# ECE 4175
# Project Five
# Potentiometer Control

**Complete by:** Wednesday February 6th for an A+

**Reference:** Chapter 9

---

## Preamble

Make a note of the current draw from the last project. It probably ranged around 50 μA.

## Overview

This project is a variation of Project Four. However, for this project, you will generate a high-priority interrupt only when it is time to take a step. You will change the loop time to be 10 ms. You will check the one-turn potentiometer every second with a **Pot** function called each time around the main loop. Each second, convert the eight-bit pot output, **ADRESH**, to a sixteen-valued (i.e., 1-16) variable **RATEINC** to be added to **RATE**. With a new value of **RATE** ranging between 1 and 800 steps/second, display this value and also use it to form **T3CHANGE**, the number of counts to be added to **TMR3H:TMR3L** at each interrupt so that the next overflow will produce the required step period. When **RATE** exceeds 800 steps/second, start over at 1 step/second.

If the pushbutton is pressed, continue to step but do not change the step rate. You will use this to pause the changing step rate while you measure that rate with the scope.

Blink the Alive LED every second.

Remove the setting, clearing, and toggling of output pins associated with earlier projects. Now toggle **RB0** at each step, to check the step rate (which will be double the frequency of the output on **RB0**).

## Pot Function

Use the following code from Figure 9-6 of the text to read the potentiometer.

```
ADCON1 = 0x0b;               // Initialize ADC for Q&L's four analog inputs
ADCON2 = 0x09;               // ADC clock with Fosc = 4 MHz; left justify result
PORTAbits.RA7 = 1;           // Power up potentiometer (Figure 3-2)
ADCON0 = 0x03;               // Power up ADC; select pot input; start conversion
while (ADCON0bits.GO_DONE);  // Wait for completion of conversion
PORTAbits.RA7 = 0;           // Power down potentiometer
ADCON0bits.ADON = 0;         // Power down ADC
                             // Read value from high byte, ADRESH
```