2019

Digital to Analog Conversion

Kevin Lopez

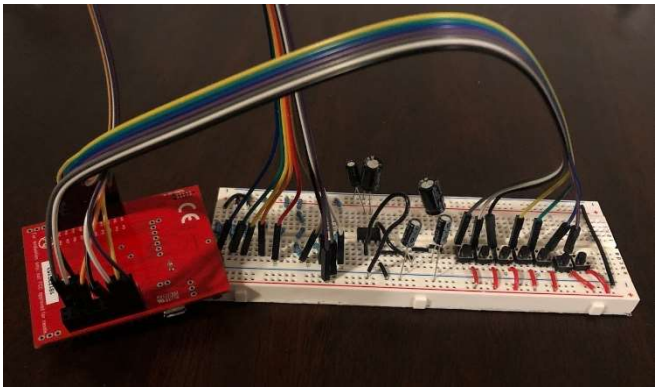
February 27, 2019

Piano using Binary weighted DAC

Table of contents

Table of Figures

Introduction



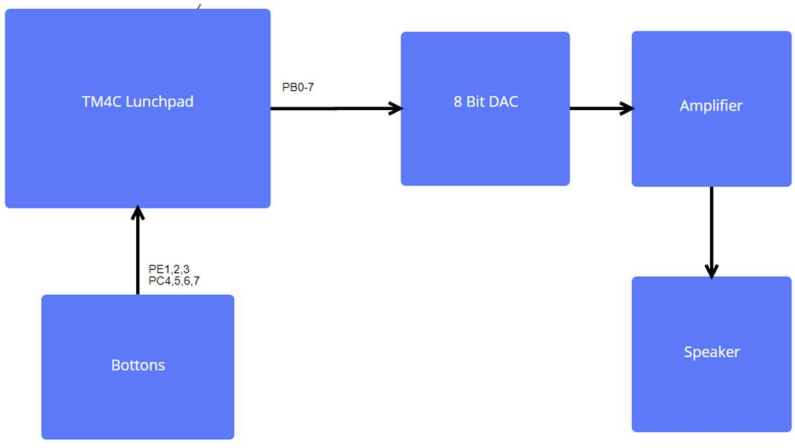
The objective of this project was to create a piano using our own DAC, and TM4C. For this project, we created a Binary weighted DAC and was controlled using the TM4C microcontroller. We used various concepts of the ARM Processor like GPIO, timer, Interrupts, and Digital to analog conversion. For this project, we created a sin wave and did so using 256 points of precision. The range of our DAC was from 0v to 3.3 volts. We created our own waveform for our DAC using excel, using various methods like ifs, sums, vlookup, and others.

Operations

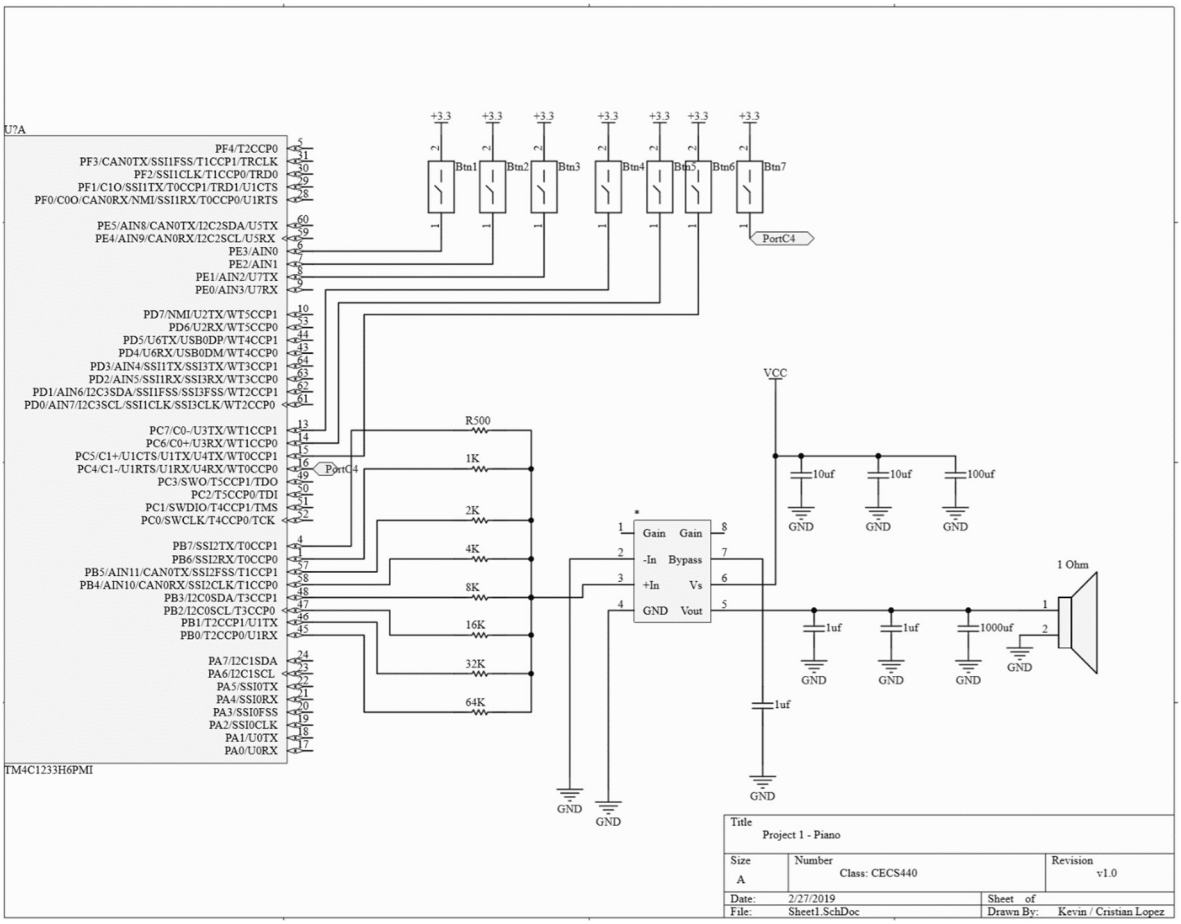
This piano has the ability to produce a sawtooth, triangle, square, and sine wave in the four different modes. In order to go through the modes, you have to press on switch one of the TM4C. We added an additional piano mode that is enabled on the last mode and is used with seven buttons, aka a piano mode. The sine wave was used for the production of the notes. We made the C, D, E, F, G, A, and B notes by modifying the frequency, and could hear each different note by holding on the different buttons.

Hardware

Hardware Block Diagram



Schematic



List of Components

- TMC4C Launchpad
- LM386 Operational amplifier
- 1 Ohm Speaker
- 500, 1k, 2k and up to 64k 1% tolerance resistors
- Buttons, wires, and capacitors
- Power supply, oscilloscope and other components for debugging/testing.

Hardware Explanation

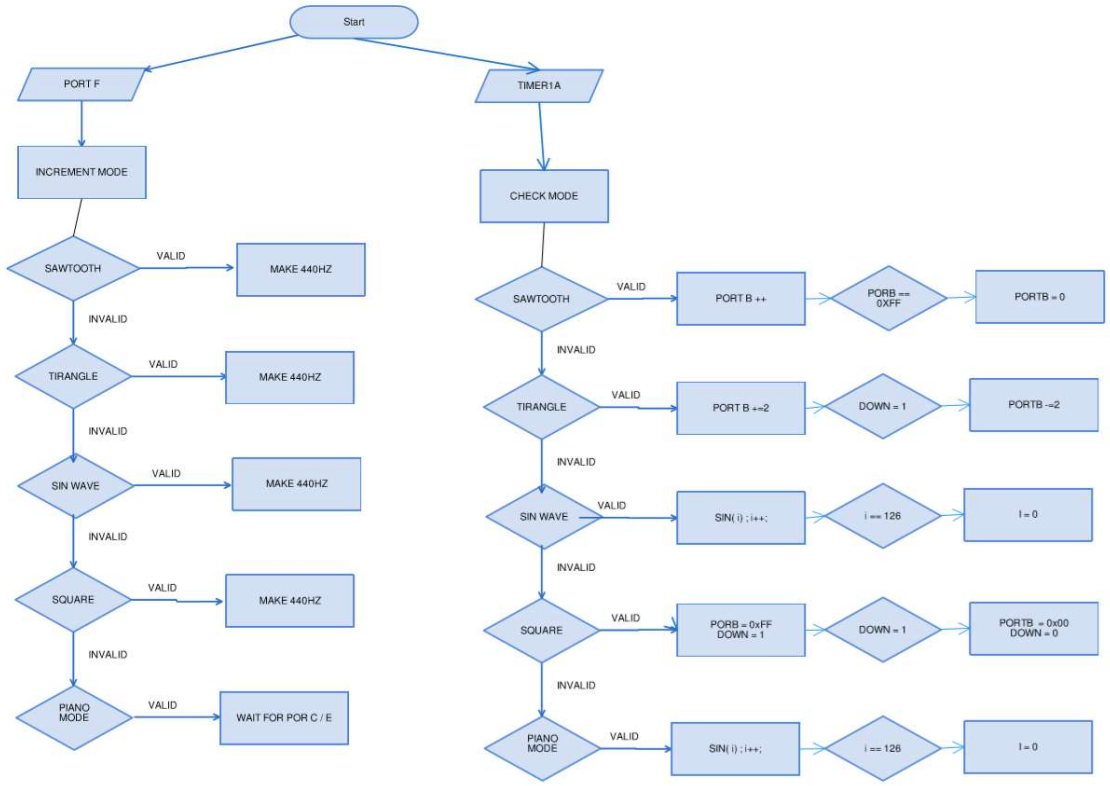
The main object used for this lab was the TM4C microcontroller and was used for the purpose of creating a DAC using the binary weighted resistors that varied from 500, and powers of two up to 64k. We used an LM326 as our amplifier since it helped us make our signal be augmented up to 5v, or voltage being supplied into. We used a speaker so we could hear such frequencies, and we used this speaker since It was the one that fitted our budget.

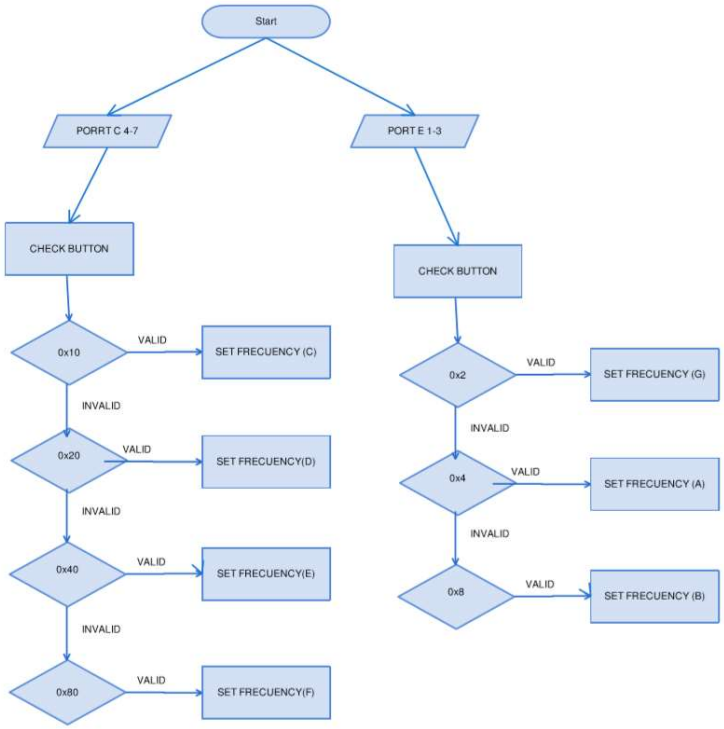
Software

Approach

With the help of GPIO, and TIMER interrupts we were able to make a successful and complete working project. We used an interrupt for Port F, that would change the modes of the waves or go into the piano mode, in that same handler we would set the frequency that our timer would be incrementing the waveform. In the Timer interrupt, we would check what mode we were on and we increment depending on. If we ever get into the piano mode, then we would be going into the port C or E handler and check for what button was pressed and set the frequency on that. Since we were able to get this project to only purely on interrupts, we did not have to modify the frequency much so it would be the same as the one be outputted in the waveform, and just letting us just multiply the frequency of the timer by 1.18 in the sin wave and divided by 1.18 in the others.

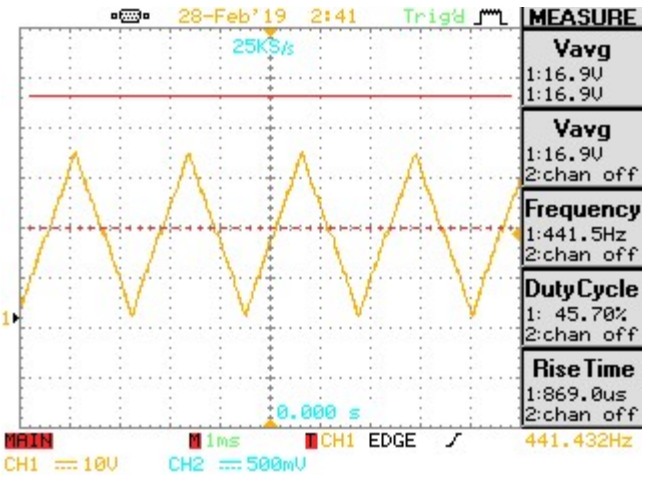
Software Flowchart



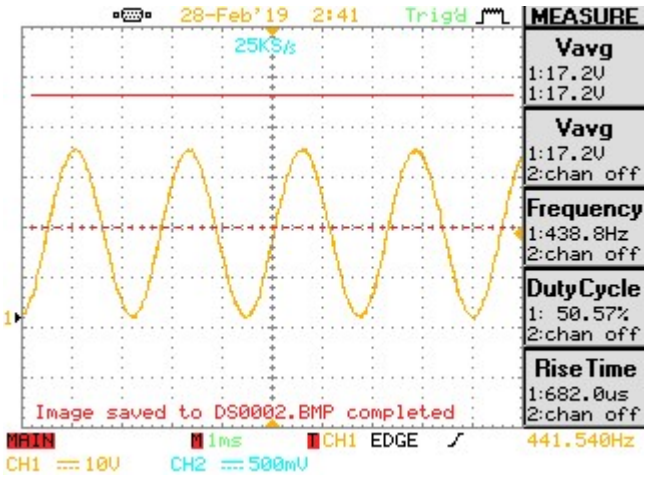


Waveforms

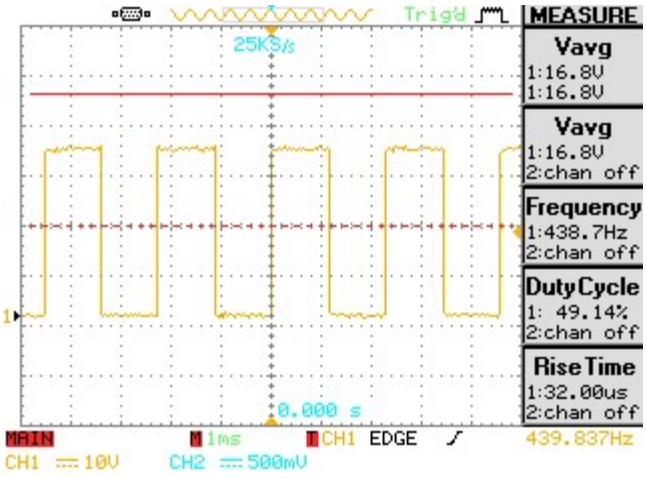
Triangle Wave



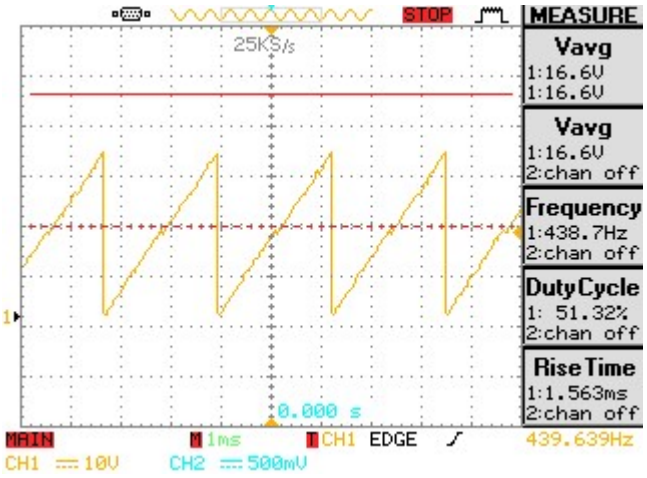
Sine Wave



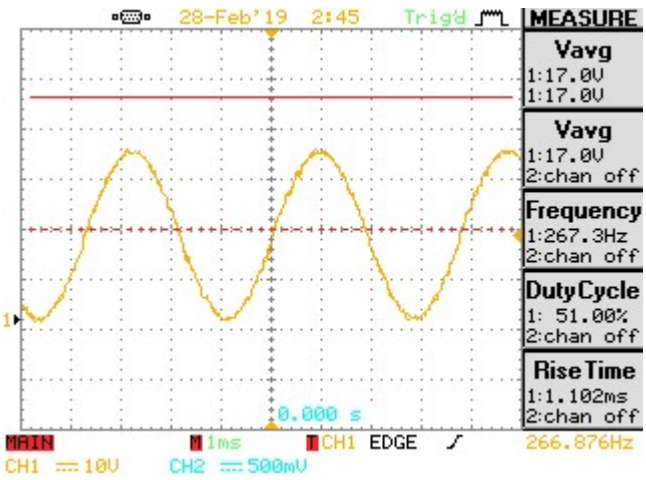
Square Wave



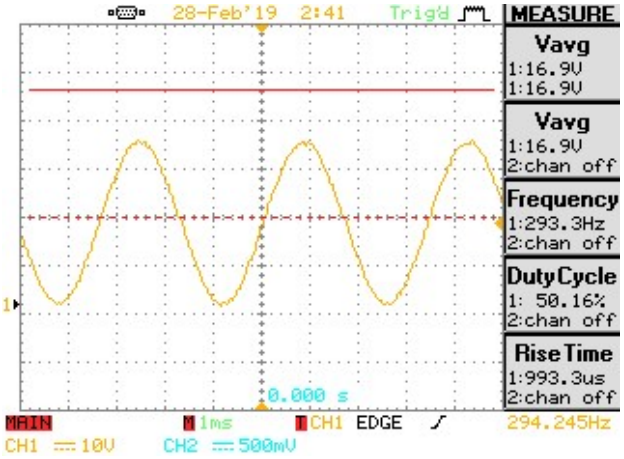
Sawtooth



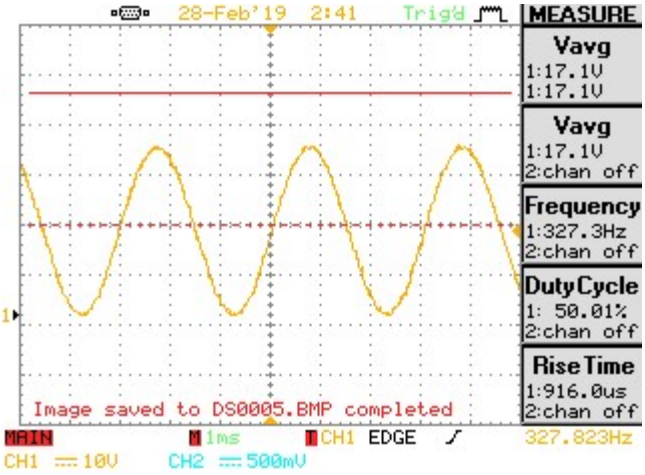
Note C – 260Hz



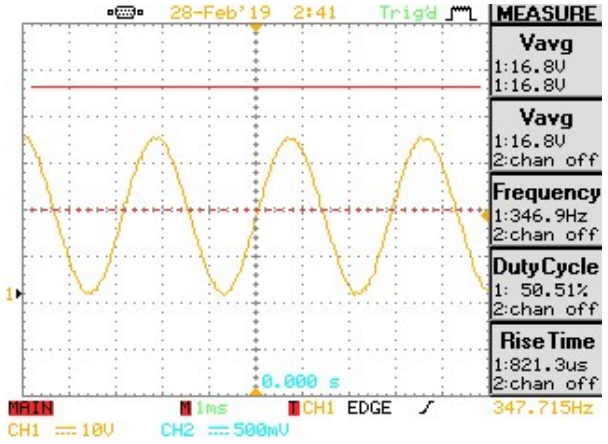
Note D – 294Hz



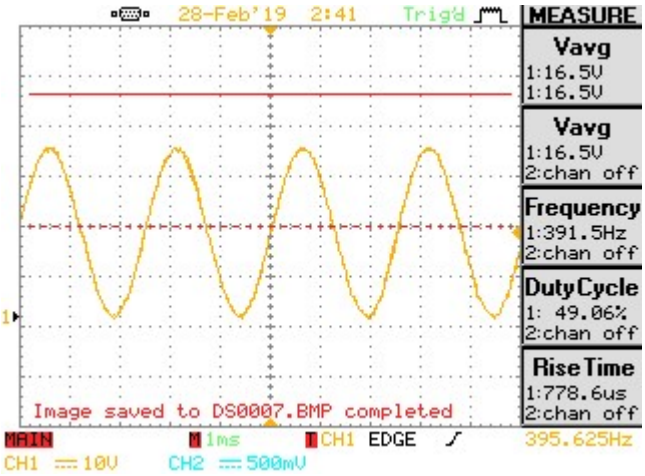
Note E – 330Hz



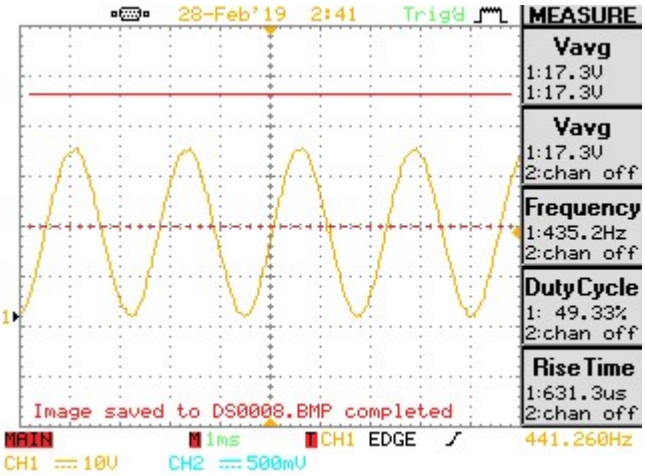
Note F - 349



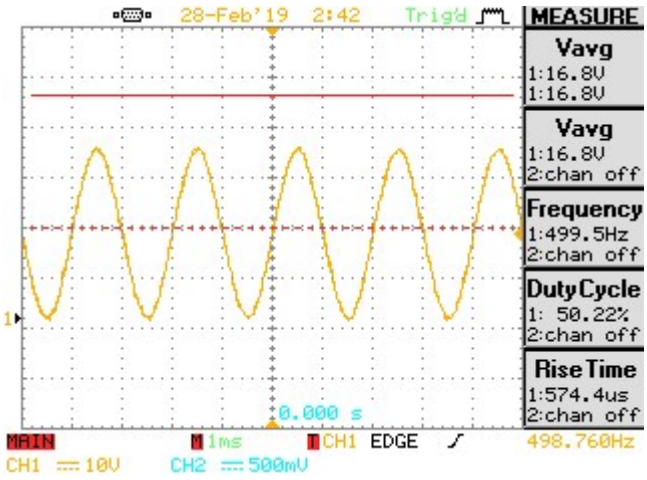
Note G - 392Hz



Note A – 440Hz



Note B – 494



Conclusion

In this project, we learned the basics of the Digital to Analog Conversion (DAC). During this process, we learned the effect of wave frequencies with the help of timers. The 8-bit weighted DAC we created helped in sending 256 different analog signals coming from a digital device that did not have a DAC built in. The different digital bits set at the input of the DAC created a variable voltage that we set for a specific amount of time which created wave frequency, then after repeating that we can transduce those frequencies to sound with the help of a speaker and an amplifier.