CS466 Lab 6 – Quadrature Decoding.
Complete By Thursday 4/10/2020.

This lab has no report but is instead the beginning for the final lab (still called #7) It's important that you get it working before moving on to #7. Start with a minimal single task template with serial working and interrupt handlers.

The motors that we will be using have incremental rotary encoders attached to them. (https://en.wikipedia.org/wiki/Rotary_encoder) When the encoder LED is powered properly 'phase-a' and 'phase-b' provide quadrature information. From the pa and pb signals you will need to keep track of motor position and velocity.

Note: You are free to work in teams of 2 in this lab. If you work in a team I only require a single report handed in.

1. ☐ Take a look at the DC motors
   a) Looking at the encoder side of the motor.. left to right. This may not be the pinout, It needs to be confirmed as the various motors use various pinouts.
      1. motor a
      2. motor b
      3. gnd
      4. +3.3v
      5. a
      6. b

   The motor-a and motor-b lines are used to apply a DC voltage across the motor. They will not be used in this lab but you need to be able to tell what is what. Using a voltmeter you can easily identify the pins for A and B as they are directly hooked to the back of the motor.

2. ☐ On your Tiva board Setup PortD pins 0 and 1 as input to receive the quadrature A and B signals respectively.

3. ☐ Setup an interrupt handler to receive edge transitions of A and B

4. ☐ In your ISR track the position of the motor based on the encoder interrupts. In your idle task, dump the position of the motor in encoder ticks.

5. ☐ Have your heartbeat display the position of the motor in encoder ticks to the serial port, say once every 500ms.

6. ☐ Use a free running timer (TIMER_0) to calculate the speed of the motor in RPM and display average speed over your report cycle with the end motor position. The Tiva Driver Library timer example (…/TiveDriver/examples/timers/timers.c) has example setups but don't generate interrupts, You're going to access a free running counter in your existing ISR to determine the time between encoder tics. The Driver Library API has another funciton, TimerValueGet() to query a timer value.

7. ☐ Add the RPM to your display.. Average the speed over the 500ms sample.