CS466 Lab 7 – Motor Servo.
Final lab, Due Midnight, May 3

Notes:

- This lab is a variation of the normal Lab7 but slightly simplified because we cannot meet with lab partners our use our lab equipment. As a result the lab can be completed with a single motor, Tiva board and
- This lab requires the use of an H-Bridge motor Driver we have the boards in the lab.
- This can still be performed as teams of two but I 'Strongly' advise that both students get the lab working.. It has been my experience that allowing an asymmetric commitment in a lab team leads to asymmetric test scores at finals time..
- Mount some kind of mechanical paddle on the motor so you can see position. It may spin fast, I recommend something that will not become too unbalanced when spinning. If both you and your lab partners are using the same code try to use similar paddles. It also helps if a motor load has a little mass. I have used something like a 1" portion of a popsicle stick that is easy to see and have some mass. Be sure to firmly attach it to the motor shaft. Otherwise you have a much more difficult control system and it will never converge with a simple PWM.

Objective:

- Integrate material we have learned in past labs.
- Use a PWM to control torque in a DC motor.
- Learn how software PID control can be used to implement motor servos.

General Description:

Operator will position motor paddle in a vertical position and turn the power on to establish the home position. Pressing S1 will energize the motor and move the paddle position to the first setting (1000). The Red LED will blink while the motor is seeking it's target position then turn off if the position is reached. Physically moving the motor out of position will cause the red led to start flashing again and the encoder will re-seek the set position. Pressing S1 will set the next element in the list, Pressing S2 will seek the prior position. Pressing both S1 and S2 concurrently will disable the servo and de-energize the motor.

Requirements:

- On the Tiva board you will implement a remedial PID control algorithm to seek motor position given from a simple master.
- Our motor has the notion of encoder tics, 4x actual encoder tics.. In this lab I will refer to the 'position' of the motor. This position is the number of tics, negative or positive, from the power-on zero position. In a larger system we would have some means to 'home' the motor to a mechanical zero that makes sense, we're skipping that.
- Store in the Tiva software a list of positions that will be sent to the servo code. In general you set the position of the servo desired and the motor servo will work to move the motor to that position.
- Position List = [1000, 2000, 3000, -5000, 10000, -1000, 0]
- The buttons on the controller should execute the following.
    - SW1 – Start/Next
        - Initially you will set the motor to a known 'zero' position and press start.
        - Once started illuminate the blue LED.
        - Subsequent presses will seek the positions stored in the actual list.
        - If the end of the list is reached continue to the start of the list.
    - SW2 – Prior
        - Nothing if SW1 not pressed first.

        - Sequence in the list in the opposite manner as SW1, moving backwards in the list
        - If decrementing off the beginning of the list continue from the last element.
    - SW1+SW2 – Motor Off
        - If both buttons are presses for 500ms put the motor in an un-powered state and turn on master red LED.
    - LED-Green
        - Strobe a 10% duty cycle heartbeat signal at 1Hz. (100ms on, 900ms off)
    - LED-Red
        - Solid when the system is idle/de-energized and ready for a start button.
        - Rapid-Blink on any SW1 press and remain blinking until the motor position has settled within 20 encoder ticks of setting
        - Off indicates that the motor has settled at it's destination and ready for a position switch.

- The Motors will be driven by 12VDC. Direction/Torque will be given by imposing a PWM and Direction system.
- The firmware watchdog should be running on both master and slave and if encountered shut down the motors. If the master hangs it should reset and initialize the remote SPI registers with a power off indication.

Prior To Lab:

- Take time to read and understand the requirements and what the system will do.
- Review the PID Controller Wiki https://en.wikipedia.org/wiki/PID_controller
- Review the Servomechanism Wiki https://en.wikipedia.org/wiki/Servomechanism
- Review the L298N Board Info Document (L298N_Board_Info.pdf)
- Because we are short on time I have provided motor and PWM code. Look first at the header files to see what each module exposes. Spend enough time to see what the motor.c and pwm.c do. Note that your application need only include motor.h.
- You will need to modify the files and/or your connections to match modules with your confiugurations.

How I would progress,

1. ☐ I have provided a motor.c/motor.h API that uses another provided PWM API. You will need to drive the integer 'static int32_t _encoder' with the quadrature encoder you constructed in Lab 6 . If you want to used one of my encoders instead you can define them back in the module.
   a) Question: How close did your Lab6 encoder match one of mine provided?

2. ☐ Allocate additional GPIO's off your master and slave for connection to the driver board. You will need two controllable GPIO's for position and the single line output for the PWM signal

3. ☐ Verify that all your GPIO's and PWM signal can operate independent of each other.
   a) This has been problematic in the past. Make sure somehow that all your signals are producing your expected output.

4. ☐ Using a simple loop in your Slave board that sequences through 2 or three motor positions once per 5 seconds or so until you get the PID tuned. I did not provide valid PID tuning parameters.

I will be updating this lab direction some. This is not a final revision. I will try not to change the above steps any so as not to cause rework.

I would like to use the servo to do something interesting... Let me know if you have any ideas..

Possibilities:

- Metronome
- Drummer
- Clock Hand
- Looking for better ideas...