Build and Analysis of a Digital Synthesizer

Adrian Jones and Nikhil Mehandru

ES52: Joy of Electronics, Part I
Spring 2014
5.12.14

Problem Statement and Solution:

Our objective was to create a synthesizer that can be controlled by user input via push buttons or Musical Instrument Digital Interface (MIDI) commands through a microcontroller.

On a higher level, music is understood to consist of pitched sounds that can be played in innumerable combinations. These sounds or notes, are waves that occur at specific frequencies. Successful operation of our synthesizer would involve sonically producing the twelve notes within an octave at the push of one of more buttons. Given that polyphony (producing more than one tone at once) would require duplication of the circuitry according to the number of desired simultaneous tones, we have decided to limit our goals to simple monophonic capability. At the highest level, our project was divided into three main parts

I) The digital side (employing push buttons and potentiometers (pots) to interface with an Arduino Due loaded with custom code)

Firstly, there are numerous synthesizers in use which can in part be characterized by the waveforms they generate, which directly affect the timbre of the sound, as well as the methods they employ to modify those waveforms. As a baseline, we decided to use the Digital to Analog (DAC) converters available on an Arduino Due development board. We programmed 3 different waveforms for output via a DAC pin. The 3 different waveforms were of the sinusoidal, triangular, and sawtooth form.

II) The analog side, first half (Handling the Arduino output to interface with a speaker or headphones via the use of a buffer operational-amplifier (op-amp), a summing op-amp with pot control, a class B power op-amp, and a TRRS jack)

Secondly, the waveforms upon generation of the various waveforms, we needed to buffer the signal. After that, we integrated volume control via an op-amp with a pot, as well as a class AB power op-amp to supply sufficient current to the load, whether it be a set of headphones or a speaker. Finally, we implemented a TRRS jack to allow for the use of headphones.

III) The analog side, second half (Adding low frequency oscillation (LFO) to introduce digital summing and digital modulation to the outputted waveforms or sounds)

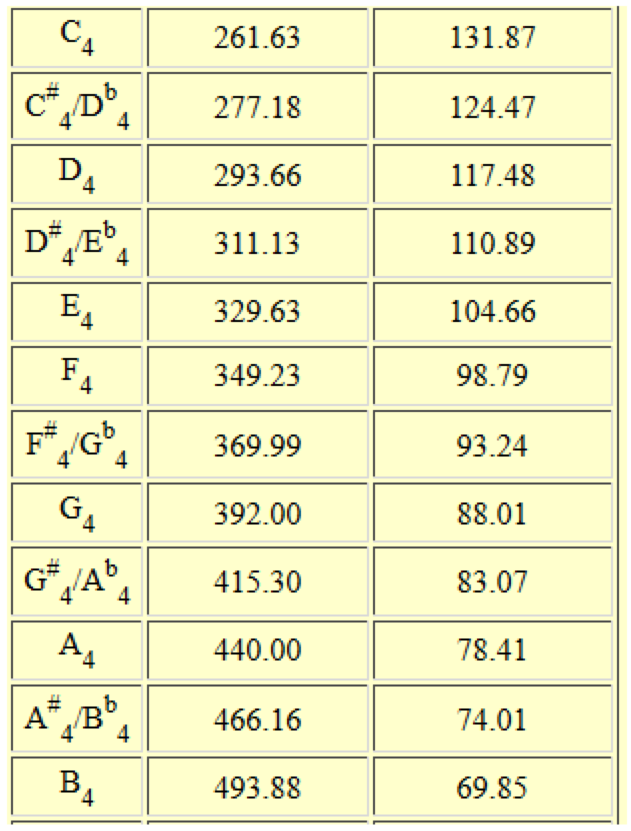
Thirdly and finally, we wanted to be able to modify the waveforms outputted by adding digital adding and modulation capabilities, effectively adding low frequency oscillation (LFO) to our synthesizer. LFO can be defined as giving an electronic signal a rhythmic pulse or sweep. We developed simple LFO by building a 555 timer into our circuit, where the output of the 555 timer was fed into an analog in pin of the Arduino for digital summing, and fed into a summing amplifier for modulation of the 555 timer’s output with the DAC1’s normal output. Using an LFO in a summer or modulator introduces complexities into the resulting sound. Of course, the specifics vary greatly depending on the type of summing or modulation, the relative frequencies of the LFO, and the signal being modulated, etc.

Description of Modules:

I) The Digital Side (Employing push buttons and potentiometers (pots) to interface with an Arduino Due loaded with custom code)

According to Western music theory, there are twelve musical pitched sounds or notes (c, c#, d, d#, e, f,…) that occur at specific frequencies. To emulate the function of a standard keyboard style synthesizer we arranged twelve push-button switches along our breadboard, wiring each to ground and a separate digital input on the Arduino, so that at the push of a button we could output a waveform at the desired frequency.

Note: Frequency (Hz), Wavelength (cm)
Frequency table for notes with A4 = 440Hz tuning



C_4	261.63	131.87
$C^{\#}_4/D^{b}_4$	277.18	124.47
D_4	293.66	117.48
$D^{\#}_4/E^{b}_4$	311.13	110.89
E_4	329.63	104.66
F_4	349.23	98.79
$F^{\#}_4/G^{b}_4$	369.99	93.24
G_4	392.00	88.01
$G^{\#}_4/A^{b}_4$	415.30	83.07
A_4	440.00	78.41
$A^{\#}_4/B^{b}_4$	466.16	74.01
B_4	493.88	69.85

Initially, we had hoped to generate these pitches in an analog manner, where depression of buttons would allow current to flow through a combination of resistors and capacitors to set the frequency of the outputted signal through a voltage controlled oscillator. However, we quickly realized that, with the accuracy of the resistors we were using being +/- 10% (whereas our requirements required 1-5% accuracy), and the difficulty of switching octaves, generating pitches or notes using analog components was not feasible within our the constraints of our project. Instead, we switched to use of the DAC built into the Arduino Due core. Further, through use of the DAC, we were able to program a variety of different waveforms (specifically, 3 different types: of the sinusoidal, triangular, and sawtooth form), whereas the analog-based waveform generation would have limited us to only square waveforms.

We coded the various waveforms by employing a set of three tables, where each table held the corresponding ‘yvalues’ for a specific waveform. The frequency of the outputted waveform was set by varying the delay after outputting a specific yvalue from a given table. Using a loop in this manner, we cycled through all 140 values in the table, where, once the 140th value in the table was reached, the counter was reset to 0 to have a continuous waveform (see Code Section 1). Note that the entire code set that was used is attached at the end of the report, entitled ‘Complete Code Set.’

We then tested our code, by reading out output using an oscilloscope. The results are shown below in Figures 1, 2, 3, 4, 5, and 6, where we pressed 2 buttons while each of the three waveforms was loaded. The first button corresponded to note c and the second to note b. One important point to note when looking at the code, though, is that the frequency values for each of the notes do not exactly match the standard tuning. However, this was deliberately done, so as to better hear the differences of each of the notes. We realized that the difficulty in hearing the difference between the notes when using the standard tuning frequencies was due to the delay that was present when the microcontroller was processing instructions to output the waveform.

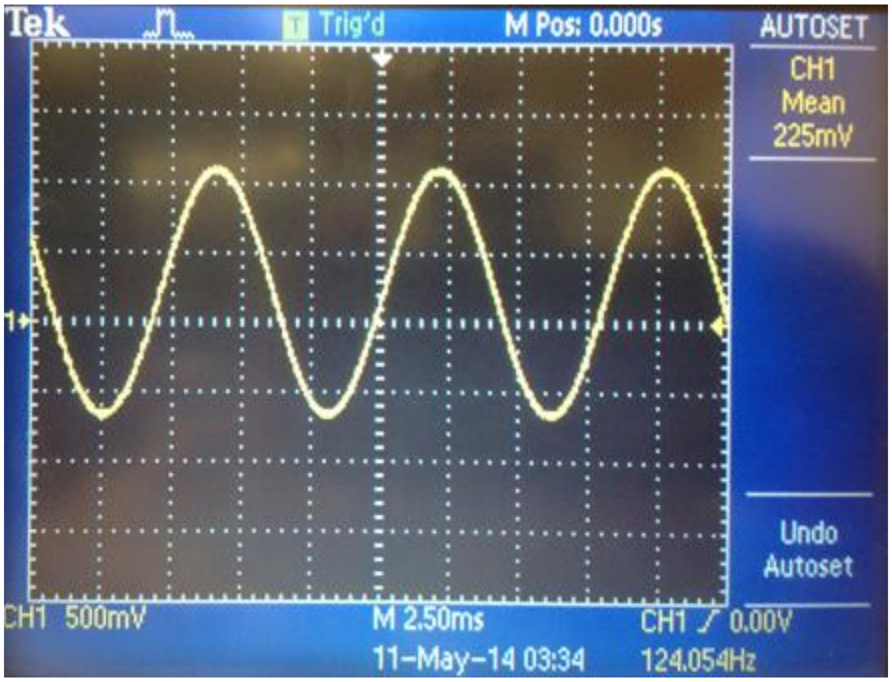


Figure 1: Sinusoidal Waveform, Note C

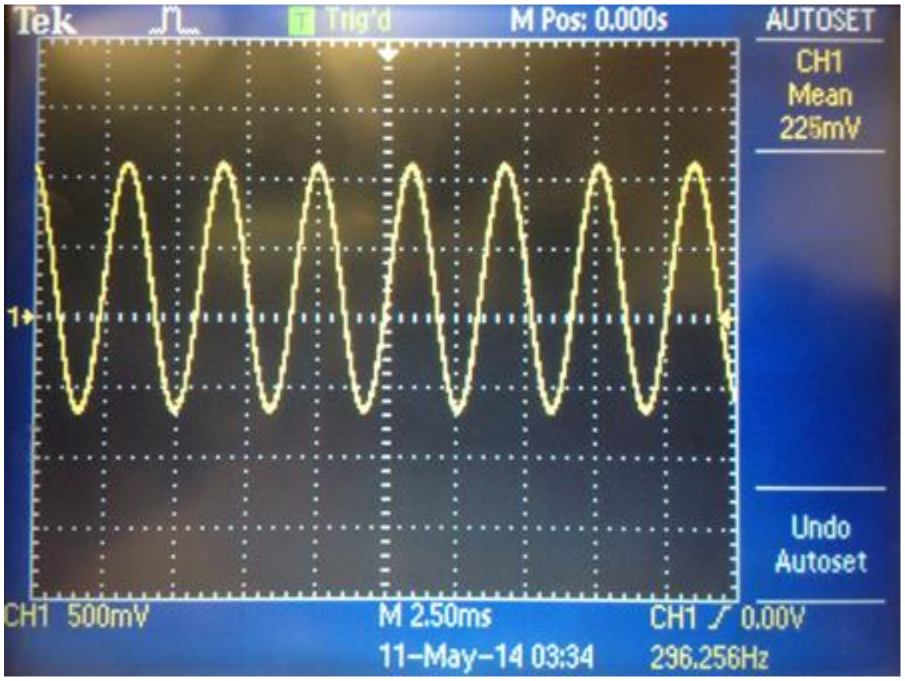


Figure 2: Sinusoidal Waveform, Note A

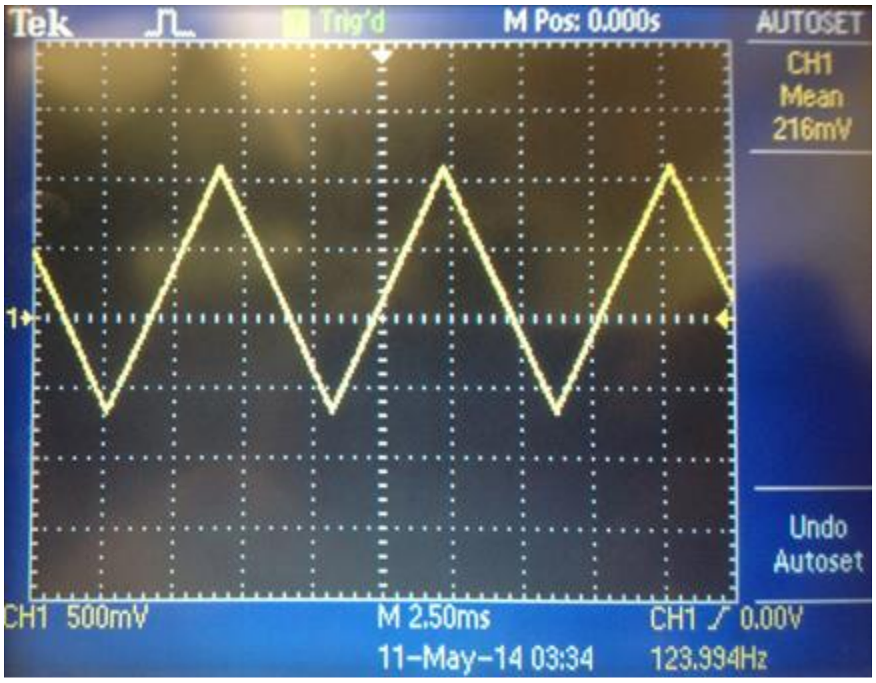


Figure 3: Triangular Waveform, Note C

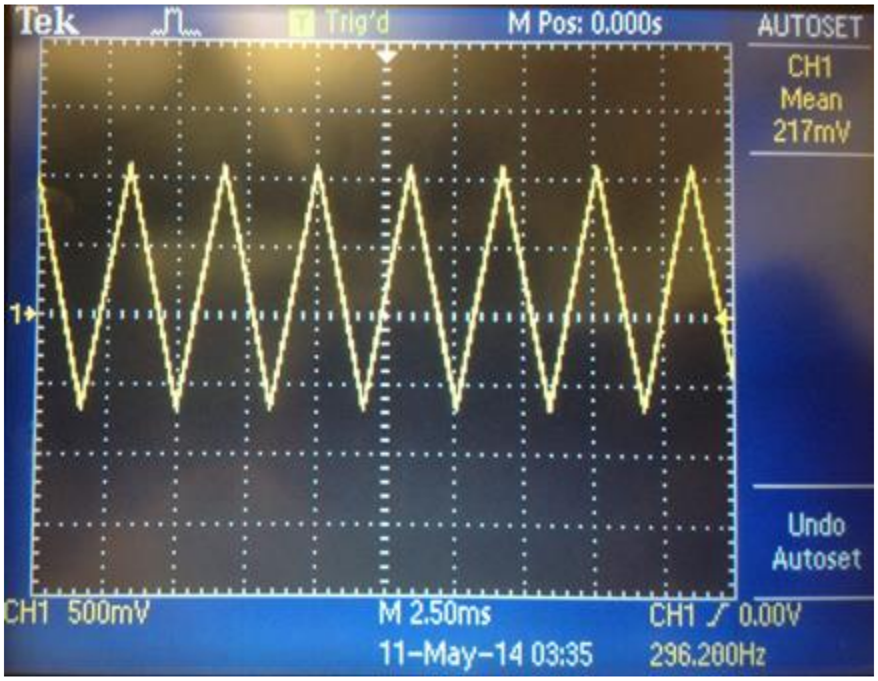


Figure 4: Triangular Waveform, Note A

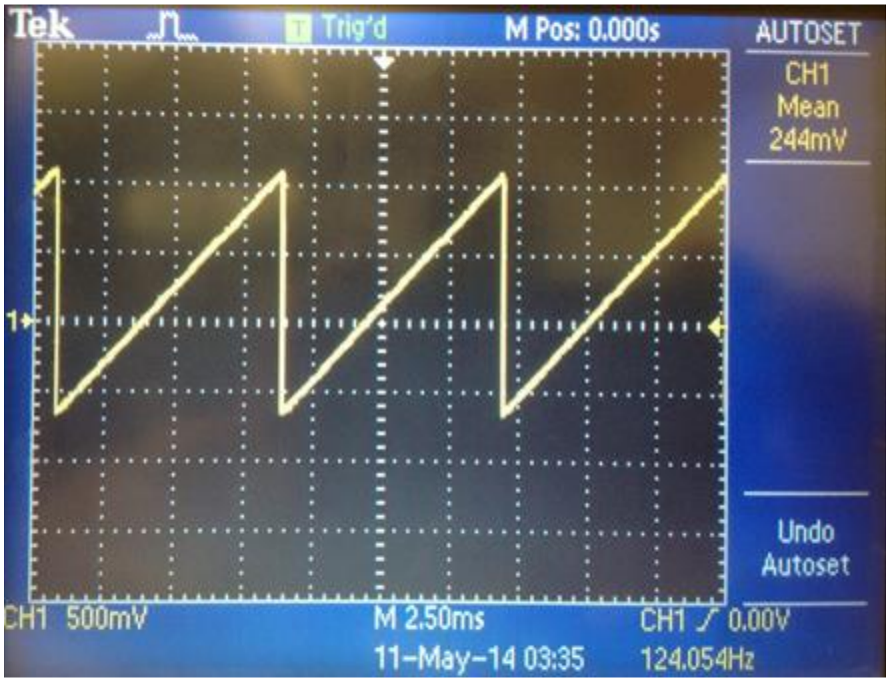


Figure 5: Sawtooth Waveform, Note C

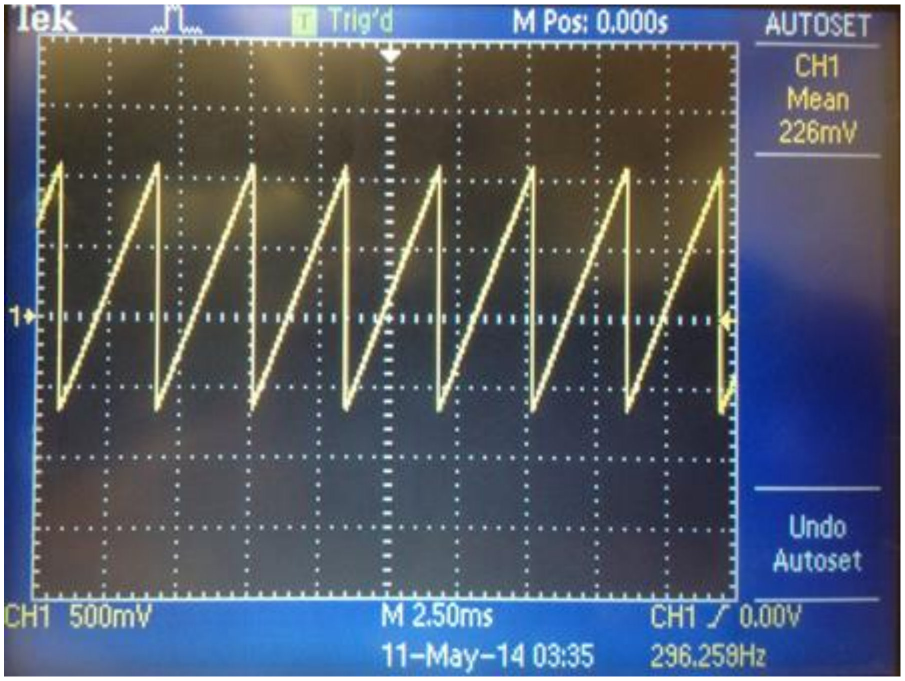


Figure 6: Sawtooth Waveform, Note A

```
// if any buttons are low (activated), write output wave form as long as button is held
void waveform_one () {
    for (int i = 0; i < 12; i++) {
        if (digitalRead(buttons[i]) == LOW) {
            int j = 0;
            // multiplying frequency by power of 6+octave variable allows the output to be in a hearable yet controllable range
            sampletime_one = 1000000.00 / (frequencies[i]*pow(2,(6+octave)));
            while (digitalRead(buttons[i]) == LOW) {
                // if the modulator switch is held then the samples will be multiplied by the analog generated square wave
                output_table = waveformsTable[waveformtype][j];
                // write the selected waveform on DAC1
                analogWrite (DAC1, output_table);
                j++;
                    // Reset the counter to repeat the wave
                    if(j == maxSamplesNum) {
                        j = 0;
                    }

                delayMicroseconds(sampletime_one);
            }
        }
    }
}
```

Code Section 1

II) The Analog Side, First Half (Handling the Arduino output to interface with a speaker or headphones via the use of a buffer operational-amplifier (op-amp), a summing op-amp with pot control, a class B power op-amp, and a TRRS jack)

To ensure the amplitude of the output of the DAC was not affected by being sent to a component with low impedance, we wired the pin directly to an op-amp that served as a buffer. We designed the low frequency oscillator within our circuit so that it could have two possible effects on our waveform, on the analog side we used a summing amplifier to add the square wave output of the low frequency oscillator to whatever waveform was outputted by the DAC.

We measured the output of the DAC before amplifying the waveform and found the peak of the wave to measure around a hundred millivolts, with lab 2 as a reference for amplifying an audio signal we included the same preamp that provided variable gain from 0 to 10 along with a class B power op-amp to provide the necessary current to the speaker/headphones. Testing the amplification stage we found the gain provided by the pre-amp to be more than sufficient as some clipping of the waveform was observed before reaching max amplification. Initially, we had powered an 8 Ohm speaker, but found the final signal to be too weak despite alterations to the amplifiers. However, by sending the output of the power amp to a TRRS jack and connecting headphones we were able to hear a much louder signal since our headphones had a much higher input impedance than the small speaker. All of the aforementioned analog circuitry can be seen in Figure 7, which displays the entire circuit.

III) The Analog Side, Second Half (Adding low frequency oscillation (LFO) to introduce analog summing and digital modulation to the outputted waveforms or sounds)

Most synthesizers worth their salt provide the user with as many ways as possible to modify the generated sound. One common way to modify sounds is through use of a low frequency oscillator; these can send an oscillating signal to modulate different parameters within the synthesizer. For instance, a low frequency oscillator could send a control signal to an amp and cause the volume of a note to fluctuate when triggered, so as to create the effect of tremolo.

LFOs are defined as oscillators that generate a modulating signal under 20 Hz and to achieve this we used a 555 timer and, using resistors and a capacitor, set its output frequency to be 16.35 Hz or the note C in the lowest octave. As previously mentioned, in our circuit our LFO when switched, provides a signal to a summing amplifier to create a ringing effect when added to the DAC's waveform. To achieve modulation of the signal digitally, we placed another switch in the path of the LFO's output to an analog input on the Arduino, and when a separate push-button is pressed along with a note, the signal outputted by the DAC is a sample of the LFO's square wave multiplied by a sample of a sine, triangle, or sawtooth wave. Lastly, a picture of our completely assembled circuit is shown in Figure 8. Following Figure 8, we have Table 1, which is our bill of materials for said circuit.

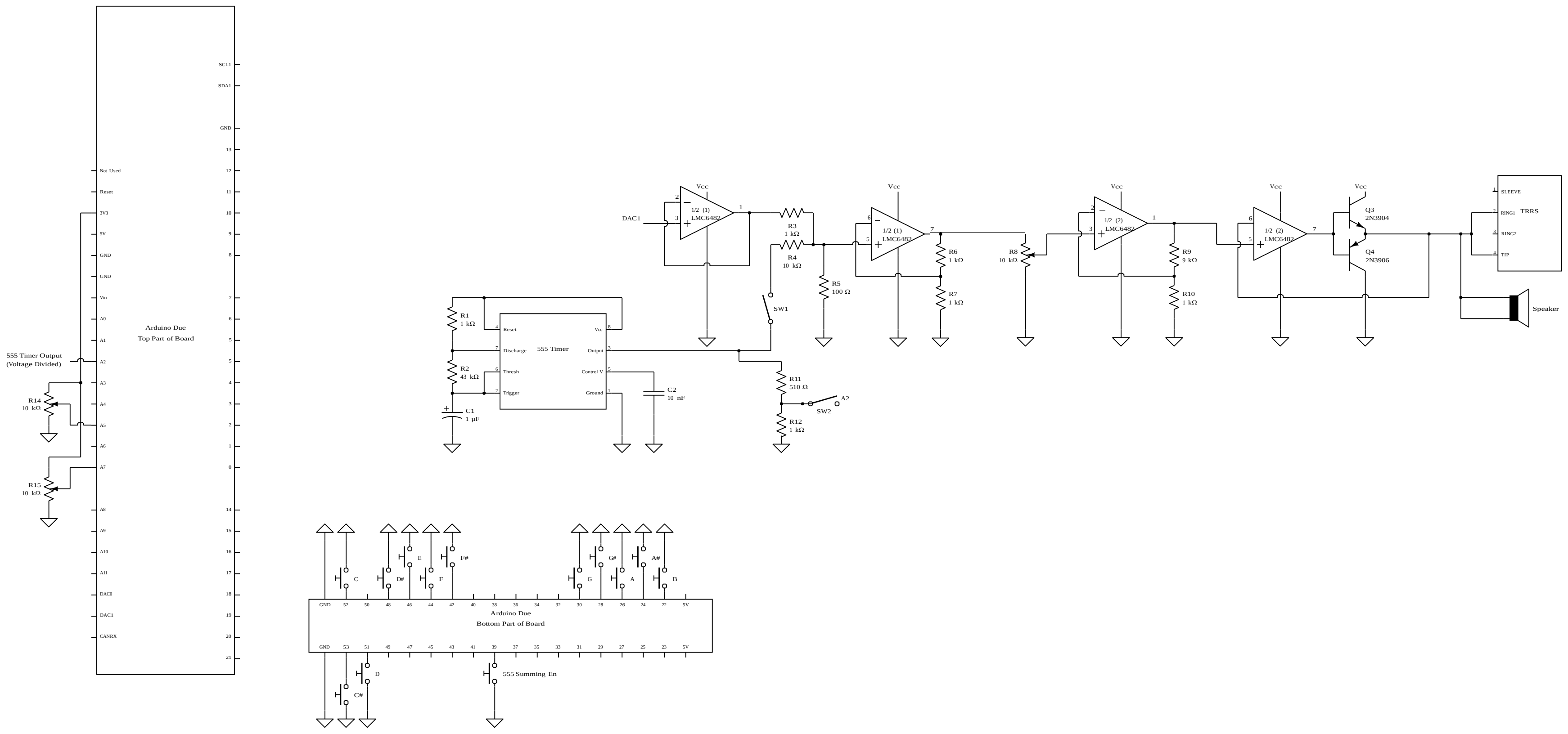


Figure 7: Complete Circuit Schematic

IV) Future Work

Through completing this project, we learned a great deal about digital to analog conversion, as well as how complex commercially available synthesizers actually are. It took us many hours to implement the functionality we have, which is quite minimal compared to professional-grade products. As such, there are plenty of things that can be done to further advance this project, with the most immediate functionality being implementation of polyphony. We had tried a number of different approaches to achieve polyphony, but they unfortunately all failed and we ran out of time before we could integrate it into our final prototype. We hope that others will find this write-up useful and possibly build upon it. Thanks for taking the time to read our project report.

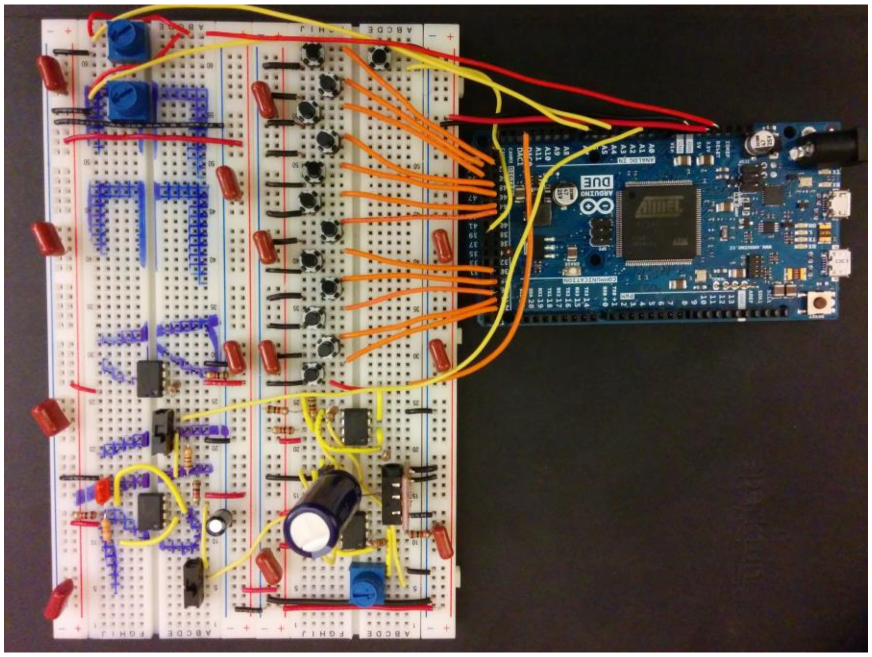


Figure 8: Photograph of Completed Circuit

Component	Vendor	Description	Unit Cost	Quantity
Arduino Due Board	Adafruit	Standard Kit	$49.95	1
TRRS 3.5mm Jack Breakout	Sparkfun	Used to hook in headphones	$3.95	1
¼ W Resistors	Amazon	10 to 1 Mohm Kit		
10 nF Capacitor	Digi-Key	Used in 555 timer setup	$0.21	1
1 µF Capacitor (Polarized)	Digi-Key	Used in 555 timer setup	$0.19	1
Mini Push Button Switch	Sparkfun	Used for user input	$0.35	12
0.5 W, 8 Ohm Speaker	Sparkfun	Used to play sound without headphones	$1.95	1
NA555DR Timer	Sparkfun	Used for addition/modulation of waveform	$0.95	1
Breadboard	Sparkfun	Needed to interface analog component with each other and with Arduino	$5.95	2
LMC6482 Operational Amplifier	Digi-Key	Used in part 2, analog circuit, first half	$1.47	2
10K Three Pin Potentiometer	Digi-Key	Used in preamp and for analog input of 555 Timer/waveform selection	$1.01	3
2N3904 npn Transistor	Digi-Key	Used in push-pull power amp	$0.19	1
2N3906 pnp Transistor	Digi-Key	Used in push-pull power amp	$0.19	1
TOTAL COST			**$79.45**	

Table 1: Bill of Materials for Entire Project

References:

1) "Arduino - DueSimpleWaveformGenerator." *Arduino - DueSimpleWaveformGenerator*. N.p., n.d. Web. 28 Apr. 2014.
2) "Electronic Synthesis." N.p., n.d. Web. 07 Apr. 2014.
3) *ES52 ISite*. N.p., n.d. Web. 28 Apr. 2014.
4) "Frequencies of Musical Notes, A4 = 432 Hz." *Frequencies of Musical Notes, A4 = 432 Hz*. N.p., n.d. Web. 7 Apr. 2014.
5) "Low-frequency Oscillation." *Wikipedia*. Wikimedia Foundation, 18 Apr. 2014. Web. 2 May 2014.
6) "Language Reference." *Arduino*. N.p., n.d. Web. 28 Apr. 2014.

7) "Ring Modulation." *Wikipedia*. Wikimedia Foundation, 05 Nov. 2014. Web. 2 May 2014.
8) "Synth Secrets, Part 10: Modulation." *Synth Secrets, Part 10: Modulation*. N.p., n.d. Web. 29 Apr. 2014.