Liam Gogley 012742858
Mark Munoz 013069795

Project 1: R2R DAC Piano

CECS 447

2018-02-26

Table of Contents

Table of Figures

I. Introduction:

The purpose of this project was to explore the construction and interfacing of a 6-bit DAC (digital to analog converter) using an “R-2R” ladder or binary weighted DAC with 1.5K resistors. Ports PA2 to PA7 were connected to the 6 bit input to the DAC, and ports PD0 - PD3 and PC4 - PC7 handle the button press options of the 4x4 keypad.

II. Operation:

The project operates from one onboard push button ‘SW1’ that cycles between 4 modes: triangle wave, square wave, sine wave, and piano mode. The first three modes operate at 262 Hz, while the piano mode has a varying frequency, using the SysTick timer to generate the sound frequency and a clock count to generate the 128 steps for each wave. In piano mode, the keypad is interfaced with port D and port C as column and row reads respectively. The resulting sound wave is fed into a 10K potentiometer, which acts as an adjustable voltage divider to the LM386 amplifier’s non-inverting input. The signal is output from the amplifier to a low impedance speaker.

III. Hardware:

The DAC chosen is an R-2R ladder with 1% 1.5K and 3K resistor values. While this yields lower accuracy to the binary weighted DAC, it offers good accuracy with little to no discernable loss in quality of the signal. To create the ladder, 1.5K resistors are placed in series for a 3K total resistance per port, and one 1.5K connecting each port to the analog voltage out port. To prevent signal attenuation and noise, the leads of the components were kept short.

The formula used to calculate the R-2R ladder DAC, with *x* denoting the 3.3v “bit”:

$$Vout = \frac{x_5}{2} + \frac{x_4}{4} + \frac{x_3}{8} + \frac{x_2}{16} + \frac{x_1}{32} + \frac{x_0}{64}$$

To make the amplifier, the circuit requires the analog signal in the non-inverting input with the inverting input grounded. Each capacitor is used for signal conditioning and power supply conditioning, larger capacitor values handle large bumps while small values short interval power interruptions and spikes. At the output of the amplifier, a DC blocking capacitor bank is used to only permit the audio wave from passing.

Hardware Block Diagram:

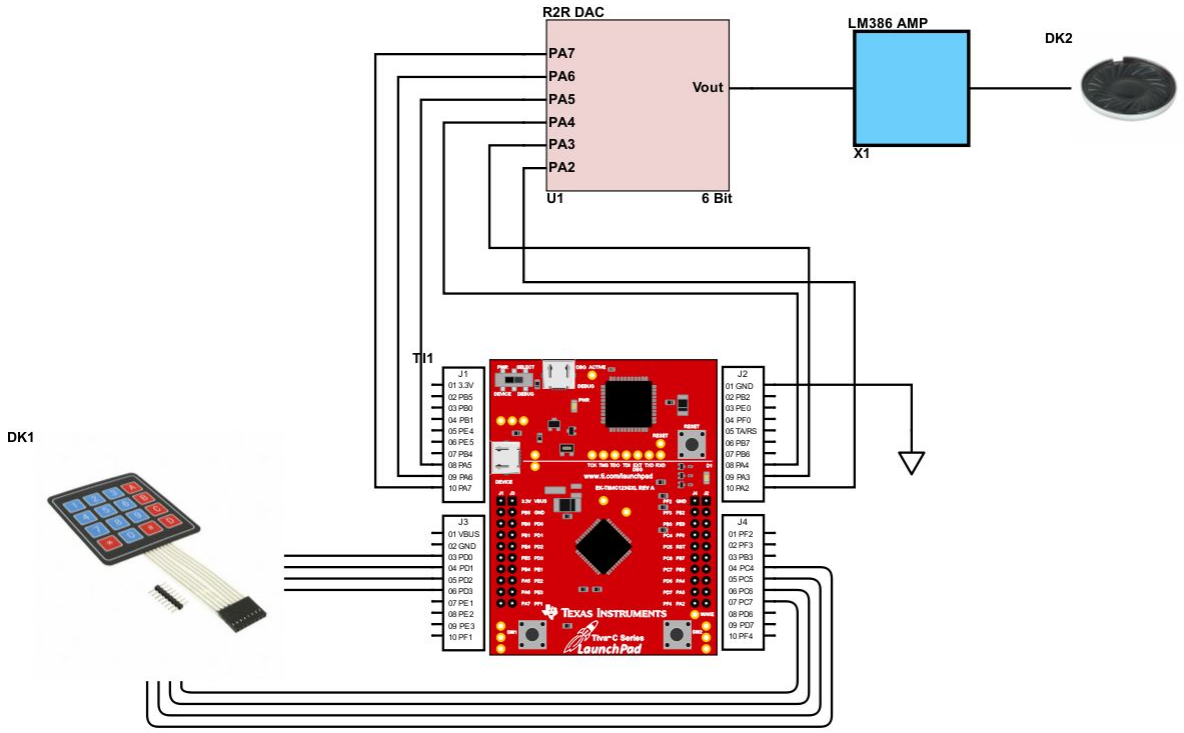

Figure 1.1: Hardware Block Diagram

R2R Schematic:

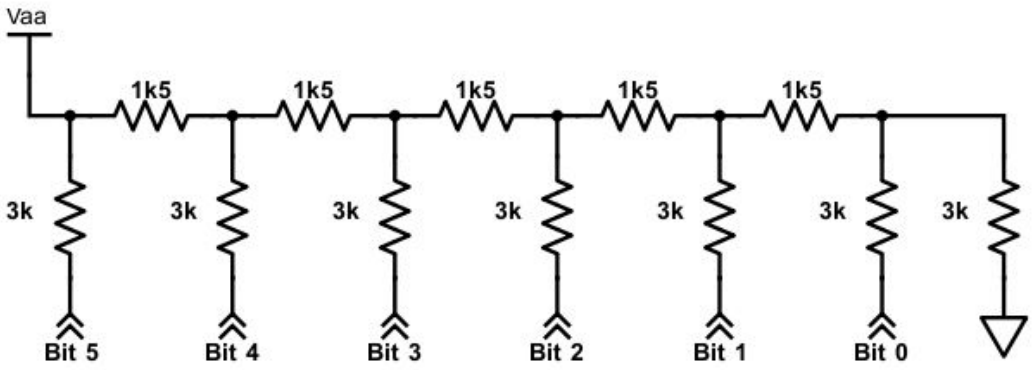

Figure 1.2: R2R Schematic

Triangle Wave:

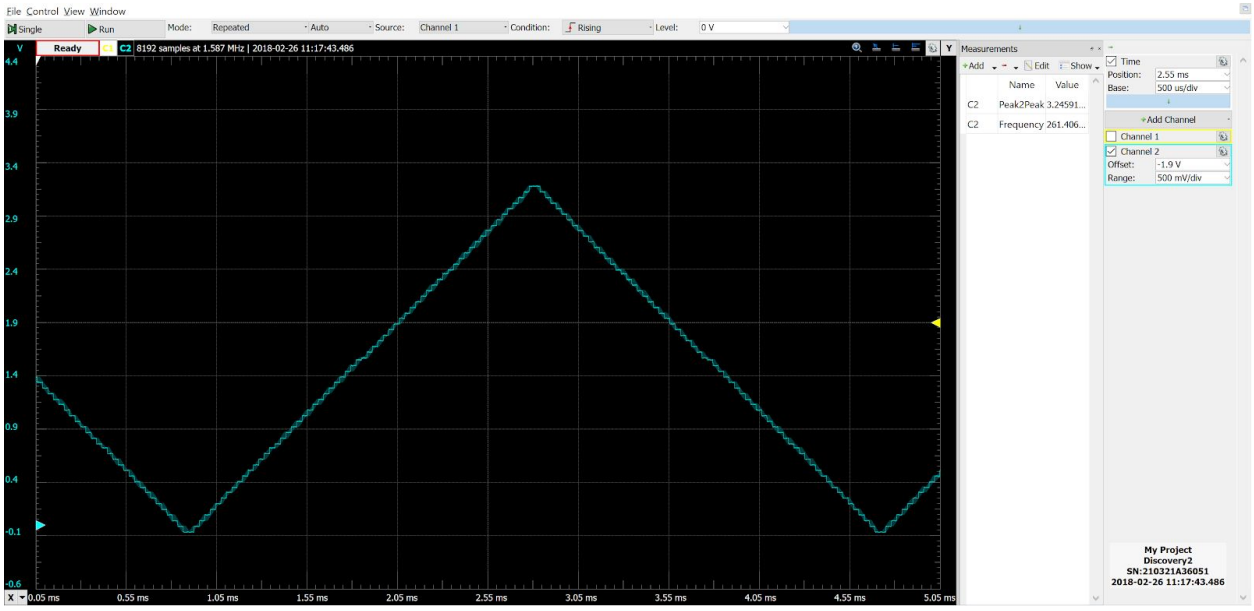

Figure 1.3: 262 Hz Triangle Wave | Vpp = 3.25V

Square Wave:

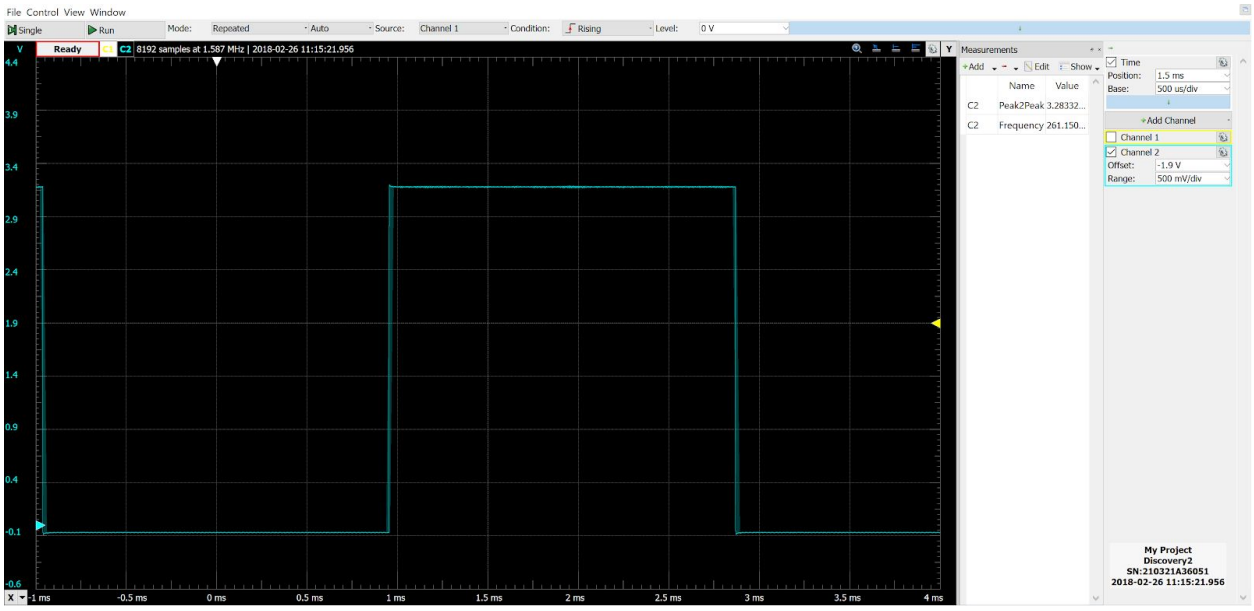

Figure 1.4: 262 Hz Square Wave | Vpp = 3.28V

Sine Wave:

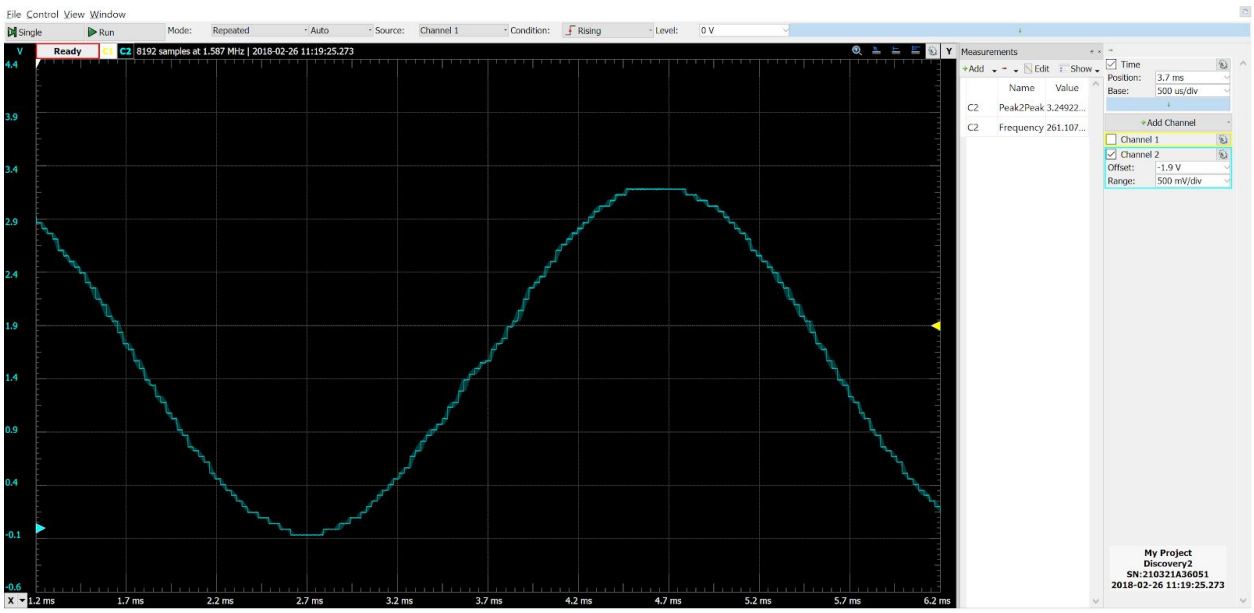

Figure 1.5: 262 Hz Sine Wave | Vpp = 3.24V

IV. Software:

To generate the triangle waveform, the SysTick handler incremented the variable clk_cnt for 64 steps, which was then assigned to input of the 6-bit dac. Next clk_cnt was decremented by one for 64 steps. The resulting output of the DAC is a triangle wave that increases one precision level for half a period, and decrements one precision level for the other half of the period, as seen in Figure 1.3.

To generate the Square wave, the DAC output was set to the highest output for clk_cnt <64 and then set to 0 for 64<clk_cnt<=128. The resulting output of the DAC can be seen in Figure 1.4.

The following equation was used to generate a 128 element sine waveform table:

$$V_{out} = 1.65 * sin(pi * \frac{x}{64}) + 1.65$$

where x is the index of the 128 element table. The value of the array element is the corresponding DAC n value that is closest to V_{out}. The same clk_cnt variable

was used to iterate through the sine table, updating the output of the DAC 128 times per waveform period, resulting in the waveform seen in Figure 1.5.

For the final piano mode, the generation of the sine wave on the DAC was performed using the same 128 element sine table, however the frequency at which the table was iterated through varied depending on which keypad button was pressed. If the keypad button ‘1’ was pressed, the SysTick reload value was modified such that the DAC output sine wave would have a frequency of 262 Hz, generating a C note. Keypad button ‘2’ would change the sine waveform frequency to be 294 Hz, generating a D note, keypad ‘3’ would output at 330 Hz(E note), and so on for an entire octave of notes(C-B). The SysTick reload value for varying frequencies was calculated using the following equation:

$$reload = \frac{Bus\ Clock\ Frequency}{Frequency\ of\ Note * Number\ of\ Samples}$$

Lastly, if no input from keypad was detected, the DAC output was set to 0v and SysTick interrupt was disabled, prevent the output of the DAC from being updated until a keypad button was pressed.

Software Flow Diagram:

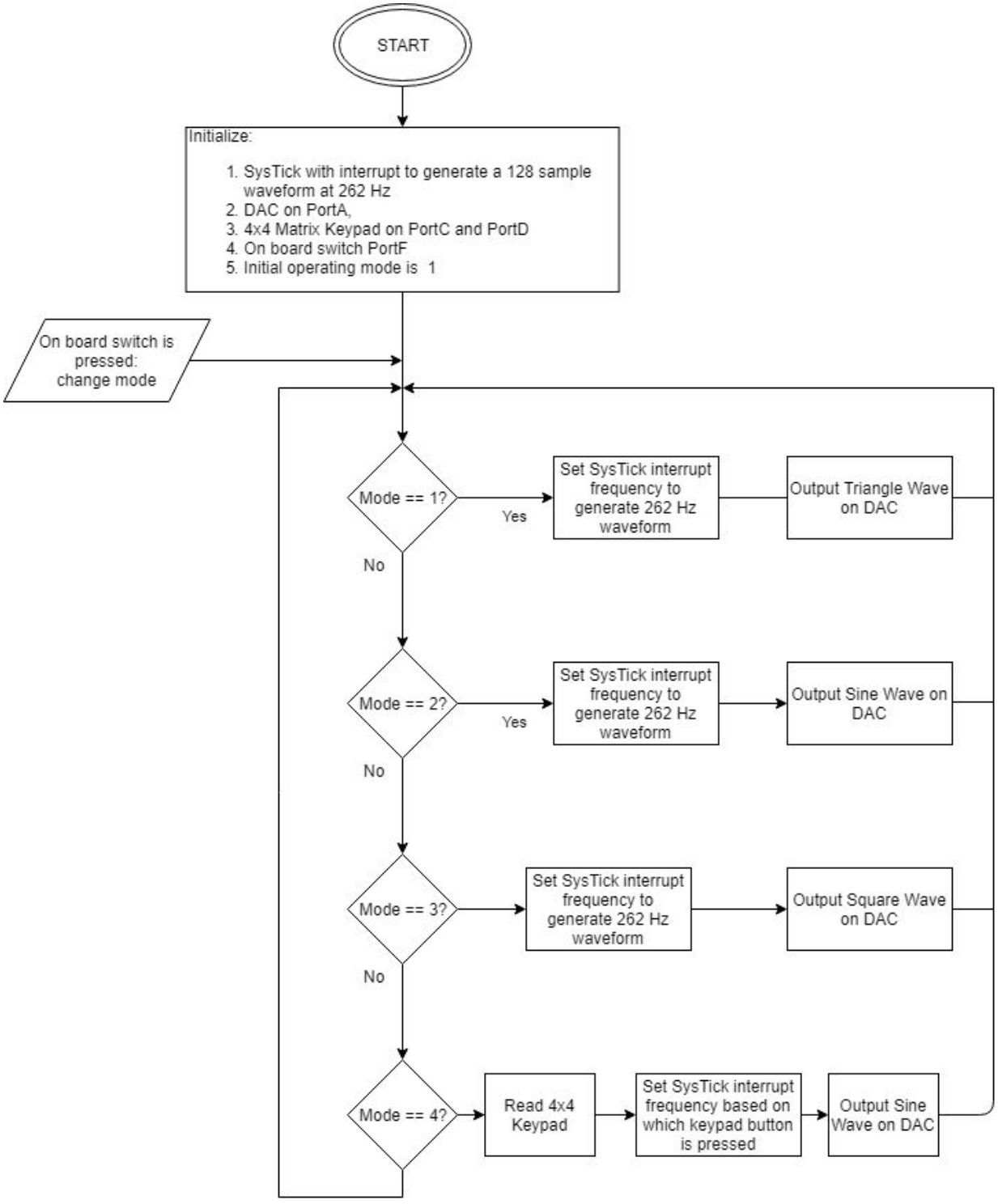

Figure 1.6: Software Flow Diagram

V. Conclusion:

In this project we experienced immediate successes with a clean output of the triangle, square, and sine waves, thanks to Mark's calculations and use of 128 steps. With little noise in our circuit, we were able to retrieve clear visuals of the expected signals, and we moved to the function of the 4x4 matrix pad. First tests of our keypad included the use of a case statement, and onboard LEDs for debugging. After unexpected LED results and unresponsive notes from testing in the lab, we modified the algorithm to the algorithm provided in class that places priority on the column and checks the row activated using an 'if' condition. This allowed our piano keys to function as expected, and we proceeded to test the notes programmed.

To keep traces short, a simple interface board was built on a perfboard which contained the R2R DAC and analog Vout. This signal was applied to a solderless breadboard which contained the amplifier circuit and speaker. Upon first test, we attempted to use the adjustable bench power supply from the lab room, but we quickly realized the amount of noise injected on the circuit. After evaluating the resultant signal with a scope, we switched our power source to the onboard VBUS for our 5 volt out.