## Physics 333, Lab 2

Goals:

Create and test your own PWM function, fold it into a function

inspect the PWM function with an oscilloscope

use a capacitor to average out the PWM signal

generate a periodic function with a PWM

use a PWM signal to drive an external component (with a transistor)

Many of these problems could be accomplished with the system-supplied “analogwrite()” routine, but it is better for you to learn how these things are made. Accordingly, please do not use analogwrite(). Code using the system-supplied analogwrite() routine **will not be considered correct**.

1. Based on the discussion in class, please write a program that generates a PWM signal that has a minimum delay (fundamental 1/0 unit) of ~10us (0.1 MHz), and a period of ~10ms. Implicitly, this is a duty level that can be specified to 1 part in 1000. Document the accuracy of your routine with a multimeter and an oscilloscope.
   1. Specifically, what is the finest resolution of average voltage you can specify between adjacent duty levels?
   2. Also, what are the on-off times in the oscilloscope output? How closely do they match the millis/micros delay times in your code?
2. Fold your PWM routine into a function. It should take some standard outputs that you define, eg, carrier period, duty level, duration of pulse, etc. Make sure the function works - it is a good way to clarify your thinking and code!
3. Use a capacitor and a load resistor, connected between the PWM lead and ground, to average out the PWM signal. Inspect the averaged and unaveraged signal on the oscilloscope. Is the capacitor’s effect uniform across duty levels? This works best if the capacitor is large.
4. Use your PWM routine to create a sawtooth function, with period = 2sec. Inspect the function with an oscilloscope w/ and w/o a capacitor to average out the signal. How accurately have you created a “perfect” sawtooth?
5. Use your PWM routine and a 3904 transistor (or similar) to drive an external load with your sawtooth. (A cpu fan is easy, but there are other options, eg a speaker at 20kHz...). Note, if you want to drive a fan, the Blum Arduino book has a reasonable recipe on pp 65-69. Please demonstrate this with a video on youtube or similar.
6. Use your PWM routine to create an analog soundwave, . Take A to be 2.5V and write your routine so that the frequency, is a user-specified argument to the function. Demonstrate this with an oscilloscope screenshot or video.