– ECE 09.321: Final Project –

Application Note - Team Dripps, Merrill, Gayed presents The Wombulator: An Electronic Bass Frequency Synthesizer - Rev. 1

Bill Dripps, James Merrill, Tim Gayed

drippsw0@students.rowan.edu
merrillj1@students.rowan.edu
gayedt6@students.rowan.edu

Rowan University

December 21, 2018

1 Design Overview

This AN describes an electronic frequency synthesizer capable of producing sawtooth waves with frequencies that range from 27.5 to 110 Hz. Series connected RLC notch filters were then used to remove harmonic content from the sawtooth wave.

1.1 Design Features

The synthesizer presents the following features:

- 2-octave frequency range
- User selectable harmonic filtration

1.2 Featured Applications

- Music Production/Performance

1.3 Design Resources

https://github.com/RU09342-F18/intro-to-embedded-final-project-auditory-wombulations/tree/master

1.4 Block Diagram

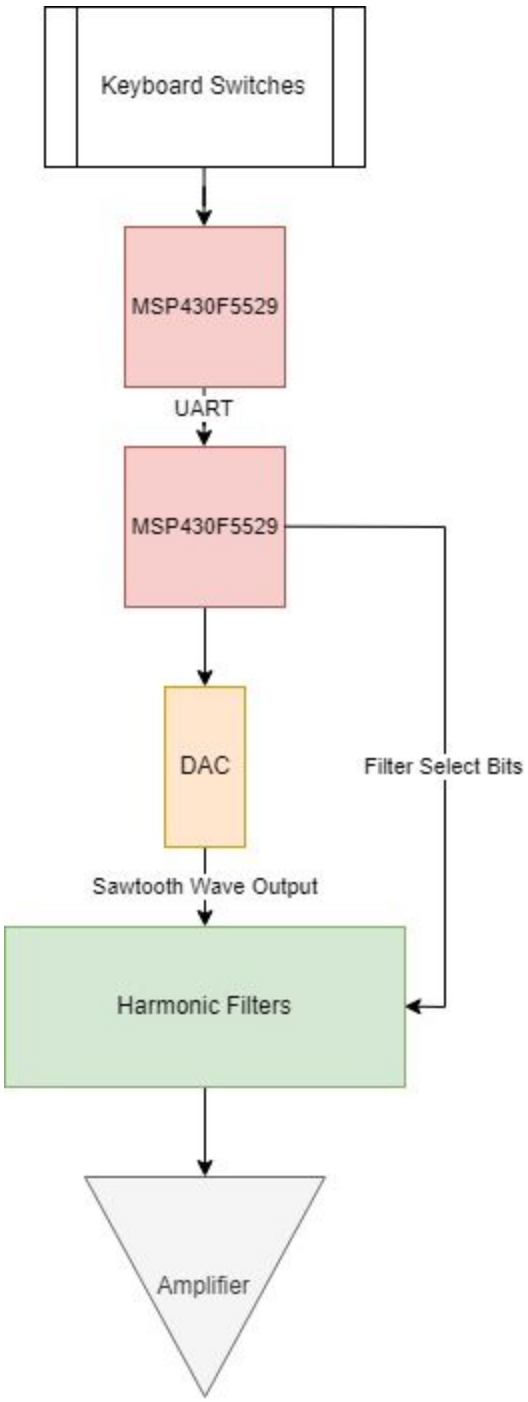


Figure 1. General Block Diagram

1.5 Board Image

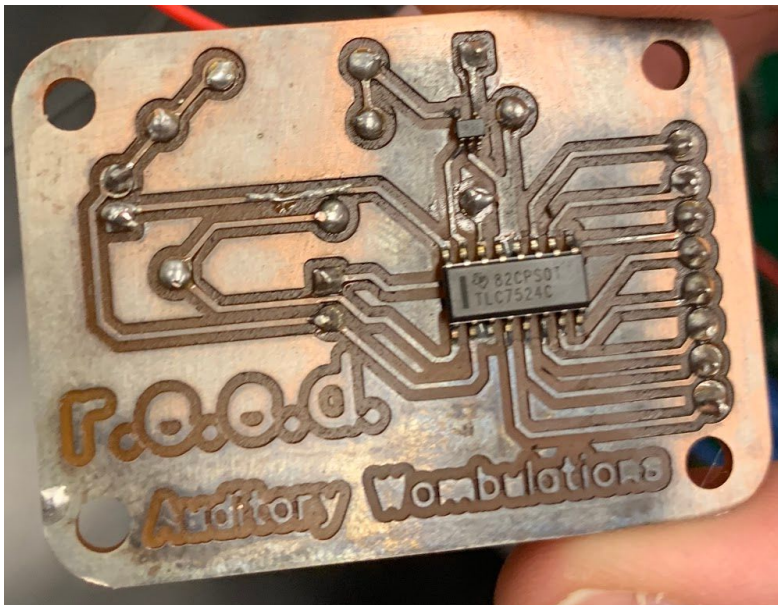


Figure 2. DAC board

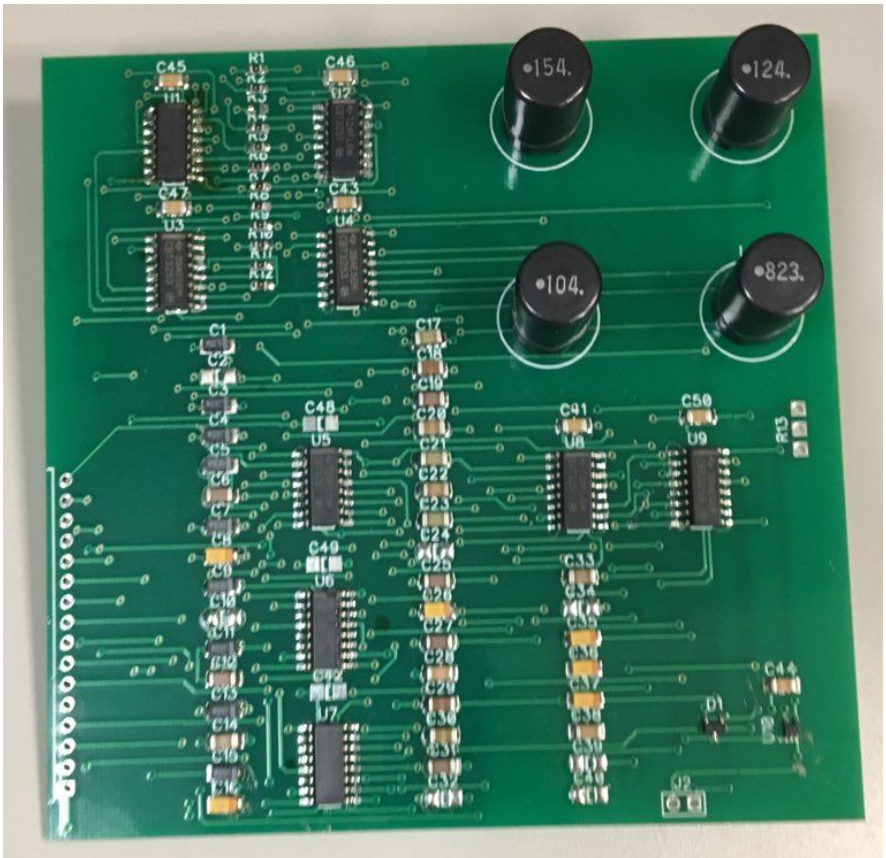


Figure 3. Harmonic Filter Board (1 of 4)

Figure 4. Filter connector board

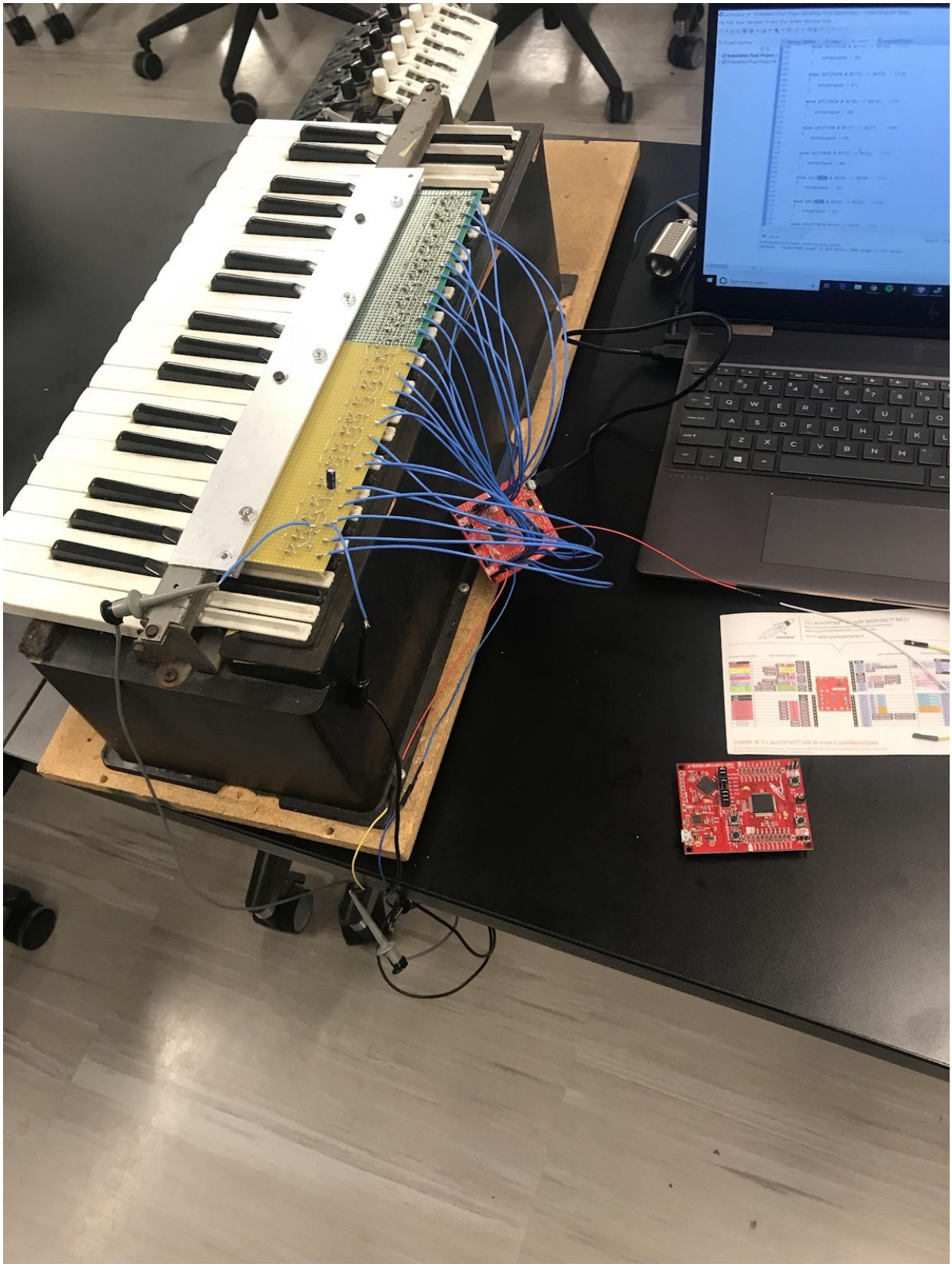


Figure 5. Keyboard and Switch interface

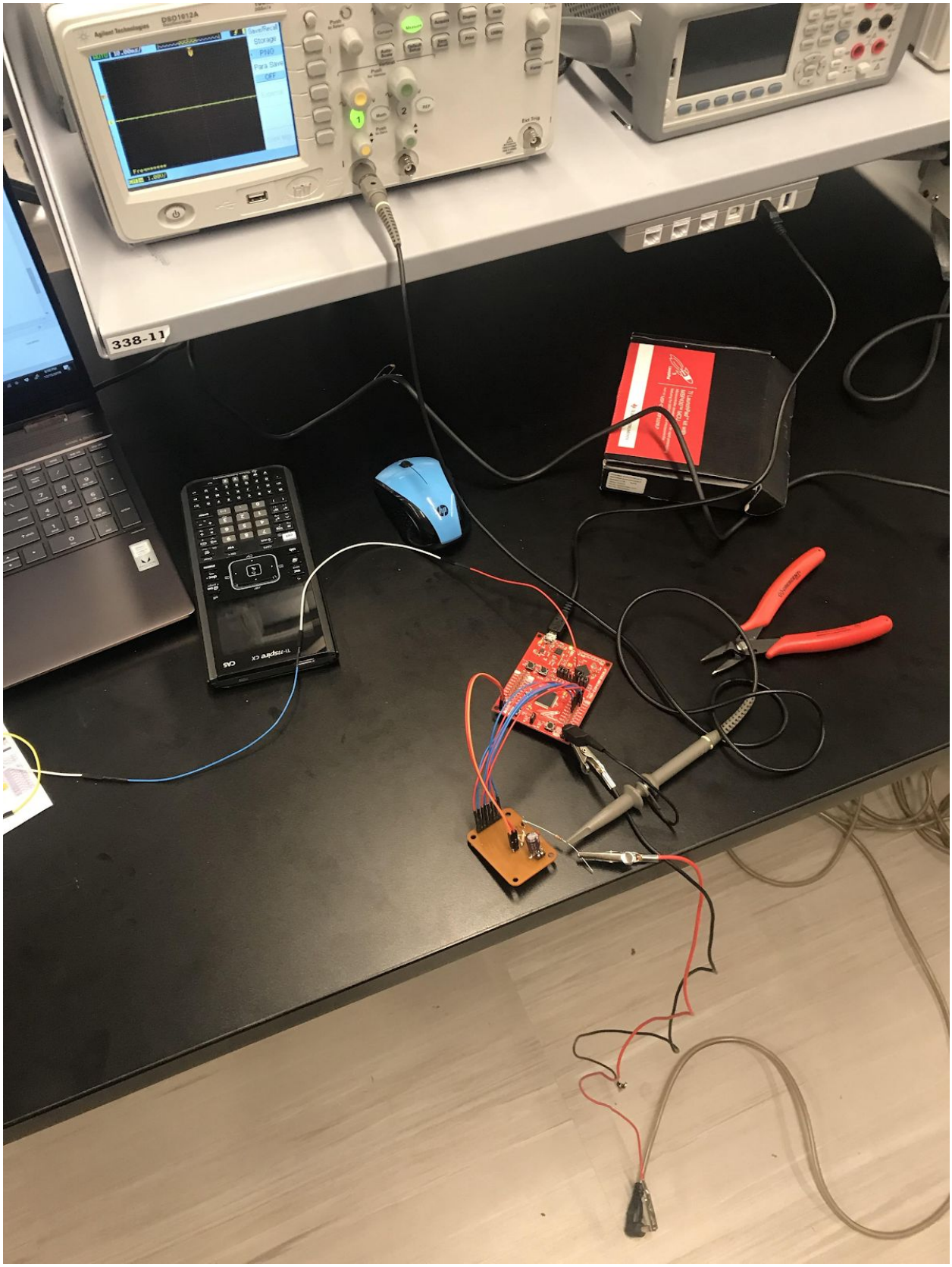


Figure 6. DAC and µC interface

Figure 7. Amplifier interface

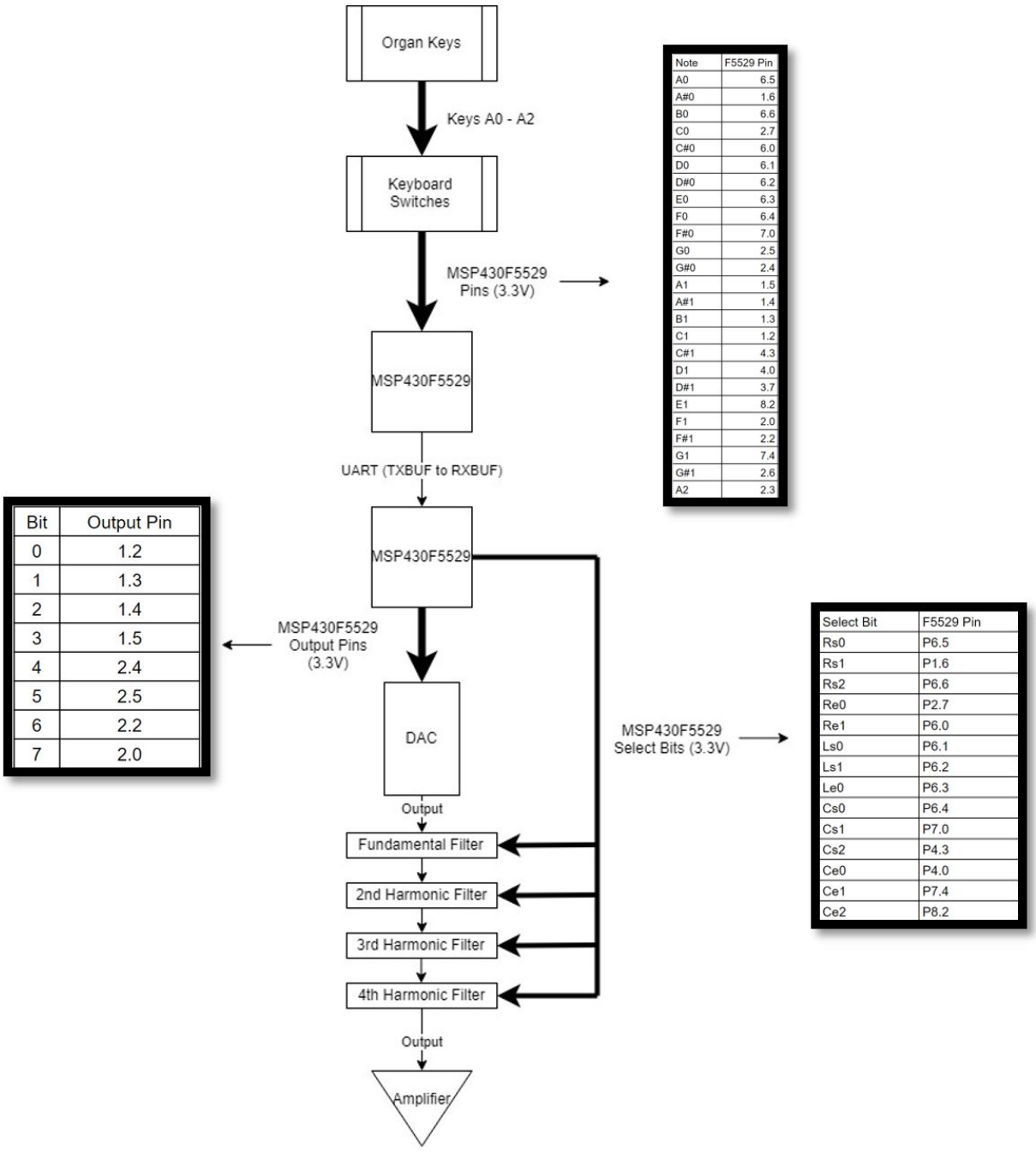
2 Key System Specifications

Parameter	Specification	Details
Baud Rate	9600	*The rate at which each bit, or pulse, is passed through UART Communication*
Note Range (Frequency Range)	A0 - A2 (27.5Hz - 110 Hz)	*Range of notes (range of frequencies) that can be played.*
Filterable Harmonics	Four (Fundamental to the Fourth)	*Number of harmonics that can be filtered to change the waveform output.*
Output Waveform	Sawtooth	*Type of base waveform outputted from the Digital-to-Analog Converter.*

3 System Description

The Wombulator uses 2-MSP430F5529 boards and an 8-bit parallel DAC to produce sawtooth waveforms of different particular frequencies. A series of buffered notch filters are then used to remove harmonically related frequencies .

3.1 Detailed Block Diagram



Note	F5529 Pin
A0	6.5
A#0	1.6
B0	6.6
C0	2.7
C#0	6.0
D0	6.1
D#0	6.2
E0	6.3
F0	6.4
F#0	7.0
G0	2.5
G#0	2.4
A1	1.5
A#1	1.4
B1	1.3
C1	1.2
C#1	4.3
D1	4.0
D#1	3.7
E1	8.2
F1	2.0
F#1	2.2
G1	7.4
G#1	2.6
A2	2.3

Bit	Output Pin
0	1.2
1	1.3
2	1.4
3	1.5
4	2.4
5	2.5
6	2.2
7	2.0

Select Bit	F5529 Pin
Rs0	P6.5
Rs1	P1.6
Rs2	P6.6
Re0	P2.7
Re1	P6.0
Ls0	P6.1
Ls1	P6.2
Le0	P6.3
Cs0	P6.4
Cs1	P7.0
Cs2	P4.3
Ce0	P4.0
Ce1	P7.4
Ce2	P8.2

Figure 8. Detailed Block Diagram

3.2 Highlighted Devices

- MSP430F5529 - *The node of the system*
- Digital-to-Analog Converter (DAC) - *Generates Sawtooth Output*
- Analog Multiplexers - *Select resistance, capacitor and inductor values for each harmonic filter*

3.3 MSP430F5529

The MSP430F5529 is the programmable processor that connects the note played by the keyboard to the synthesizer's output. In the system, two MSP430F5529's were used. The first processor was used to read the note played. This was done by first setting 25 pins on the MSP430F5529 to the input direction. Next, 25 switches were attached to the 25 keys used for the synthesizer. These switches were attached to the 3.3V source of the MSP430F5529 so that whenever a key was pressed, 3.3V could pass through the output of the switch. The output of each switch was set to a pin and depending on the key pressed, a variable containing the note played was generated. This value (which was a number from 1-25) was then transferred via UART to the second board. The second board received this value and used eight pins to drive an 8-bit DAC to produce a sawtooth waveform at the corresponding frequency. The second board also used 14 additional pins that attached to 14 select bits for the Multiplexers on the harmonic filter boards. With each note, a different filter circuit needed to be selected to filter the correct harmonic frequency.

3.4 Digital-to-Analog Converter (DAC)

The Digital-to-Analog Converter (DAC) is the critical component to the sawtooth waveform generation due to its ability to convert a series of digital inputs from the MSP430F5529 into an analog waveform. The DAC used for this project is the TLC7524CDR. This is an 8-bit DAC which means it uses eight digital inputs which act as the eight bits to an 8-bit binary number. The output of the DAC is equal to the binary equivalent of those eight bits divided by 256 (2^8) times the reference voltage (V_{REF}) set to the corresponding pin. For The Wombulator to generate a sawtooth wave, the eight bit input needs to count from 00000000 to 11111111 incrementally at the frequency desired by the note played. The input to the DAC (DB0-DB7) comes from the output pins of the second MSP430F5529.

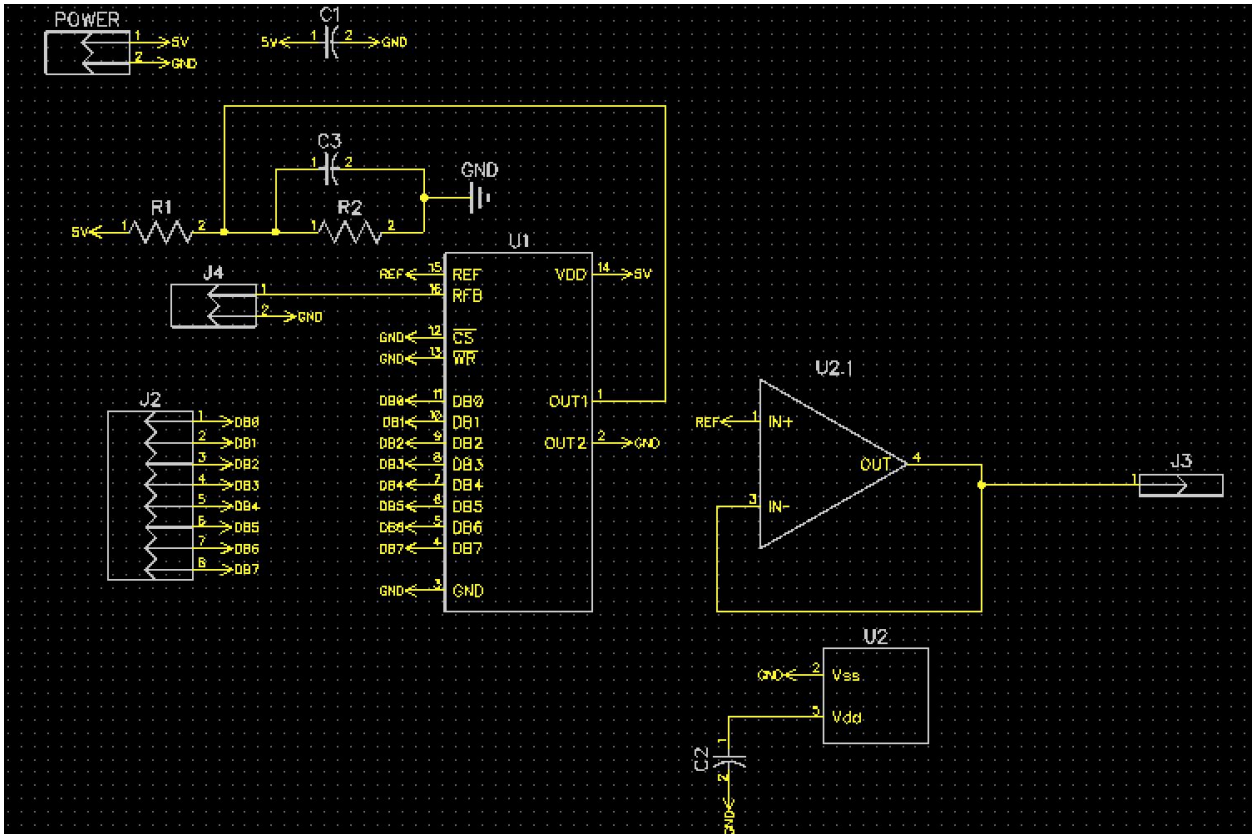


Figure 9. DAC board schematic

The figure above shows the schematic for the DAC circuit. The TLC7524CDR was powered off of 5V. The LMV321IDCKR was used as a buffer. The buffer amplifier was used with the aim of preventing the signal source from being affected by whatever currents that the load may draw. The signal is 'buffered from' load currents. The interposed buffer amplifier prevents the second circuit from loading the first circuit unacceptably and interfering with its desired operation. The capacitor was added to R2 in the schematic below to smooth

Figure 10. The figure above shows the output waveform of the DAC with the input coming from the A2 key (110Hz) from the keyboard. The output for the DAC was then connected to the filter boards where the signal was later amplified and played through the speaker.

4 System Design Theory

4.1a Fourier Analysis of a Sawtooth Wave

The Fourier series that defines a sawtooth wave can be expressed in the following manner:

$$\frac{1}{2} - \frac{1}{\pi}\sum_{n=1}^{\infty}\frac{1}{n}\sin\left(\frac{n\pi x}{L}\right)$$

Figure 11.

Source: http://mathworld.wolfram.com/FourierSeriesSawtoothWave.html

Without analyzing this equation too deeply, one quick takeaway is that the sawtooth wave is comprised of an infinite number of sine waves, each increasing in frequency by the next natural integer multiple of the fundamental frequency, but also decreasing in amplitude at rate correspondent with that of the harmonic series. The effect can be visualised using an FFT (fast fourier transform) tool, one of which can be found within the audio editing software Audacity.

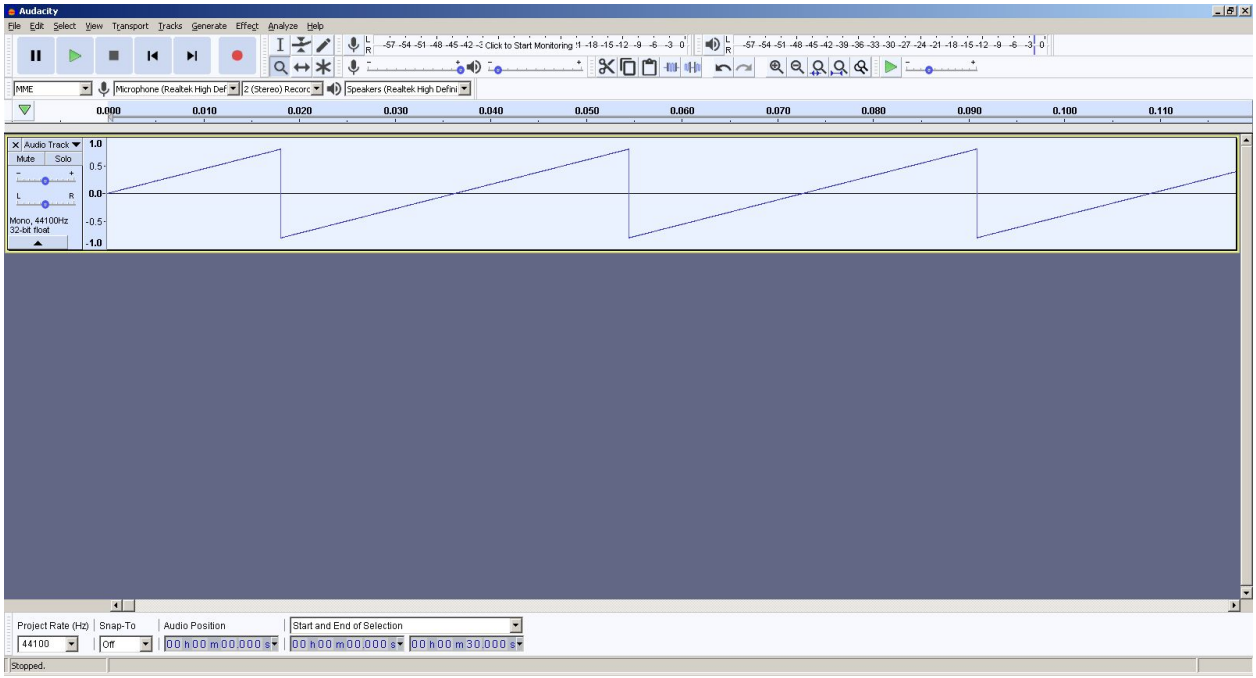


Figure 12. A 27.5 Hz Sawtooth Wave

Using Audacity's tone generator, a 27.5 Hz Sawtooth Wave can be generated as depicted. In the following image, a graph of the Fourier analysis of the wave is shown. This can be accessed under Analyze >> Plot Spectrum in the menu bar.

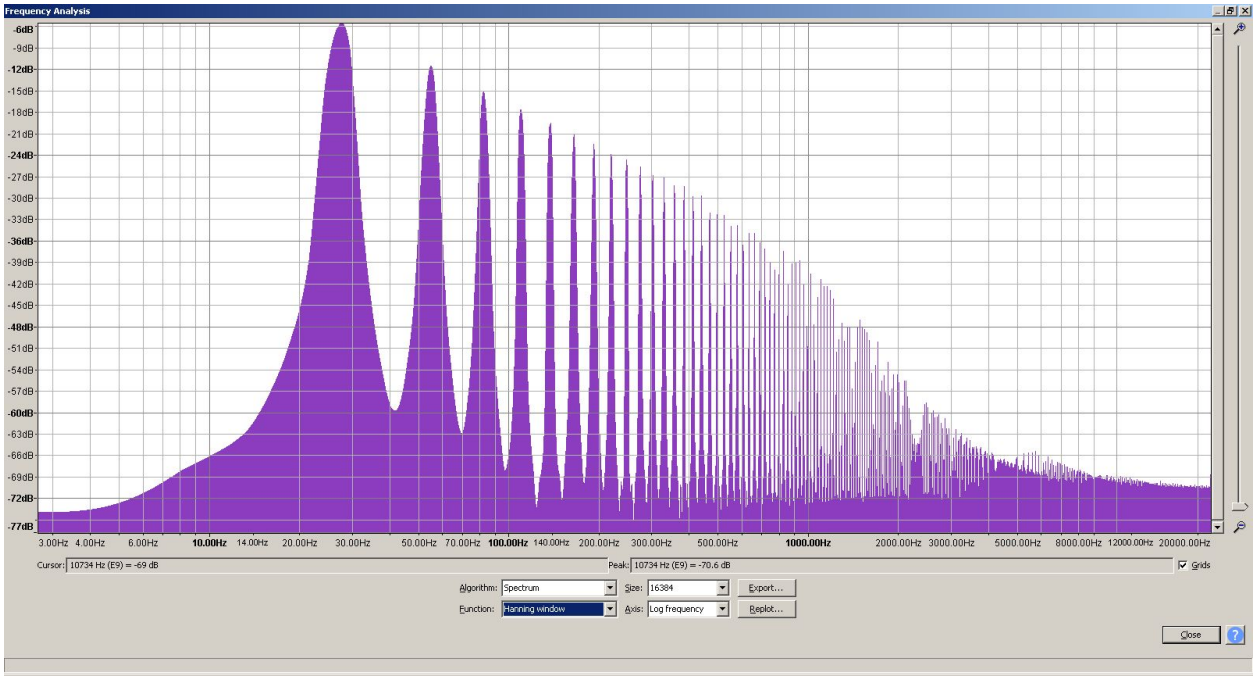


Figure 13. FFT of 27.5 Hz Sawtooth Wave

In the FFT graph, it can be seen that there are peaks at 27.5, 55, 82.5, 110, 137.5, etc. Hz, with each peak decreasing in maximum height than that of the previous one. Of note is that each successive frequency is a multiple of the fundamental frequency. This

corresponds with what might be expected based on the mathematical equation given for the harmonic content of a sawtooth wave. Next is the FFT graph of a triangle wave of the same frequency (27.5 Hz).

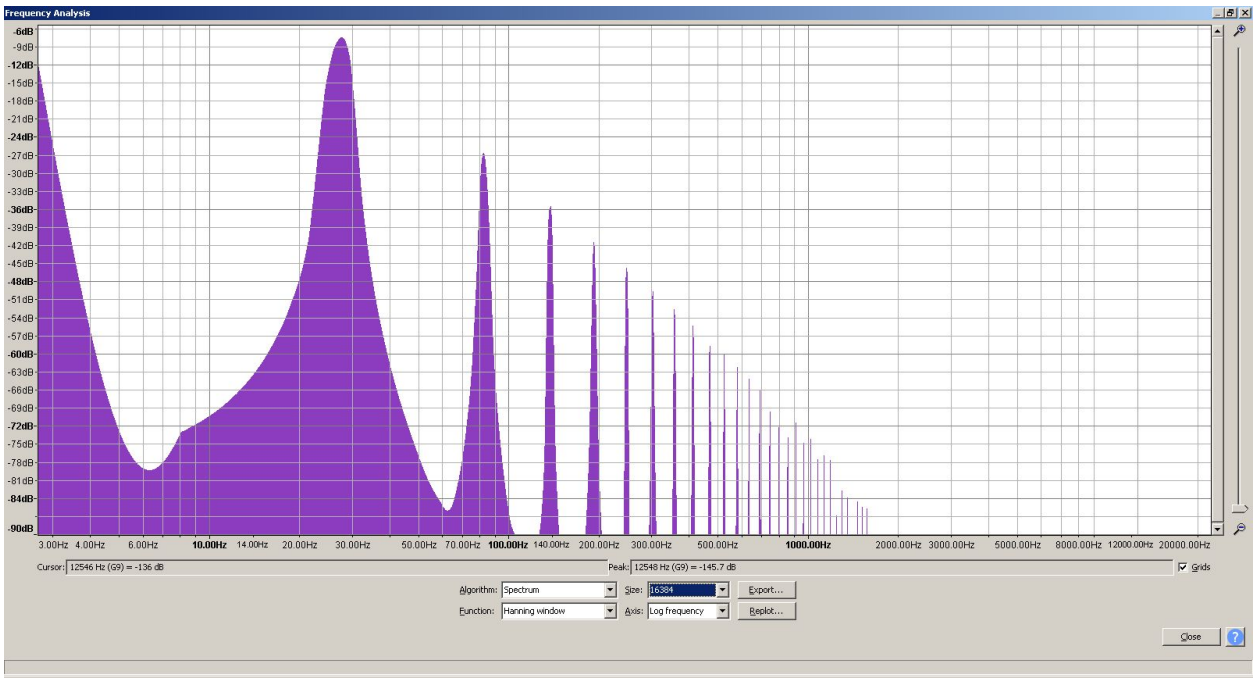


Figure 14. 27.5 Hz Triangle Wave FFT

Looking at the Fourier Transform plot of a Triangle wave, it becomes immediately obvious that it is made of a different composition of waves than the sawtooth. Comparing the two, the triangle wave seems to have peaks at every other instance that the sawtooth has, and the peaks seem to decay at a faster rate than that of the sawtooth. So if one were to remove particular harmonic content from the sawtooth wave, it would appear as though the sawtooth wave could be transformed into a triangle wave. Using the Effect >> Notch Filter tool, with the Q set to 0.5 and the center frequency set to 55, 110 and 165 Hz results in the following waveform:

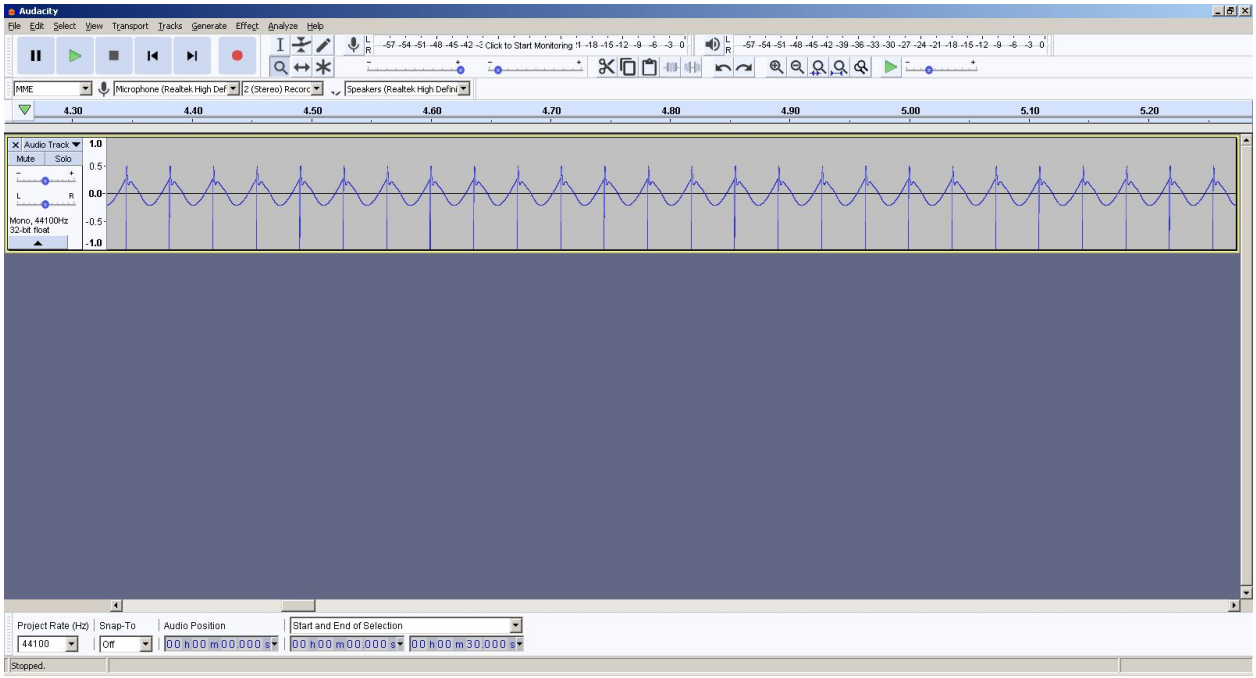


Figure 15. Notching out the 2nd, 4th and 6th Harmonic Frequencies from the Sawtooth

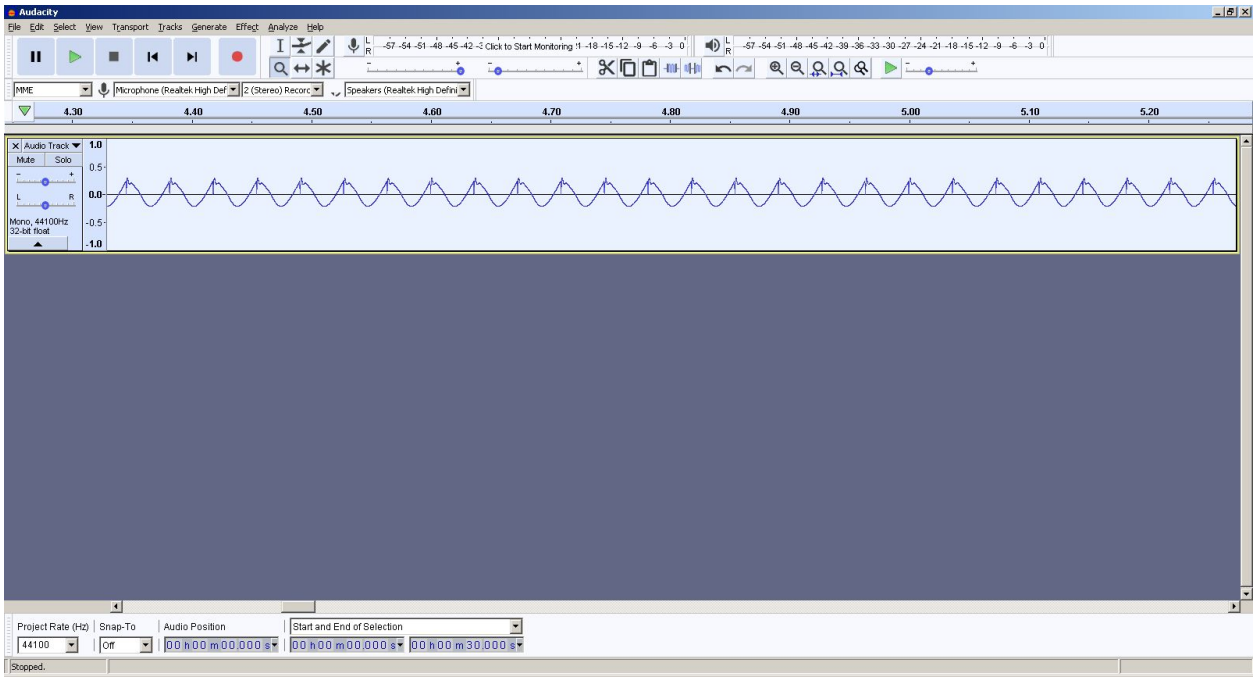


Figure 16. Applying a Low pass filter to remove spikes

After applying the notch filtering to the sawtooth wave and using a low pass filter to remove the negative amplitude spikes, the result is a waveform that looks like something between a triangle and a sine wave. The resultant waveform had an even greater drop off rate in the next couple of present harmonics, and also a greater quantity of higher frequency content.

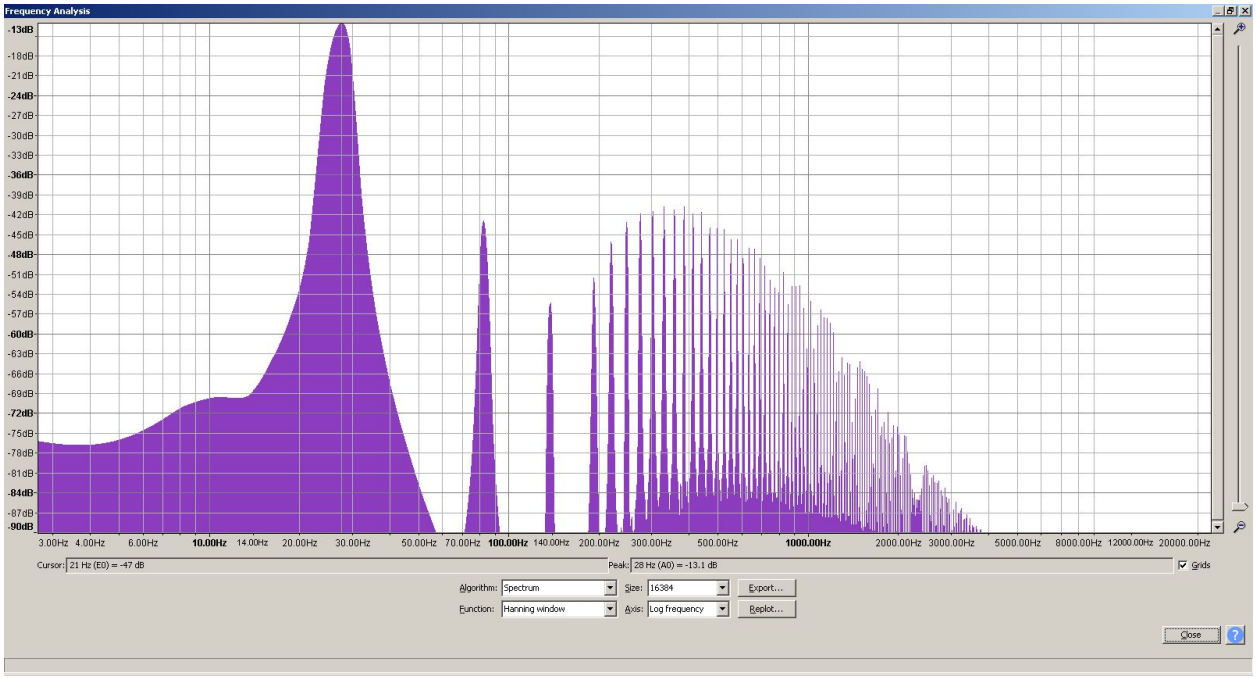


Figure 17. Notch and Low Pass filters applied.

With this information in mind, designing a filter around a series of notch filters, each tuned to the harmonic series of the fundamentals of particular sawtooth waves becomes the next task.

4.1b Notching the Sawtooth Wave

A series RLC notch filter was chosen as the basis for the notch filter design because they require few components and are simple to construct. The center frequency is determinable by the equation $f = 1 \div 2\Pi\sqrt{LC}$, the damping ratio is calculated using the equation $\zeta = (R \div 2) \cdot \sqrt{C \div L}$, and the quality factor, Q, can be found by the relation $Q = 1 \div 2\zeta$. (source: https://en.wikibooks.org/wiki/Circuit_Theory/RLC_Circuits). In this case, a target damping ratio of 1 was chosen so as to minimize ringing in the filter circuit, which results in a Q factor of 0.5. The low Q factor will result in wide filter bandwidths, which leads to some inaccuracy in filtration, as one filter will have some effect on the next filter's center frequency.

As an example, a calculation of the 27.5 Hz filter will be shown here. To start, it should be noted that in order to have a low resonant frequency between an inductor and a capacitor, large values will have to be used. The largest practical and readily available inductor from a major supplier is at present, a 150 mH inductor. With this in mind as a design constraint, the equation $f = 1 \div 2\Pi\sqrt{LC}$ can be rearranged to solve for C in the form $C = (1 \div 2\Pi f)^{2} \cdot (1/L)$, where f is 27.5 Hz and L is 0.15 H. C is found to be in this case 223.5 µF, but in practice 220 µF is an acquirable capacitor. Aiming to find a resistor that will provide a damping ratio of 1, the equation $\zeta = (R \div 2) \cdot \sqrt{C \div L}$ can be rearranged to $R = 2 \div \sqrt{C/L}$, and using the values of 220 µF and 150 mH for C and L, respectively, results in a resistance of 52.2 Ω, or practically 51 Ω. Using the practical values to solve for ζ and thus Q, a value of 0.977 is found for ζ and 0.512 for Q. Calculating each of these RLC combinations by hand would prove to be quite tedious, so a calculator designed to quickly come up with practical values for these

components was used. That calculator can be found at: http://sim.okawa-denshi.jp/en/RLCbekeisan.htm

Using that tool, the following list of RLC combinations was generated:

Fundamental							
R (Ω)	R designator	L (H)	L designator	C (F)	C designator	Capacitors (µF)	f (Hz)
51	0	0.150	0	220u	0	220	27.5
56	1	0.150	0	200u	1	100+100	29.14
	1	0.150	0	180u	2	150+33	30.87
62	2	0.150	0	160u	3	150+10	32.7
	0	0.120	1	180u	2		34.65
	1	0.120	1	160u	3		36.71
75	4	0.150	0	110u	7		38.89
	0	0.100	2	150u	4	150	41.2
68	3	0.120	1	110u	7	100+10	43.65
	1	0.100	2	120u	6	100+22	46.25
	0	0.082	3	130u	5	100+33	49
91	6	0.150	0	62u	8	47+15	51.91
100	7	0.150	0	56u	9	22+33	55
	6	0.120	1	62u	8		58.27
	6	0.120	1	56u	9		61.74
120	9	0.150	0	39u	13	33+5.6	65.41
110	8	0.120	1	43u	12	33+10	69.3
	8	0.120	1	39u	13		73.42
82	5	0.082	3	51u	10	47+3.9	77.78
	5	0.082	3	47u	11	47	82.41
	8	0.100	2	33u	15	33	87.31
	6	0.082	3	36u	14	15+22	92.5
150	10	0.120	1	22u	16	22	98
	10	0.120	1	20u	17	10+10	103.83
160	11	0.120	1	18u	18	10+8.2	110
2nd Harmonic							
R (Ω)	R designator	L (H)	L designator	C (F)	C designator	Capacitors (µF)	f (Hz)
100	0	0.150	0	56u	0	22+33	55

110	1	0.150	0	51u	1	47+3.9	58.28
	1	0.150	0	47u	2	47	61.74
120	2	0.150	0	39u	3	33+5.6	65.4
	0	0.120	1	47u	2		69.3
	1	0.120	1	39u	3		73.42
150	4	0.150	0	27u	7		77.78
	0	0.100	2	36u	4	15+22	82.4
130	3	0.120	1	27u	7	22+4.7	87.3
	1	0.100	2	30u	6	22+8.2	92.5
	0	0.082	3	33u	5	33	98
180	6	0.150	0	16u	8	15+1	103.82
200	7	0.150	0	15u	9	15	110
	6	0.120	1	16u	8		116.54
	6	0.120	1	15u	9		123.48
240	9	0.150	0	10u	13	8.2+1.8	130.82
220	8	0.120	1	11u	12	8.2+2.7	138.6
	8	0.120	1	10u	13		146.84
160	5	0.082	3	13u	10	8.2+4.7	155.56
	5	0.082	3	12u	11	8.2+3.9	164.82
	8	0.100	2	8.2u	15	8.2	174.62
	6	0.082	3	9.1u	14	8.2+0.82	185
300	10	0.120	1	5.6u	16	5.6	196
	10	0.120	1	4.7u	17	4.7	207.66
330	11	0.120	1	4.3u	18	2.2+2.2	220

3rd Harmonic

R (Ω)	R designator	L (H)	L designator	C (F)	C designator	Capacitors (µF)	f (Hz)
150	0	0.150	0	24u	0	12+12	82.5
160	1	0.150	0	22u	1	22	87.42
	1	0.150	0	20u	2	10+10	92.61
180	2	0.150	0	18u	3	10+8.2	98.1
	0	0.120	1	20u	2		103.95
	1	0.120	1	18u	3		110.13
220	4	0.150	0	12u	7	12	116.67
	0	0.100	2	16u	4	8.2+8.2	123.6
200	3	0.120	1	12u	7		130.95

	1	0.100	2	13u	6	6.8+6.8	138.75
	0	0.082	3	15u	5	15	147
270	6	0.150	0	6.8u	8	6.8	155.73
300	7	0.150	0	6.2u	9	3.9+2.2	165
	6	0.120	1	6.8u	8		174.81
	6	0.120	1	6.2u	9		185.22
360	9	0.150	0	4.3u	13	2.2+2.2	196.23
330	8	0.120	1	4.7u	12	4.7	207.9
	8	0.120	1	4.3u	13		220.26
240	5	0.082	3	5.6u	10	5.6	233.34
	5	0.082	3	5.1u	11	3.3+1.8	247.23
	8	0.100	2	3.6u	15	1.8+1.8	261.93
	6	0.082	3	3.9u	14	3.9	277.5
430	10	0.120	1	2.4u	16	1.2+1.2	294
	10	0.120	1	2.2u	17	2.2	311.49
510	11	0.120	1	2u	18	1+1	330

4th Harmonic

R (Ω)	R designator	L (H)	L designator	C (F)	C designator	Capacitors (µF)	f (Hz)
100	0	0.082	0	27u	0	12+15	110
110	1	0.082	0	22u	1	22	116.56
	1	0.082	0	20u	2	10+10	123.48
120	2	0.082	0	18u	3	10+8.2	130.8
	0	0.068	1	20u	2		138.6
	1	0.068	1	18u	3		146.84
150	4	0.082	0	13u	7	6.8+6.8	155.56
	0	0.047	2	20u	4	10+10	164.8
130	3	0.068	1	13u	7		174.6
	1	0.047	2	16u	6	10+5.6	185
	0	0.039	3	16u	5	8.2+8.2	196
180	6	0.082	0	7.5u	8	3.9+3.3	207.64
200	7	0.082	0	6.2u	9	3.3+2.7	220
	6	0.068	1	7.5u	8		233.08
	6	0.068	1	6.2u	9		246.96
240	9	0.082	0	4.7u	13	4.7	261.64
220	8	0.068	1	4.7u	12	4.7	277.2

	8	0.068	1	4.7u	13		293.68
160	5	0.039	3	6.8u	10	6.8	311.12
	5	0.039	3	6.2u	11	3.3+2.7	329.64
	8	0.047	2	4.3u	15	2.2+2.2	349.24
	6	0.039	3	4.7u	14	4.7	370
300	10	0.068	1	2.4u	16	1.2+1.2	392
	10	0.068	1	2.2u	17	2.2	415.32
330	11	0.068	1	2u	18	1+1	440
5th Harmonic							
R (Ω)	R designator	L (H)	L designator	C (F)	C designator	Capacitors (µF)	f (Hz)
150	0	0.082	0	16u	0	8.2+8.2	137.5
160	1	0.082	0	15u	1	15	145.7
	1	0.082	0	13u	2	8.2+4.7	154.35
180	2	0.082	0	12u	3	12	163.5
	0	0.068	1	13u	2		173.25
	1	0.068	1	12u	3		183.55
220	4	0.082	0	8.2u	7	8.2	194.45
	0	0.047	2	13u	4	8.2+4.7	206
200	3	0.068	1	8.2u	7		218.25
	1	0.047	2	10u	6	10	231.25
	0	0.039	3	11u	5	5.6+5.6	245
270	6	0.082	0	4.7u	8	4.7	259.55
300	7	0.082	0	3.9u	9	3.9	275
	6	0.068	1	4.7u	8		291.35
	6	0.068	1	3.9u	9		308.7
360	9	0.082	0	2.7u	13	2.7	327.05
330	8	0.068	1	3u	12	1.5+1.5	346.5
	8	0.068	1	2.7u	13		367.1
240	5	0.039	3	4.3u	10	2.2+2.2	388.9
	5	0.039	3	3.9u	11	3.9	412.05
	8	0.047	2	2.8u	15	1.8+1	436.55
	6	0.039	3	3u	14	1.5+1.5	462.5
430	10	0.068	1	1.5u	16	1.5	490
	10	0.068	1	1.4u	17	.68+.68	519.15
510	11	0.068	1	1.2u	18	1.2	550

6th Harmonic							
R (Ω)	R designator	L (H)	L designator	C (F)	C designator	Capacitors (µF)	f (Hz)
180	0	0.082	0	11u	0	5.6+5.6	165
200	1	0.082	0	10u	1	10	174.84
	1	0.082	0	9.1u	2	8.2+0.82	185.22
200	2	0.082	0	8.2u	3	8.2	196.2
	0	0.068	1	9.1u	2		207.9
	1	0.068	1	8.2u	3		220.26
240	4	0.082	0	5.6u	7	5.6	233.34
	0	0.047	2	9.1u	4	8.2+.82	247.2
220	3	0.068	1	5.6u	7		261.9
	1	0.047	2	6.8u	6	6.8	277.5
	0	0.039	3	7.5u	5	3.9+3.3	294
330	6	0.082	0	3.3u	8	3.3	311.46
360	7	0.082	0	2.8u	9	1.3+1.5	330
	6	0.068	1	3.3u	8		349.62
	6	0.068	1	2.8u	9		370.44
390	9	0.082	0	2u	13	1+1	392.46
360	8	0.068	1	2.2u	12	2.2	415.8
	8	0.068	1	2u	13		440.52
220	5	0.039	3	3u	10	1.5+1.5	466.68
	5	0.039	3	2.7u	11	2.7	494.46
	8	0.047	2	2u	15	1+1	523.86
	6	0.039	3	2.1u	14	1.5+.56	555
510	10	0.068	1	1u	16	1	588
	10	0.068	1	.95u	17	.47+.47	622.98
560	11	0.068	1	0.85u	18	.47+.39	660
7th Harmonic							
R (Ω)	R designator	L (H)	L designator	C (F)	C designator	Capacitors (µF)	f (Hz)
91	0	0.039	0	18u	0	10+8.2	192.5
100	1	0.039	0	16u	1	8.2+8.2	203.98
	1	0.039	0	14u	2	8.2+5.6	216.09

110	2	0.039	0	12u	3	12	228.9
	0	0.033	1	14u	2		242.55
	1	0.033	1	12u	3		256.97
130	4	0.039	0	8.8u	7	3.9+4.7	272.23
	0	0.022	2	14u	4	8.2+5.6	288.4
120	3	0.033	1	8.8u	7		305.55
	1	0.022	2	11u	6	5.6+5.6	323.75
	0	0.018	3	12u	5	12	343
180	6	0.039	0	5.1u	8	2.7+2.2	363.37
200	7	0.039	0	4.3u	9	2.2+2.2	385
	6	0.033	1	5.1u	8		407.89
	6	0.033	1	4.3u	9		432.18
220	9	0.039	0	3u	13	1.5+1.5	457.87
160	8	0.033	1	3.3u	12	3.3	485.1
	8	0.033	1	3u	13		513.94
120	5	0.018	3	4.7u	10	4.7	544.46
	5	0.018	3	4.3u	11	2.2+2.2	576.87
	8	0.022	2	3u	15	1.5+1.5	611.17
	6	0.018	3	3.3u	14	3.3	647.5
300	10	0.033	1	1.6u	16	.82+.82	686
	10	0.033	1	1.5u	17	1.5	726.81
330	11	0.033	1	1.3u	18	1.3	770

8th Harmonic

R (Ω)	R designator	L (H)	L designator	C (F)	C designator	Capacitors (µF)	f (Hz)
100	0	0.039	0	13u	0	4.7+8.2	220
110	1	0.039	0	12u	1	12	233.12
	1	0.039	0	11u	2	5.6+5.6	246.96
120	2	0.039	0	10u	3	10	261.6
	0	0.033	1	11u	2		277.2
	1	0.033	1	10u	3		293.68
150	4	0.039	0	6.8u	7	6.8	311.12
	0	0.022	2	11u	4	5.6+5.6	329.6
130	3	0.033	1	6.8u	7		349.2
	1	0.022	2	8.2u	6	8.2	370

	0	0.018	3	9.1u	5	8.2+.82	392
180	6	0.039	0	3.7u	8	2.2+1.5	415.28
200	7	0.039	0	3.3u	9	3.3	440
	6	0.033	1	3.7u	8		466.16
	6	0.033	1	3.3u	9		493.92
240	9	0.039	0	2.4u	13	1.2+1.2	523.28
220	8	0.033	1	2.5u	12	1.3+1.2	554.4
	8	0.033	1	2.4u	13		587.36
160	5	0.018	3	3.6u	10	1.8+1.8	622.24
	5	0.018	3	3.3u	11	3.3	659.28
	8	0.022	2	2.4u	15	1.2+1.2	698.48
	6	0.018	3	2.6u	14	1.3+1.3	740
300	10	0.033	1	1.2u	16	1.2	784
	10	0.033	1	1.1u	17	.56+.56	830.64
330	11	0.033	1	1u	18	1	880

Figure 18. RLC Filter Specs for Each Harmonic

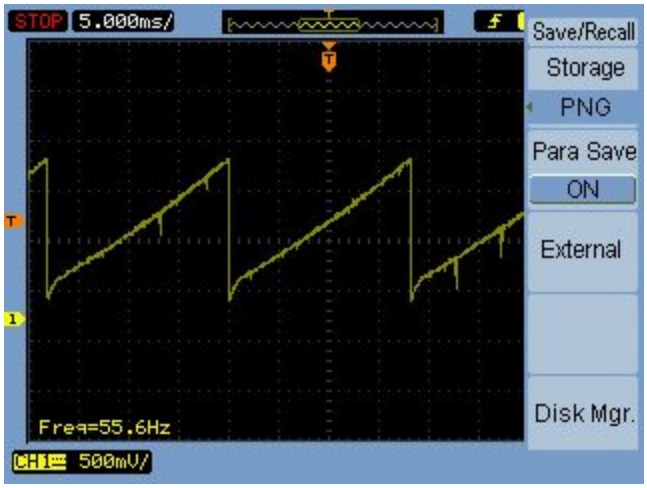


Figure 19. Fundamental Frequency filtered out. Note that the negative voltage spike that would normally be encountered here is removed using a diode, to protect the op amp from receiving a signal outside of its specified range

Figure 20. 2nd Harmonic notch

Figure 21. 3rd Harmonic notch

Figure 22. 4th Harmonic notch

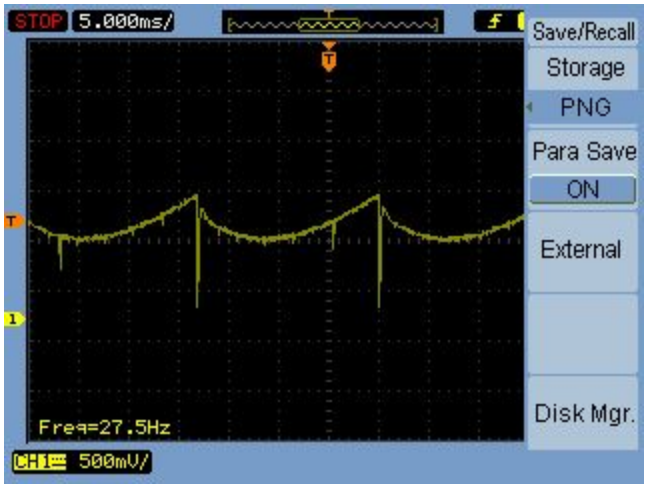


Figure 23. Fundamental, 2nd, 3rd, 4th Harmonics notched

While 8 harmonic filter boards were initially calculated for, time constraints and damage during assembly restricted the filter to contain only the first 4 harmonic filters.

4.2 Producing the Correct Frequency Sawtooth Wave

To produce a sawtooth wave, an external 8-bit DAC was attached to eight pins on the MSP430F5529 where each pin functioned as a binary bit in an 8-bit counter. These pins would turn on and off depending on how many times an internal clock would reset. The clock reset interrupt would trigger a counter that would increment up to 255 (2^8 - 1). Depending on which bit of the counter the pin corresponded to, the counter would be modularly divided by a number (2, 4, 8, 16, 32, 64, 128 or 256) and if the number was greater than half of the divisor, the corresponding pin would turn on. This created the 8-bit counter.

Since the clock was configured to be in Up mode, the speed of the 8-bit counter and therefore the waveform could be changed by changing the CCR0 value. However, the fastest normal clock setting on the MSP430F5529, SMCLK, runs at 1 MHz. Although this is much faster then what is needed, when 1 MHz is divided by 256 (counter) and then the CCR0 is changed to try and tune each note frequency, the increments between CCR0 value would cause a change in frequency higher than acceptable and notes were multiple cents off the correct pitch. T0 counter this, SMCLK was sped up to its fastest possible frequency, 25 MHz, using the Unified Clock System (UCS) Registers. With a higher clock speed, changes in the CCR0 would correspond to smaller frequency changes allowing for each note to be properly tuned.

4.3 Filter Selection

The filter boards' control pins are connected in parallel, so that the same control codes are used on each board to select the correct resistor, inductor and capacitor combination for the particular note being played. The selection of the individual components is accomplished through the use of analog multiplexer ICs. A list of the control codes is provided below.

Header Values	Note ->	A 0	A# 0	B 0	C 1	C# 1	D 1	D# 1	E 1	F 1	F# 1	G 1	G# 1	A 1	A# 1	B 1	C 2	C# 2	D 2	D# 2	E 2	F 2	F# 2	G 2	G# 2	A 2
0	5vdc																									
1	Gnd																									
2	Rs0	0	1	1	0	0	1	0	0	1	1	0	0	1	0	0	1	0	0	1	1	0	0	0	0	1
3	Rs1	0	0	0	1	0	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1	1
4	Rs2	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	0	0	0
5	Re0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	1	1
6	Re1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1	0	0	0
7	Ls0	0	0	0	0	1	1	0	0	1	0	1	0	0	1	1	0	1	1	1	1	0	1	1	1	1
8	Ls1	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0
9	Le0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	Cs0	0	1	0	1	0	1	1	0	1	0	1	0	1	0	1	1	0	1	0	1	1	0	0	1	0
11	Cs1	0	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1
12	Cs2	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0	1	1	0	0	0
13	Ce0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	Ce1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
15	Ce2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0

Figure 24. Filter Board control codes

4.4 Hardware Switch Debouncing

The snap-action switches used to interface the keyboard with the µC had a relatively long bounce time of between 500 and 600 µseconds. To combat this, an RC charge up network was used to smooth the bounce out over the span of 1 ms. Below are a schematic, as well as oscilloscope depictions of the bounce and debounce outputs.

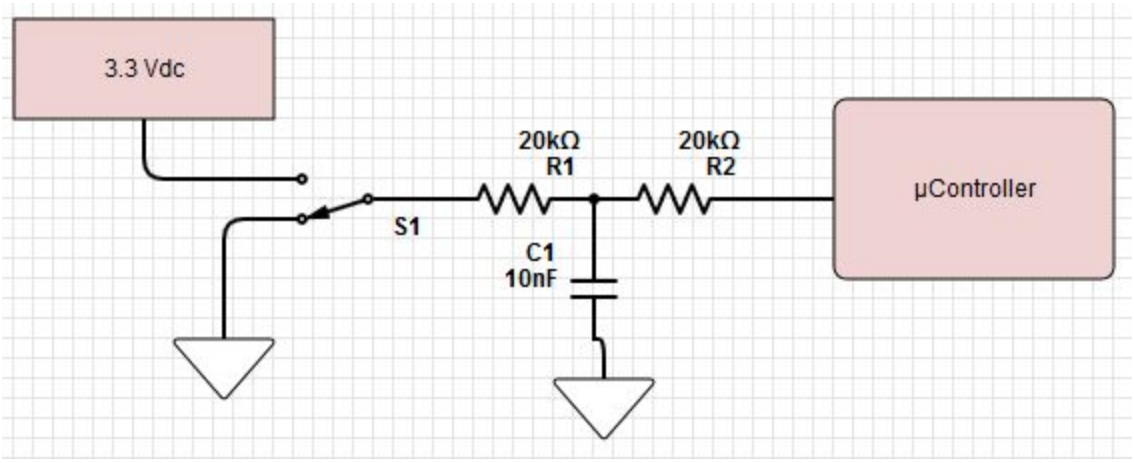


Figure 25. Key Input Debouncing Schematic

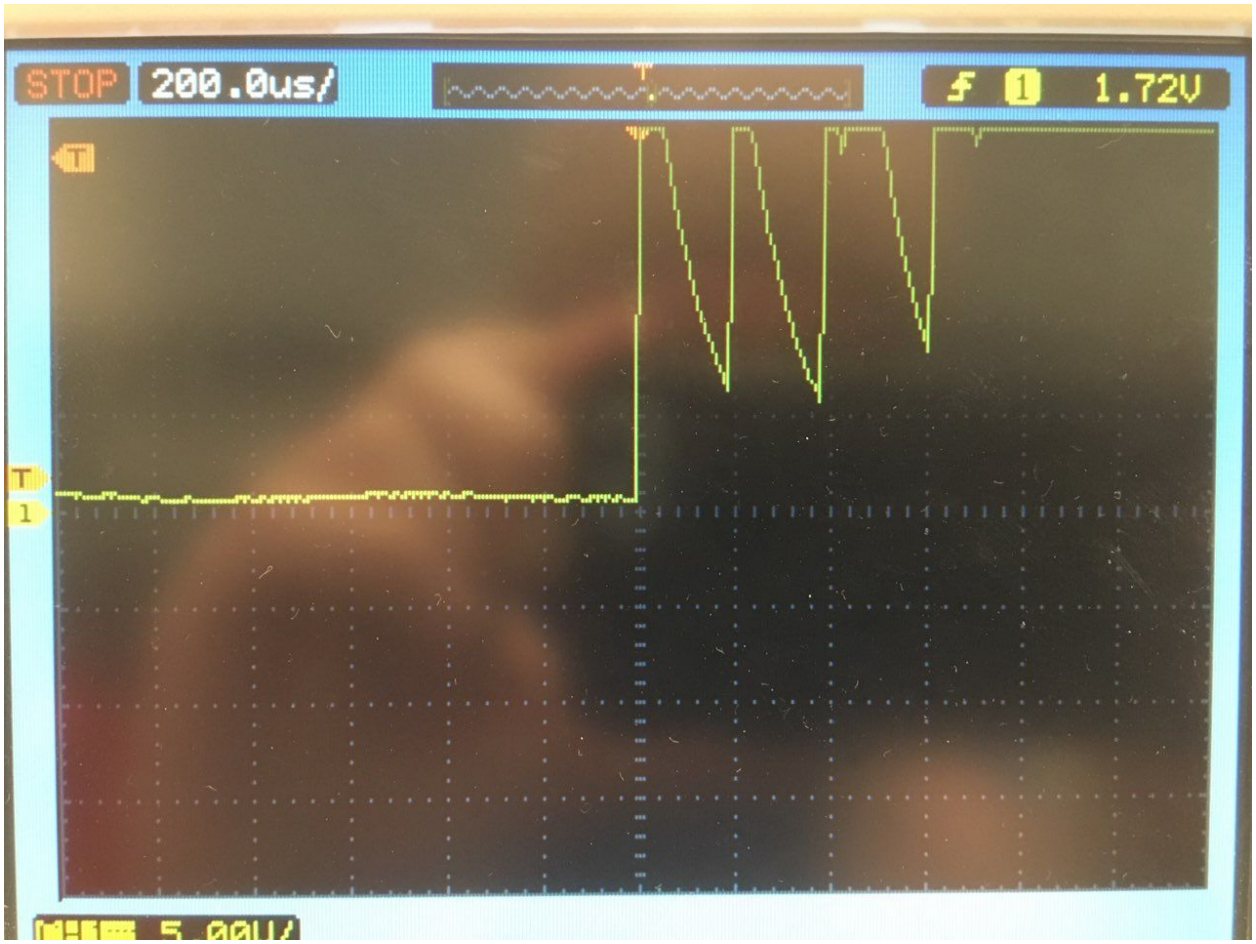


Figure 26. Keyboard Input without Hardware Debouncing

Figure 27. Keyboard Input Graph with Hardware Debouncing

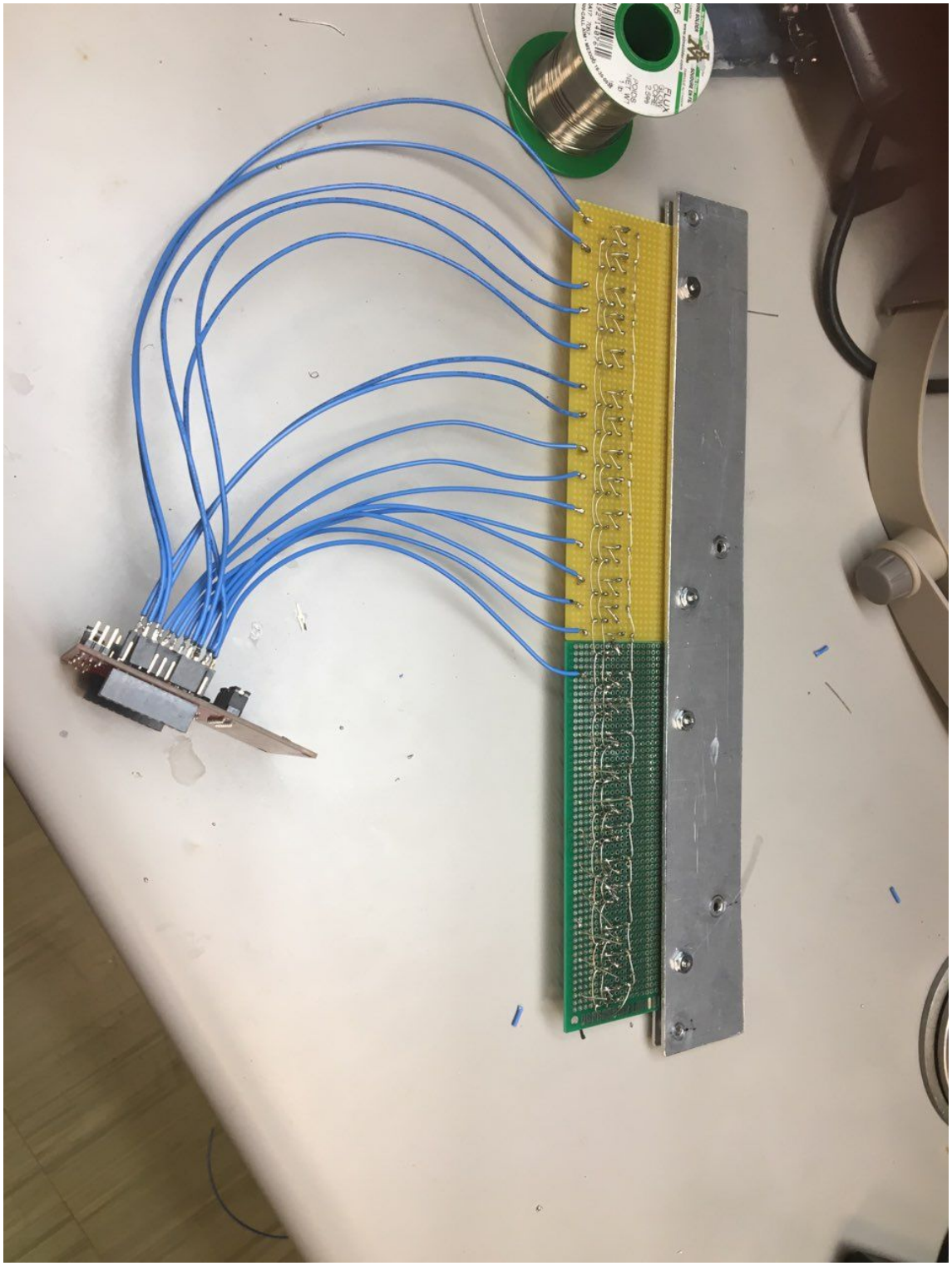


Figure 28. Partly Assembled Switch Panel with Hardware Debouncing

5 Getting Started/How to use the device

To use The Wombulator, make sure each MSP430F5529 is connected to power. This power can either be applied via USB from a computer or outlet, or from a power supply to the corresponding power pins. Next, make sure that the each pin corresponds to the correct key, DAC input and MUX select bit. These should not be an issue as the synthesizers uses female headers which are soldered on to connecting wires. Next, make sure that the output from the DAC is connected to the first filter board. This connection is not soldered so this would more likely be an issue. Next, make sure that the tip and sleeve of the ¼ in. cable (which is connected to the amplifier) are connected to the output of the fourth filter board and ground respectively. Lastly, double check that each ground from all the components are connected.

To make sure the system is working before the amplifier is turned out, set up an oscilloscope probe on the end of the fourth filter board, make sure all of the filter potentiometers are turned off and watch the output as you play the synthesizer. If sawtooth waves appear at the proper frequencies, the synthesizer is working. When you turn on the amplifier, a low-end sawtooth sound should output.

6 Test Setup

As described in the previous section, to test if The Wombulator is working properly, first, make sure all of the filter potentiometers are all the way off. There shouldn't be any filtration happening for the testing stage. Next, attach an oscilloscope probe to the output of the fourth filter board and play a note on the keyboard. A slightly noisy saw wave should appear. Now, twist each of the potentiometers on each board. If the saw wave changes its shape, the system is working properly.

7 Conclusion and Future Work

7.1 Conclusion

The design and construction of the Wombulator proved to be a very intensive and challenging feat. For the project to match up with signal processing theory, precise calculations of resistor, capacitor, and inductor values were required for each of the 25 keys in the design. After careful calculation, the design then had to be made with careful considerations for the physical constraints of the keyboard as well as for the electronic hardware used in the design. After creating the design, all the components of the Wombulator had to be fabricated, put together, developed, and finally tested.

Because of the complexity of the project, design took over a month to complete. There were several bugs along the way that required in depth troubleshooting to correct. This project served to provide several lessons on the importance of component layout during PCB design as well as preparing careful consideration for the mounting of the hardware together after fabrication.

The Wombulator left out some additional features that will be explained in the next section. Some of these features include audio signal effects that were left out due to time constraints during the assembly of the project. Some features of operation were simplified due to physical limitations of hardware and their design. For example, the Wombulator can only play one note at a time. Anytime more than one key is pressed, the higher key is selected. This is due to a design choice which does not allow for the combination of multiple notes.

Overall, despite some of the limitations of the design, all of the features promised in the proposal were delivered upon. All of the keyboard inputs work as intended and the external DAC works as expected as well. Even with fewer than anticipated filter boards on the final design due to complications during fabrication, the filtering stage works as intended.

7.2 Future Work

Some features that this synthesizer lacked are the inclusion of any effects, velocity or pressure sensitive keys, multiple tone generators, polyphony (the ability to play more than one note at a time), or countless others. Some effects could have included tremolo (amplitude modulation), vibrato (frequency modulation), ADSR generator (signal envelope shaping), ring modulator (frequency heterodyning), phaser/flanger (phase shift modulation), or any others. Some of these could be easily implemented with some minor design changes, for example, the tremolo could be implemented by adding a low frequency oscillator (< 20 Hz) to the DAC circuit, which would be used to modulate the reference voltage by a user selectable amplitude at a user selectable rate.

Improvements to the design could include having a Q control for the filter boards, which would be a matter of adding a potentiometer to the resistor selection circuit. Some initial and inconclusive testing has shown that a Q factor of up to 5 could be achieved using the components already in the circuit. The limiting factor is the current limit through the inductors used, which was 35 mA. However, using this type of filter topology and aiming to achieve a Q factor of that magnitude would introduce ringing into the final signal, due to being an underdamped RLC circuit, which might produce a musically pleasing tones, but caution would have to be taken to ensure that the circuit does not enter an unstable state while it uncontrollably oscillates.

The wiring and case left much to be desired by the time this project was submitted. Even though the device was functional and met its basic design requirements, exposed wiring and open cases are visually ugly, and invitations for electrical failures due to shorts from external debris or other materials interfering with the circuitry. A well designed case would benefit this project.

The amplifier used in this project had an input buffer in the form of an RC high pass filter, which had an impedance of 3.9 MΩ and a corner frequency of 40 Hz. (source: https://support.fender.com/hc/en-us/article_attachments/115007682006/Workingman_s_10__Complete_.pdf) Since The Wombulator produces frequencies down to 27.5 Hz, this meant that some of the audio information was being attenuated in passing through the amplifier's buffer. To combat this, an amplifier capable of accepting and producing lower frequencies on the input and output, respectively, should instead be used.

8 Design Files

8.1 Additional Design Files

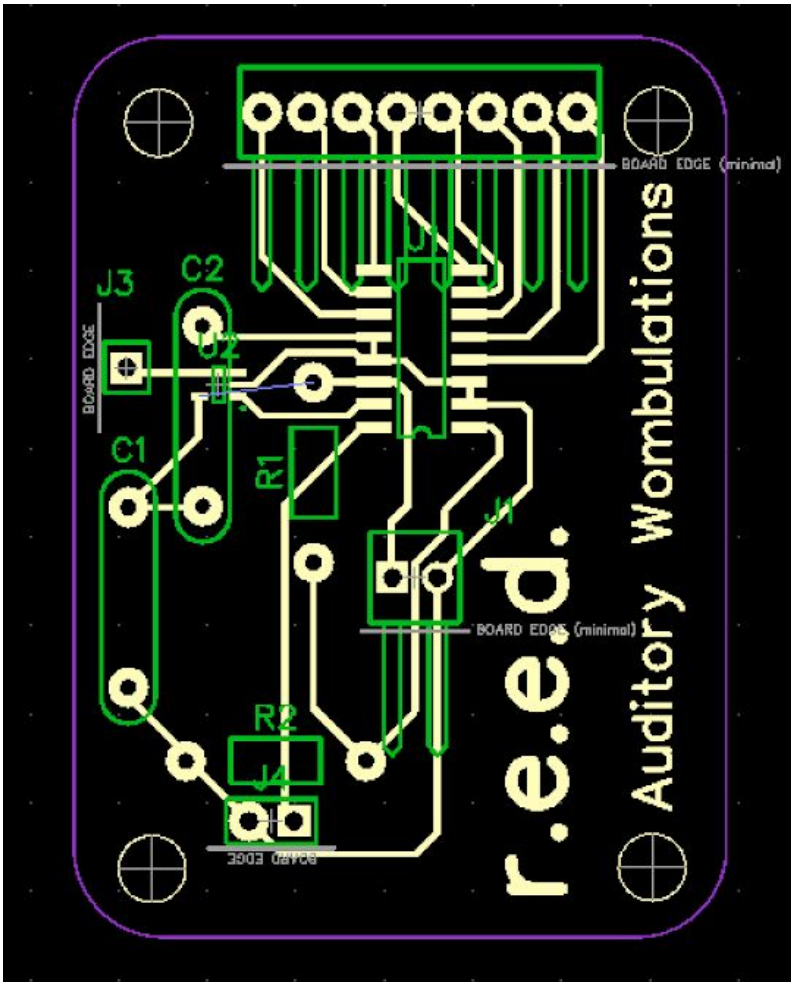


Figure 29. DAC PCB Design Layout

8.2 Additional Schematics

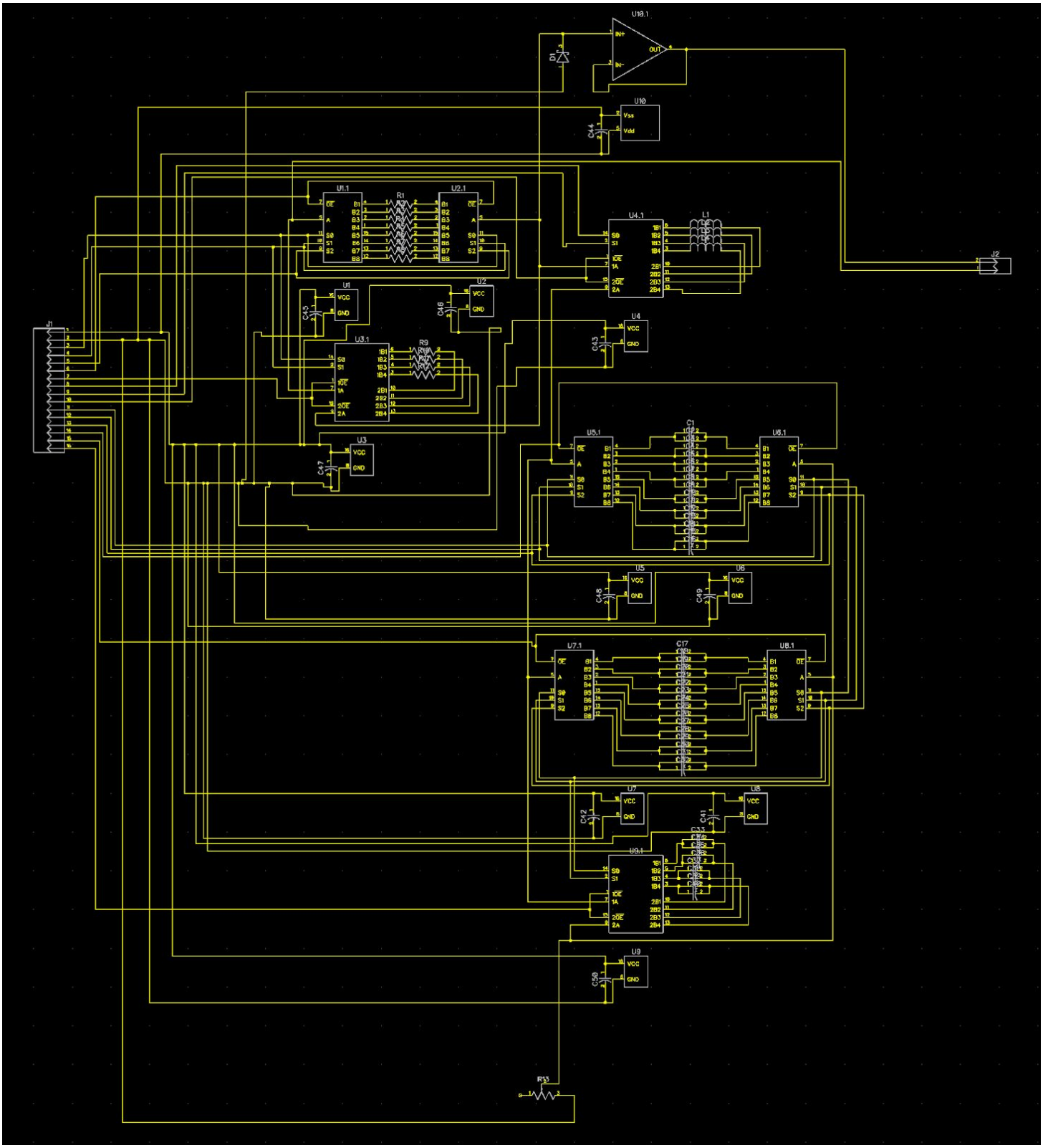


Figure 30. Filter Board Schematic

8.3 Bill of Materials

Quantity	Part Number	Manufacturer	Description
1	ESR03EZPJ510	ROHM Semiconductor	Thick Film Resistors - SMD 0603 51ohm 5% Anti Surge AEC-Q200
5	ESR03EZPJ101	ROHM Semiconductor	Thick Film Resistors - SMD 0603 100ohm 5% Anti Surge AEC-Q200
6	ESR03EZPJ121	ROHM Semiconductor	Thick Film Resistors - SMD 0603 120ohm 5% Anti Surge AEC-Q200
8	ESR03EZPJ221	ROHM Semiconductor	Thick Film Resistors - SMD 0603 220ohm 5% Anti Surge AEC-Q200
7	ESR03EZPJ331	ROHM Semiconductor	Thick Film Resistors - SMD 0603 330ohm 5% Anti Surge AEC-Q200
1	ESR03EZPJ750	ROHM Semiconductor	Thick Film Resistors - SMD 0603 75ohm 5% Anti Surge AEC-Q200
2	ERJ-PA3J271V	Panasonic	Thick Film Resistors - SMD 0603 270ohm 5% Anti-Surge AEC-Q200
8	SDR03EZPJ201	ROHM Semiconductor	Thick Film Resistors - SMD 0603 200ohm 5% Anti Surge AEC-Q200
3	SDR03EZPJ511	ROHM Semiconductor	Thick Film Resistors - SMD 0603 510ohm 5% Anti Surge AEC-Q200
1	SDR03EZPJ820	ROHM Semiconductor	Thick Film Resistors - SMD 0603 82ohm 5% Anti Surge AEC-Q200
1	SDR03EZPJ561	ROHM Semiconductor	Thick Film Resistors - SMD 0603 560ohm 5% Anti Surge AEC-Q200
6	SDR03EZPJ241	ROHM Semiconductor	Thick Film Resistors - SMD 0603 240ohm 5% Anti Surge AEC-Q200
6	ESR03EZPJ301	ROHM Semiconductor	Thick Film Resistors - SMD 0603 300ohm 5% Anti Surge AEC-Q200
6	ESR03EZPJ151	ROHM Semiconductor	Thick Film Resistors - SMD 0603 150ohm 5% Anti Surge AEC-Q200
5	ESR03EZPJ361	ROHM Semiconductor	Thick Film Resistors - SMD 0603 360ohm 5% Anti Surge AEC-Q200
1	ESR03EZPJ391	ROHM Semiconductor	Thick Film Resistors - SMD 0603 390ohm 5% Anti Surge AEC-Q200
1	ESR03EZPJ680	ROHM Semiconductor	Thick Film Resistors - SMD 0603 68ohm 5% Anti Surge AEC-Q200

4	ESR03EZPJ131	ROHM Semiconductor	Thick Film Resistors - SMD 0603 130ohm 5% Anti Surge AEC-Q200
5	ESR03EZPJ111	ROHM Semiconductor	Thick Film Resistors - SMD 0603 110ohm 5% Anti Surge AEC-Q200
7	ESR03EZPJ161	ROHM Semiconductor	Thick Film Resistors - SMD 0603 160ohm 5% Anti Surge AEC-Q200
7	ESR03EZPJ181	ROHM Semiconductor	Thick Film Resistors - SMD 0603 180ohm 5% Anti Surge AEC-Q200
2	ESR03EZPJ910	ROHM Semiconductor	Thick Film Resistors - SMD 0603 91ohm 5% Anti Surge AEC-Q200
2	ESR03EZPJ431	ROHM Semiconductor	Thick Film Resistors - SMD 0603 430ohm 5% Anti Surge AEC-Q200
1	ESR03EZPJ560	ROHM Semiconductor	Thick Film Resistors - SMD 0603 56ohm 5% Anti Surge AEC-Q200
1	TMCMA0G227MTRF	Vishay	Tantalum Capacitors - Solid SMD 220uF 4V 20% A case
3	TMCMA0G157MTRF	Vishay	Tantalum Capacitors - Solid SMD 150uF 4V 20% A case
5	TMCMA0J107MTRF	Vishay	Tantalum Capacitors - Solid SMD 100uF 6.3V 20% A case
5	JMK316AC6476ML-T	Taiyo Yuden	Multilayer Ceramic Capacitors MLCC - SMD/SMT 47uF 6.3V X6S +/-20% 1206 Gen Purp
9	C3216X5R0J336M130AC	TDK	Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 6.3V 33uF X5R 20% T: 1.3mm
10	JMK316AB7226KL-TR	Taiyo Yuden	Multilayer Ceramic Capacitors MLCC - SMD/SMT 22uF 6.3V X7R 10% 1206
8	C3216X7S0J156M160AB	TDK	Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 6.3V 15uF X7S 20% T: 1.6mm
20	TAJA106M006SNJ	AVX	Tantalum Capacitors - Solid SMD 6.3volts 10uF 20%
30	C1206C825K4PACTU	KEMET	Multilayer Ceramic Capacitors MLCC - SMD/SMT 16V 8.2uF X5R 1206 10%
22	C1206C565K8RACTU	KEMET	Multilayer Ceramic Capacitors MLCC - SMD/SMT 10V 5.6uF X7R 1206 10%
14	885012208002	Wurth Electronics	Multilayer Ceramic Capacitors MLCC - SMD/SMT WCAP-CSGP 4.7uF 1206 10% 6.3V MLCC
10	C1206C395K3PACTU	KEMET	Multilayer Ceramic Capacitors MLCC - SMD/SMT 25V 3.9uF X5R 1206 10%

12	293D335X9016A8T	Vishay	Tantalum Capacitors - Solid SMD 3.3uF 16volts 10% A case Molded
6	C1206C275K3PACTU	KEMET	Multilayer Ceramic Capacitors MLCC - SMD/SMT 25V 2.7uF X5R 1206 10%
18	C1206C225K3REC7210	KEMET	Multilayer Ceramic Capacitors MLCC - SMD/SMT 25V 2.2uF X7R 1206 10%
9	12063C185K4Z2A	AVX	Multilayer Ceramic Capacitors MLCC - SMD/SMT 25V 1.8uF X7R 1206 10% Tol
15	CGA5L2X7R1E155M160AA	TDK	Multilayer Ceramic Capacitors MLCC - SMD/SMT CGA 1206 25V 1.5uF X7R 20% AEC-Q200
8	1206ZC135KAT2A	AVX	Multilayer Ceramic Capacitors MLCC - SMD/SMT 10V 1.3uF X7R 1206 10% Tol
13	GCG31MR71E125MA01L	Murata	Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 1.2uF 25volts X7R 20%
2	VJ1206V684ZXAPW1BC	Vishay	Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 0.68uF 50volts Y5V +80-20%
5	VJ1206Y564KXJTW1BC	Vishay	Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 0.56uF 16volts X7R 10%
1	VJ1206Y394MXJTW1BC	Vishay	Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 0.39uF 16volts X7R 20%
7	VJ1206Y824KXJTW1BC	Vishay	Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 0.82uF 16volts X7R 10%
3	VJ1206Y474KXATW1BC	Vishay	Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 .47uF 50volts X7R 10%
9	C3216X6S1A685M085AB	TDK	Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 10V 6.8uF X6S 20% T: 0.85mm
15	LMV321IDCKR	Texas Instruments	Operational Amplifiers - Op Amps Low V R/R
48	SN74CBT3251D	Texas Instruments	Multiplexer Switch ICs 1-of-8 FET
24	SN74CBT3253D	Texas Instruments	Multiplexer Switch ICs Dual 1-of-4 FET
8	SBAT54CLT1G	ON Semiconductor	Schottky Diodes & Rectifiers SS DUAL COMMON CATHODE
80	885012208009	Wurth Electronics	Multilayer Ceramic Capacitors MLCC - SMD/SMT WCAP-CSGP 0.1uF 1206 10% 10V MLCC
8	M20-9751646	Harwin	Headers & Wire Housings 16 SIL HORIZONTAL PIN HEADER TIN

3	LHL10TB154J	Taiyo Yuden	Fixed Inductors 150mH 300Ohms +/-5%Tol 28mA
3	LHL10TB124J	Taiyo Yuden	Fixed Inductors 120mH 260Ohms +/-5%Tol 30mA
3	LHL10NB104J	Taiyo Yuden	Fixed Inductors 100mH 240Ohms +/-5%Tol 31mA
6	LHL10TB823J	Taiyo Yuden	Fixed Inductors 82mH 210Ohms +/-5%Tol 33mA
5	LHL10NB393J	Taiyo Yuden	Fixed Inductors 39000uH 84Ohms +/-5%Tol 53mA
2	LHL08TB333J	Taiyo Yuden	Fixed Inductors 33mH 100Ohms +/-5%Tol 40mA
2	LHL08TB223J	Taiyo Yuden	Fixed Inductors 22mH 82Ohms +/-5%Tol 44mA
2	LHL10TB183J	Taiyo Yuden	Fixed Inductors 18mH 38Ohms +/-5%Tol 81mA
1	TLC7524CDR	Texas Instruments	Digital to Analog Converters - DAC 8-Bit 0.1 us MDAC Parallel Input
30	ZMA00A080S06PC	C&K Switches	Basic / Snap Action Switches Switch Snapact Spdt .1A Roll T/H
10	885012208036	Wurth Electronics	Multilayer Ceramic Capacitors MLCC - SMD/SMT WCAP-CSGP 1uF 1206 10% 16V MLCC
5	RLB9012-473KL	Bourns	Fixed Inductors 47000uH 10% .08A
5	RLB0913-683K	Bourns	Fixed Inductors 68mH 10% .045A Non-SHLD 8.5x12.5mm
4	PTD902-2015K-A104	Bourns	Potentiometers 100K AUDIO
50	CFR-25JT-52-20K	Yageo	Carbon Film Resistors 1/4W 20K Ohm 1%
25	K103K10X7RH5TH5	Vishay	Multilayer Ceramic Capacitors MLCC - Leaded .01uF 100volts 10% 5.0mm X7R

9 Code

9.1 Reading the Note Played

```
// The following code was programmed into the MSP430F5529 that read the note being played
// depending on the key pressed and then sent the value of the note played over to the other
// MSP430F5529 over UART.

#include <msp430.h>

unsigned volatile int noteplayed = 0;

void main(void)
 {
    WDTCTL = WDTPW + WDTHOLD;           //stop watchdog timer

    volatile unsigned int i;

    // Note Pin Input
    P6DIR &= ~BIT5; //A0
    P1DIR &= ~BIT6; //A#0
    P6DIR &= ~BIT6; //B0
    P2DIR &= ~BIT7; //C1
    P6DIR &= ~BIT0; //C#1
    P6DIR &= ~BIT1; //D1
    P6DIR &= ~BIT2; //D#1
    P6DIR &= ~BIT3; //E1
    P6DIR &= ~BIT4; //F1
    P7DIR &= ~BIT0; //F#1
    P2DIR &= ~BIT5; //G1
    P2DIR &= ~BIT4; //G#1
    P1DIR &= ~BIT5; //A1
    P1DIR &= ~BIT4; //A#1
    P1DIR &= ~BIT3; //B1
    P1DIR &= ~BIT2; //C2
    P4DIR &= ~BIT3; //C#2
    P4DIR &= ~BIT0; //D2
    P3DIR &= ~BIT7; //D#2
    P8DIR &= ~BIT2; //E2
    P2DIR &= ~BIT0; //F2
    P2DIR &= ~BIT2; //F#2
    P7DIR &= ~BIT4; //G2
    P2DIR &= ~BIT6; //G#2
    P2DIR &= ~BIT3; //A2

    //Configure UART
    P3SEL |= BIT3 + BIT4;                       // Board to Board Select Bits
    P4SEL |= BIT4 + BIT5;                       // Computer to Board Select Bits
    UCA1CTL1 = UCSWRST;                         // initialize USCI
    UCA1CTL1 |= UCSSEL_1;                       // set to use ACLK (UCSSEL_21
    UCA1BR0 = 3;                                // Baud Rate is 9600, should be 104 when running
normal SMCLK
    UCA1BR1 = 0;                                // set to 0
    UCA1MCTL |= UCBRS_3 + UCBRF_0;              // Modulation UCBRSx=1, UCBRFx=0
    UCA1CTL1 &= ~UCSWRST;                       // initialize USCI
    UCA1IE |= UCRXIE;                           // enable USCI_A1 RX interrupt

    while(1)
    {
        //__bis_SR_register(GIE);                 // Test Setup

        // Configure pins to read if key is pressed and pass the key value of UART
         if((P2IN & BIT3) == BIT3)   //A2
         {
```

```
    noteplayed = 25;
}

else if((P2IN & BIT6) == BIT6)   //G#2
{
    noteplayed = 24;
}
else if((P7IN & BIT4) == BIT4)   //G2
{
    noteplayed = 23;
}
else if((P2IN & BIT2) == BIT2)   //F#2
{
    noteplayed = 22;
}
else if((P2IN & BIT0) == BIT0)   //F2
{
    noteplayed = 21;
}
else if((P8IN & BIT2) == BIT2)   //E2
{
    noteplayed = 20;
}
else if((P3IN & BIT7) == BIT7)   //D#2
{
    noteplayed = 19;
}
else if((P4IN & BIT0) == BIT0)   //D2
{
    noteplayed = 18;
}
else if((P4IN & BIT3) == BIT3)   //C#2
{
    noteplayed = 17;
}
else if((P1IN & BIT2) == BIT2)   //C2
{
    noteplayed = 16;
}
else if((P1IN & BIT3) == BIT3)   //B1
{
    noteplayed = 15;
}
else if((P1IN & BIT4) == BIT4)   //A#1
{
    noteplayed = 14;
}
else if((P1IN & BIT5) == BIT5)   //A1
{
    noteplayed = 13;
}
else if((P2IN & BIT4) == BIT4)   //G#1
{
    noteplayed = 12;
}
else if((P2IN & BIT5) == BIT5)   //G1
{
    noteplayed = 11;
}
else if((P7IN & BIT0) == BIT0)   //F#1
{
    noteplayed = 10;
}
else if((P6IN & BIT4) == BIT4)   //F1
{
    noteplayed = 9;
```

```
        }
        else if((P6IN & BIT3) == BIT3)   //E1
        {
            noteplayed = 8;
        }
        else if((P6IN & BIT2) == BIT2)   //D#1
        {
            noteplayed = 7;
        }
        else if((P6IN & BIT1) == BIT1)   //D1
        {
            noteplayed = 6;
        }
        else if((P6IN & BIT0) == BIT0)   //C#1
        {
            noteplayed = 5;
        }
        else if((P2IN & BIT7) == BIT7)   //C1
        {
            noteplayed = 4;
        }
        else if((P6IN & BIT6) == BIT6)   //B0
        {
            noteplayed = 3;
        }
        else if((P1IN & BIT6) == BIT6)   //A#0
        {
            noteplayed = 2;
        }
        else if((P6IN & BIT5) == BIT5)   //A0
        {
            noteplayed = 1;
        }

       else //no key is being pressed
       {
           noteplayed = 0;
       }

       UCA1TXBUF = noteplayed; //put the noteplayed on the TXBUF to be sent to the
output/filter board
    }
}
```

9.2 Outputting Proper Sawtooth Wave and Select Bits

```
// The following code was programmed into the second MSP430F5529 which received the note
played
// over UART and then outputted the digital inputs to the DAC at the corresponding note
// frequency.  As well as outputting the proper frequency sawtooth wave, the proper select
bits
// for the four harmonic filter boards were also programmed for each note.  The filter for
each
// note needed to be different because the harmonics of each note change as the note does.

#include <msp430.h>
#include <math.h>

volatile unsigned int counter = 0;
volatile unsigned int noteplayed = 1;

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;               // stop watchdog timer
```

```
    // Speed up SMCLK for 8-Bit Sawtooth
    UCSCTL3 |= SELREF__REFOCLK;             // Set DCO FLL reference = REFO
    UCSCTL4 |=  XT1OFF | XT2OFF;            // Set ACLK = REFO
    __bis_SR_register(SCG0);                // Disable the FLL control loop
    UCSCTL0 = 0x0000;                       // Set lowest possible DCOx, MODx
    UCSCTL1 = DCORSEL_7;                    // Select DCO range 24MHz operation
    UCSCTL2 = 762u;                         // Set DCO Multiplier for 16MHz
                                                // (N + 1) * FLLRef = Fdco
                                                // (488 + 1) * 32768 = 16.023552MHz
                                                // Set FLL Div = fDCOCLK/2
    UCSCTL4 = SELA__REFOCLK | SELS__DCOCLK | SELM__DCOCLK;

    //Configure UART
    P3SEL |= BIT3 + BIT4;                   // Board to Board Select Bits
    P4SEL |= BIT4 + BIT5;                   // Computer to Board Select Bits
    UCA1CTL1 = UCSWRST;                     // initialize USCI
    UCA1CTL1 |= UCSSEL_1;                   // set to use ACLK (UCSSEL_21
    UCA1BR0 = 3;                            // Baud Rate is 9600, should be 104 when running
normal SMCLK
    UCA1BR1 = 0;                            // set to 0
    UCA1MCTL |= UCBRS_3 + UCBRF_0;          // Modulation UCBRSx=1, UCBRFx=0
    UCA1CTL1 &= ~UCSWRST;                   // initialize USCI
    UCA1IE |= UCRXIE;                       // enable USCI_A1 RX interrupt

    __bis_SR_register(SCG0);                // Enable the FLL control loop

    // Initialize Outputs   // DAC Input:
    P1DIR |= BIT2;          // 1
    P1DIR |= BIT3;          // 2
    P1DIR |= BIT4;          // 3
    P1DIR |= BIT5;          // 4
    P2DIR |= BIT4;          // 5
    P2DIR |= BIT5;          // 6
    P2DIR |= BIT2;          // 7
    P2DIR |= BIT0;          // 8

    // Initialize Select Bits
    P6DIR |= BIT5;          //RS0
    P1DIR |= BIT6;          //RS1
    P6DIR |= BIT6;          //RS2
    P2DIR |= BIT7;          //RE0
    P6DIR |= BIT0;          //RE1
    P6DIR |= BIT1;          //LS0
    P6DIR |= BIT2;          //LS1
    P6DIR |= BIT3;          //LE0
    P6DIR |= BIT4;          //CS0
    P7DIR |= BIT0;          //CS1
    P4DIR |= BIT3;          //CS2
    P4DIR |= BIT0;          //CE0
    P7DIR |= BIT4;          //CE1
    P8DIR |= BIT2;          //CE2

    TB0CTL = TBSSEL_2 + MC_1 + ID_0;    // Timer is in continuous mode, uses SMCLK and divide
by 4
    TB0CCR0 = 0;                        // Initialize CCR0
    TB0CCTL0 = CCIE;                    // enable interrupt for the clock

    while(1)
    {
        __bis_SR_register(GIE);                 // Test Setup
        switch(noteplayed)
        {
            case 1: TB0CCR0 = 1797;// A0
                    P6OUT &= ~BIT5;         //RS0
                    P1OUT &= ~BIT6;         //RS1
```

```
        P6OUT &= ~BIT6;         //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT &= ~BIT1;         //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT &= ~BIT0;         //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 2: TB0CCR0 = 1696;// A#0
        P6OUT |= BIT5;          //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT &= ~BIT1;         //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT &= ~BIT0;         //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 3: TB0CCR0 = 1601;// B0
        P6OUT |= BIT5;          //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT &= ~BIT1;         //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT &= ~BIT0;         //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 4: TB0CCR0 = 1511;// C1
        P6OUT &= ~BIT5;         //RS0
        P1OUT |= BIT6;          //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT &= ~BIT1;         //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT &= ~BIT0;         //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 5: TB0CCR0 = 1426;// C#1
        P6OUT &= ~BIT5;         //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT &= ~BIT7;         //RE0
```

```
        P6OUT |= BIT0;          //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT &= ~BIT0;         //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 6: TB0CCR0 = 1346;// D1
        P6OUT |= BIT5;          //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT &= ~BIT0;         //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 7: TB0CCR0 = 1271;// D#1
        P6OUT &= ~BIT5;         //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT |= BIT6;          //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT |= BIT3;          //CS2
        P4OUT &= ~BIT0;         //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 8: TB0CCR0 = 1199;// E1
        P6OUT &= ~BIT5;         //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT &= ~BIT1;         //LS0
        P6OUT |= BIT2;          //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT |= BIT3;          //CS2
        P4OUT &= ~BIT0;         //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 9: TB0CCR0 = 1132;// F1
        P6OUT |= BIT5;          //RS0
        P1OUT |= BIT6;          //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT |= BIT1;          //LS0
```

```
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT |= BIT3;          //CS2
        P4OUT &= ~BIT0;         //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 10:TB0CCR0 = 1059;// F#1
        P6OUT |= BIT5;          //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT &= ~BIT1;         //LS0
        P6OUT |= BIT2;          //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT |= BIT3;          //CS2
        P4OUT &= ~BIT0;         //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 11:TB0CCR0 = 1008;// G1
        P6OUT &= ~BIT5;         //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT |= BIT2;          //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT |= BIT3;          //CS2
        P4OUT &= ~BIT0;         //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 12:TB0CCR0 = 952;// G#1
        P6OUT &= ~BIT5;         //RS0
        P1OUT |= BIT6;          //RS1
        P6OUT |= BIT6;          //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT &= ~BIT1;         //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT &= ~BIT4;         //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 13:TB0CCR0 = 898;// A1
        P6OUT |= BIT5;          //RS0
        P1OUT |= BIT6;          //RS1
        P6OUT |= BIT6;          //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT &= ~BIT1;         //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
```

```
        P6OUT |= BIT4;          //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT &= ~BIT4;         //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 14:TB0CCR0 = 848;// A#1
        P6OUT &= ~BIT5;         //RS0
        P1OUT |= BIT6;          //RS1
        P6OUT |= BIT6;          //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT |= BIT3;          //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT &= ~BIT4;         //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 15:TB0CCR0 = 800;// B1
        P6OUT &= ~BIT5;         //RS0
        P1OUT |= BIT6;          //RS1
        P6OUT |= BIT6;          //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT &= ~BIT4;         //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 16:TB0CCR0 = 767;// C2
        P6OUT |= BIT5;          //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT |= BIT7;          //RE0
        P6OUT &= ~BIT0;         //RE1
        P6OUT &= ~BIT1;         //LS0
        P6OUT &= BIT2;          //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT |= BIT3;          //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT &= ~BIT4;         //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 17:TB0CCR0 = 713;// C#2
        P6OUT &= ~BIT5;         //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT |= BIT7;          //RE0
        P6OUT &= ~BIT0;         //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT &= ~BIT0;         //CS1
```

```
        P4OUT |= BIT3;          //CS2
        P4OUT |= BIT0;         //CE0
        P7OUT &= ~BIT4;          //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 18:TB0CCR0 = 673;// D2
        P6OUT &= ~BIT5;         //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT |= BIT7;          //RE0
        P6OUT &= ~BIT0;         //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT |= BIT3;          //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT &= ~BIT4;         //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 19:TB0CCR0 = 635;// D#2
        P6OUT |= BIT5;          //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT |= BIT6;          //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT |= BIT2;          //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT &= ~BIT4;         //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 20:TB0CCR0 = 599;// E2
        P6OUT |= BIT5;          //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT |= BIT6;          //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT |= BIT2;          //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT &= ~BIT4;         //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 21:TB0CCR0 = 566;// F2
        P6OUT &= ~BIT5;         //RS0
        P1OUT &= ~BIT6;         //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT |= BIT7;          //RE0
        P6OUT &= ~BIT0;         //RE1
        P6OUT &= ~BIT1;         //LS0
        P6OUT |= BIT2;          //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT |= BIT3;          //CS2
        P4OUT |= BIT0;          //CE0
```

```
        P7OUT &= ~BIT4;         //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 22:TB0CCR0 = 534;// F#2
        P6OUT &= ~BIT5;         //RS0
        P1OUT |= BIT6;          //RS1
        P6OUT |= BIT6;          //RS2
        P2OUT &= ~BIT7;         //RE0
        P6OUT |= BIT0;          //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT |= BIT2;          //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT |= BIT3;          //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT &= ~BIT4;         //CE1
        P8OUT |= BIT2;          //CE2
    break;
case 23:TB0CCR0 = 504;// G2
        P6OUT &= ~BIT5;         //RS0
        P1OUT |= BIT6;          //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT |= BIT7;          //RE0
        P6OUT &= ~BIT0;         //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT &= ~BIT2;         //CE2
    break;
case 24:TB0CCR0 = 476;// G#2
        P6OUT &= ~BIT5;         //RS0
        P1OUT |= BIT6;          //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT |= BIT7;          //RE0
        P6OUT &= ~BIT0;         //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT |= BIT4;          //CS0
        P7OUT &= ~BIT0;         //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT &= ~BIT2;         //CE2
    break;
case 25:TB0CCR0 = 449;// A2
        P6OUT |= BIT5;          //RS0
        P1OUT |= BIT6;          //RS1
        P6OUT &= ~BIT6;         //RS2
        P2OUT |= BIT7;          //RE0
        P6OUT &= ~BIT0;         //RE1
        P6OUT |= BIT1;          //LS0
        P6OUT &= ~BIT2;         //LS1
        P6OUT &= ~BIT3;         //LE0
        P6OUT &= ~BIT4;         //CS0
        P7OUT |= BIT0;          //CS1
        P4OUT &= ~BIT3;         //CS2
        P4OUT |= BIT0;          //CE0
        P7OUT |= BIT4;          //CE1
        P8OUT &= ~BIT2;         //CE2
```

```
                break;
            default: TB0CCR0 = 65354;       //default return 0
        }
    }
}

    //Sawtooth Timer Interrupt Vector (Goes into the DAC)
    #pragma vector = TIMER0_B0_VECTOR
    __interrupt void TIMERTEST(void)
    {
        if(noteplayed > 0)          // If there is a note being played
        {
            // Create an 8-Bit counter
            if(counter > 255)
                counter = 0;

            counter = counter + 1;

            // 1's Bit
            if(counter % 2 == 1)
                P1OUT |= BIT2;
            else
                P1OUT &= ~BIT2;

            // 2's Bit
            if(counter % 4 >= 2)
                P1OUT |= BIT3;
            else
                P1OUT &= ~BIT3;

            // 4's Bit
            if(counter % 8 >= 4)
                P1OUT |= BIT4;
            else
                P1OUT &= ~BIT4;

            // 8's Bit
            if(counter % 16 >= 8)
                P1OUT |= BIT5;
            else
                P1OUT &= ~BIT5;

            // 16's Bit
            if(counter % 32 >= 16)
                P2OUT |= BIT4;
            else
                P2OUT &= ~BIT4;

            // 32's Bit
            if(counter % 64 >= 32)
                P2OUT |= BIT5;
            else
                P2OUT &= ~BIT5;

            // 64's Bit
            if(counter % 128 >= 64)
                P2OUT |= BIT2;
            else
                P2OUT &= ~BIT2;

            // 128's Bit
            if(counter % 256 >= 128)
                P2OUT |= BIT0;
            else
                P2OUT &= ~BIT0;
        }
```

```
        else // if nothing is being played
        {
            // Turn off all pins
            P1OUT &= ~BIT2;
            P1OUT &= ~BIT3;
            P1OUT &= ~BIT4;
            P1OUT &= ~BIT5;
            P2OUT &= ~BIT4;
            P2OUT &= ~BIT5;
            P2OUT &= ~BIT0;
            P2OUT &= ~BIT2;
        }
    }

// Receiving UART Interrupt
#pragma vector=USCI_A1_VECTOR
__interrupt void Temp_control(void)
{
  switch(__even_in_range(UCA1IV,4))
  {
      case 0:break;                             // Vector 0 - no interrupt
      case 2:                                   // Vector 2 - RXIFG
          noteplayed = UCA1RXBUF;               // Store noteplayed into the RXBUF
          break;
      case 4:break;                             // Vector 4 - TXIFG
      default: break;
  }
}
```