CS466 Lab 6 – Quadrature Decoding.
This is the first part of the final lab and should not take longer than a week.
Be sure to complete this lab before starting lab7

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 pico board setup two pins GPIO6 (pico-9) and GPIO7 (pico10) 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. ☐ You are free to apply up to 12 Volts on the motor but simply move it by hand (slowly) until you have the tic counts working.

7. ☐ Use a free running timer to calculate the speed of the motor in RPM and display average speed over your report cycle with the end motor position. We are working on a method to geta timer running at near the system clock speed.

   The clock speed that we are running the Pico at is 125MHZ. Some simple math shows that this is a timer-tick every 8ns.. (pretty accurate). Rumor is (I have not tested this) that you have a 64 bit access to the most accurate value on the Pico

```
// Some hardware timer code that I grabbed off the
// internet.   May not even work, you will need to test
// if the timer works with some reliability before
// using it,

#include "hardware/timer.h"

// Read lower 32 bits directly (TIMERAWL register)
uint32_t low = timer_hw->timerawl;  // 32 bit reg
// Read upper 32 bits (TIMERAWH register)
uint32_t high = timer_hw->timerawh;  // 32 bit reg
// Combine into 64-bit value
uint64_t raw_ticks = ((uint64_t)high << 32) | low;
```

Be careful that this timer can roll over. Calculate how often this timer will roll over and you will need to design a mitigation to account for it. Allowing a discontinuity like this into the feedback system that you are designing in lab 7 will add digital noise and make your system less stable.

Also remember as I said in class that each transition on the encoder has to cause 4 transitions on the very cheap quadrature encoder. They cannot be assumed to be perfectly 90 degrees out of phase.

Your ISR should keep the tick-to-tick time encountered in a circular array that continuously is updated every ISR invocation… Leave the RPM calculation to run in task mode. When called from your heartbeat it can simply sum the circular array then divide by the array size to get an average tick-to-tick time over the ARRAY_SIZE number of samples. Because the system and sensor is based on detecting the timing of 4 edges, I strongly recommend that you make the ARRAY_SIZE a multiple of 4.

How should you deal with data sharing? Do you need to use the full 64 bits?

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