified it did not present an issue to the function of the circuit so it's cause was not investigated further.

The experimental and simulated results can be compared against the design specifications, which are summarised in Table 5.1. The critical design requirement for 48 dB of attenuation at the Nyquist frequency of 7.8 kHz was achieved, but only just at 48.4 dB. The design ripple of 0.1 dB (with the intent to keep it under 0.5 dB) was technically met. However, when considering what was really observed both in the transient and frequency response, 0.5 dB was not achieved. So, a ripple of 2.98 dB should be accepted as the accurate value.

Table 5.1: Summary of specifications from design, simulation and experiment. 'Full' refers to the results of Figure 5.6, and 'Filter' to Figure 5.7. Trans. Is for the transient response.

	Design	Simulation	Exp. Full Bode	Exp. Filter Bode	Exp. Full Trans.
Stopband Attenuation (dB)	54	54	48.4	53.68	52.8
Passband Frequency (Hz)	3.4	3.4	3.890	3.846	N/A
Ripple (dB)	0.1	0.7	0.46	2.98	N/A

Through comparison of the performance of each stage individually the cause of the peak seen in Figure 5.7 might be understood. Table 5.2 is a summary of the natural frequency and peak gain in the frequency response. The calculated values come from MATLAB (Figure B.6), and the simulated and experimental results are in Appendix C. Each experimental stage had a natural frequency approximately 100 Hz to 200 Hz higher than what was calculated in MATLAB. Stage A and C had a similar gain, whereas stage B had a gain 0.7 dB lower than calculated, which may explain where the peak at the end of the passband comes from. Overall, each stage's frequency response is the expected shape, but the summation of slight deviations give rise to an overall frequency response that is noticeably different from expectation.

Table 5.2: Summary of natural frequency and peak gain for each stage in ideal

	Calc. A	Calc. B	Calc. C	Sim. A	Sim. B	Sim. C	Exp. A	Exp. B	Exp. C
Natural Frequency (kHz)	3.613	2.837	1.744	3.59	2.75	1.74	3.715	3.055	1.950
Peak Gain (dB)	13.35	2.17	-4.30	13.17	3.11	-4.30	13.17	2.84	-4.57

In summary the circuit met the critical specification of 48 dB attenuation at 7.8 kHz. However, due to the discrepancy in the results, more testing should be done to validate or refute the results, or to determine if or how the amplifier is affecting the frequency response.

6. Conclusion and Future Improvements

The aim of designing, implementing and evaluating the input signal conditioning circuitry for a digital voice recorder was successfully achieved. The circuit was designed and implemented on a breadboard and experimentally evaluated. The circuit met the requirements of recording human speech while attenuating all signals above the Nyquist frequency of 7.8 kHz by the minimum of 48 dB as required by the Teensy 2.0's ADC sampling frequency and sampling depth. Simulation showed that even with component tolerance considered the filter will still meet the specification.

While the circuit successfully captured the narrowband speech frequency range, this was achieved using a sampling rate typically designated for wideband speech. To truly meet wideband specifications, a significantly steeper filter response is necessary; simulations indicate this would require an 18th-order Chebyshev filter. Although an elliptical filter could reduce the requirement to an 8^{th} order, it would necessitate a more complex filter topology and calculations.

Discrepancies observed in the experimental results indicate a need for further validation of the experimental data and circuit performance, to confirm or refute these results.

The provided Electret microphone and TL974 quad op-amp were used in the design, with the only other components that were not resistors or capacitors being a 250 kΩ trim-pot and a LM358P op-amp IC, both being low-cost components.

While the microphone testing gave results it was done with a piezoelectric buzzer that does not accurately represent the human speech. Future improvements would use a better audio source and have a testing environment with lower ambient noise.

References

[1] Texas Instruments, “TL97x Output Rail-To-Rail Very-Low-Noise Operational Amplifiers,” Texas Instruments Incorporated, Dallas, 2015.

[2] Same Sky, “CMA-6542TF-K,” Same Sky, Lake Oswego, 2024.

[3] Texas Instruments Inc., “Industry-Standard Dual Operational Amplifiers,” Texas Instruments Inc., Dallas, 2024.

[4] Mouser Electronics, “Texas Instruments LM358P,” Mouser Electronics, [Online]. Available: https://au.mouser.com/ProductDetail/Texas-Instruments/LM358P?qs=X1HXWTtiZ0QtOTT8%252bVnsyw%3D%3D. [Accessed 15 May 2025].

[5] R. Quam, I. Martinez, C. Lorenzo, A. Bonmati, M. Rosa, P. Jarabo and J. L. Arsuaga, “Studying audition in fossil hominins: A new approach to the evolution of language?,” *Psychology of Language,* pp. 1-37, October 2012.

[6] Telecommunication Standardization Sector of ITU, “Vocabulary for performance, quality of service and quality of experience,” Geneva, 2017.

[7] P. Horowitz and W. Hill, The Art of Electronics, Third Edition ed., New York, New York: Cambridge University Press, 2016.

[8] K. Howard, “Ringing False: Digital Audio's Ubiquitous Filter Page 2,” 2025 Stereophile, 5 January 2006. [Online]. Available: https://www.stereophile.com/content/ringing-false-digital-audios-ubiquitous-filter-page-2. [Accessed 14 May 2024].

[9] S. Winder, Analog and digital filter design, 2nd ed., Woburn: Elsevier Science, 2002.

[10] A. Waters, Active Filter Design, 1st ed., London: MACMILLAN EDUCATION LTD, 1991.

[11] The MathWorks, Inc., *Signal Processing Toolbox,* Natick, Massachusetts: The MathWorks, Inc., 2024.

[12] The MathWorks, Inc., *Control System Toolbox,* Natick, Massachusetts: The MathWorks, Inc, 2024.

[13] Texas Instruments Inc., “Analysis of the Sallen-Key Architecture,” Texas Instruments Incorporated, Dallas, 2002.

[14] J. Banks, Frequency Response and Introduction to Filters: Week 8 Lecture Notes, Brisbane, 2025.

[15] TDK Corporation, “Piezoelectronic buzzers,” TDK Corporation, Peachtree City, 2022.

[16] Texas Instruments Inc., “TL082 Wide Bandwidth Dual JFET Input Operational Amplifier,” Texas Instruments Inc., Dallas, 2013.

[17] J. G. Švec and S. Granqvist, “Tutorial and Guidelines on Measurement of,” *Journal of Speech, Language, and Hearing Research,* vol. 61, no. 3, pp. 441-461, 2018.

Appendix A. Experimental Plots

Figure A.1: Oscilloscope capture of frequency response for the analogy conditioning circuit from 10 Hz to 20 kHz.

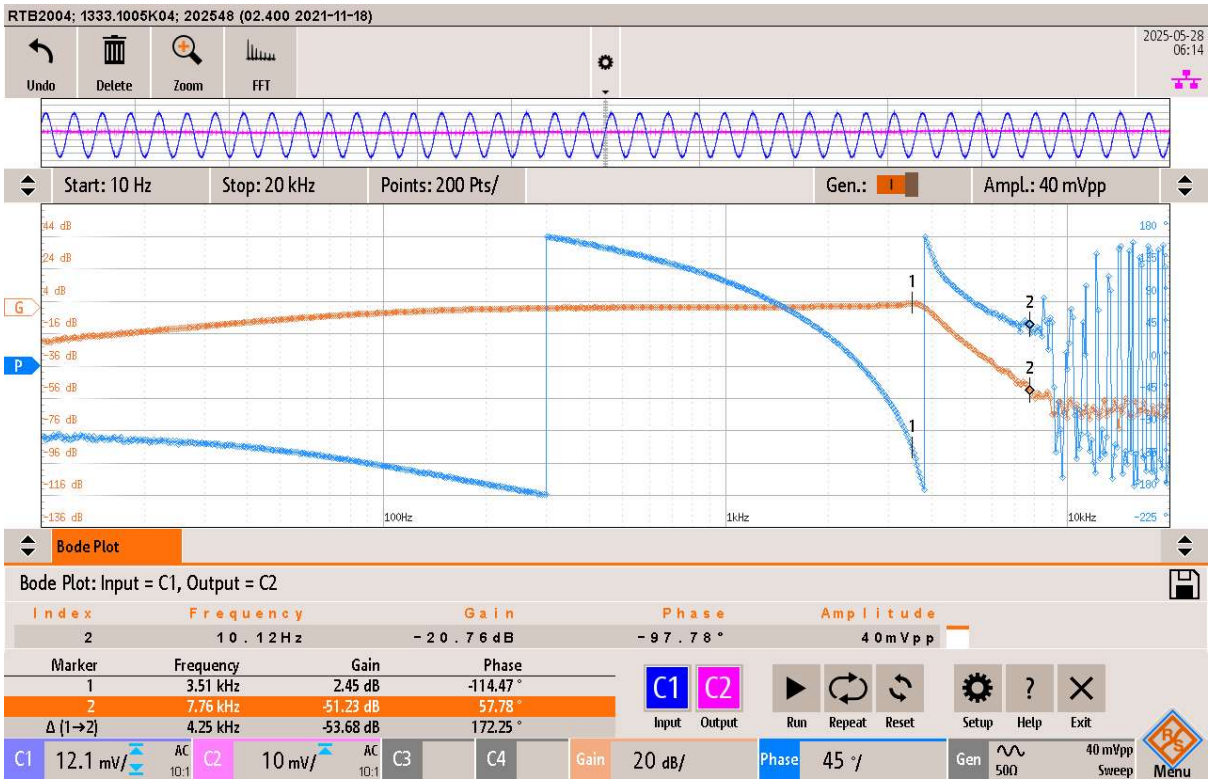

Figure A.2: Oscilloscope capture of frequency response for the low pass filter from 10 Hz to 20 kHz.

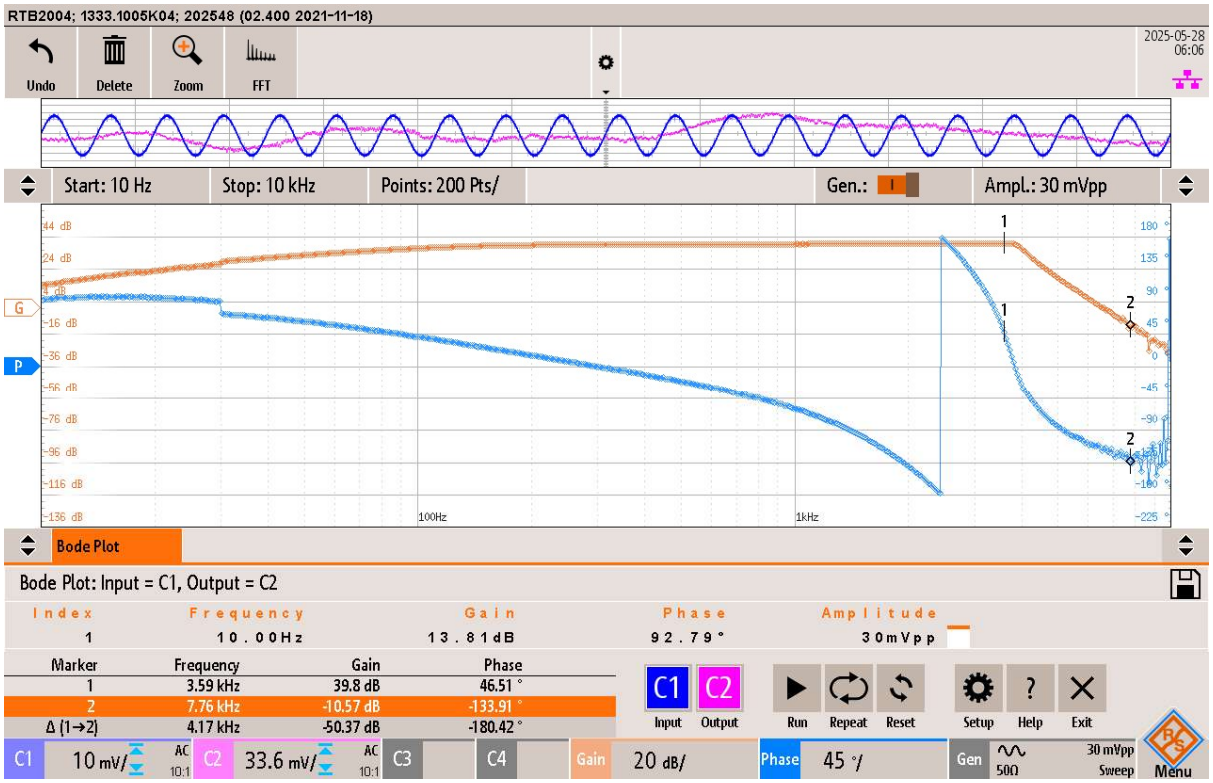

Figure A.3: Oscilloscope capture of frequency response for the analogy conditioning circuit from 10 Hz to 20 kHz, done at 30 mV instead of 40 mV.

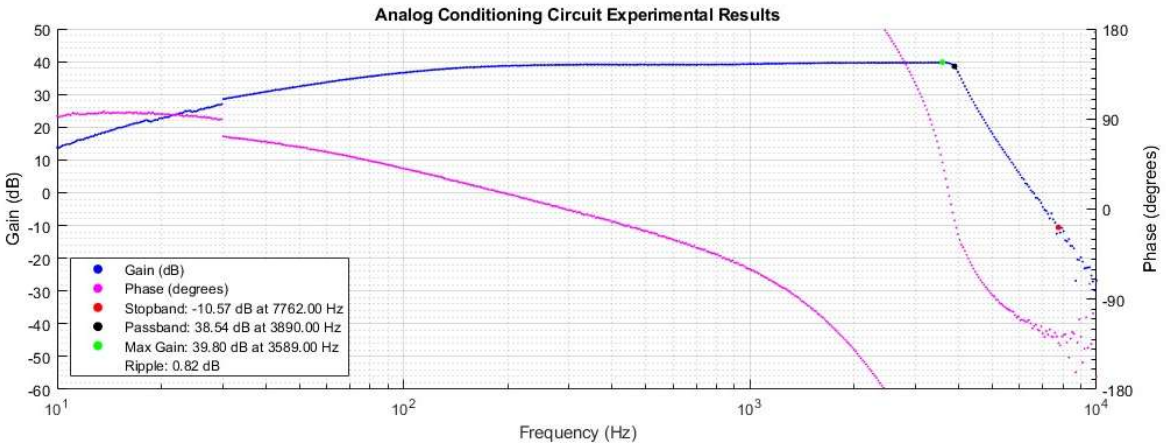

Figure A.4: Oscilloscope capture of frequency response for the analogy conditioning circuit from 10 Hz to 20 kHz, done at 30 mV instead of 40 mV. This is the MATLAB output from the data for Figure A.3.

Appendix B. Supplemental Figures

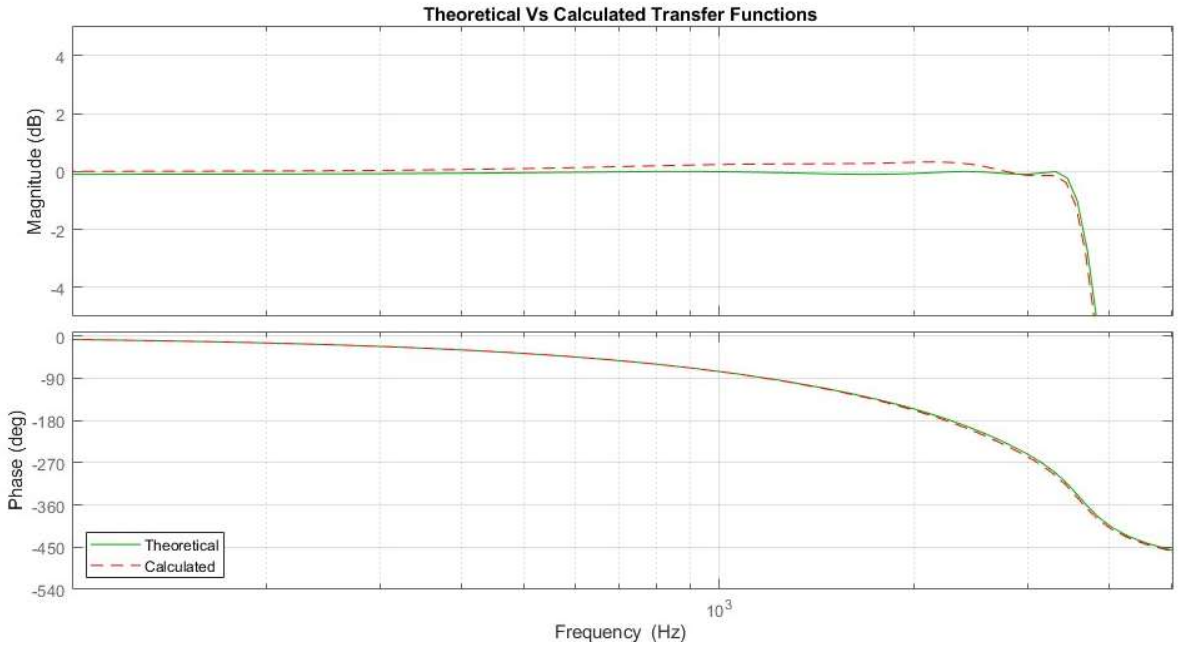

Figure B.1: Zoomed in view of Figure 3.3 from 1 kHz to 5 kHz showing that the theoretical and calculated transfer functions differ slightly

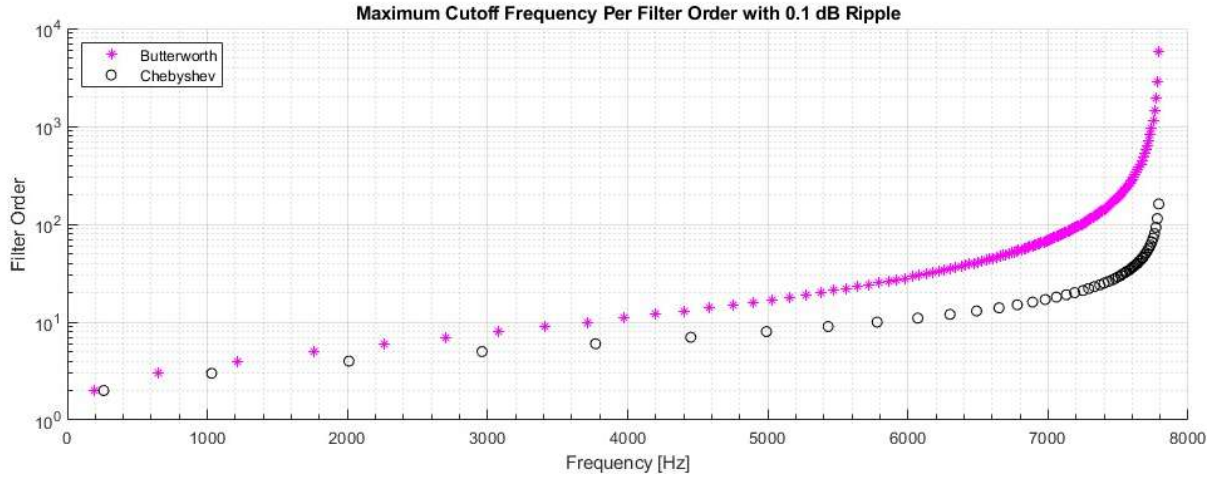

Figure B.2: The maximum frequency that Butterworth and Chebyshev filters can achieve. The at order at 7790 Hz is 5788 and 161 for Butterworth and Chebyshev respectively. Note that the filter order is in logarithmic scale.

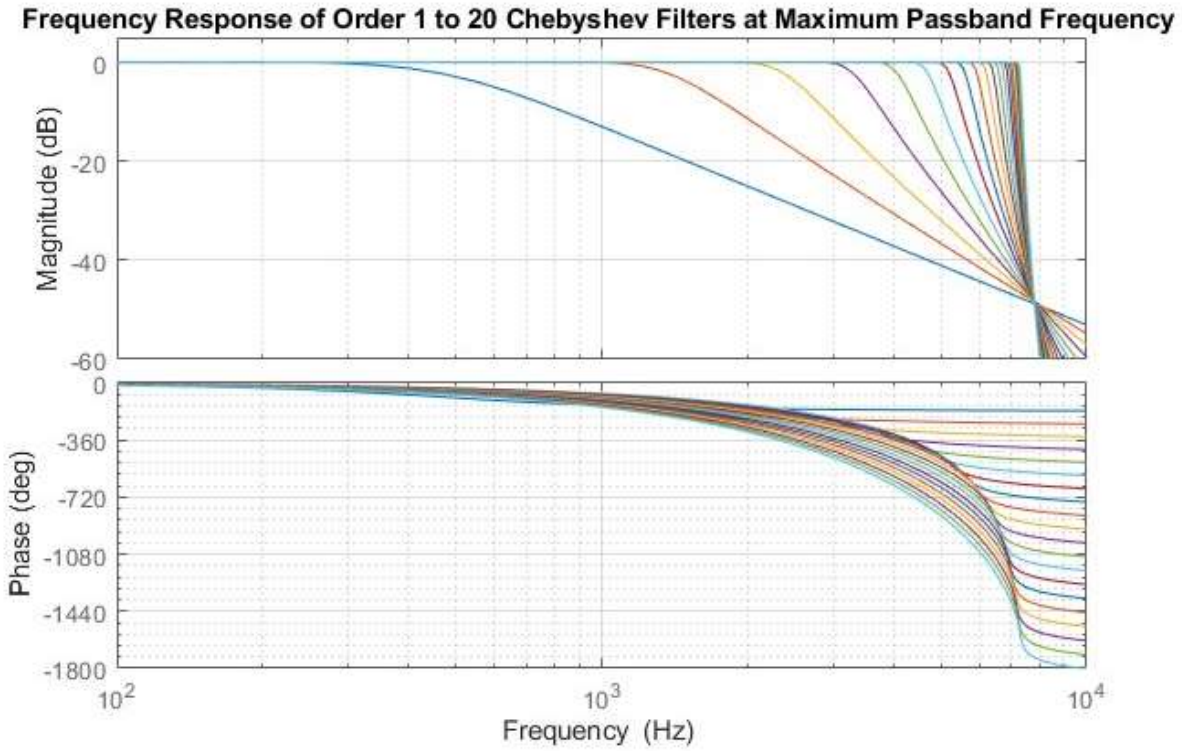

Figure B.3: The frequency response of Chebyshev filters of orders 1 to 20.

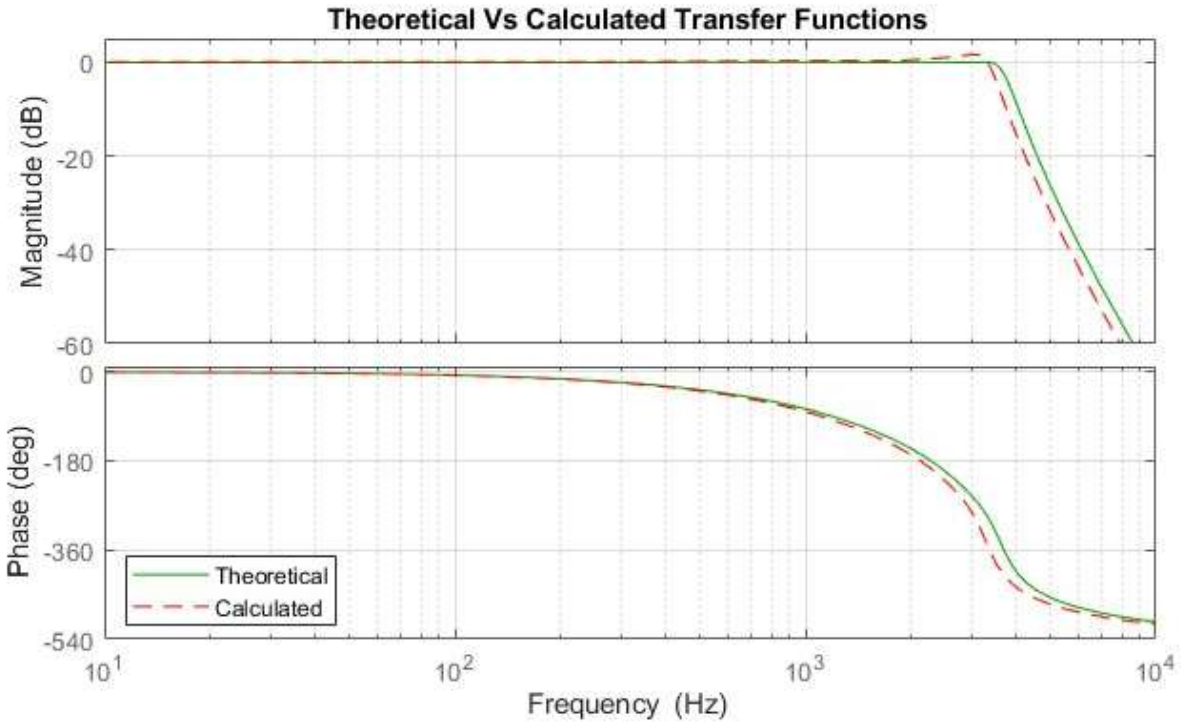

Figure B.4: Plot showing the theoretical and calculated transfer functions of the lowpass filter section when poor component tuning.

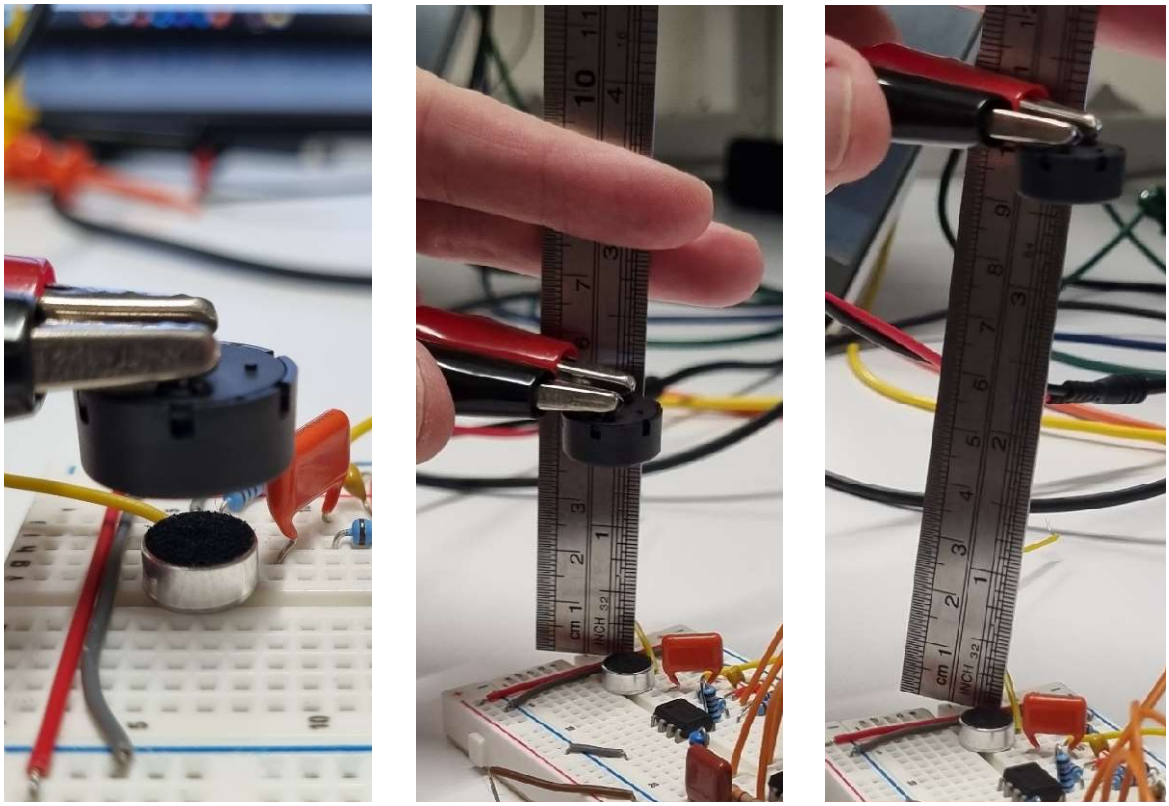

Figure B.5: Buzzer positioned approximately 1 cm (left), 5 cm (middle) and 10 cm (right) above the microphone.

```
Gain at natural frequencies: Stage A = 13.3506 dB, Stage B = 2.1666 dB, Stage C = -4.29737 dB
```

Figure B.6: MATLAB output for the gain at the natural frequencies of each stage.

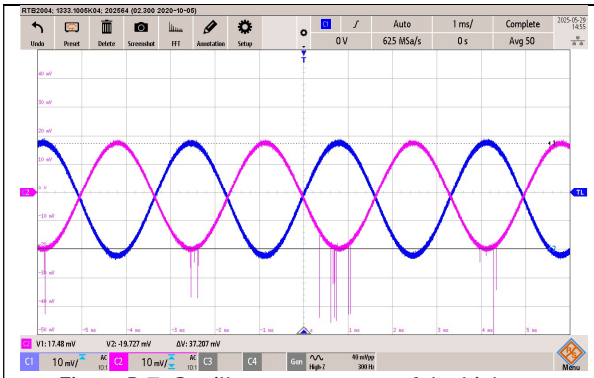

Figure B.7: Oscilloscope capture of the highpass response of the microphone capacitor at 300 Hz. Gain is -0.63 dB.

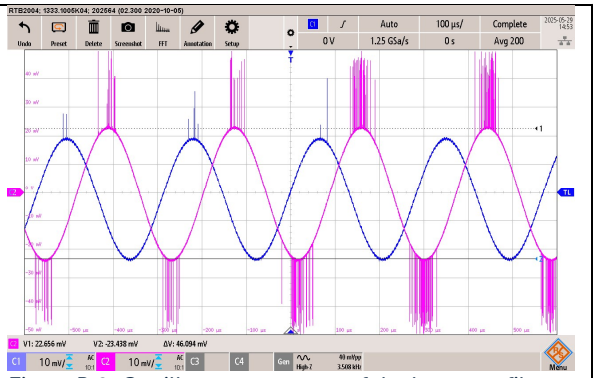

Figure B.8: Oscilloscope capture of the low pass filter at 3.508 kHz. Gain is 1.2 dB.

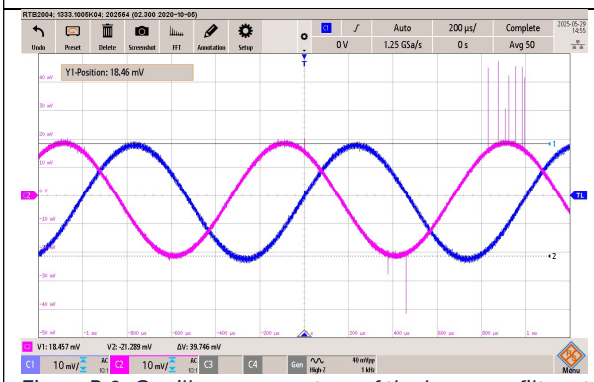

Figure B.9: Oscilloscope capture of the low pass filter at 1 kHz. Gain is -0.05 dB.

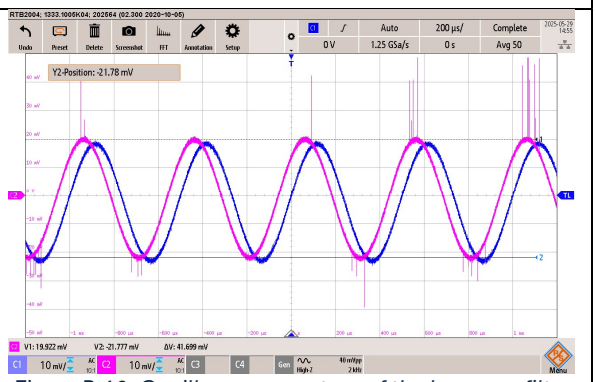

Figure B.10: Oscilloscope capture of the low pass filter at 2 kHz. Gain is 0.36 dB.

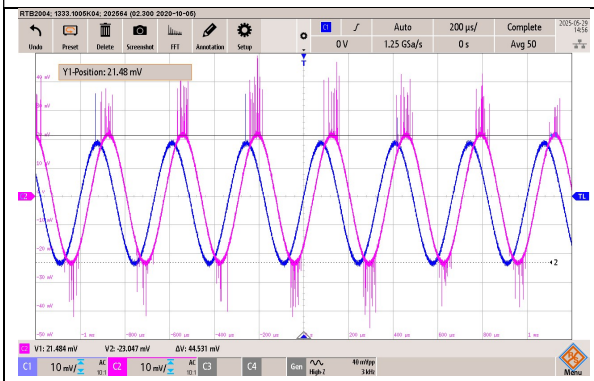

Figure B.11: Oscilloscope capture of the low pass filter at 3 kHz. Gain is 0.93 dB.

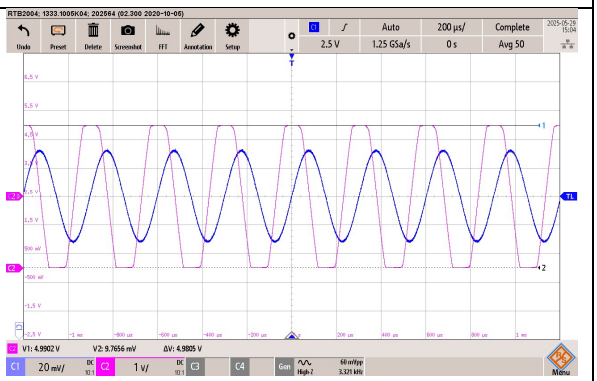

Figure B.12: Oscilloscope capture showing the signal being clipping near 0 V and 5 V.

Appendix C. Individual Filter Stage Analysis

Each individual stage was simulated and experimentally tested. Each stage performed experimentally as expected based off the simulations. The csv data was used to determine the natural frequency and peak gain of each stage. The natural frequency and peak gain for the simulation and experimental data is in Table 5.2.

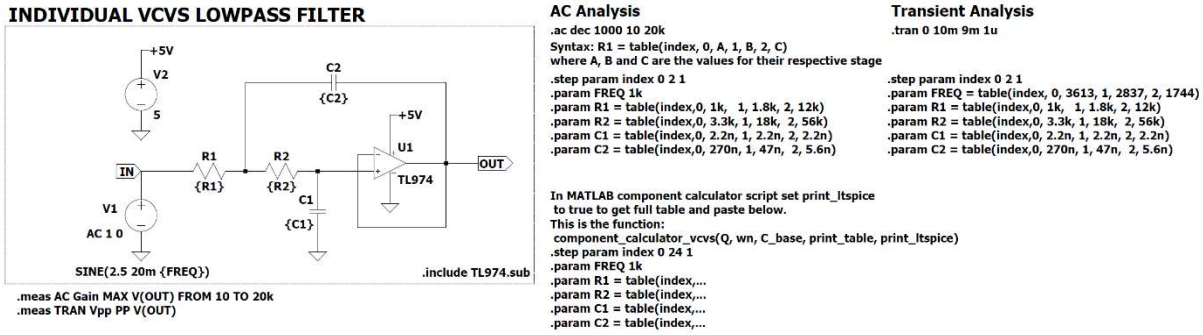

Figure C.1: Individual VCVS lowpass filter to simulate each stage in isolation.

Figure C.2: Simulated frequency response of stage A, B and C from 10 Hz to 20 kHz. Stage A is green, Stage B is blue, and Stage C is red.

Figure C.3: Oscilloscope capture of frequency response of stage A.

Figure C.4: Oscilloscope capture of frequency response of stage B.

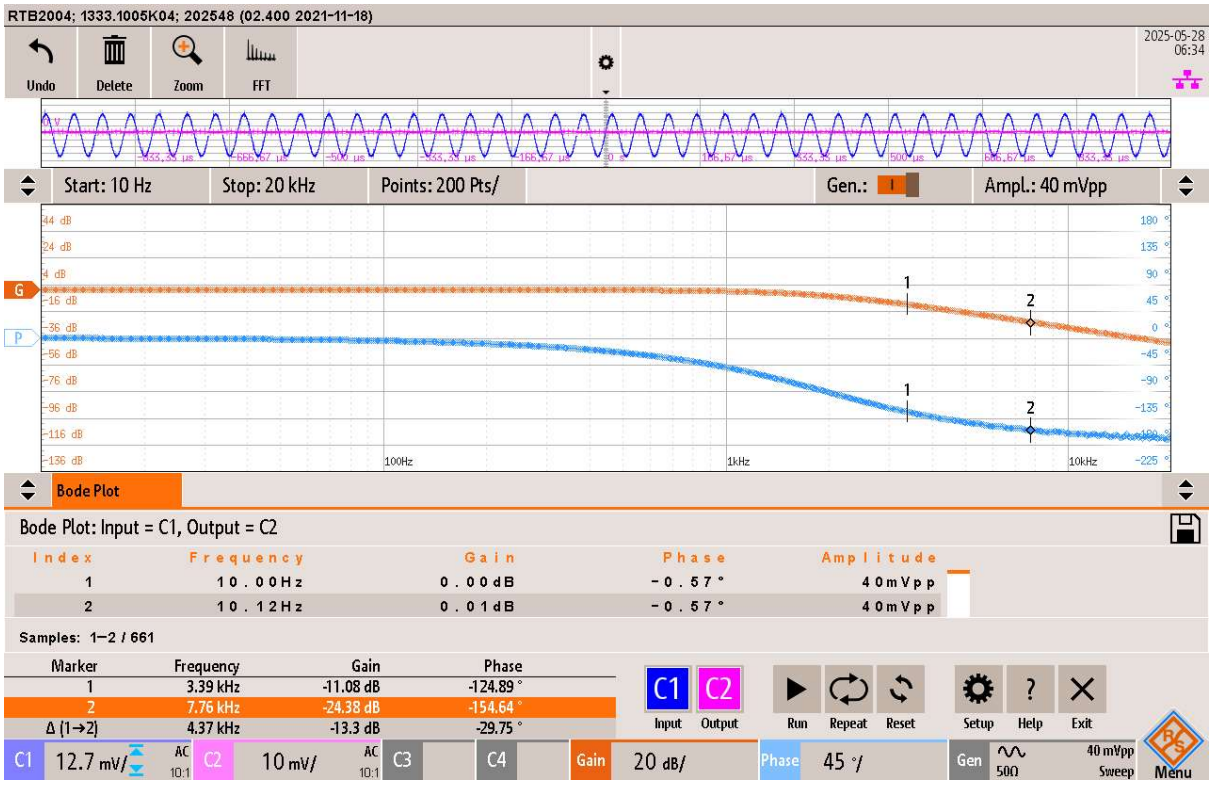

Figure C.5: Oscilloscope capture of frequency response of stage C.

Appendix D. Supplemental Tables

Table D.1: Table generated by MATLAB of the lines selected for stages A, B and C in that order.

Table showing the stage lines selected

k	wn_ack	Q_ack	wn_err	Q_err	Avg_wn_Q_err	R1	R2	C1	C2	R1_rnd	R2_rnd	C1_rnd	C2_rnd	R1_err	R2_err	C2_err	Avg_err
12	22587	4.6801	0.51224	1.0197	0.76596	1038.3	3283.3	2.2e-09	2.587e-07	1000	3300	2.2e-09	2.7e-07	3.6866	0.50805	4.3693	2.8546
23	17277	1.3288	3.0859	0.21146	1.6487	1898.7	17250	2.2e-09	4.367e-08	1800	18000	2.2e-09	4.7e-08	5.1968	4.3493	7.625	5.7237
16	10990	0.60822	0.24788	1.4608	0.85433	12260	56905	2.2e-09	5.421e-09	12000	56000	2.2e-09	5.6e-09	2.1186	1.5897	3.302	2.3368

Table D.2: Table generated by MATLAB for component tuning of Stage A.

Stage A: wn = 22702.8 rad/s, fn = 3613.27 Hz, Q = 4.6329

k	wn_ack	Q_ack	wn_err	Q_err	Avg_wn_Q_err	R1	R2	C1	C2	R1_rnd	R2_rnd	C1_rnd	C2_rnd	R1_err	R2_err	C2_err	Avg_err
0	22842	4.5227	0.61196	2.3793	1.4956	2160.8	2160.8	2.2e-09	1.8888e-07	2200	2200	2.2e-09	1.8e-07	1.8142	1.8142	4.702	2.7768
1	22842	4.5227	0.61196	2.3793	1.4956	2057.2	2264.4	2.2e-09	1.8932e-07	2200	2200	2.2e-09	1.8e-07	6.9403	2.8429	4.921	4.9014
2	25253	4.5	11.231	2.8686	7.0497	1954.1	2367.5	2.2e-09	1.9063e-07	1800	2200	2.2e-09	1.8e-07	7.887	7.0739	5.5738	6.8449
3	22795	4.4313	0.40472	4.3516	2.3782	1852	2469.6	2.2e-09	1.9282e-07	1800	2700	2.2e-09	1.8e-07	2.8058	9.328	6.6487	6.2609
4	22795	4.4313	0.40472	4.3516	2.3782	1751.2	2570.4	2.2e-09	1.9592e-07	1800	2700	2.2e-09	1.8e-07	2.787	5.042	8.1264	5.3185
5	22795	4.4313	0.40472	4.3516	2.3782	1652.2	2669.4	2.2e-09	1.9996e-07	1800	2700	2.2e-09	1.8e-07	8.943	1.148	9.9808	6.6906
6	22587	4.7916	0.51224	3.4249	1.9686	1555.5	2766.1	2.2e-09	2.0497e-07	1500	2700	2.2e-09	2.2e-07	3.5676	2.3897	7.3352	4.4308
7	22587	4.7916	0.51224	3.4249	1.9686	1461.3	2860.3	2.2e-09	2.1099e-07	1500	2700	2.2e-09	2.2e-07	2.6476	5.6038	4.2696	4.1736
8	22587	4.7916	0.51224	3.4249	1.9686	1370	2951.6	2.2e-09	2.1809e-07	1500	2700	2.2e-09	2.2e-07	9.4885	8.5239	0.87514	6.2958
9	22842	4.4222	0.61196	4.5487	2.5803	1281.9	3039.7	2.2e-09	2.2633e-07	1200	3300	2.2e-09	2.2e-07	6.3853	8.5617	2.7969	5.9146
10	22842	4.4222	0.61196	4.5487	2.5803	1197.1	3124.5	2.2e-09	2.3578e-07	1200	3300	2.2e-09	2.2e-07	0.24512	5.6159	6.6943	4.1851
11	20619	4.899	9.1805	5.7432	7.4619	1115.8	3205.8	2.2e-09	2.4654e-07	1200	3300	2.2e-09	2.7e-07	7.5432	2.9395	9.5158	6.6662
12	22587	4.6801	0.51224	1.0197	0.76596	1038.3	3283.3	2.2e-09	2.587e-07	1000	3300	2.2e-09	2.7e-07	3.6866	0.50805	4.3693	2.8546
13	22587	4.6801	0.51224	1.0197	0.76596	964.49	3357.1	2.2e-09	2.7237e-07	1000	3300	2.2e-09	2.7e-07	3.6816	1.701	0.86893	2.0839
14	24943	4.4232	9.8659	4.5262	7.196	894.52	3427.1	2.2e-09	2.8768e-07	820	3300	2.2e-09	2.7e-07	8.3308	3.708	6.1443	6.061
15	22561	4.89	0.62259	5.5502	3.0864	828.37	3493.2	2.2e-09	3.0476e-07	820	3300	2.2e-09	3.3e-07	1.0108	5.5313	8.2803	4.9408
16	22561	4.89	0.62259	5.5502	3.0864	766.02	3555.6	2.2e-09	3.2379e-07	820	3300	2.2e-09	3.3e-07	7.0462	7.1879	1.9176	5.3839
17	22790	4.3548	0.38407	6.003	3.1935	707.42	3614.2	2.2e-09	3.4493e-07	680	3900	2.2e-09	3.3e-07	3.8758	7.9084	4.3285	5.3709
18	20964	4.7342	7.6601	2.1855	4.9228	652.47	3669.1	2.2e-09	3.6838e-07	680	3900	2.2e-09	3.9e-07	4.2189	6.2924	5.8698	5.4604
19	23101	4.4118	1.7536	4.7732	3.2634	601.09	3720.5	2.2e-09	3.9435e-07	560	3900	2.2e-09	3.9e-07	6.8351	4.8243	1.1023	4.2539
20	23101	4.4118	1.7536	4.7732	3.2634	553.13	3768.5	2.2e-09	4.2308e-07	560	3900	2.2e-09	3.9e-07	1.2412	3.4905	7.8188	4.1835
21	22970	4.5283	1.1762	2.2573	1.7167	508.49	3813.1	2.2e-09	4.5484e-07	470	3900	2.2e-09	4.7e-07	7.5688	2.2787	3.3331	4.3935
22	22970	4.5283	1.1762	2.2573	1.7167	467	3854.6	2.2e-09	4.8992e-07	470	3900	2.2e-09	4.7e-07	0.64337	1.1778	4.0659	1.9624
23	23101	4.5866	1.7536	0.9996	1.3766	428.51	3893.1	2.2e-09	5.2864e-07	390	3900	2.2e-09	5.6e-07	8.9869	0.1776	5.9316	5.032
24	23101	4.5866	1.7536	0.9996	1.3766	392.87	3928.7	2.2e-09	5.7137e-07	390	3900	2.2e-09	5.6e-07	0.73111	0.73111	1.9892	1.1505

Table D.3: Table generated by MATLAB for component tuning of Stage B.

Stage B: wn = 17827.1 rad/s, fn = 2837.27 Hz, Q = 1.33157

k	wn_ack	Q_ack	wn_err	Q_err	Avg_wn_Q_err	R1	R2	C1	C2	R1_rnd	R2_rnd	C1_rnd	C2_rnd	R1_err	R2_err	C2_err	Avg_err
0	17408	1.3056	2.3521	1.9517	2.1519	9574.2	9574.2	2.2e-09	1.5603e-08	10000	10000	2.2e-09	1.5e-08	4.4472	4.4472	3.8653	4.2533
1	17408	1.3056	2.3521	1.9517	2.1519	9115.3	10033	2.2e-09	1.5639e-08	10000	10000	2.2e-09	1.5e-08	9.7059	0.3303	4.0862	4.7074
2	19224	1.2992	7.8341	2.4324	5.1332	8658.5	10490	2.2e-09	1.5747e-08	8200	10000	2.2e-09	1.5e-08	5.2949	4.6708	4.7448	4.9035
3	19224	1.2992	7.8341	2.4324	5.1332	8205.8	10943	2.2e-09	1.5928e-08	8200	10000	2.2e-09	1.5e-08	0.070749	8.6142	5.8291	4.838
4	17549	1.2823	1.5614	3.7022	2.6318	7759.3	11389	2.2e-09	1.6185e-08	8200	12000	2.2e-09	1.5e-08	5.6795	5.3638	7.3198	6.121
5	17592	1.3744	1.3204	3.2162	2.2683	7320.9	11828	2.2e-09	1.6518e-08	6800	12000	2.2e-09	1.8e-08	7.1147	1.4579	8.9714	5.848
6	17592	1.3744	1.3204	3.2162	2.2683	6892.2	12256	2.2e-09	1.6932e-08	6800	12000	2.2e-09	1.8e-08	1.3376	2.0907	6.3089	3.2457
7	17592	1.3744	1.3204	3.2162	2.2683	6474.9	12674	2.2e-09	1.743e-08	6800	12000	2.2e-09	1.8e-08	5.0213	5.3146	3.2726	4.5362
8	19385	1.3323	8.7396	0.053586	4.3966	6070.3	13078	2.2e-09	1.8016e-08	5600	12000	2.2e-09	1.8e-08	7.7478	8.2436	0.089384	5.3603
9	19385	1.3323	8.7396	0.053586	4.3966	5679.7	13469	2.2e-09	1.8697e-08	5600	12000	2.2e-09	1.8e-08	1.4034	10.905	3.7263	5.3448
10	17339	1.2726	2.7404	4.4275	3.5839	5304	13844	2.2e-09	1.9478e-08	5600	15000	2.2e-09	1.8e-08	5.5799	8.3472	7.5864	7.1712
11	17119	1.3478	3.9709	1.2195	2.5952	4944.1	14204	2.2e-09	2.0366e-08	4700	15000	2.2e-09	2.2e-08	4.9372	5.6017	8.0223	6.187
12	17119	1.3478	3.9709	1.2195	2.5952	4600.5	14548	2.2e-09	2.137e-08	4700	15000	2.2e-09	2.2e-08	2.1635	3.1073	2.946	2.7389
13	18793	1.2797	5.4191	3.8937	4.6564	4273.5	14875	2.2e-09	2.25e-08	3900	15000	2.2e-09	2.2e-08	8.7406	0.84108	2.2208	3.9342
14	18793	1.2797	5.4191	3.8937	4.6564	3963.5	15185	2.2e-09	2.3764e-08	3900	15000	2.2e-09	2.2e-08	1.6022	1.2178	7.4243	3.4147
15	16964	1.4177	4.8412	6.4688	5.655	3670.4	15478	2.2e-09	2.5176e-08	3900	15000	2.2e-09	2.7e-08	6.255	3.0883	7.2449	5.5294
16	18442	1.3469	3.4485	1.1481	2.2983	3394.2	15754	2.2e-09	2.6748e-08	3300	15000	2.2e-09	2.7e-08	2.774	4.7877	0.94306	2.8349
17	18442	1.3469	3.4485	1.1481	2.2983	3134.5	16014	2.2e-09	2.8494e-08	3300	15000	2.2e-09	2.7e-08	5.2807	6.3316	5.2433	5.6186
18	18442	1.3925	3.4485	4.5769	4.0127	2891	16257	2.2e-09	3.0431e-08	2700	15000	2.2e-09	3.3e-08	6.6074	7.7343	8.4424	7.5947
19	18442	1.3925	3.4485	4.5769	4.0127	2663.3	16485	2.2e-09	3.2576e-08	2700	15000	2.2e-09	3.3e-08	1.3769	9.0087	1.3008	3.8955
20	16835	1.3043	5.5649	2.0444	3.8047	2450.9	16698	2.2e-09	3.495e-08	2700	18000	2.2e-09	3.3e-08	10.165	7.8002	5.5788	7.8481
21	17156	1.3116	3.766	1.4962	2.6311	2253	16895	2.2e-09	3.7573e-08	2200	18000	2.2e-09	3.9e-08	2.354	6.538	3.7967	4.2296
22	17156	1.3116	3.766	1.4962	2.6311	2069.2	17079	2.2e-09	4.0471e-08	2200	18000	2.2e-09	3.9e-08	6.3215	5.3912	3.6355	5.1161
23	17277	1.3288	3.0859	0.21146	1.6487	1898.7	17250	2.2e-09	4.367e-08	1800	18000	2.2e-09	4.7e-08	5.1968	4.3493	7.625	5.7237
24	17277	1.3288	3.0859	0.21146	1.6487	1740.8	17408	2.2e-09	4.7199e-08	1800	18000	2.2e-09	4.7e-08	3.4028	3.4028	0.42247	2.4093

Table D.4: Table generated by MATLAB for component tuning of Stage C.

Stage C: wn = 10963.1 rad/s, fn = 1744.84 Hz, Q = 0.59946

k	wn_ack	Q_ack	wn_err	Q_err	Avg_wn_Q_err	R1	R2	C1	C2	R1_rnd	R2_rnd	C1_rnd	C2_rnd	R1_err	R2_err	C2_err	Avg_err
0	11247	0.61237	2.5848	2.154	2.3694	34582	34582	2.2e-09	3.1623e-09	33000	33000	2.2e-09	3.3e-09	4.5751	4.5751	4.3544	4.5016
1	10345	0.61024	5.6357	1.7987	3.7172	32925	36240	2.2e-09	3.1696e-09	33000	39000	2.2e-09	3.3e-09	0.22925	7.6164	4.1147	3.9868
2	10345	0.61024	5.6357	1.7987	3.7172	31274	37890	2.2e-09	3.1915e-09	33000	39000	2.2e-09	3.3e-09	5.5174	2.9299	3.3997	3.949
3	11437	0.60217	4.3236	0.45133	2.3875	29639	39525	2.2e-09	3.2282e-09	27000	39000	2.2e-09	3.3e-09	8.9053	1.328	2.2227	4.152
4	11437	0.60217	4.3236	0.45133	2.3875	28027	41138	2.2e-09	3.2802e-09	27000	39000	2.2e-09	3.3e-09	3.6634	5.1963	0.60461	3.1548
5	11437	0.60217	4.3236	0.45133	2.3875	26443	42721	2.2e-09	3.3477e-09	27000	39000	2.2e-09	3.3e-09	2.1063	8.7107	1.426	4.081
6	10418	0.58958	4.9688	1.6477	3.3083	24895	44270	2.2e-09	3.4316e-09	27000	47000	2.2e-09	3.3e-09	8.4569	6.1675	3.8346	6.153
7	11542	0.57076	5.2777	4.7869	5.0323	23387	45777	2.2e-09	3.5325e-09	22000	47000	2.2e-09	3.3e-09	5.932	2.6716	6.5812	5.0616
8	10617	0.62049	3.1586	3.5075	3.333	21926	47238	2.2e-09	3.6513e-09	22000	47000	2.2e-09	3.9e-09	0.33719	0.50443	6.81	2.5505
9	10617	0.62049	3.1586	3.5075	3.333	20515	48649	2.2e-09	3.7893e-09	22000	47000	2.2e-09	3.9e-09	7.2376	3.3899	2.9219	4.5165
10	11737	0.59579	7.0622	0.61231	3.8373	19158	50006	2.2e-09	3.9476e-09	18000	47000	2.2e-09	3.9e-09	6.0458	6.0114	1.2048	4.4207
11	11737	0.59579	7.0622	0.61231	3.8373	17858	51306	2.2e-09	4.1276e-09	18000	47000	2.2e-09	3.9e-09	0.79431	8.3931	5.5149	4.9008
12	9795.1	0.6271	10.654	4.6107	7.6324	16617	52547	2.2e-09	4.3312e-09	18000	56000	2.2e-09	4.7e-09	8.3231	6.5704	8.5157	7.8031
13	10730	0.59665	2.1265	0.46892	1.2977	15436	53728	2.2e-09	4.56e-09	15000	56000	2.2e-09	4.7e-09	2.8249	4.2281	3.0694	3.3741
14	10730	0.59665	2.1265	0.46892	1.2977	14316	54848	2.2e-09	4.8163e-09	15000	56000	2.2e-09	4.7e-09	4.7762	2.1001	2.4156	3.0973
15	11997	0.5572	9.4259	7.0492	8.2376	13258	55907	2.2e-09	5.1025e-09	12000	56000	2.2e-09	4.7e-09	9.4858	0.16677	7.8874	5.8467
16	10990	0.60822	0.24788	1.4608	0.85433	12260	56905	2.2e-09	5.421e-09	12000	56000	2.2e-09	5.6e-09	2.1186	1.5897	3.302	2.3368
17	10990	0.60822	0.24788	1.4608	0.85433	11322	57843	2.2e-09	5.7749e-09	12000	56000	2.2e-09	5.6e-09	5.9905	3.1855	3.0289	4.0683
18	12039	0.57205	9.816	4.5727	7.1944	10442	58722	2.2e-09	6.1675e-09	10000	56000	2.2e-09	5.6e-09	4.2366	4.6353	9.201	6.0243
19	10925	0.63037	0.34356	5.1557	2.7497	9620	59544	2.2e-09	6.6023e-09	10000	56000	2.2e-09	6.8e-09	3.9503	5.9525	2.995	4.2993
20	12065	0.58683	10.052	2.1076	6.0799	8852.6	60312	2.2e-09	7.0833e-09	8200	56000	2.2e-09	6.8e-09	7.3714	7.1492	3.9998	6.1734
21	10987	0.64441	0.21805	7.4982	3.8581	8138	61026	2.2e-09	7.6151e-09	8200	56000	2.2e-09	8.2e-09	0.76192	8.2364	7.6815	5.5599
22	12065	0.59991	10.052	0.074683	5.0634	7474	61690	2.2e-09	8.2024e-09	6800	56000	2.2e-09	8.2e-09	9.0175	9.2241	0.028883	6.0902
23	10949	0.55501	0.12932	7.4146	3.7719	6858	62306	2.2e-09	8.8507e-09	6800	68000	2.2e-09	8.2e-09	0.84604	9.1382	7.3518	5.7787
24	9914.7	0.61291	9.5633	2.2435	5.9034	6287.7	62877	2.2e-09	9.566e-09	6800	68000	2.2e-09	1e-08	8.1482	8.1482	4.5374	6.9446

Table D.5: Summary table of the V_{PP} for different values of R_L, frequencies and buzzer heights.

	R_L = 4.7 kΩ									R_L = 10 kΩ								
Buzzer Height (cm)	10			5			1			10			5			1		
Frequency (kHz)	2	2.8	5	2	2.8	5	2	2.8	5	2	2.8	5	2	2.8	5	2	2.8	5
V_{PP} (mV)	N/A	16	3	12	30	7	93	160	13	11	55	6	12	25	6	100	168	16

Appendix E. Additional Equations

Calculation for Microphone Capacitor

The microphone capacitor blocks DC current and will act as a passive highpass filter. Designed for a gain of -0.5 dB at 400 Hz, and a 10 kΩ resistor, the capacitor value is rounded up to 120 nF for an E12 capacitor. Equation (14) is the transfer function of a passive lowpass RC filter, with the x substitution to make it easier to solve. Solving for x and then f_c, it can then be put into equation (16) for the cutoff frequency to find the capacitor value.

$$Gain = 20\log_{10}\frac{{}^{f}/_{f_c}}{\sqrt{1+\left({}^{f}/_{f_c}\right)^2}} \tag{14}$$

$$x = {}^{f}/_{f_c}$$

$$-0.5 = 20\log_{10}\frac{x}{\sqrt{1+x^2}}$$

$$x = 2.8628$$

$$f_c = \frac{300\ Hz}{2.8638} \approx 105\ Hz \tag{15}$$

$$f_c = \frac{1}{2\pi RC} \tag{16}$$

$$C = \frac{1}{2\pi * 10\ k\Omega * 105\ hz} \approx 152\ nF \tag{17}$$

Appendix F. MATLAB Code

Appendix F.1. Filter Order Analysis

```
%% Filter Order Analysis
% Requires the 'Control System Toolbox' and 'Signal Processing Toolbox'.
% This script is designed to analyse the order of filter needed based on
% the cutoff frequency.
% The first section calculates the order needed for a given passband
% frequency, and attenuation in the passband and stopband.
% The second section calculates the max attenuation possible for a chosen
% cutoff frequency.

% ----- Clear Commands ----- %
clc;              % Clear the Command Window
clear;            % Clear variables
close all;        % Close all figures

%% Max Passband Frequency for Nth Order
%----- Filter Specifications -----%
fs = 7800;              % stopband frequency (15625/2), Hz
Amin = -20*log10(1/2^8); % stopband attenuation, approx 48 dB
Amax = 1;             % passband attenuation (ripple), dB

% ----- Calculations ----- %
results = []; % results for butt and cheb order per fp
fprintf(1, 'fs = %g Hz, A_min = %.4g dB, A_max = %g dB\n', fs, Amin, Amax)
for j = 100:10:fs % passband frequency, goes from 100->fs in steps of 10 Hz
    fp = j;
    ws = 2*pi*fs;
    wp = 2*pi*fp;

    % Calculate the order and wn for each passband frequency
    [n_butt, wn_butt] = buttord(wp, ws, Amax, Amin, 's');
    [n_cheb, wn_cheb] = cheb1ord(wp, ws, Amax, Amin, 's');

    results = [results, struct( ...
        'fp', fp, ...
        'n_butt', n_butt, ...
        'n_cheb', n_cheb ...
        )];
end
%disp(struct2table(results))

% ----- Find max fp for a given filter order ----- %
butt_ord = [];
cheb_ord = [];
for j = 2:length(results) % start at 2 since the previous index is checked
% Check if the current and previous orders are the same
% If not, append the previous order and its freq to the struct
    if results(j).n_butt ~= results(j-1).n_butt
        butt_ord = [butt_ord, struct( ...
        'n_butt', results(j-1).n_butt, ...
        'fp_max', results(j-1).fp)];
    end

    if results(j).n_cheb ~= results(j-1).n_cheb
        cheb_ord = [cheb_ord, struct( ...
        'n_cheb', results(j-1).n_cheb, ...
        'fp_max', results(j-1).fp)];
    end
end
%disp(struct2table(butt_min_ord)) % Displays the data as a table
```

```
%disp(struct2table(cheb_min_ord)) % Displays the data as a table

% ---- Plot Filter Orders for Max fp ----- %
x1 = [butt_ord.fp_max]; % Turn struct into vectors for plotting
y1 = [butt_ord.n_butt];
x2 = [cheb_ord.fp_max];
y2 = [cheb_ord.n_cheb];

fig1 = figure; % Create new figure to plot on
scatter(x1, y1, 'magenta', '*');    % Butterworth Plot
hold on;
scatter(x2, y2, 'black', 'o');      % Chebyshev Plot

xlabel('Frequency [Hz]');
ylabel('Filter Order');
legend('Butterworth', 'Chebyshev', 'Location', 'northwest');
title(sprintf('Maximum Cutoff Frequency Per Filter Order with %g dB Ripple', Amax));
grid on;
%set(gca, 'YScale', 'log'); % Sets the y-axis to log scale, useful for
%                           showing the full scale, since the filter
%                           order grows exponentially as fp approached fs
ax = gca;               % get current axis
ax.YMinorTick = 'on';   % Turn on minor ticks
ax.YMinorGrid = 'on';   % Turn on minor grid lines
ax.YLim = [0, 20];      % Set y limit, max ends up being in the 200's
ax.XMinorTick = 'on';   % Turn on minor ticks
ax.XMinorGrid = 'on';   % Turn on minor grid lines

% Control size and aspect ratio of the plot for consistency
% Make sure the x-y ratios are the same for both functions
fig1.Units = 'inches';
fig1.Position = [1 1 12 4];  % Plot size, x-pos, y-pos, x-width, y-width

hold off;

% ----- Plot Transfer Functions ----- %
% For each fp in in butt/cheb_min_ord loop through and plot the
%   transfer function

fig2 = figure; % Create new figure to plot on
max_ord = 20;
for j = 1:max_ord
    fp = cheb_ord(j).fp_max;      % passband frequency, Hz
    ws = 2*pi*fs;
    wp = 2*pi*fp;

    [~, wn_cheb] = cheb1ord(wp, ws, Amax, Amin, 's');
    [b, a] = cheby1(cheb_ord(j).n_cheb, Amax, wn_cheb,'low','s');
    H = tf(b,a);
    h = bodeplot(H);
    hold on;
end

setoptions(h, ...
    'YLim', {[-60, 5], [-1800, 10]}, ...
    'XLim', [100, 10000], ...
    'FreqUnits','Hz', ...
    'PhaseVisible','on');
title(sprintf(['Frequency Response of Order 1 to %g Chebyshev Filters at Maximum ' ...
    'Passband Frequency with %g dB Ripple'], max_ord, Amax));
grid on;
ax = gca;               % get current axis
ax.XMinorTick = 'on';   % Turn on minor ticks
ax.XMinorGrid = 'on';   % Turn on minor grid lines
ax.YMinorTick = 'on';   % Turn on minor ticks
ax.YMinorGrid = 'on';   % Turn on minor grid lines
```

```
% Control size and aspect ratio of the plot for consistency
% Make sure the x-y ratios are the same for both functions
fig2.Units = 'inches';
fig2.Position = [1 1 12 8];  % Plot size, x-pos, y-pos, x-width, y-width

hold off;

%% Max Attenuation For Given Order and Passband Frequency
% Set the desired passband frequency fp.
% This will then go through each order of filter and find the max
%   attenuation possible before reaching the stopband frequency.
% Do for both Butterworth and Cheyshev. Was easiest to just copy paste code.

fp = 3400; % Hz, Set this based off previous sections analysis
wp = 2*pi*fp; % rad/s
max_order = 20; % Maximum order of filter to check

butt_attenuation_results(max_order) = struct('Order', [], 'Amin', []);
% Butterworth
for i = 1:max_order
    order = i;
    for j = 1:1000 % Will check up to 1000 dB of attenuation
    % Iterate through attenuation, calculate the order needed.
    % Once n_butt exceeds order exit loop.
        Amin = j;
        [n_butt, ~] = buttord(wp, ws, Amax, Amin, 's');
        if n_butt > order % Once calculated order exceeds order
            butt_attenuation_results(i).Order = order;
            butt_attenuation_results(i).Amin = Amin - 1;
            break
        end
    end
end
%disp(struct2table(butt_attenuation_results))

% Chebyshev
cheb_attenuation_results(max_order) = struct('Order', [], 'Amin', []);
for i = 1:max_order
    order = i;
    for j = 1:1000 % Will check up to 200 dB of attenuation
    % Iterate through attenuation, calculate the order needed.
    % Once n_butt exceeds order exit loop.
        Amin = j;
        [n_cheb, ~] = cheb1ord(wp, ws, Amax, Amin, 's');
        if n_cheb > order % Once calculated order exceeds order
            cheb_attenuation_results(i).Order = order;
            cheb_attenuation_results(i).Amin = Amin - 1;
            break
        end
    end
end
%disp(struct2table(cheb_attenuation_results))

fig3 = figure; % Create new figure to plot on
x1 = [butt_attenuation_results.Order]; % Turn struct into vectors for plotting
y1 = [butt_attenuation_results.Amin];
x2 = [cheb_attenuation_results.Order];
y2 = [cheb_attenuation_results.Amin];
scatter(x1, y1, 'magenta', '*');   % Butterworth Plot
hold on;
scatter(x2, y2, 'black', 'o');   % Chebyshev Plot

xlabel('Filter Order');
ylabel('Maximum Attenuation [dB]');
legend('Butterworth', 'Chebyshev', 'Location', 'northwest');
```

```
title(sprintf('Max Attenuation for Nth Order Filter with %g dB Ripple', Amax));
grid on;
ax = gca;               % get current axis
ax.YMinorTick = 'on';   % Turn on minor ticks
ax.YMinorGrid = 'on';   % Turn on minor grid lines
ax.XMinorTick = 'on';   % Turn on minor ticks
ax.XMinorGrid = 'on';   % Turn on minor grid lines

% Control size and aspect ratio of the plot for consistency
% Make sure the x-y ratios are the same for both functions
fig3.Units = 'inches';
fig3.Position = [1 1 12 4];  % Plot size, x-pos, y-pos, x-width, y-width

hold off;
```

Appendix F.2. Component Value Calculations

```
%% Component Value Calculations
% Requires the 'Control System Toolbox' and 'Signal Processing Toolbox'.
% This script is used to calculate the component values for an analog lowpass
%   filter with a Chebyshev response and VCVS topology.
% Should first run through Filter_order_analysis to get help work out filter specs.
% I recommend hitting publish to more easily see the large tables generated.

% ----- Table Notes ----- %
% The 'component_calculator_sallen_key' functions will output a table of
% tuned component values.
% k: Tuning iteration.
% wn_ack: Natural frequency of stage from rounded component values.
% Q_ack: Q-factor of stage from rounded component values.
% wn_Error: Percentage error between input wn and wn_ack.
% Q_Error: Percentage error between input Q and Q_ack.
% R1, R2, C1, C2: Ideal component values for the stage.
% R1_Rounded -> C2_rounded: Closest E12 components to the ideal ones.
% R1_Error -> C2_Error: Percentage error between ideal and rounded components.
% Avg_Error: Average of R1_Error, R2_Error and C2_Error. C1's error is zero.

% ----- Clear Commands ----- %
clc;              % Clear the Command Window
clear;            % Clear variables
close all;        % Close all figures

% ----- Filter Specifications ----- %
fs = 7800;      % stopband frequency (15625/2), Hz
fp = 3400;      % passband frequency, Hz
ws = 2*pi*fs;   % stopband frequency, rad/s
wp = 2*pi*fp;   % passband frequency, rad/s

C_base = 2.2e-9; % Set value for C1 of each stage, rule of thumb is 10/fp uF
                 % 10/3400 uF ~ 2.94 nF

Amin = -20*log10(1/2^8);  % ~ 48 dB
Amax = 0.1; % passband attenuation (ripple), dB

fprintf(1,['\nThree Stage Chebyshev Response, ' ...
            'Sallen-Key Topology Lowpass Filter\n' ...
            'Passband Frequency: %g Hz\n' ...
            'Stopband Frequency: %g Hz\n' ...
            'Maximum Passband Gain: %g dB\n' ...
            'Minimum Stopband Attenuation: %.4g dB\n\n'],...
    fp,fs,Amax,Amin);

% ----- Calculate Chebyshev Order & Transfer Function ----- %
[n_cheb, wn_cheb] = cheb1ord (wp, ws, Amax, Amin, 's'); % Order and natural freq
[b, a] = cheby1(n_cheb, Amax, wn_cheb,'low','s'); % Transfer func coefficients
H = tf(b,a); % Transfer Function
```

```
[z, p, k] = tf2zpk (b, a); % Get the poles
[magnitude_at_ws, ~, ~] = bode(H, ws); % Takes H and ws and returns magnitude
fprintf(1, 'Chebyshev Filter Order: %g\n', n_cheb); % Sanity check it's 6th Order
fprintf(1, 'Transfer Function Attenutation at %g Hz = %.4g dB\n\n', ...
    fs, -20*log10(magnitude_at_ws)); % Convert magnitude to dB

%%%%% stage A %%%%%
% Use poles to get 2nd order transfer function coefficients.
% Use the coefficients to get the Q-factor and natural frequency.
% Input those along with a base capacitor values to get a table of tuned components.
a1A = -p(1)-p(2);
a2A = p(1)*p(2);
wnA = sqrt(a2A);
QA = wnA/a1A;
fprintf(1,'Stage A: wn = %g rad/s, fn = %g Hz, Q = %g\n', ...
           wnA, wnA/(2*pi), QA);
stageAresults = component_calculator_vcvs(QA, wnA, C_base, true, false);

%%%%% stage B %%%%
a1B = -p(3)-p(4);
a2B = p(3)*p(4);
wnB = sqrt(a2B);
QB = wnB/a1B;
fprintf(1,'Stage B: wn = %g rad/s, fn = %g Hz, Q = %g\n', ...
           wnB, wnB/(2*pi), QB);
stageBresults = component_calculator_vcvs(QB, wnB, C_base, true, false);

%%%%% stage C %%%%
a1C = -p(5)-p(6);
a2C = p(5)*p(6);
wnC = sqrt(a2C);
QC = wnC/a1C;
fprintf(1,'Stage C: wn = %g rad/s, fn = %g Hz, Q = %g\n', ...
           wnC, wnC/(2*pi), QC);
stageCresults = component_calculator_vcvs(QC, wnC, C_base, true, false);

%% ----- Compare Transfer Functions -----
% Takes the real rounded component values of each stage and calculates the
% transfer function, then multiplies all three to get the final function.
% The theoretical one calculated at the start is plotted against the real
% one.

% Select the desired line for each stage (input k+1)
stageAline = 13; % 13 for actual value, 12 for high error plot
stageBline = 24; % 24 for actual value, 16 for high error plot
stageCline = 17; % 17 for actual value, 13 for high error plot

% This outputs a table showing the three selected lines
a = stageAresults(stageAline); % Extract desired line
b = stageBresults(stageBline);
c = stageCresults(stageCline);
fields = fieldnames(a); % Get the fields for each column
combined_stage_lines = struct();
for i = 1:length(fields)
    field_name = fields{i}; % Get the actual field name string
    combined_stage_lines(1).(field_name) = a.(field_name);
    combined_stage_lines(2).(field_name) = b.(field_name);
    combined_stage_lines(3).(field_name) = c.(field_name);
end
fprintf(1, 'Table showing the stage lines selected\n')
disp(struct2table(combined_stage_lines));


% Calculate the transfer function for each stage
H_stageA = transfer_function_vcvs(1, ...
    stageAresults(stageAline).R1_rnd, stageAresults(stageAline).R2_rnd,...
```

```
    stageAresults(stageAline).C1_rnd, stageAresults(stageAline).C2_rnd);

H_stageB = transfer_function_vcvs(1, ...
    stageBresults(stageBline).R1_rnd, stageBresults(stageBline).R2_rnd,...
    stageBresults(stageBline).C1_rnd, stageBresults(stageBline).C2_rnd);

H_stageC = transfer_function_vcvs(1, ...
    stageCresults(stageCline).R1_rnd, stageCresults(stageCline).R2_rnd,...
    stageCresults(stageCline).C1_rnd, stageCresults(stageCline).C2_rnd);

% Combine the transfer functions
H_Calculated = H_stageA * H_stageB * H_stageC;

% Gives the magnitude at the natural frequencies
[magA, ~, ~] = bode(H_stageA, wnA);
magA_db = 20*log10(magA);
[magB, ~, ~] = bode(H_stageB, wnB);
magB_db = 20*log10(magB);
[magC, ~, ~] = bode(H_stageC, wnC);
magC_db = 20*log10(magC);
fprintf(1, ['Gain at natural frequencies: Stage A = %g dB, Stage B = %g dB, ' ...
        'Stage C = %g dB\n'], ...
        magA_db, magB_db, magC_db);

% Plot theoretical vs calculated transfer functions
fig1 = figure;
fig1.Units = 'inches';
fig1.Position = [1 1 12 6];  % x, y, width, height (in inches)
h = bodeplot(H, 'g-', H_Calculated, 'r--');
title('Theoretical Vs Calculated Transfer Functions');
setoptions(h, ...
    'FreqUnits','Hz', ...
    'PhaseVisible','on', ...
    'XLim', [10, 10000], ...
    'YLim', {[-60, 10], [-540, 10]});
grid on;
legend('Theoretical', 'Calculated', 'Location', 'southwest');


%% ----- Functions -----
% Contains small functions used above
% component_calculator_vcvs is large so gets own file.

function transfer_function = transfer_function_vcvs(K, R1, R2, C1, C2)
% Takes component values for a vcvs key lowpass filter and returns
% the transfer function.
    num = [0 0 K/(R1*R2*C1*C2)];    % numerator
    den = [1 ...                    % denominator
           1/(R1*C2) + 1/(R2*C2) + (1-K)/(R2*C1) ...
           1/(R1*R2*C1*C2)];
    transfer_function = tf(num,den);
end
```

Appendix F.3. Component Calculator VCVS

```
function results = component_calculator_vcvs ...
                           (Q, wn, C_base, print_table, print_ltspice)
% Function for component tuning for VCVS filter.
% Q is the Q-factor for the filter
% wn is the natural frequency of the filter, rad/s
% C_base is the base capacitor value, fro which the other component values
%   are derived from, Farads
% print_table is a Boolean that will display the results from the function,
%   default should be true
% print_ltspice is a Boolean that will display a spice directive input to
```

```
%   simulate all the sets of components. Check the section below for more
%   info.

    results = []; % Initialize output
    for k = 0:24
        % Taken from the lecture notes by Dr Jasmine Banks
        m = 10^(k / 24);
        n = ((Q * (1 + m)) / sqrt(m))^2;
        C1 = C_base;
        C2 = n * C1;
        R1 = 1 / (wn * C1 * sqrt(m * n));
        R2 = m * R1;

        % Round to E12
        R1_rounded = round_to_E12(R1);
        R2_rounded = round_to_E12(R2);
        C1_rounded = round_to_E12(C1);
        C2_rounded = round_to_E12(C2);

        wn_ack = 1 / sqrt(R1_rounded*R2_rounded*C1_rounded*C2_rounded);
        Q_ack = sqrt(R1_rounded*R2_rounded*C1_rounded*C2_rounded) ...
                / (R2_rounded*C1_rounded + R1_rounded*C1_rounded);
        wn_percentage_error = percent_error(wn, wn_ack);
        Q_percentage_error = percent_error(Q, Q_ack);
        avg_wn_Q_error = (wn_percentage_error + Q_percentage_error) / 2;

        R1_pecentage_error = percent_error(R1, R1_rounded);
        R2_pecentage_error = percent_error(R2, R2_rounded);
        C1_pecentage_error = percent_error(C1, C1_rounded);
        C2_pecentage_error = percent_error(C2, C2_rounded);
        Average_error = ...
                (R1_pecentage_error + R2_pecentage_error + C2_pecentage_error) / 3;

        % Store results
        results = [results; struct(...
            'k', k, ...
            'wn_ack', wn_ack, ...
            'Q_ack', Q_ack, ...
            'wn_err', wn_percentage_error, ...
            'Q_err', Q_percentage_error, ...
            'Avg_wn_Q_err', avg_wn_Q_error, ...
            'R1', R1, ...
            'R2', R2, ...
            'C1', C1, ...
            'C2', C2, ...
            'R1_rnd', R1_rounded, ...
            'R2_rnd', R2_rounded, ...
            'C1_rnd', C1_rounded, ...
            'C2_rnd', C2_rounded, ...
            'R1_err', R1_pecentage_error, ...
            'R2_err', R2_pecentage_error, ...
            'C2_err', C2_pecentage_error, ...
            'Avg_err', Average_error...
        )];
    end

    % This generates strings for LTSpice to look at the response for each
    % set of components.
    % Input '.step param index 1 24 1' (.step param <name> <start> <stop>
    % <increment>) as a spice directive, followed by the
    % generated strings. It creates one string per component.
    r1_table = ".param R1 = table(index";
    r2_table = ".param R2 = table(index";
    c1_table = ".param C1 = table(index";
    c2_table = ".param C2 = table(index";
    for i = 1:length(results)
```

```
        r1 = results(i).R1_rnd;
        r2 = results(i).R2_rnd;
        c1 = results(i).C1_rnd;
        c2 = results(i).C2_rnd;

        r1_table = sprintf('%s,%d,%.3g', r1_table, i-1, r1);
        r2_table = sprintf('%s,%d,%.3g', r2_table, i-1, r2);
        c1_table = sprintf('%s,%d,%.3g', c1_table, i-1, c1);
        c2_table = sprintf('%s,%d,%.3g', c2_table, i-1, c2);
    end

    % Print for copy-paste into LTSpice
    if print_ltspice == true
        fprintf('%s)\n', r1_table);
        fprintf('%s)\n', r2_table);
        fprintf('%s)\n', c1_table);
        fprintf('%s)\n\n', c2_table);
    end

    % Display the results as a table, useful to decide what components to use
    if print_table == true
        disp(struct2table(results))
    end
end

function rounded = round_to_E12(value)
% Finds the nearest E12 value for the input component value.
% Generates all the E12 values from pico to Mega.
% Finds the index of the generated value with the smallest difference to
% the input value, then uses the index to return the rounded value.
    E12 = [1.0 1.2 1.5 1.8 2.2 2.7 3.3 3.9 4.7 5.6 6.8 8.2];
    decades = -12:6;
    values = [];
    for d = decades
        values = [values, E12 * 10^d];
    end
    [~, index] = min(abs(values - value));
    rounded = values(index);
end

function err = percent_error(actual, approx)
    err = abs(actual - approx) / actual * 100;
end
```

Appendix F.4. Experimental Data Plotting

```
%% Experimental Data Plotting
% This script takes the CSV files generated by the oscilloscopes and
% creates bode plots.
% They are scatter plots, so to change the thickness of the lines change
% the size parameter in the scatter function.
% Colour can also be changed in the same function.

% ----- Clear Commands ----- %
clc;              % Clear the Command Window
clear;            % Clear variables
close all;        % Close all figures

fs = 7800; % Stopband frequency, for placing a marker on the plots

%% Stage A
stageAexp = readtable("STAGEA.CSV"); % Converts CSV to table
stageAexpTitle = 'Stage A Experimental Results';
exp_bode_plot(fs, stageAexp, stageAexpTitle, false, true);

%% Stage B
```

```
stageBexp = readtable("STAGEB.CSV"); % Converts CSV to table
stageBexpTitle = 'Stage B Experimental Results';
exp_bode_plot(fs, stageBexp, stageBexpTitle, false, true);

%% Stage C
stageCexp = readtable("STAGEC.CSV"); % Converts CSV to table
stageCexpTitle = 'Stage C Experimental Results';
exp_bode_plot(fs, stageCexp, stageCexpTitle, false, false);

%% Microphone Output to Stage C Output
in2coutExp = readtable("IN2COUT.CSV"); % Converts CSV to table
in2coutExpTitle = 'Low Pass Filter Experimental Results';
exp_bode_plot(fs, in2coutExp, in2coutExpTitle, true, true);

%% Microphone Output to Amplifier Output
in2outExp = readtable("IN2OUT.CSV"); % Converts CSV to table
in2outExpTitle = 'Analog Conditioning Circuit Experimental Results';
exp_bode_plot(fs, in2outExp, in2outExpTitle, true, true);
%% Function
function fig = exp_bode_plot(stopband_freq, data_table, plot_title, markers, gain_marker)
    fig = figure;
    title(plot_title);

    % Set log scale for x-axis and enable hold
    set(gca, 'XScale', 'log');
    hold on;

    % Use yyaxis to set up dual y-axes
    yyaxis left
    gainPlot = scatter(data_table.FrequencyInHz, data_table.GainInDB, ...
                       3, "blue", "filled", ...
                       'DisplayName', 'Gain (dB)');
    ylabel('Gain (dB)');
    ylim([-60 40]);
    yticks([-60 -50 -40 -30 -20 -10 0 10 20 30 40]);

    passband_min_freq = 300;
    passband_max_freq = 3600; % Tune this just below the passband freq that is visible
    % Find indices corresponding to the passband frequency range
    passband_indices = find(data_table.FrequencyInHz >= passband_min_freq & ...
                            data_table.FrequencyInHz <= passband_max_freq);

    % Gain values in passband
    passband_gains = data_table.GainInDB(passband_indices);

    % Max Passband Gain marker
    if gain_marker == true
        max_passband_gain = max(passband_gains);
        max_passband_gain_marker_index = ...
                    find(data_table.GainInDB == max_passband_gain, 1, 'last');
        max_gain_markerPlot = ...
                scatter(data_table.FrequencyInHz(max_passband_gain_marker_index), ...
                        data_table.GainInDB(max_passband_gain_marker_index), ...
                        15, 'go', 'filled', ...
                        'DisplayName', ...
                        sprintf('Max Gain: %.2f dB at %.2f Hz', ...
                        data_table.GainInDB(max_passband_gain_marker_index), ...
                        data_table.FrequencyInHz(max_passband_gain_marker_index)) ...
                        );
    end

    % Add markers at stopband frequency and passband frequency
    if markers == true
        % Passband frequency
        % Find the lowest gain passband
        lowest_passband_gain = min(passband_gains);
```

```
    % Get the first index that is lower than the lowest passband gain
    passband_marker_index = find(data_table.GainInDB < lowest_passband_gain & ...
                                 data_table.FrequencyInHz >= passband_max_freq, ...
                                 1, 'first');

    % Add a marker at this point
    passband_markerPlot = scatter(data_table.FrequencyInHz(passband_marker_index), ...
                                  data_table.GainInDB(passband_marker_index), ...
                                  15, 'ko', 'filled', ...
                                  'DisplayName', ...
                                  sprintf('Passband: %.2f dB at %.2f Hz', ...
                                  data_table.GainInDB(passband_marker_index), ...
                                  data_table.FrequencyInHz(passband_marker_index) ...
                                  ));

    % Stopband Frequency
    stopband_marker_index = find(data_table.FrequencyInHz < stopband_freq, 1, 'last');
    stopband_markerPlot = scatter(data_table.FrequencyInHz(stopband_marker_index), ...
                          data_table.GainInDB(stopband_marker_index), ...
                          15, 'ro', 'filled', ...
                          'DisplayName', ...
                          sprintf('Stopband: %.2f dB at %.2f Hz', ...
                          data_table.GainInDB(stopband_marker_index), ...
                          data_table.FrequencyInHz(stopband_marker_index) ...
                          ));
end

yyaxis right
phasePlot = scatter(data_table.FrequencyInHz, ...
                    data_table.PhaseIn_, ...
                    3, "magenta", "filled", ...
                    'DisplayName', 'Phase (degrees)');

max_passband_gain = max(passband_gains);
lowest_passband_gain = min(passband_gains);
ripple = abs(max_passband_gain - lowest_passband_gain);
ripple_in_legend = scatter(0, 0, 'white', 'DisplayName', ...
                           sprintf('Ripple: %.2f dB', ripple));

ylabel('Phase (degrees)');
ylim([-180 180]);
yticks([-180 -90 0 90 180]);

% Common X-axis
xlabel('Frequency (Hz)');
xlim([10 20000]);

% Grid and minor ticks
grid on;
ax = gca;
ax.YMinorTick = 'on';
ax.YMinorGrid = 'on';
ax.XMinorTick = 'on';
ax.XMinorGrid = 'on';

% Set both y-axes to black, they default to blue and orange
ax.YAxis(1).Color = 'k'; % Left axis
ax.YAxis(2).Color = 'k'; % Right axis

% legend handles must be in the order you want them shown
if markers == true && gain_marker == true
    legend([gainPlot, phasePlot, ...
            stopband_markerPlot, ...
            passband_markerPlot, ...
            max_gain_markerPlot, ...
```

```
                ripple_in_legend], ...
               'Location', 'southwest');
    elseif markers == false && gain_marker == true
            legend([gainPlot, phasePlot, ...
                max_gain_markerPlot], ...
               'Location', 'southwest');
    else
        legend([gainPlot, phasePlot,], 'Location', 'southwest');
    end

    % Control size and aspect ratio of the plot for consistency
    fig.Units = 'inches';
    fig.Position = [1 1 12 4];  % Plot size, x-pos, y-pos, x-width, y-width

    hold off;
end
```

Appendix G. LTSpice Netlist

Appendix G.1. Analog Signal Conditioning Circuit

```
XU1 N010 A_OUT +5V 0 A_OUT TL974
V2 IN 0 SINE(2.78 20m {FREQ}) AC 1 0
R6 N010 N005 {R2A}
R5 N005 BIAS {R1A}
C3 A_OUT N005 {C2A}
C2 0 N010 {C1A}
V1 +5V 0 5
XU2 N008 B_OUT +5V 0 B_OUT TL974
R8 N008 N001 {R2B}
R7 N001 A_OUT {R1B}
C5 B_OUT N001 {C2B}
C4 0 N008 {C1B}
XU3 N006 C_OUT +5V 0 C_OUT TL974
R10 N006 N002 {R2C}
R9 N002 B_OUT {R1C}
C7 C_OUT N002 {C2C}
C6 0 N006 {C1C}
XU4 N011 N004 +5V 0 BIAS lmx58_lm2904
R1 N004 N007 10K
R2 BIAS N004 10K
C1 N007 IN 150n
XU5 N009 N003 +5V 0 OUT TL974
R11 N003 C_OUT 10K
R12 OUT N003 1Meg
R14 0 N012 10k
R13 +5V N012 10k
R4 0 N011 10k
R3 +5V N011 10k
C_TL974 +5V 0 100n
C_TL082 +5V 0 100n
R§Trimpot N012 N009 85k
.include TL974.sub
.tran 0 10m 2m 1u
.param FREQ=300
* .step param run 1 100 1\n.param R1A=mc(1k, 0.05)    R2A=mc(3.3k, 0.05)
C1A=mc(2.2n, 0.10)   C2A=mc(270n, 0.10)\n.param R1B=mc(1.8k, 0.05)
R2B=mc(18k, 0.05)    C1B=mc(2.2n, 0.10)   C2B=mc(47n, 0.10)\n.param
R1C=mc(12k, 0.05)   R2C=mc(56k, 0.05)    C1C=mc(2.2n, 0.10)   C2C=mc(5.6n,
0.10)
.param R1A=1k,    R2A=3.3k,  C1A=2.2n,  C2A=270n
.param R1B=1.8k,  R2B=18k,   C1B=2.2n,  C2B=47n
.param R1C=12k,   R2C=56k,   C1C=2.2n,  C2C=5.6n
* Stage A
* Stage B
* Stage C
* Amplifier
* DC Offset
* Microphone Signal
* +5V Power & Bypass Capcitors
* Input Signal Conditioning Circuit Simulation
* .step param FREQ list 1k 2133 3.4k 7.8k
* .ac dec 1000 10 20k
.meas AC Rp_full PP mag(V(OUT)) FROM 300 TO 3400
.meas AC Rp_filter PP mag(V(C_OUT)) FROM 300 TO 3400
.meas AC MaxGain_mag MAX mag(V(OUT)) FROM 300 TO 3400
```

```
.meas AC MaxGain_freq FIND frequency WHEN mag(V(OUT))=MaxGain_mag
.meas AC skirt_full PP mag(V(OUT)) FROM 3400 TO 7800
.meas AC skirt_filter PP mag(V(C_OUT)) FROM 3400 TO 7800
.meas AC Amin MAX mag(V(C_OUT)) FROM 7800
.meas AC fp_low FIND mag(V(C_OUT)) AT 300
.include lmx58_lm2904.lib
* Trimpot to tune input bias current
.backanno
.end
```

Appendix G.2. Individual Circuit

```
XU1 N002 OUT +5V 0 OUT TL974
V1 IN 0 SINE(2.5 20m {FREQ}) AC 1 0
R2 N002 N001 {R2}
R1 N001 IN {R1}
C2 OUT N001 {C2}
C1 0 N002 {C1}
V2 +5V 0 5
.include TL974.sub
* .tran 0 10m 9m 1u
.ac dec 1000 10 20k
* In MATLAB component calculator script set print_ltspice \n to true to get
full table and paste below.\nThis is the function: \n
component_calculator_vcvs(Q, wn, C_base, print_table, print_ltspice)
* INDIVIDUAL VCVS LOWPASS FILTER
* .step param index 0 24 1\n.param FREQ 1k\n.param R1 =
table(index,...\n.param R2 = table(index,...\n.param C1 =
table(index,...\n.param C2 = table(index,...
.step param index 0 2 1
.param FREQ 1k
.param R1 = table(index,0, 1k,   1, 1.8k, 2, 12k)
.param R2 = table(index,0, 3.3k, 1, 18k,  2, 56k)
.param C1 = table(index,0, 2.2n, 1, 2.2n, 2, 2.2n)
.param C2 = table(index,0, 270n, 1, 47n,  2, 5.6n)
* Syntax: R1 = table(index, 0, A, 1, B, 2, C)\nwhere A, B and C are the
values for their respective stage
* AC Analysis
* Transient Analysis
.meas AC Gain MAX V(OUT) FROM 10 TO 20k
.meas TRAN Vpp PP V(OUT)
* .step param index 0 2 1\n.param FREQ = table(index, 0, 3613, 1, 2837, 2,
1744)\n.param R1 = table(index,0, 1k,   1, 1.8k, 2, 12k)\n.param R2 =
table(index,0, 3.3k, 1, 18k,  2, 56k)\n.param C1 = table(index,0, 2.2n, 1,
2.2n, 2, 2.2n)\n.param C2 = table(index,0, 270n, 1, 47n,  2, 5.6n)
.backanno
.end
```