CECS 447 Microprocessor + Microcontrollers III

Fall 2019

Project 1: DAC – Tone Generator

Umar Khan

Ahmed Hayat

09/30/2019

Table of Contents

Table of Figures

Introduction

In this project, the goal was to create a tone generator using DAC (Digital to Analog Conversion) and TM4C123 microcontroller by Texas Instruments. The project incorporates the basic understanding of the ARM processor, GPIO, timers, interrupts, digital to analog conversion and analog to digital conversion. The required parts for the project was to create an 8 bit DAC using binary weighted or R2R logic. We used R2R logic for our 8-bit DAC. We created an excel sheet which included 256 rows and then tested the output values generated in our DAC with the expected values generated in the excel sheet for saw tooth, triangle, sine, and square waves. The frequency required for this project was 60 Hz.

Operation

Once the code was loaded on to the launch pad, we had to press reset to turn on the system. There were 5 required modes. We used the button PF0 to switch between the modes. When mode was set to 1, it would generate saw tooth wave. Similarly, when mode was set to 2, 3, 4, it would generate the waves for triangle, sine, and square wave respectively. We used the oscilloscope to check the output of the DAC in each mode and captured the image from the function generator.

When the modes were switched, the output of the different waves was heard using the speaker. Each mode had different sound as they all had the different waveform. Later when the tone generator was incorporated in the project, we used sine wave form to generate notes based on the ADC values. A potentiometer was used to adjust the ADC values and frequencies ranging from 262 Hz to 494 Hz.

Hardware

Hardware Block Diagram

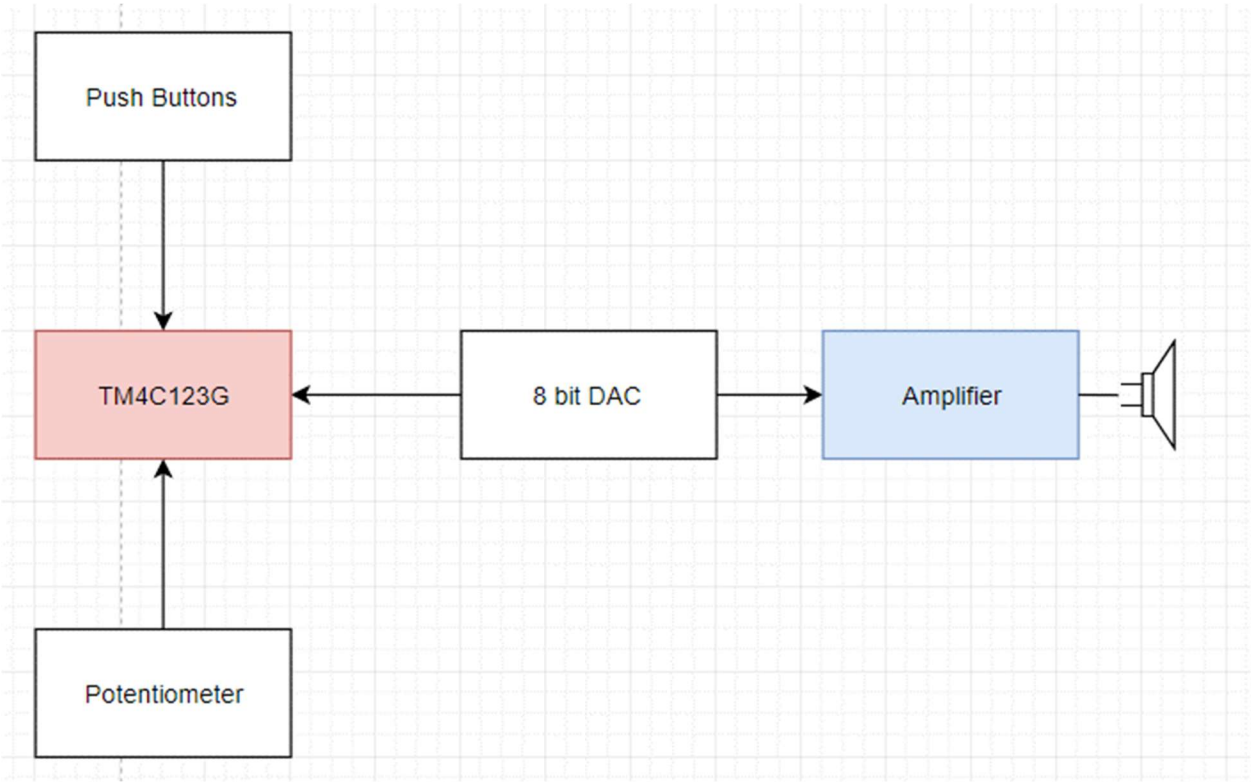

Figure 1 - Block Diagram

Schematics

8-Bit DAC

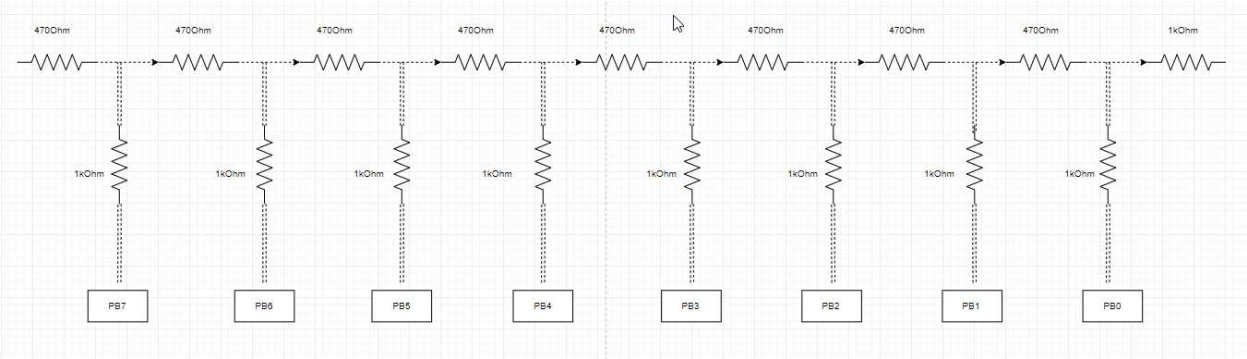

Figure 2 - 8-bit DAC

LM386

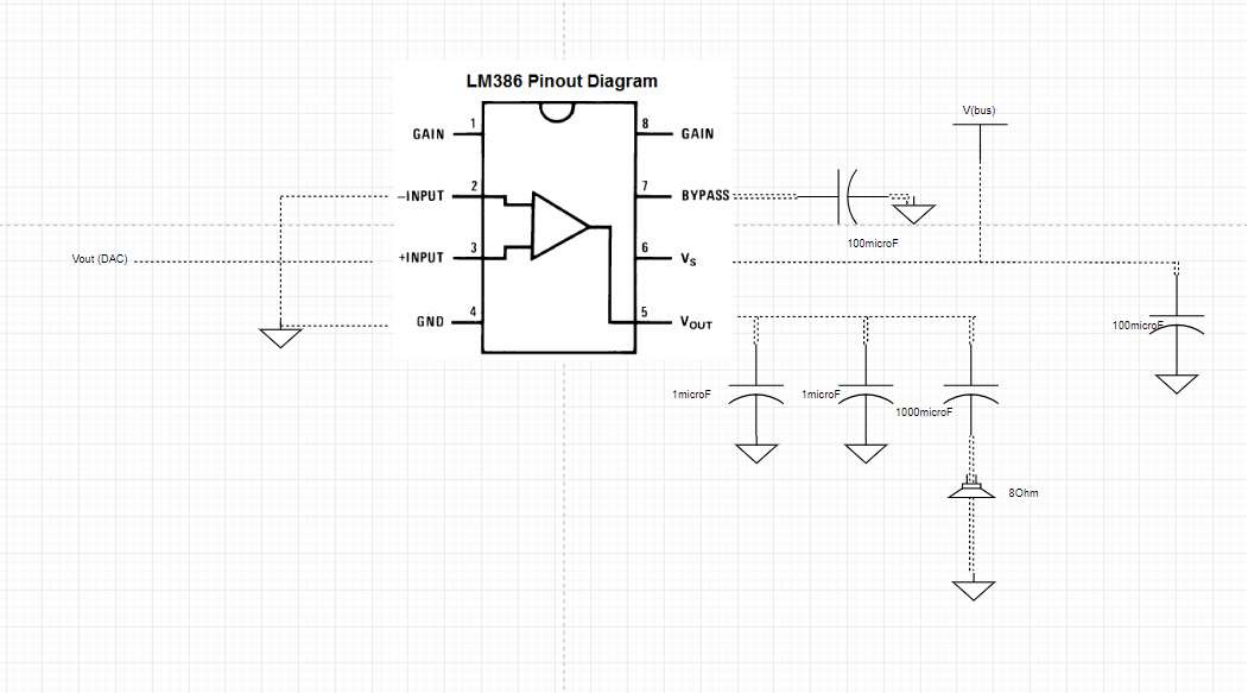

Figure 3 - LM386

Full Schematic

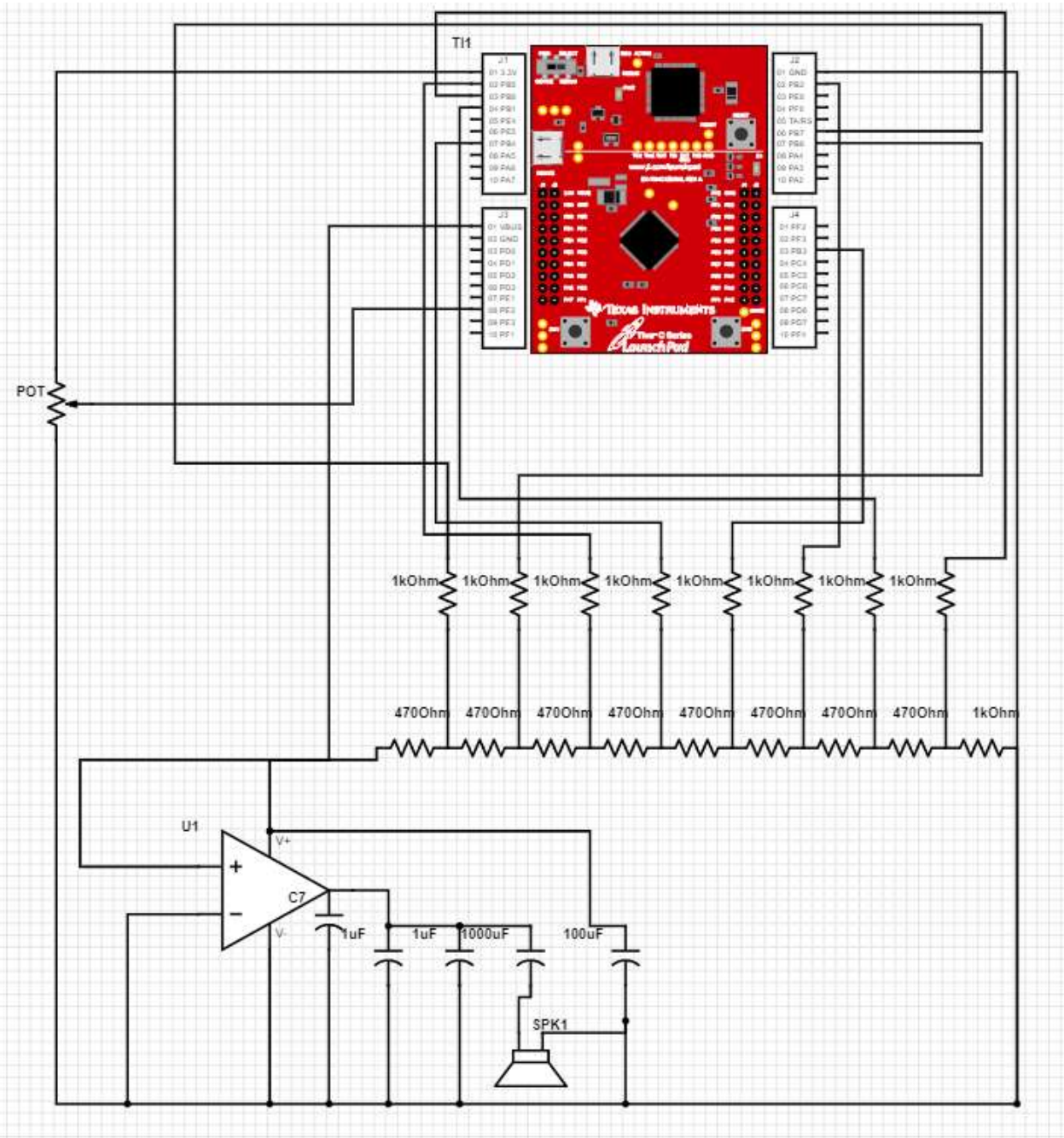

Figure 4 - Full Schematic

Oscilloscope Outputs

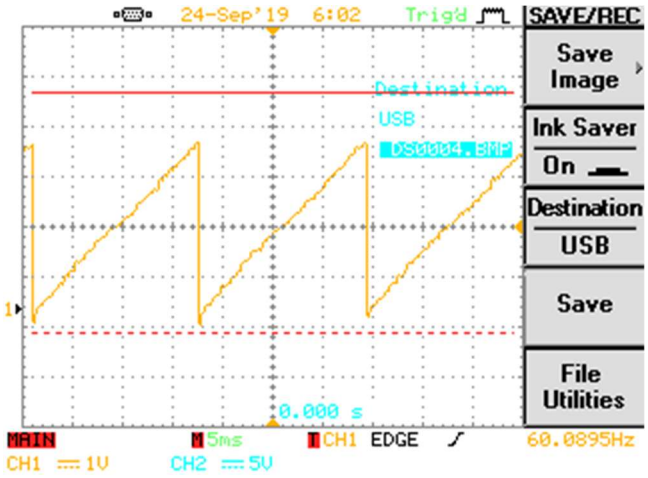

Figure 5 - Sawtooth Wave

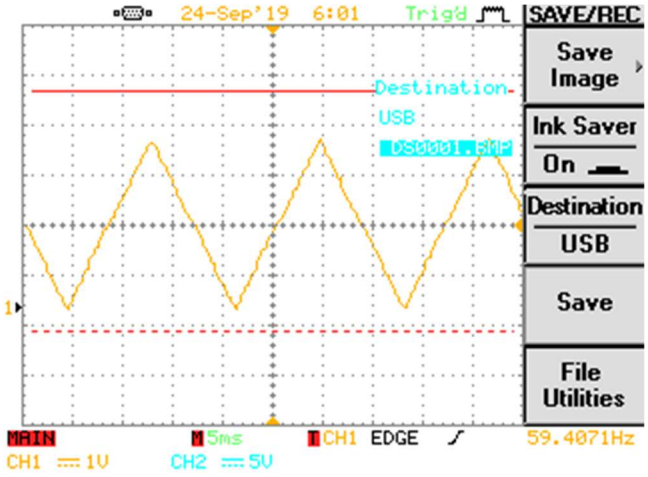

Figure 6 - Triangle Wave

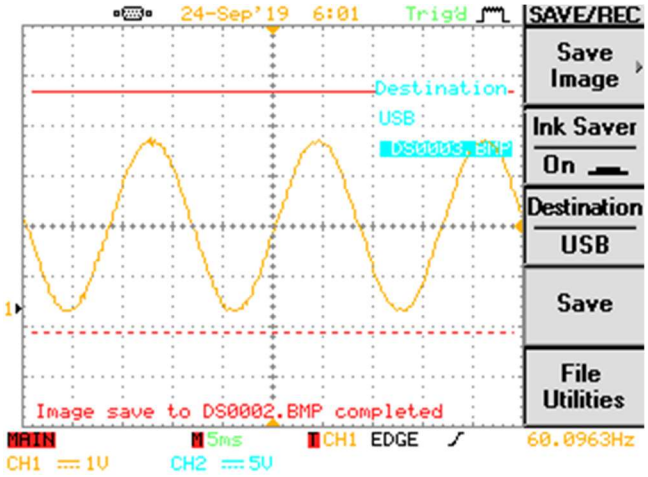

Figure 7 - Sine Wave

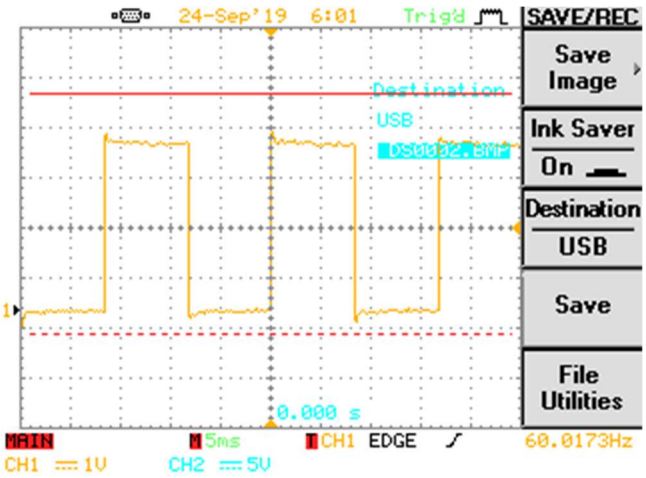

Figure 8 - Square Wave

Components

- TM4C123G (Texas Instrument Launch pad) – to run the program on a microcontroller
- LM386 – used as an amplifier
- Speaker
- Oscilloscope
- Resistors
- Capacitors
- Wires
- Potentiometer

Software

Software Approach

The software is designed to accomplish the following requirements of the project. Firstly, to generate a 60 Hz sawtooth, triangle, square and sine wave. Secondly, to generate a sine wave ranging from 262 Hz to 464 Hz. Thirdly, to use a switch to switch between the waveforms or modes.

At the beginning, we configure PortF, PortB, Systick, PLL, and ADC0. PortF is used for the on board Switch and LEDs. PortB (0-7) are used to connect to 8-bit DAC. Systick with interrupt mode is used to generate the timing of the system. PLL is set to generate 50 MHZ clock. Additionally, ADC0_InitSWTriggerSeq3_Ch1() is used to configure ADC0 and PortE (2) as an analog input which is connected to the potentiometer.

The software flowchart is shown in Fig 9, the main program starts by initializing PortF, PortB, Systick, PLL, ADC0 and sets the mode one. Then the program enters the super loop. GPIOPortF_Handler (void) toggles the mode when SW1 is pressed and it stays on the same mode till the switch pressed again.

Mode 1: To generate sawtooth the for loop counts up from 0 to 255, the value is sent PortB which is connected to the 8-bit DCA. The delay between the counts is set to 63 microseconds to generate 60 Hz sawtooth wave.

Mode 2: To generate triangle the for loop counts from 0 to 255 and counts down from 255 to 0. Each time count value is sent PortB, which is connected, to the 8-bit DCA. The delay between the counts is set to 11 microseconds to generate 60 Hz triangle wave.

Mode 3: To generate 60 Hz square wave, PortB is set to 0x00 for half of the period (8.33 ms) of the time and PortB is set to 0xFF the remaining half of the period(8.33ms).

Mode 4: The equation $\boldsymbol{y\ (x) = 1.65\sin(2\pi x/256) + 1.65}$ is used to generate 256 points using excel. These points are matched to the truth table of the 8-bit DAC and the corresponding values of the truth table are stored in SineWave array. Then the program loops through the array and outputs it to PortB. The delay between the outputs is set to 62 microseconds to generate 60 Hz sine wave.

Mode 5: Tone generator mode generates sine waves from 262Hz to 494Hz depending on the value of potentiometer connected at PE2. The function freqCal() reads the ADC0 and calculates the delay time

required to generate the corresponding sine wave. Accordingly, the program loops through the SineWave array and outputs it to PortB.

Software Flow Diagram

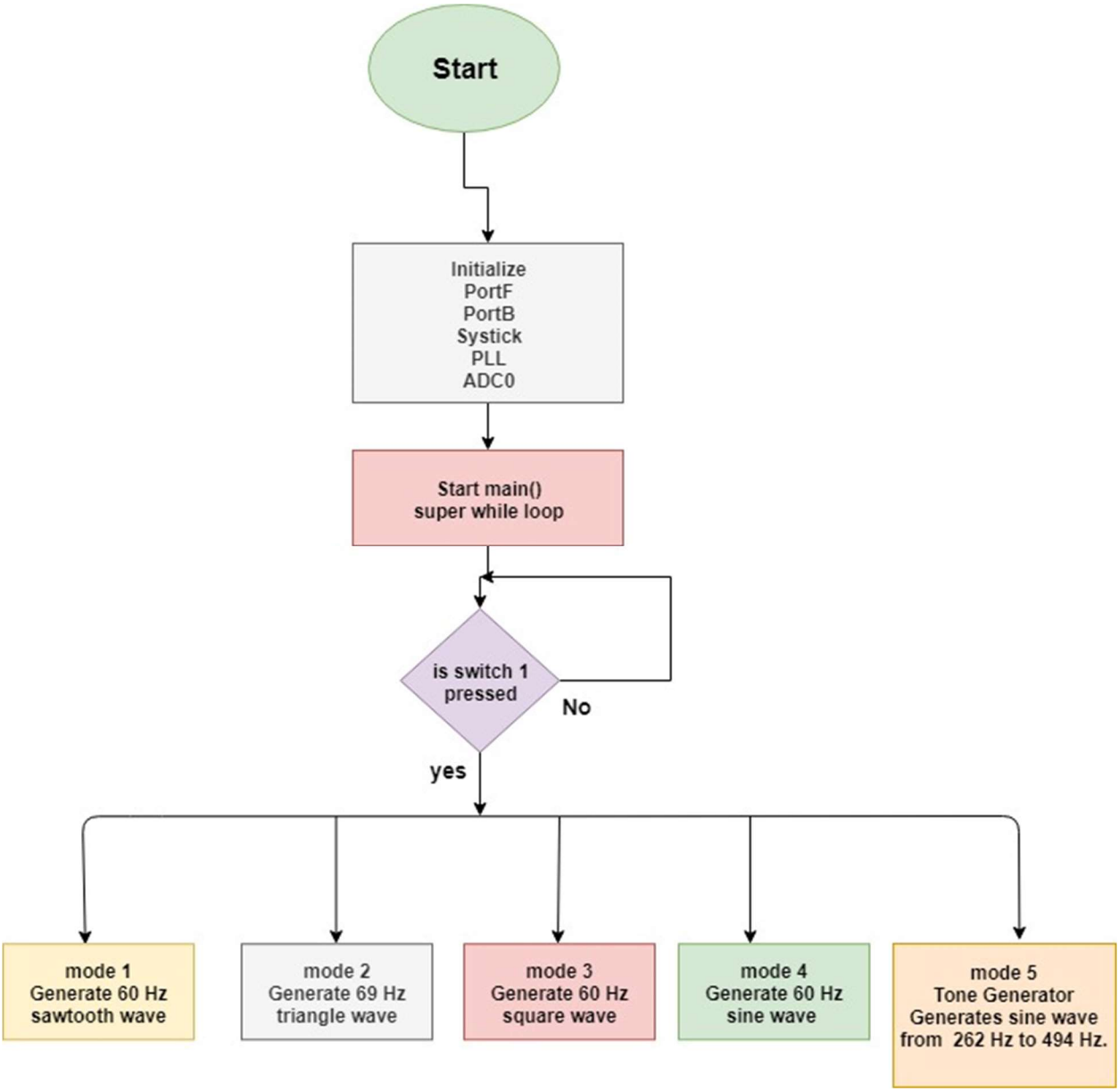

Figure 9 - Software Flow

Conclusion

We also ran to some issue with DAC output, which was resolved by making sure all the resistors are attached in correct manner to the breadboard. An issue we had with mode 5 was that the frequency incrementing and decrementing value was around 10 Hz. This was the result of the linear mapping. However, in the debugger we were able increment/decrement less than 10 Hz.

An attempt was made on the extra credit section. We were able to change the LEDs but it was only when we had to reset every time we were in the range of the required frequencies. We will be following up on that with John Yu.