

CMPE2150 Project 03

AJ Armstrong

27 Nov 2024

Table of contents

Basic Feedback Sensing	1
Proportional Control	2

Basic Feedback Sensing

Use the circuit and start with the software project you were working with in the associated Self Assessment. In case you need the PWM specifications again, here they are:

Your initial program should have the following characteristics:

- Set up the 16-bit PWM channel (Channel67) to drive the Enable pin, with the following characteristics:
 - Channels 6 and 7 concatenated into a single 16-bit PWM channel, available on pin 109
 - The period is 10,000 clock cycles to allow for an easy conversion between percent duty cycle and the required PWMDTY value
 - The frequency is $500Hz$ (Hint: to get this speed, you'll probably need to set the prescaler exponent to 0)
- Pressing the “Left” pushbutton will drive the motor in reverse (clockwise as seen when looking down the shaft, not the encoder)
- Pressing the “Right” pushbutton will drive the motor forward (counterclockwise)

Make the following adaptations to the code:

- Start with the duty cycle set to 40%.
- Pressing the “Up” button will speed up the motor
- Pressing the “Down” button will slow down the motor

- The duty cycle should never go outside of the bounds of 0% and 100%
- Write an Interrupt Service Routine that responds to a falling edge of Timer Channel 0 (i.e. Encoder C_1), set up for Input Capture. This ISR should do the following:
 - Return the current Timer Capture value to the main program for use in determining the frequency of pulses from the Encoder.
 - Check to see if Encoder C_2 is already LOW, which would indicate reverse motion; either way, a flag should be set for the main program to interpret as forward or reverse motion.
 - Indicate to the main program through the use of a flag that the ISR has been run.
 - Clear the Timer flag.
- In the main program, calculate the gearhead output shaft's rotational velocity in RPM and display this on the top line of the Seven Segment Display.
- In the main program, write "0000" to the bottom line of the Seven Segment Display when the motor is turning forward and "9999" when the motor is turning in reverse.

Ask your instructor to grade your work out of three marks up to this point, and record the grade assigned below. If your instructor isn't available, attach a picture of the Seven Segment Displays while the motor is running in reverse. _____

Proportional Control

Once you have the basic sensing and display functionality described above working properly, continue with the following proportional speed control code, which should attempt to maintain a constant rotational velocity even when a load is applied to the main shaft:

- When the middle pushbutton switch is pressed, store the current frequency of pulses from the Encoder as the Set Point.
- Calculate the Error Signal between subsequent readings of the pulse frequency and the stored Set Point.
- Adjust the PWM duty cycle percentage value to match the calculated Error Signal. (In other words, multiply the Error Signal by 100, the way you would to set the PWM duty cycle to a particular percent, since the period is 10,000.)

- Your duty cycle should never exceed 100%.

To test your control system, observe the waveform generated by the Pulse Width Modulator.

- Unloaded, the duty cycle should be approximately 40%.
- Apply a load to the shaft by squeezing it with your fingers to try to stop it. (Alternatively, you could gently hold your finger against the rotating magnetic encoder wheel). As you load the motor, the duty cycle should increase, but the rotational velocity should remain essentially constant (as long as the duty cycle is less than 100%).
- Apply sufficient load that the duty cycle reaches 100%, and verify that the rotational velocity decreases for loads beyond the normal range.

Once your system is working satisfactorily, ask your instructor to grade it out of two marks, and record the grade assigned here. If your instructor isn't available, attach a picture of the PWM output signal when the motor is loaded heavily, but not beyond the normal range.
