TOUCHPAD-CONTROLLED PARAMETRIC EQUALIZER

By

Farsheed Hamidi-Toosi
Anthony Mangognia
Alexander Spektor

ECE 445, SENIOR DESIGN PROJECT

SPRING 2005

TA: Chad Carlson

May 3, 2005

Project No. 24

ABSTRACT

This paper discusses the design and testing of a touchpad-controlled parametric equalizer for audio applications. After an overview of the features and benefits of such a device, the document divides it into three subcomponents: the touchpad, the touchpad ⇔ DSP interface, and the DSP filter. The design and testing of each subcomponent is then discussed in detail, followed by testing procedures and results for the finished product. Also included is a cost analysis, detailing both parts and labor costs for the project.

TABLE OF CONTENTS

1. INTRODUCTION

Before reviewing the full details of the project design, it is important to understand its purpose, usefulness to the consumer, and overall design.

1.1 Project Purpose

Our project is a "Touchpad-Controlled Parametric Equalizer" for audio applications. We selected the project because current interfaces for equalizers are awkward; they involve cumbersome knobs and sliders. The project is exciting because it provides a new and improved method for musicians and sound engineers to interface with their equalizers so that they can achieve the perfect sound quickly during recording sessions and, more importantly, live performances.

The goal of the project is to provide musicians, sound engineers, and sound designers with a parametric equalizer filter for audio signals. It functions by allowing the user to move his or her finger on a touchpad surface, changing the individual parameters of the filter. More specifically, the horizontal position of the fingertip on the touchpad surface selects the center frequency of the filter and the vertical position selects the amplitude boost or attenuation for that particular frequency. Of course, there is a smooth curve around this boosted/cut frequency, for a natural-sounding filter effect. Sharp discontinuities are avoided, as they typically create unpleasant audio artifacts. The bandwidth of the filter is selected by the average of the total pressure exerted by the user onto the touchpad.

1.2 Benefits and Features

Product Benefits to the End Customer

- Ease of control: Simple finger movements on a flat surface are easier than the traditional knob/slider approach to control over parametric equalizers.
- Instant feedback: As soon as the parameters are changed via the touchpad, the user gets auditory feedback.
- Easy to visualize and understand: The position of the finger corresponds to the position of the curve on a typical filter frequency response graph.
- Large control surface: The large area of the touchpad allows for more precise control.

Product Features

- Real-time parametric equalization of audio signals through a DSP.
- Independent control over frequency, boost/attenuation, and bandwidth of the filter through separate state variable IIR filter design.
- External tuning circuit to ensure filter accuracy.

1.3 Subprojects

The product is based on a modular design, consisting of the touchpad, the touchpad interface, and the DSP, as seen in the block diagram of Figure 1.1. The separation of the project in this manner allows a fair distribution of project design and execution among the members of a three-person team like ours.

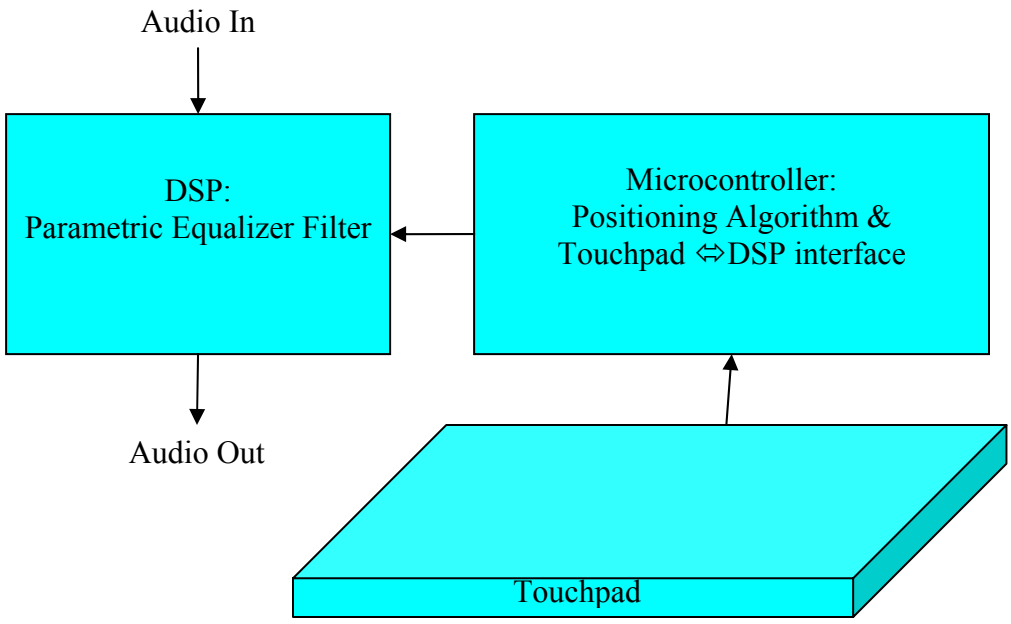


Figure 1.1 Project Block Diagram

1.3.1 Touchpad

The touchpad subproject includes the touchpad and its enclosure, as well as the four pressure sensors and the circuitry necessary to amplify sensor signals and eliminate unwanted DC bias. The input to the touchpad is physical contact with the user and the output is amplified sensor data, which is sent to the microcontroller.

1.3.2 Microcontroller

The microcontroller subproject includes the circuitry necessary to collect touchpad sensor data, analyze the data, and transmit it serially at RS-232 standards to the DSP. The input to the microcontroller is the sensor data and the output is positioning data, which is sent to the DSP.

1.3.3 DSP

The DSP consists of a Texas Instruments TMS320C54x with a RS-232 serial input, and 1/8 inch audio inputs and outputs (provided via in-lab adaptors). The DSP receives serial data from the microcontroller, generates filter coefficients, and filters audio in real-time. Thus, there are two inputs to the DSP: the audio and positioning data from touchpad, and there is one output: the filtered audio.

2. DESIGN CONSIDERATIONS

In designing the project, we had to choose among several options for the three subprojects.

2.1 Touchpad Design Considerations

The basic purpose of the touchpad is to provide the user with a finite 3-dimensional surface which will allow the user to use as a tool to interact with an imaginary space. This space consists of a frequency range along one direction (X-direction), an amplitude range along another direction (Y-direction), and bandwidth along the other direction (Z-direction), as seen in Figure 2.1

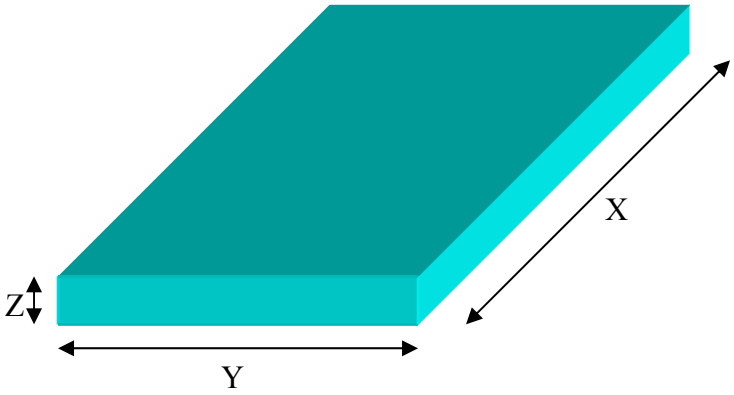


Figure 2.1 Three-dimensional touchpad space

The initial consideration for the touchpad was to use a commercial computer input device. Three major limitations prevented this consideration from being realized. First, the price of such a device is rather prohibitive—on the order of $100. Second, the area of the commercially-available touchpads is quite small—typically less than $10in^2$. Third, the commercial touchpads only allow for two dimensions of detection—the horizontal and vertical.

Since the commercial touchpad was not a viable option, our own design had to be substituted. The final touchpad designed for this system uses the method of sensing pressure at the four corners of the pad, which is translated into a set of voltages that are constantly sent to—and periodically sampled by—a microcontroller, which calculates positioning data.

The touchpad consists of three main components: the actual touchpad box, pressure-sensing devices, and an amplification stage to amplify the output of the pressure-sensing devices.

2.2 Interface Design Considerations

The primary requirement for the interface between the touchpad and the DSP filter is the availability of four analog inputs and a serial output. A secondary requirement is flexibility and affordability. The correct choice for this particular project was obvious.

The Microchip PIC16F877A microcontroller was chosen to function as the brains of the touchpad ⇔ DSP interface because of its in-lab availability, affordability, and suitability to our needs. With more than four analog inputs (with 10-bit A/D conversion) and an on-board UART for serial transmission, the PIC is ideal for this application [1]. Moreover, the PIC may be programmed in a form of Assembly, which minimizes run-time cycles and brings us closer to real-time operation.

2.3 DSP Filter Design Considerations

The DSP chosen was a Texas Instruments TMS320C54x model, a low-cost yet powerful DSP that has been used in embedded applications such as cell phones. The interface and board for the DSP chip was provided as an all-in-one unit by the University. There are two analog inputs and six analog outputs, primarily designed for multi-channel processing of stereo audio streams. Using the native TI programming environment, Code Composer, the program is loaded into the DSP memory through a standard parallel port and is executed. In addition to the stereo analog inputs, there is a serial port which allows the DSP to communicate with external devices by receiving or transmitting data.

While the DSP can be programmed in either C or Assembly, C was ultimately chosen due to its flexibility as a coding language and because of several built-in C function calls to the serial port.

Initially, we planned on using a Chamberlin filter to implement our parametric equalizer, but we found that it created undesirable resonant frequencies at the 3 dB corner frequencies when used as a notch filter. Therefore the much more reliable, Mitra-Regalia filter was used, which is used in the industry for parametric equalizer applications. The nice feature of this algorithm is that it produces symmetric filter outputs, with a bell like shape.

One obstacle we encountered was that the Mitra-Regalia second-order IIR filter produced instability when implemented using the direct form II model, due to feedback problems and quantization errors on the fixed-point DSP. However, by using an all-pass lattice configuration, stability was ensured. This is because the all-pass lattice ensures pole-zero cancellation even when implemented on fixed-point processors.

3. DESIGN DETAILS

Having gone over the design strategies for each subproject, it is now important to gain a firm understanding of each block's design details. In reviewing the specifics detailed in this chapter, please be sure to review the circuit diagram found in Appendix A.

3.1 Touchpad

As described in the design considerations, the touchpad converts pressure sensor data to a signal that is ready for analog-to-digital conversion in a microcontroller. The general flow of this process can be seen in Figure 3.1.

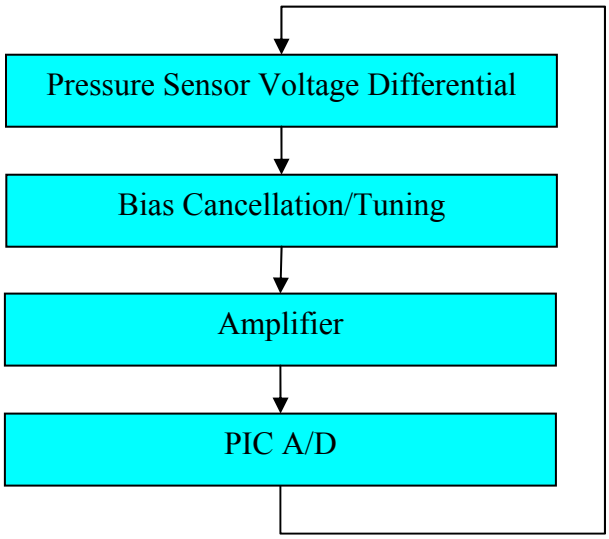


Figure 3.1 Touchpad Process Flow

3.1.1 Touchpad Box

The touchpad box not only provides a surface to which a finger can be placed to choose a frequency range to amplify or attenuate, but it also houses all of the circuitry for the touchpad and the components that allow the touchpad to interface with the DSP. The only components that are external to the touchpad are the DSP device, including the wiring to connect to the DSP (DB-9 connector), the touchpad tuning circuit, and the wiring to provide power to the touchpad. The box has the following dimensions: 12in (X-dir), 8in (Y-dir), and 4in (Z-dir). On one side (8in × 4in) the box includes a slit along the Y-direction that allows the touchpad board to be slid out in order to provide easy access to the internal components (Figure 3.2). In the interior, on the same side as just mentioned and the side opposite it, there are height-adjustable sensor mounts (Figure 3.3). These mounts provide a platform for the pressure sensors to be mounted on, slits for the leads on the pressure sensors to extend through, and screws through the sides that can be loosened to adjust the height of the pressure sensors and tightened to hold the mounts in place. The mounts are in an L-bracket shape, with one side in the X-Y direction that includes the slits and the surface to which the pressure sensors can be mounted to. The other side of the mounts is for support and is also used to insert the screws through in order to hold each mount in place. On one of the sides that is in the X-Z direction (8in × 4in), a square hole at the bottom center was cut to provide an opening for power cables and the DB-9 connector. The overall design may be seen in Figure 3.4.

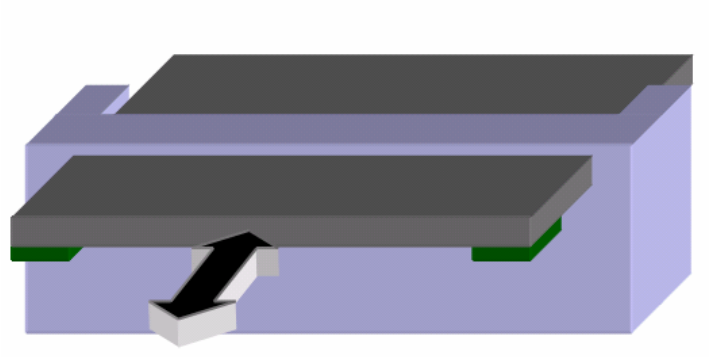


Figure 3.2 Slide-in Touchpad

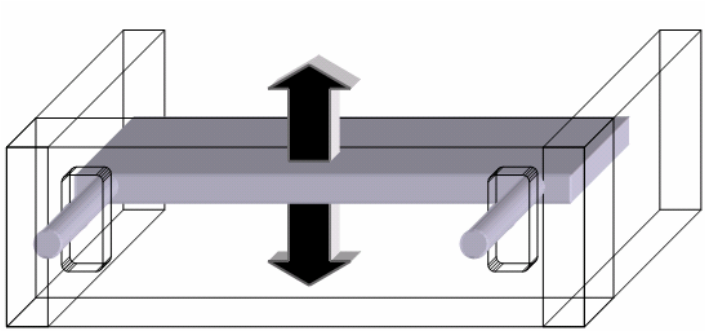


Figure 3.3 Height-Adjustable Sensor Mount

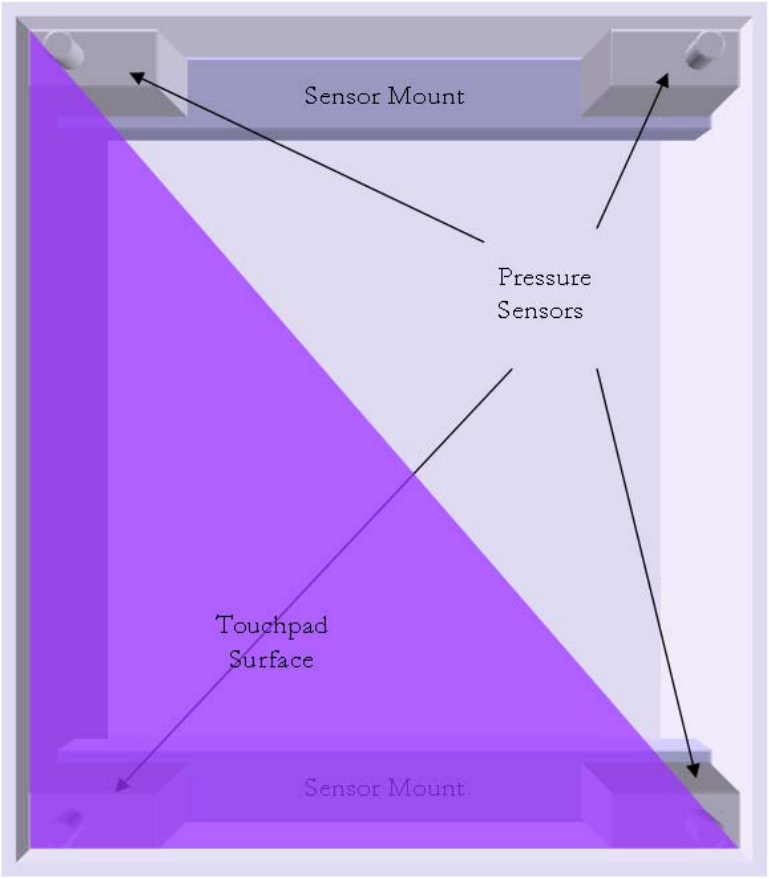


Figure 3.4 Bird's Eye View

3.1.2 Pressure Sensors

There are four pressure sensors located in each of the four corners of the touchpad box. The pressure sensors are mounted to the L-shaped bracket mounts, as described earlier. This project uses the 125PC30G1 pressure transducers, which are made by Honeywell [2]. They consist of a Wheatstone bridge with a variable resistance that changes with pressure. The pressure sensors require a +10V power supply across pins 1 and 3. When pressure is applies a differential voltage can be detected at the two output pins (pins 2 and 4). After some testing (to be discussed in more detail later in this paper), it was found that the pressures sensors were working in a pressure range of 0-30 psi, with a linear differential voltage ratio of about 8.33 mV/psi.

3.1.3 BIAS Cancellation/Tuning

It is desired that when entering the microcontroller A/D, the pressure sensor signal should be a voltage between 0V and +5V. Unfortunately, the accumulation of bias voltage and its amplification can lead to a signal that is confusing to the A/D. To combat this, the two outputs of each pressure sensor are sent through their own tuning circuit, which is designed to offset the DC bias voltage before it is amplified in the amplification stage. The tuning circuit consists of a pull-down 1MΩ resistor to ground from each pressure sensor output, with one of the resistors in series with a 100KΩ potentiometer. The four potentiometers sit on a separate board outside of the box, allowing final calibration once the touchpad is in its place.

3.1.4 Amplifier

The amplifiers used to bring the pressure sensor output to suitable levels were of the instrumentation type, namely the Analog Devices AD622AN. For this device, amplification ranges from 2x to 1000x, and can be chosen with an external resistor between pins 1 and 8. Another helpful aspect of these amplifiers is that they can be powered with a large range of voltages between ±2.6 V and ±15 V, allowing easy integration with our circuit [3].

3.2 Touchpad ⇔ DSP Filter Interface

The interface between the touchpad and the DSP is based around a Microchip PIC16F877A microcontroller running at a common 20MHz [3]. The interface circuit takes four analog inputs from the pressure sensors in the touchpad, converts the data to a digital format, performs calculations on the data, and sends the calculation results to the DSP via a MAX232 line driver, as seen in the workflow diagram of Figure 3.5.

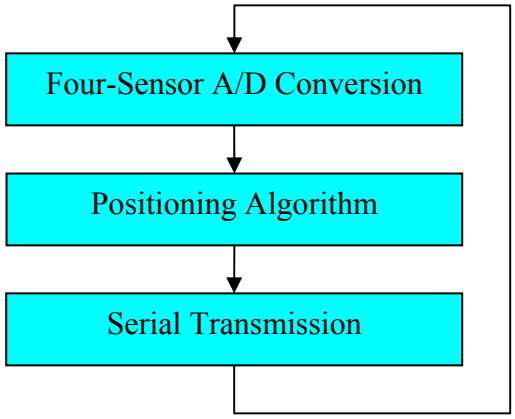


Figure 3.5 Interface Process Flow

3.2.1 A/D Conversion

The data from the pressure sensors in the touchpad are provided as a voltage with the lower voltage reference level of ground. The upper voltage reference level is provided with a voltage divider circuit, where the pin Vref+ on the microcontroller is connected between a pull-down resistor to ground and a pull-up resistor to +10V.

Using these reference voltages, the microcontroller performs analog to digital conversion, returning a ten-bit-long result, where '0000000000' represents ground-level voltage and '1111111111' represents Vref+-level voltage. Such a conversion allows for 1024 different readings, a resolution that is unnecessarily high for this project. Moreover, having ten bits for each reading makes it necessary to use two 8-bit words to store the readings. This additional storage requirement translates to the need for additional bytes for multiply and divide algorithms. In order to reduce the A/D result to a more manageable form, the 10-bit string is truncated to include only the 8 most significant bits.

Each sensor's data is read eight times per cycle and averaged. This procedure acts as a quick software low-pass filter for the sensor data, which tends to have some high-frequency variations.

3.2.2 Positioning Algorithm

Once the analog to digital conversion/averaging process is complete, the microcontroller is left with four byte-long values: *S0*, *S1*, *S2*, and *S3*. These four bytes represent the voltage levels at each of the four pressure sensors, and are in a counter-clockwise order starting with the bottom-left sensor. This bottom-left sensor is assigned to be the geometric origin, and the three remaining sensors are placed in the first quadrant of a standard geometric plane. In this quadrant, sensor *S1* is placed at (0, Y_{max}), sensor *S2* is placed at (X_{max}, Y_{max}), and sensor *S3* is placed at (X_{max}, 0), as seen in Figure 3.6.

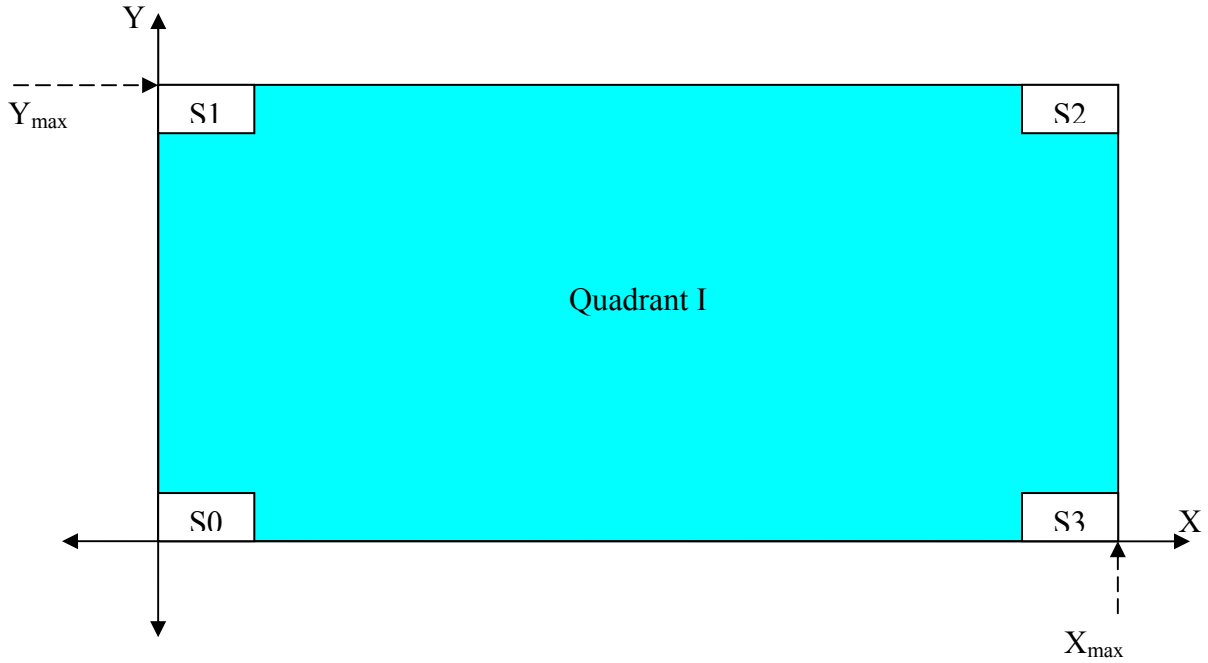


Figure 3.6 Positioning of Touchpad

Define X and Y to be the position of the finger on the touchpad. A simple weighting-averaging algorithm can reliably locate the finger based on the four pressure sensor readings. First, a center-of-mass calculation is performed on each edge of the touchpad, yielding the initial projection of the X and Y position of the finger onto each edge. These calculations are based on equations (3.1) – (3.4).

$$X_1 = S3\ X_{max} / (S3 + S0) \quad (3.1)$$
$$X_2 = S2\ X_{max} / (S2 + S1) \quad (3.2)$$
$$Y_1 = S1\ Y_{max} / (S1 + S0) \quad (3.3)$$
$$Y_2 = S2\ Y_{max} / (S2 + S3) \quad (3.4)$$

A rough idea of the finger's position can be determined from numerical averages, as seen in equations (3.5) and (3.6).

$$X_{avg} = (X_1 + X_2) / 2 \quad (3.5)$$
$$Y_{avg} = (Y_1 + Y_2) / 2 \quad (3.6)$$

However, it may be assumed that readings on the side closer to the finger are more accurate. Therefore, the X and Y position are based on a weighted average, where the edge that is closest to the finger's position weighs more, as in equations (3.7) and (3.8).

$$X = X_1\ (Y_{max} - Y_{avg}) + X_2\ (Y_{avg}) \quad (3.7)$$
$$Y = Y_1\ (X_{max} - X_{avg}) + Y_2\ (X_{avg}) \quad (3.8)$$

3.2.3 Serial Transmission

Once the positioning coordinates have been determined, the microcontroller transmits the data to the DSP. The data are sent in a serial form at 38,400kbps, with 8 data bits, one stop bit, no parity checking, and no flow control. These settings are chosen because they are the default for the DSP, and changing the settings in the microcontroller to accommodate the DSP is easier than doing the reverse.

The data are sent in a redundant form to ensure accurate transmission. The format for the data frame is seen in Figure 3.7.

'230'	'230'	'230'	S0
S1	S2	S3	X
X	X	Y	Y
Y	'232'	'232'	'232'

Figure 3.7 Data Frame Format

Note that at first, "230" is sent three times as a start sequence, indicating that a frame is about to begin. This sequence is followed by the four sensor values, which are used for overall pressure calculation. Next, the positioning data are sent three times to ensure accuracy. Finally, "232" is sent three times to indicate the end of the data frame. The start and stop sequence values were chosen because they are greater than 200, which is the maximum position that may be sent.

3.2.4 TTL → RS-232 Conversion

The serial data to be transmitted are available only at TTL voltage levels, ranging from 0 to +5V. The standard for serial data transmission that we use is RS-232, which needs data to be between -12V and +12V. Hence, we use the MAX232 RS-232 line driver to convert from TTL to RS-232 levels, as seen in Figure 3.8 [4].

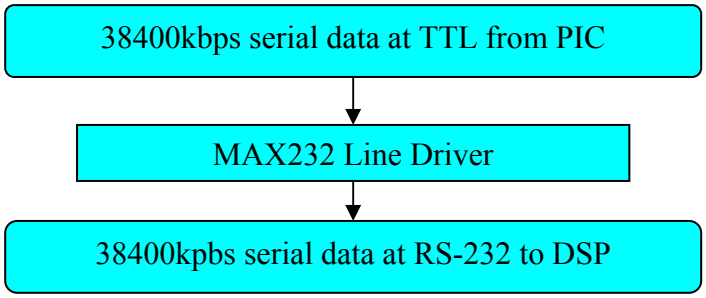


Figure 3.8 TTL → RS-232 Conversion

3.3 DSP Filter

The basic process flow for the DSP can be seen in Figure 3.9. The DSP receives data frames from the microcontroller, decodes the data, basing the filter coefficients on the X- and Y-position, and applies the filter to audio.

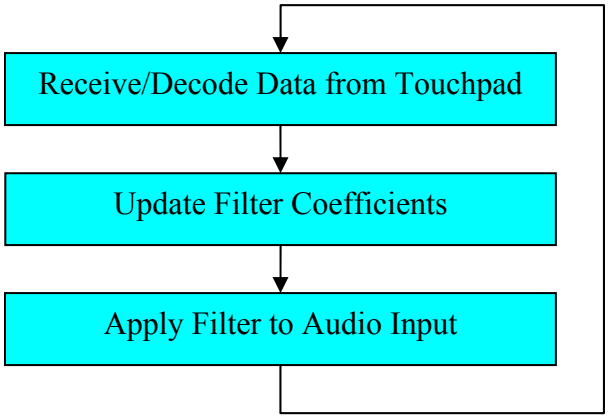


Figure 3.9 DSP Process Flow

3.3.1 Data Frame Capture

Once the DSP is running, it checks the serial port receive buffer every clock cycle to see if the PIC has sent any new information. Once the DSP receives the start bytes of the data sequence from the PIC, it begins loading a circular array buffer with the data sequence. Once the array buffer is full, it checks to see if the start bytes are at the beginning of the buffer. If it is, then that means that the entire data sequence is in the buffer and the data contained is reliable enough to start sending the data to the filter and to start filtering.

3.3.2 Parsing the Data Frame

Recall from Figure 3.7 that data were sent from the DSP in a certain format. By using redundancy, we minimized any error that could occur in the byte sequence as they are sent from the microcontroller to the DSP. Once the 16 byte stream is loaded into the data sequence array of the DSP, the DSP can parse the data sequence by using relative indexing. The value of the X coordinate is pulled from Data[8]. This

will allow 1 byte of forward and backward byte shift error if the sequence is slightly off. The value for the Y coordinate is pulled from Data[11], and the sensor values are pulled from Data[3]…Data[6]. Refer to Table 3.1 for received data → array mapping.

Table 3.1 DSP Received Data Mapping

Data[0]	← START BYTE
Data[1]	← START BYTE
Data[2]	← START BYTE
Data[3]	← SENSOR0 VALUE (BYTE)
Data[4]	← SENSOR1 VALUE (BYTE)
Data[5]	← SENSOR2 VALUE (BYTE)
Data[6]	← SENSOR3 VALUE (BYTE)
Data[7]	← X BYTE
Data[8]	← X BYTE
Data[9]	← X BYTE
Data[10]	← Y BYTE
Data[11]	← Y BYTE
Data[12]	← Y BYTE
Data[13]	← END BYTE
Data[14]	← END BYTE
Data[15]	← END BYTE

3.3.3 Converting the Data to Filter Coefficients

Once the four pressure sensor values are received, as well as the X- and Y-coordinates of where the finger is pressing down on the touchpad, the data-to-filter coefficient routine must begin. This essentially maps the X-value to a coefficient, which represents the filter amplitude, the Y-value to a center frequency coefficient, and it sums the four pressure readings and uses that to control the bandwidth of the filter.

3.3.4 Filter Algorithm

A second-order IIR filter was implemented on the DSP, based on a filter design by Mitra-Regalia. The Direct Form II implementation of the filter is shown in Figure 3.10 [5].

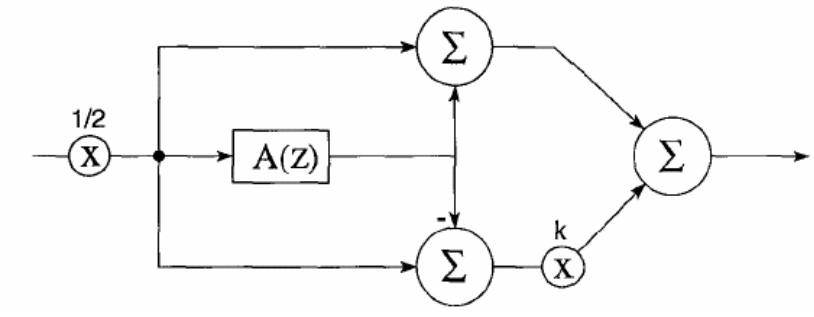


Figure 3.10 Mitra-Regalia Topology.

The input is a monophonic audio source directly connected via BNC cables to the DSP. The output is a monophonic audio out, also directly connected to the DSP. The block labeled A(z) is a second-order all-pass filter, implemented using a lattice structure. The block diagram is shown in Figure 3.11.

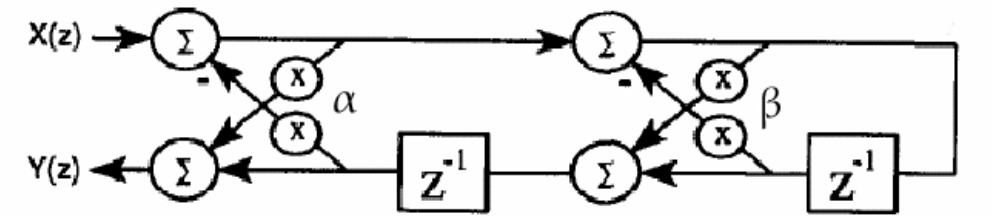


Figure 3.11 2nd order All-Pass Lattice, A(z)

The advantage of the all-pass lattice structure over standard direct form II is that it remains stable even when the coefficients are changed quickly. This is because the lattice structure ensures that the poles and zeros cancel each other, even if there is quantization error. The result is a stable and reliable filter. The all-pass lattice will not affect the magnitude of the input, but will change the phase of the input signal in such a way that when its output is added to the original input, as performed in the Mitra-Regalia topology, the phases will cancel at the desired frequency. The variable k controls whether the input is boosted or attenuated at the desired frequency [6]. The equations for the three parameters are listed as (3.9)-(3.11).

$$k = 10^{(\text{gain [dB]} / 20\text{ dB})} \tag{3.9}$$

$$\beta = -\cos(\omega c) \tag{3.10}$$

$$\alpha = \frac{(1 - \tan(\text{BW[rad]}/2)}{(1 + \tan(\text{BW[rad]}/2)} \tag{3.11}$$

Each parameter of the filter can be changed independently without affecting the other parameters. This is an essential feature for any parametric system. Please refer to Appendix B, Figures B.1–B.3, to see how each parameter affects the filter.

3.3.5 Data → Parameter Mapping

Once the data from the PIC has been parsed and stored in variables, they must be converted to the above coefficients via a mapping algorithm.

3.3.5.1 Gain Mapping

The number of data points expected from the PIC to control gain (Y) is an integer number from 0 to 50. Mapping a Y-value of 25 to a gain of zero resulted in a gain of 1. The equation (3.12) is used for mapping Y → K.

$$K = Y * 0.4 \tag{3.12}$$

When Y = 25, the resulting K = 1, causing a gain of 1. The reason it was implemented in such a way is because the DSP cannot store multiplier values in memory that are greater than 1 because it uses fractional arithmetic on 16 bit registers. The solution to this problem was to make a looping function that added 0.04 to the value of K each iteration. Therefore, to achieve any arbitrary gain we simply execute the loop Y number of times. If the loop is executed Y = 25 number of times, the gain is equivalent to 1. If Y = 50 the gain is 2. If Y = 1, the gain is 0.04.

3.3.5.2 Bandwidth Mapping

Alpha is preset so that the bandwidth of the filter is 100Hz when the DSP first starts running. This provides immediately audible results, and seems to be the best starting bandwidth from a usability standpoint, based on usability tests. The bandwidth is determined from the sum of the four raw sensor values based on their pressures. Since each sensor value ranges anywhere from 0-100, a maximum of 400 values could be used for the bandwidth. The way these 400 values are converted to be a number from 0 – 1 was a simple multiplication by 80 on the DSP. Since the DSP uses fixed fractional point arithmetic using 16 bits, a decimal value of 32768 (2^{15}) corresponds to unity. By multiplying 400 by 80, we shift 0 – 400 values into the range of 0 – 32000 fractional, or 0 – 1 for practical purposes. A value of 0 corresponds to no pressure; a value of 400 corresponds to maximum pressure on the touchpad. As the pressure increases, the value for alpha is decreased to zero, which causes the bandwidth to increase as pressure increases.

3.3.5.3 Center Frequency Mapping

Center frequency is mapped via a logarithmic scheme. Such a scheme is implemented because human hearing is approximately logarithmic and most of the information in audio resides below 1 kHz. Therefore, the 200 different frequency values coming from the PIC are mapped logarithmically to cover the full range of hearing from 20-20 kHz. By linearly mapping 128 values to cover 20-1 kHz, 64 values to cover 1 kHz to 11 kHz, and 8 values to cover the remaining spectrum, a crude log approximation is implemented. The result is a very intuitive frequency sweep as the user sweeps his or her finger from left to right on the touchpad. Such a pseudo-logarithmic implementation is commonly seen in industry standard equalizers. Such a scheme also allows for less frequency resolution on average, with the lower frequencies having a much higher frequency resolution than the lower ones. This allows the resolution at lower frequencies to be around 100Hz, but this is suitable since the bandwidth is fixed at 100Hz to start. The result is an audibly-smooth filter, as the frequency is changed in real-time. Another trick used to smooth the transitions from frequency to frequency is to average the current value with the last value. This solution eliminated any audible pops or clicks as the IIR filter adjusted to the parameter changes.

3.4 5V Power Supply Circuit

A minor, yet important detail must be mentioned here, in regard to the availability of a +5V rail to power the PIC, MAX232, and oscillator chips. This voltage is available because of the addition of a MC7805T voltage regulator, which converts the +10V input to +5V. For details about this small circuit, please refer to Appendix A [7].

4. DESIGN VERIFICATION

In order to verify our project design, we first tested individual components, and then performed tests on the overall product.

4.1 Component Testing

Individual attention was paid to some of the components to ensure that they contribute correctly to the overall design.

4.1.1 Pressure Sensor

Being the main input to the whole system the pressure sensors were the components that all other components relied on. Thus, it was important to know how they functioned and whether they worked up to the standards of precision that would be necessary for the rest of system to work properly. After the basics were known, such as the supply voltage required, the pressure range the device was operating in was necessary to determine. Also, it would be helpful to know the change in voltage that corresponds with a change in pressure. This would help determine the amount of amplification necessary. Another important thing to consider was whether the pressure sensors would arrive at the same voltage each time a pressure was applied. Stated differently, the current state of the pressure sensors should not depend on the previous state of the pressure sensors, no hysteresis, if the rest of our device was to function the same way every time it was used. The test that was designed to determine how the pressure sensors functioned is described in the following paragraphs.

The pressure sensor was powered according to the specifications. The output of the pressure sensors was sent to an instrumentation amplifier with a known amplification selected. The output from this was then sent to an oscilloscope. In order to determine the specific pressure being applied, the pressure sensor was hooked up to sphygmomanometer by using a tube that had a pump (the usual hand pump on a sphygmomanometer) at one end that branched into two tubes, one of which connected to the actual sphygmomanometer and the other which connected right onto the knob of the pressure sensor.

At this point the pressure was pumped up to different pressures, moving up from zero pressure to 300 mmHg in increments of 10 mmHg. When 300 mmHg was reached, the pressure was then released in increments of 10 mmHg back down to zero pressure. This test was repeated 5 times and the results were plotted in Figure 4.1.

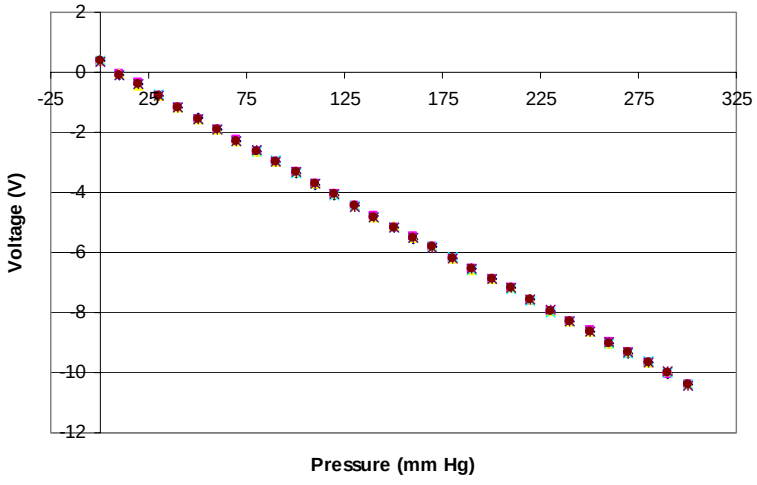


Figure 4.1 Voltage Differential Created by Applied Pressure

As can be seen from Figure 4.1, there is no hysteresis occurring with these pressure sensors. It was also determined that the average finger pressure that would be applied to the touchpad would not exceed 270 mmHg and the pressure sensors worked fine in this range, producing that same difference in output at both low pressures and high pressures in this range. Also, it was concluded that the pressure sensors were functioning in the pressure range of 0-30 psi (well beyond the pressure range expected from the finger, 1 psi corresponding to 51.7 mmHg), which was one of the three possibilities on the data sheet [2], and that there was a change in voltage of around 8.33 volts for every psi.

4.1.2 Latency

The target of less than 100 milliseconds of latency for the overall system was achieved. Each component of the system was tested for a delay at its individual stage. The pressure sensors did not cause significant delay, and the PIC was programmed in assembly, which optimized its cycle time to a nanosecond range. The DSP had some latency initially due to extensive for-loop calls and unnecessary branching operations. By eliminating these loops, the DSP achieved approximately 30000 cycles. Considering the DSP processor operates in the MHz range, this is only about 30 ms of delay. Usability tests confirmed that the total system delay was negligible and did not interfere with device operation.

4.1.3 Filter Algorithm Test

The procedure for testing the filter algorithm was to simulate different values for bandwidth, gain and center frequency in Matlab, and compare with real tests performed on the system with an oscilloscope.

The filter algorithm was coded in Matlab, and simulated to test the filter with an impulse signal as well as broadband Gaussian white noise. The first test fixed the center frequency to 11.025 kHz and the gain to 0.5. It then varied the value of bandwidth coefficient alpha from 0.1 to 0.9. This corresponds to decreasing the bandwidth from 10.2 kHz to 700Hz. The simulations matched the expected results when compared to theoretical tests from previously published papers. An impulse was sent in to retrieve the impulse response of the filter, which shows an accurate response of the filter. Please refer to Figure C.1 in Appendix C for the results.

Next, the bandwidth coefficient was fixed at 0.75, the center frequency was fixed at 11.025 kHz, and gain was varied from -12dB to +12dB. Again, our simulations matched predictions accurately. Please refer to Figure C.2 in Appendix C for the results.

Lastly, center frequency was varied from 5072 Hz to 16977 Hz with a fixed $\alpha = 0.8$ and a fixed GAIN = 0.5. The simulations verified that our algorithm was implemented properly. Please refer to Figure C.3 in Appendix C for the results.

After the impulse response was determined, Gaussian white noise at 2 watts was used as an input, with $\alpha = 0.75$, $\beta = -0.5$, and $k = 0.25$. This resulted in a notch filter at 7350 Hz. Since the ear responds to intensity fluctuations it is also useful to examine the power spectrum of the filter, noted below. The power spectrum shows a more accurate depiction of how the filter is actually working. Note the phase of the filter, which is approximately linear, except some slight nonlinear effects at very low and very high frequencies due to the bilinear transformation. Please refer to Figures C.4 and C.5 in Appendix C for the results of the FFT and Power Spectrum results, respectively. The linear-nonlinear effects may be seen in Figure C.6 in Appendix C.

The same test was successfully repeated with $k = 2$. Please refer to Figures C.7 and C.8 for the results.

4.2 Finished Product Tests

After the algorithm was coded on the DSP and integrated with the rest of the circuit, testing of the individual filter parameters was necessary to determine if the filter was working properly on the DSP.

4.2.1 Gain Tests

The input was an 11.025 kHz sine wave, 200mV P-P. Bandwidth was fixed at 500 Hz. Several values for the gain coefficient were hard coded, tested and tabulated below. The results matched our simulations approximately (within 5% error), and may be seen in Table 4.1.

Table 4.1 Gain Test

Input frequency, sine	Input peak-to-peak voltage	Gain (V/V)	Output peak-to-peak voltage	% error
11.025 kHz	200 mV	0	5 mV	1.0
11.025 kHz	200 mV	0.25	48.6 mV	2.8
11.025 kHz	200 mV	0.5	104.3 mV	4.3
11.025 kHz	200 mV	0.75	155.7 mV	3.8
11.025 kHz	200 mV	1	202.5 mV	1.3
11.025 kHz	200 mV	2	415.9 mV	3.9

The amplitude resolution for this filter was coded to be 0.24 dB. This is based on the fact that 50 different possible amplitude values were coming from the PIC, and these 50 values were spread across a +6dB and -6dB range. The ±6 dB was decided upon after extensive usability tests that determined it was in reasonable audible sensitivity. However, minor adjustments could extend the range from this 12 dB spread or beyond. By increasing the dB spread and increasing the number of amplitude values sent by the microcontroller, such a high resolution would be maintained. This amplitude resolution out performed our design review specifications, which specified an amplitude resolution of 2 – 3 dB.

The testing procedure of the amplitude response when controlled by the touchpad was accurately devised. Simply testing three values for 6 dB, 0 dB and -6 dB gain was sufficient for trusting the functionality of the system. When the finger was located at the maximum (Y = 50) coordinate value, the gain was equivalent to 2, or 6 dB. In the center (Y = 0) the gain was 1, or 0 dB. When Y = 0, the filter response showed a gain of about 0.5, or -6 dB. The only problem was that the lip of the box covered the edges of the touchpad by about a ½ of an inch. To really get ±6 dB the user would have to slide their finger under the edge of the box.

4.2.2 Frequency Test

Single frequency tests as well as a frequency sweep test were used to test the filter response on the DSP. Fixing the GAIN = 2, and the bandwidth to 100 Hz, allowed us to view the filter response due to broad band white noise using the FFT feature on the oscilloscope. The settings on the oscilloscope were: 20 kHz span centered at 10 kHz. The refresh time was on the order of milliseconds. The sweep feature simply was a loop in the code that swept the center frequency coefficient from 20 to 20 kHz over and over. By viewing the FFT output, we could see a resonant peak sweep from roughly 20 to 20 kHz, although the noisy output made it difficult to determine exact frequency values. For exact frequency

tests, we set the gain to 0 and the bandwidth to 100 Hz, then tested four different frequencies over the entire audio spectrum. By examining where the gain dropped to zero for these different test frequencies, we could determine where the filter was cutting out and if it matched what we expected. You can note that the expected frequency was less accurate at very low or very high frequencies, which is a result of the bilinear transformation and quantization error in the filter and DSP. The results of the test may be seen in Figure 4.2.

Table 4.2 Center Frequency Test

Expected center frequency	Actual center Frequency
100 Hz	115 Hz
1 kHz	893 Hz
11.025 kHz	11.025 kHz
18 kHz	17.8 kHz

4.2.3 Bandwidth Tests

Bandwidth was set to 100 Hz when the filter was initialized, however it could be set to as low as 10 Hz. Very small bandwidths would result in resonant "whistling" when boosting and there would not be audible results when cutting frequencies at such a high Q. Therefore a larger starting bandwidth was desired. When the user presses down, the total pressure of the touchpad increases the bandwidth to a maximum of 22050 Hz, essentially flattening out the center of the bandpass/notch filter. The sensitivity of this feature was in a usable range, where pressing hard produces dramatic results but slight fluctuations in pressure will not cause undesired or unexpected bandwidth variations.

4.2.4 Usability Tests

The most difficult to quantify, yet important, aspect of this design was its usability. We wanted the user to feel that by simply moving his or her finger on the touchpad, he or she would be able to hear audible results immediately. However, we did not want the pad to be so sensitive that it would be hard to use. Therefore, we tested with our ears what setting worked best for us, however extensive "tweaking" would be necessary before this design could be mass produced. In general, we found a ±6 dB gain range was adequate for a perceptual boost/cut, a pseudo-log frequency scaling provided a more natural frequency sweep across the auditory spectrum, and that an increased perception of filter boost/cut was enhanced when the bandwidth increased as the user pressed harder on the touchpad.

5. COST

Although the product is designed for musicians and sound engineers, whose other equipment is usually quite pricey, we did not want this product to be out of reach for hobbyists and amateurs. Therefore, it was important to keep a close watch on which parts we used and their prices as we designed the product.

5.1 Parts

Please refer to Table 5.1 for full information on the parts used for this project.

Table 5.1 Part cost analysis

Name	Manufacturer	Model Number	Cost	Q	Total	Source
Breadboard	Global Spec.	103-1350	$3.73	1	$3.73	ECE Store
Capacitor, 0.1uF	Generic	N/A	$0.05	10	$0.50	445 Lab
Capacitor, 1uF	Generic	N/A	$0.05	2	$0.10	445 Lab
DB-9 Connector, F	Generic	N/A	$2.53	1	$2.53	Part Shop
DB-9 Serial Cable	CNT	MC128-06	$2.18	1	$2.18	445 Lab
Diode	Generic	1N4001	$0.10	8	$0.80	445 Lab
DSP	Texas Instr.	54x	$30.00	1	$30.00	445 Lab
In-Amplifier	Analog Dev.	AD622AN	$5.13	4	$20.52	Order
Metal Box	Machine Shop	N/A	$90.00	1	$90.00	Machine Shop
Microcontroller	Microchip	PIC16F877A	$6.92	1	$6.92	445 Lab
Oscillator	FOX	F1100F	$1.40	1	$1.40	Part Shop
Potentiometer, 100K	Bourns	3296	$2.50	4	$10.00	Part Shop
Pressure Transducer	Honeywell	125PC30G1	$5.00	4	$5.00	Part Shop
Resistor, 56	Generic	N/A	$0.05	4	$0.20	Part Shop
Resistor, 330	Generic	N/A	$0.05	1	$0.05	445 Lab
Resistor, 3k	Generic	N/A	$0.05	1	$0.05	445 Lab
Resistor, 1M	Generic	N/A	$0.05	8	$0.40	445 Lab
Resistor, 220	Generic	N/A	$0.05	4	$0.20	445 Lab
RS-232 Line Driver	Maxim IC	MAX232CPE	$3.31	1	$3.31	Part Shop
Solder Board	RadioShack	276-147	$4.59	1	$4.59	ECE Store
Voltage Regulator	ON Semi.	MC7805CT	$0.57	1	$0.57	Part Shop

5.2 Cost Analysis

The total cost of the parts is $183.05, or $93.05 without the box, and $63.05 without the box and the DSP. Considering that a significant portion of the cost of the box is the labor cost, the actual box part cost may be reduced to something on the order of $10, for a DSP-free total of $73.05.

The overall cost for the project may be analyzed using equation (5.1).

$$\text{Cost} = \text{Parts} + (\text{ideal salary (hourly rate)} \times \text{actual hours spent} \times 2.5) \tag{5.1}$$

So, the overall cost for the project with the assumption of $50/hour salary is seen in equation (5.2).

$$\text{Cost} = \$183.50 + (\$50/\text{hour} \times 10\ \text{hours/week} \times 13\ \text{weeks} \times 2.5) = \boxed{\$16{,}433.50} \tag{5.2}$$

6. CONCLUSIONS

6.1 Accomplishments

The goal of building a realizable, comparatively low cost touchpad controlled parametric equalizer was achieved. The parameters of bandwidth, gain, and center frequency were controlled by a touchpad with three degrees of freedom. In our tests, the X-direction controlled center frequency, the Y-direction controlled gain, and the Z-direction (total pressure) controlled the bandwidth of the filter. By implementing a boost/attenuation of ±dB, a pseudo-log frequency sweep, and a dramatic bandwidth increase as the user pressed down, we achieved desirable and marketable results that any musician or sound engineer would be delighted with. Due to the simple construction of the touchpad, it is a cinch to operate and maintain. The sliding touchpad surface allows easy "under-the-hood" access to the microprocessor and circuit board components, and an external tuning circuit ensures accurate operation from session to session.

For sound engineers, this device provides limitless possibilities for equalizing in a recording studio. By having multiple boxes controlling multiple microphone inputs, a sound engineer could control the various equalizations of the inputs without having to fiddle with knobs or sliders. In a live situation, this device would help deal with feedback problems. If the system begins to create audio feedback the sound engineer can quickly sweep and catch it before it reaches a damaging volume level.

6.2 Uncertainties

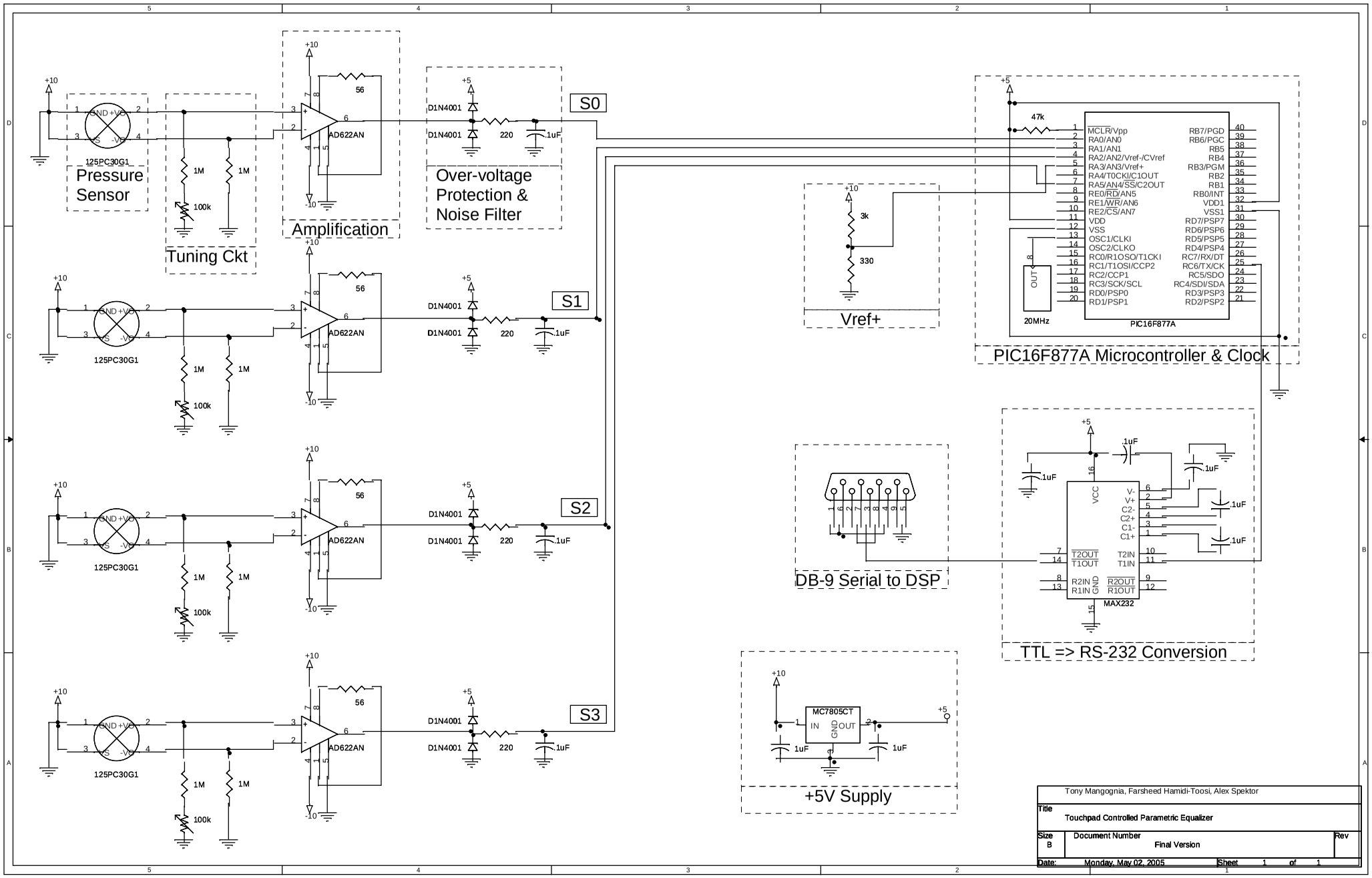
One uncertainty about our design was that it was roughly four inches thick. Such a large thickness is undesirable by today's super-slim standards. This did not adversely affect the overall operation, but it might be an inconvenience for the touchpad's target audience. In addition, the pressure sensors used were of poor quality. They required a large gain to amplify the sensor signals which introduced a lot of noise into the system. The solution to this problem would be to use better, higher quality pressure sensors.

6.3 Future Work / Alternatives

In the future, we could make this device more portable by incorporating a 120 AC transformer so that the device could be plugged into the wall. Alternatively, it could run on batteries, and a smaller DSP could be integrated into the design so that it all fits in one box.

For musicians, this device could be used as a control device for a myriad of audio effects. The popularity of existing audio effects boxes, despite no major advancements since the 1970s, means that interest is guaranteed. Perhaps the application of a touchpad controller for echo, distortion, and other effects would open up new sonic possibilities for keyboardists, guitarists, and musicians alike. And the growing popularity of home recording studios means that a more intuitive input device for technical components is needed.

Since the real innovation behind this project is the touchpad itself, we can extend the control aspects of this project to other fields. Kids, the elderly, and the disabled could use the touchpad as an input device to control a number of different devices. The advantage of this touchpad over other commercially available touchpads is its large area and affordability.



APPENDIX A - CIRCUIT DIAGRAM

APPENDIX B – IMPACT OF PARAMETERS ON FILTER SHAPE

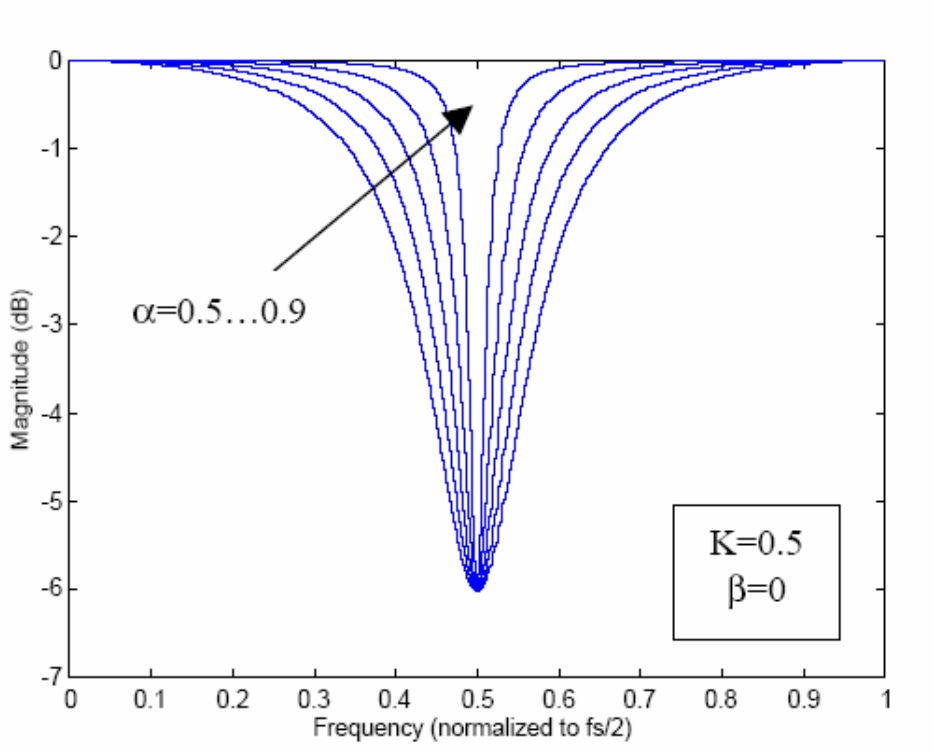


Figure B.1 Changing alpha changes the 3dB bandwidth of the filter

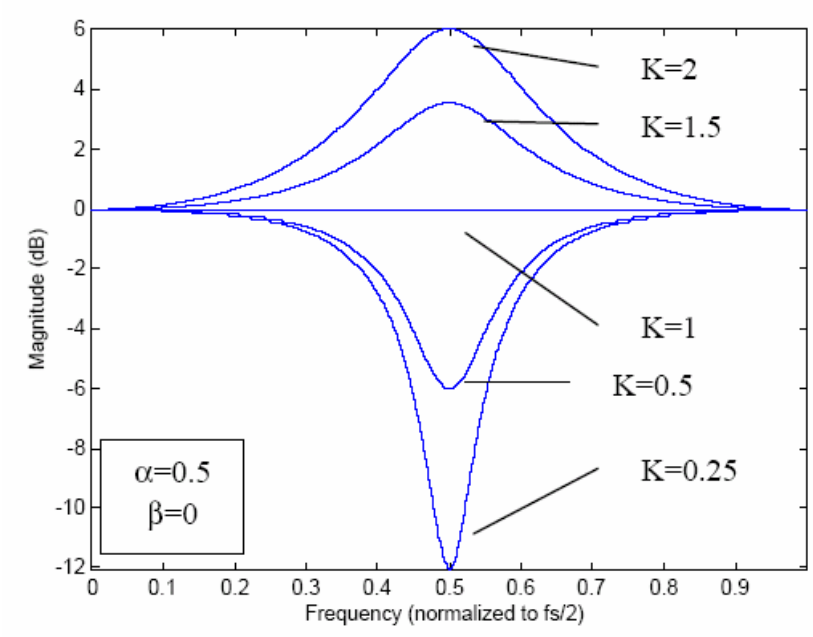


Figure B.2 Changing K changes if the filter boosts or cuts by a certain gain factor

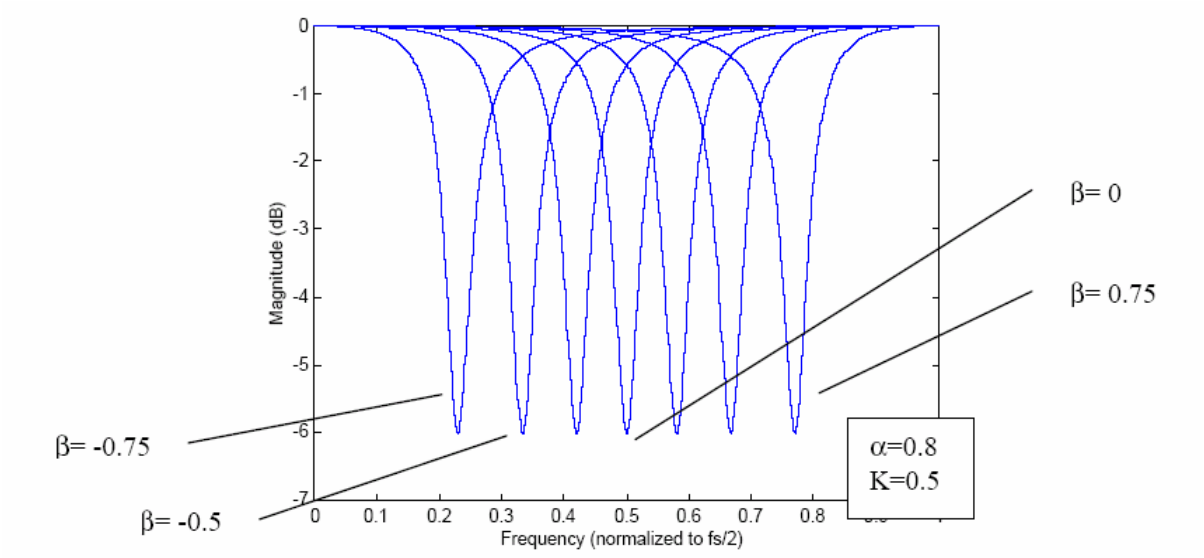


Figure B.3 Changing beta results in a change in center frequency

APPENDIX C – TEST RESULTS

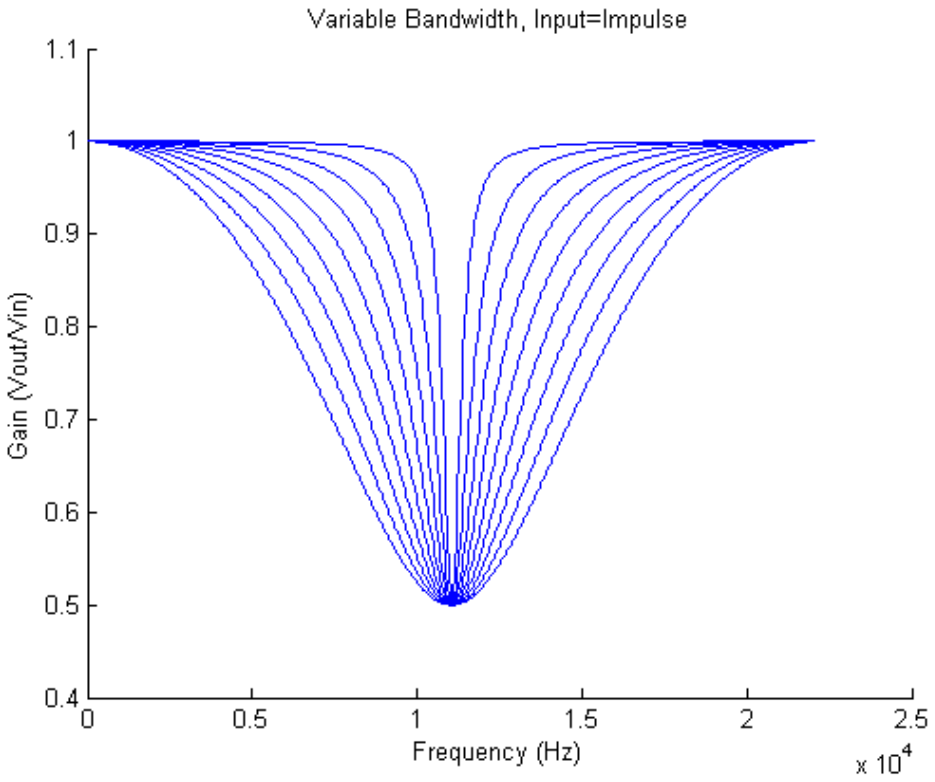


Figure C.1 Filter Response with the Bandwidth Varying from 10.2 kHz to 700Hz.

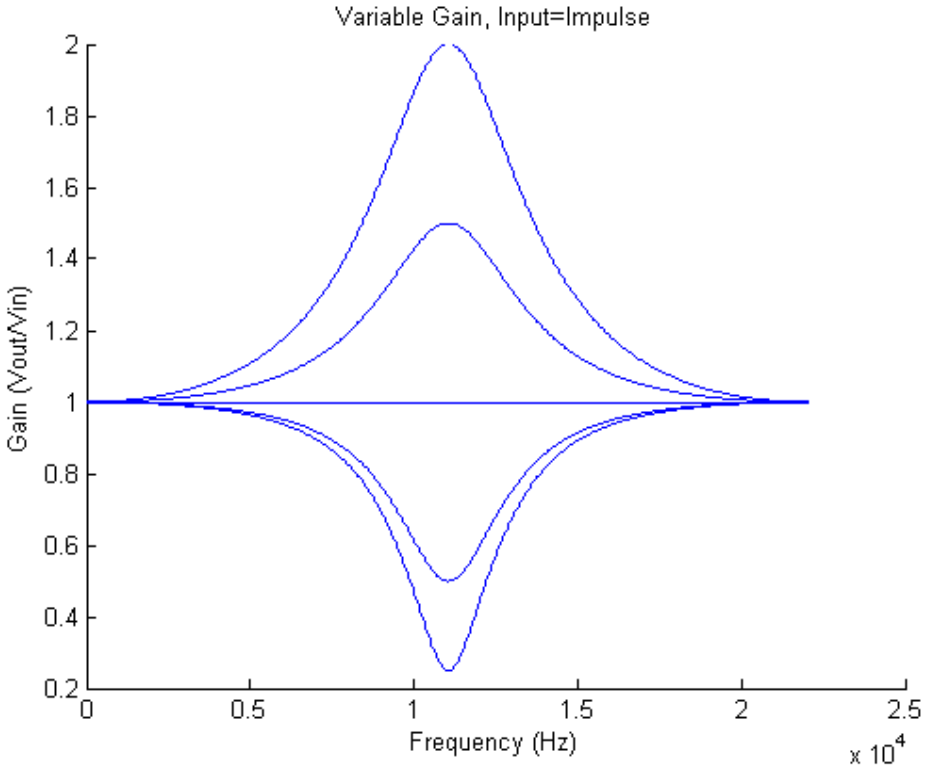


Figure C.2 Filter Response with Varied Gain

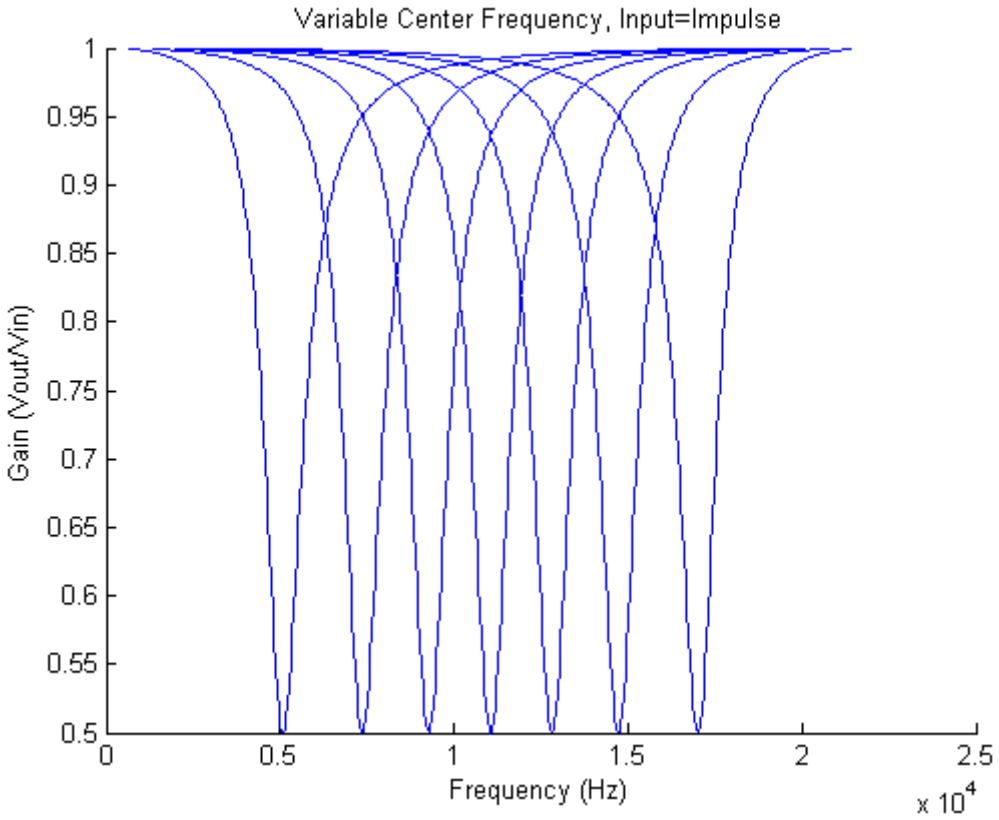


Figure C.3 Filter Response with Varied Center Frequency

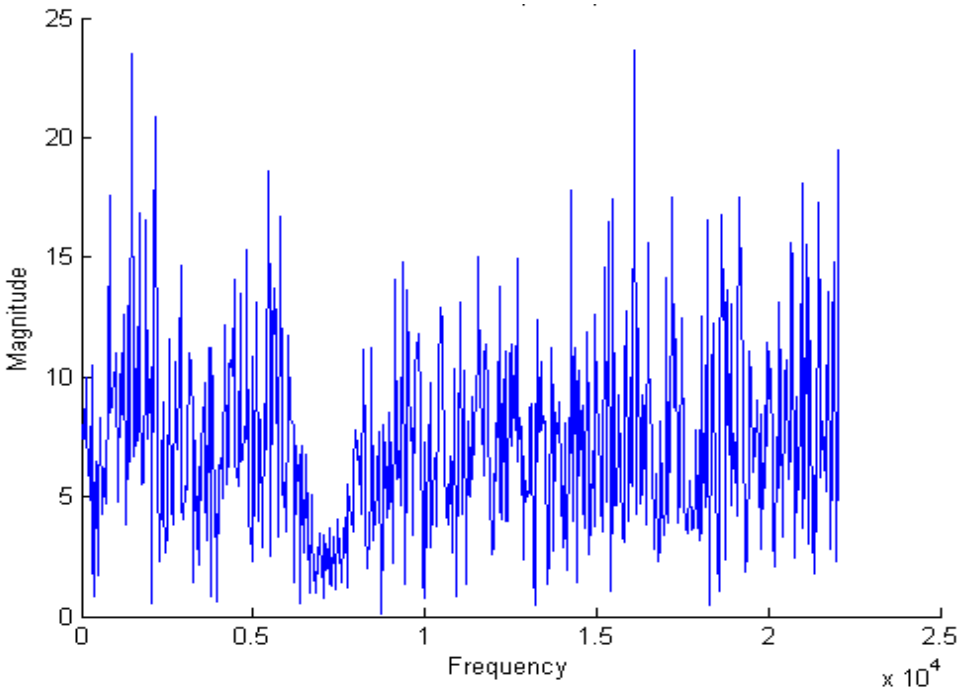


Figure C.4 FFT of Gaussian White Noise Input, $\alpha = 0.75$, $\beta = -0.5$, $k = 0.25$

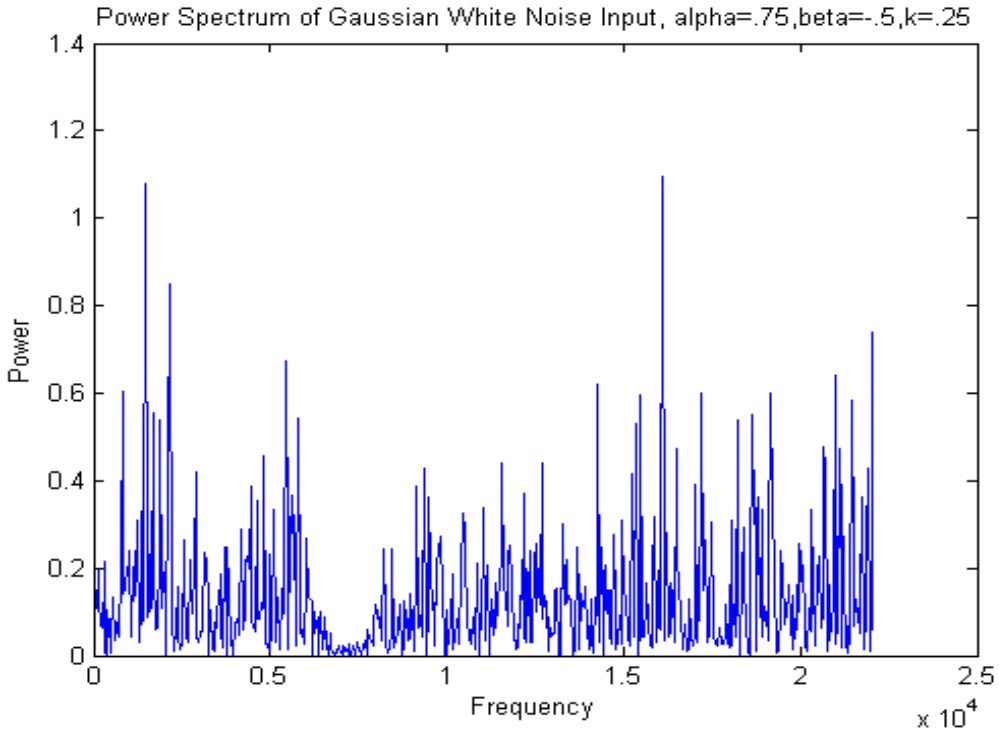


Figure C.5 Power Spectrum of Gaussian White Noise Input, α = 0.75, β = -0.5, k = 0.25

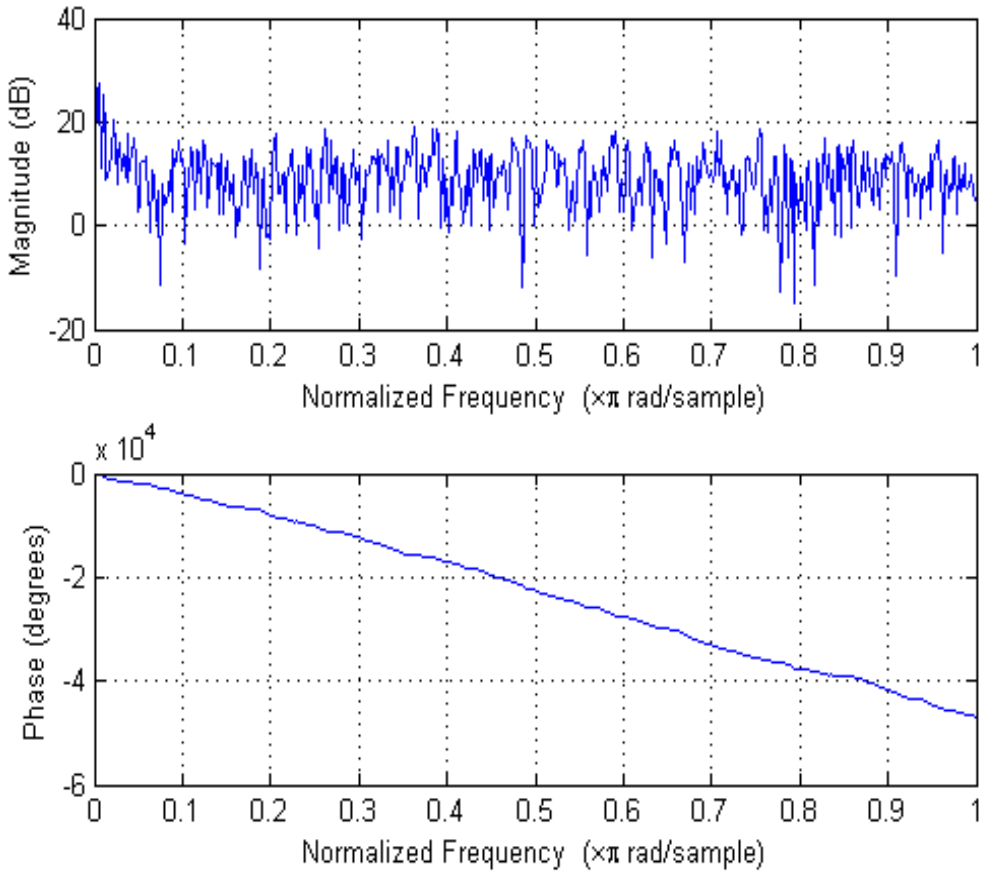


Figure C.6 Center Linearity & Lateral Curvature

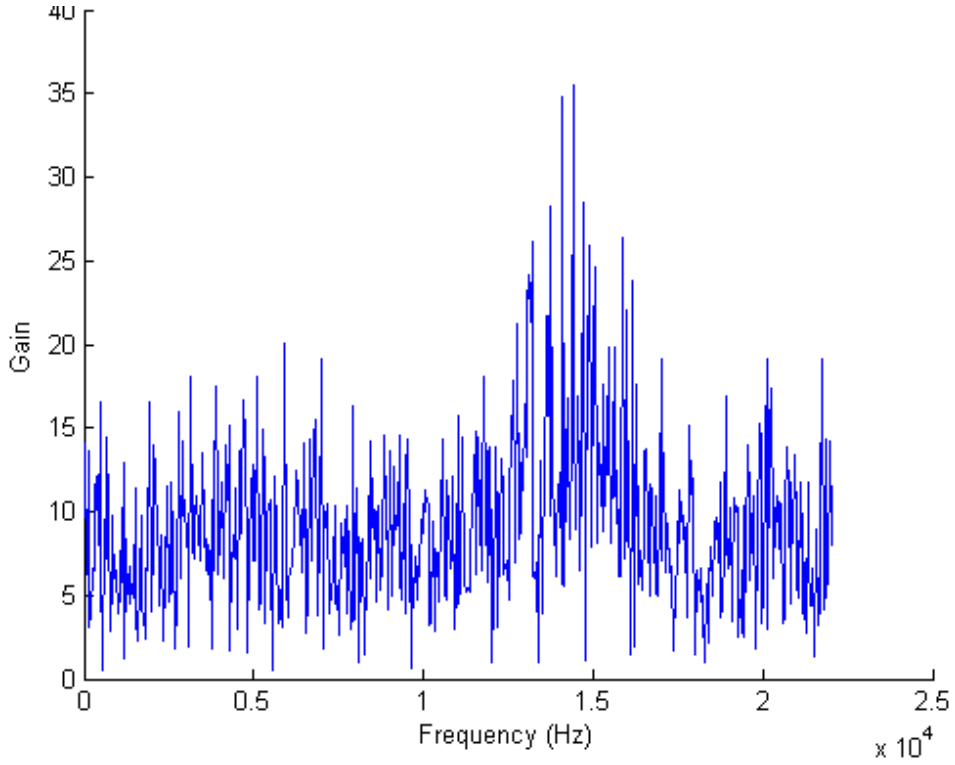


Figure C.7 FFT of Gaussian White Noise Input, α = 0.75, β = -0.5, k = 2

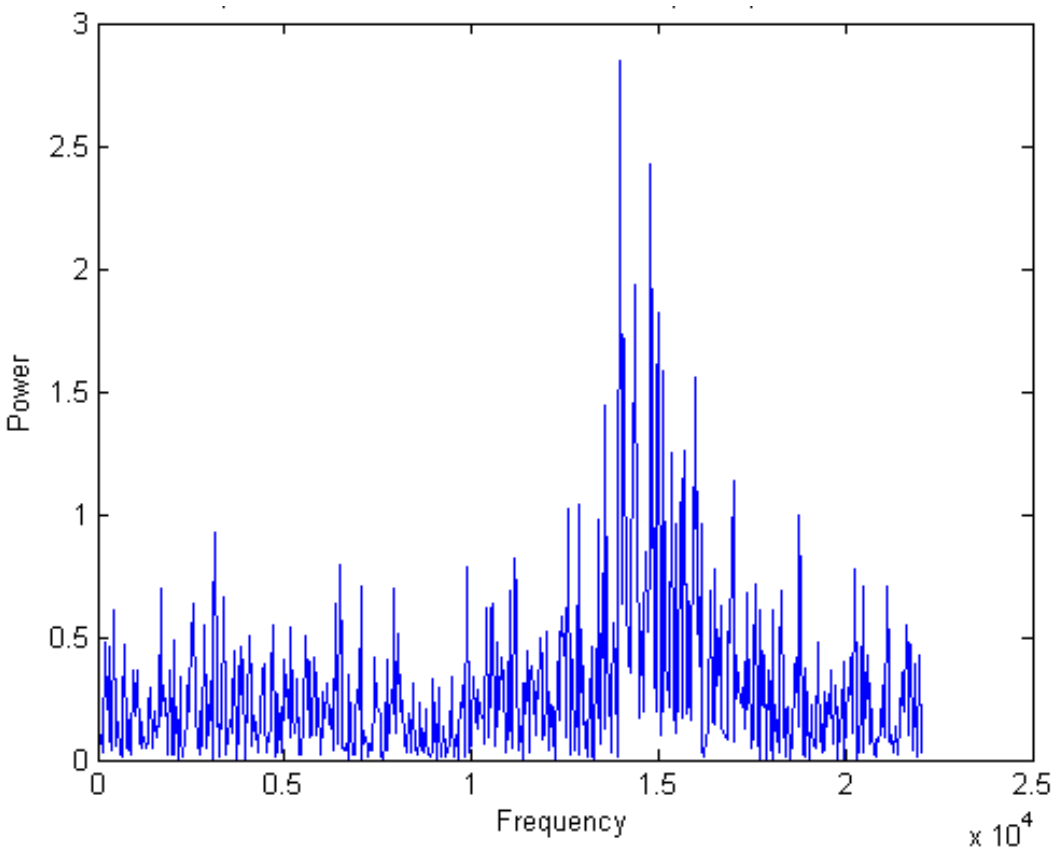


Figure C.8 Power Spectrum of Gaussian White Noise Input, α = 0.75, β = -0.5, k = 2

REFERENCES

[1] Datasheet, *PIC16F877A 40 Pin Enhanced Flash Microcontroller*, Microchip Technology, 2003.

[2] Datasheet, *125PC30G1 Pressure Transducer*, Honeywell Inc.

[3] Datasheet, *AD622AN Low Cost Instrumentation Amplifier*, Analog Devices Inc., 1999.

[4] Datasheet, *MAX232 +5V Powered Multichannel RS-232 Line Drive/Receiver*, Maxim Intergrated Products, 2001.

[5] "The Equivalence of Various Methods of Computing Biquad Coefficients for Audio Parametric Equalizers," Robert Bristow-Johnson, Wave Mechanics, Inc., http://www.harmony-central.com/Effects/Articles/EQ_Coefficients/EQ-Coefficients.pdf

[6] "IIR Filters and Equalizers," University of Colorado, Spring 2003, http://ece-www.colorado.edu/~ecen4002/lab4_2004.pdf

[7] Datasheet, *MC7805T Voltage Regulator*, ON Semiconductor, 2000.