Lab 5 – Part 2
Final Project: Motor Speed Control

Introduction and Objectives

In this lab you will control the speed of a motor. Figure 1 shows the hardware setup, which is the same as for Week 1 of Lab 4. You will use the potentiometer on your evaluation board to set the desired speed of the motor, and you will control the speed through the PWM output of the HCS12. You will measure the speed of the motor using an input capture pin, and display the desired and actual speeds on the terminal..

1. The Lab

It is up to you how you design the system to accomplish the goal of this lab, nonetheless, here are some guidelines to assist you in ensuring proper operation of the system.

1. Build the circuit shown in Figure 1.

2. Design a real-time routine that gets executed every 8ms. Develop a method to verify the timing of that routine, e.g., increment LEDs.

3. Program the A/D converter to read the value from the pot either the one on the microcontroller board or an external one. In the routine developed in Part 1, read the A/D converter (use 8-bit mode). Again develop a method to display the results and verify the operation of the A/D converter as you change the input voltage.

4. Set up the PWM to generate a 50 kHz PWM signal on one of the four PWM channels. Set it up for high polarity. It will be easiest to set PWPERx to 255. Verify that the PWM works. In the real-time routine, write the eight most significant bits to the A/D value you read to PWDTYx. The motor speed should change as you use the pot to vary the voltage on the A/D.

5. Measure the speed of the motor by determining the time between two falling edges of the optical encoder. In your main program display this time on the LCD display. You can use floating point arithmetic to convert this time into RPM. Display the RPM value on the LCD display. What is the maximum motor speed?

6. Measure the speed for several different duty cycles by varying the voltage with the pot. Plot speed vs. duty cycle.

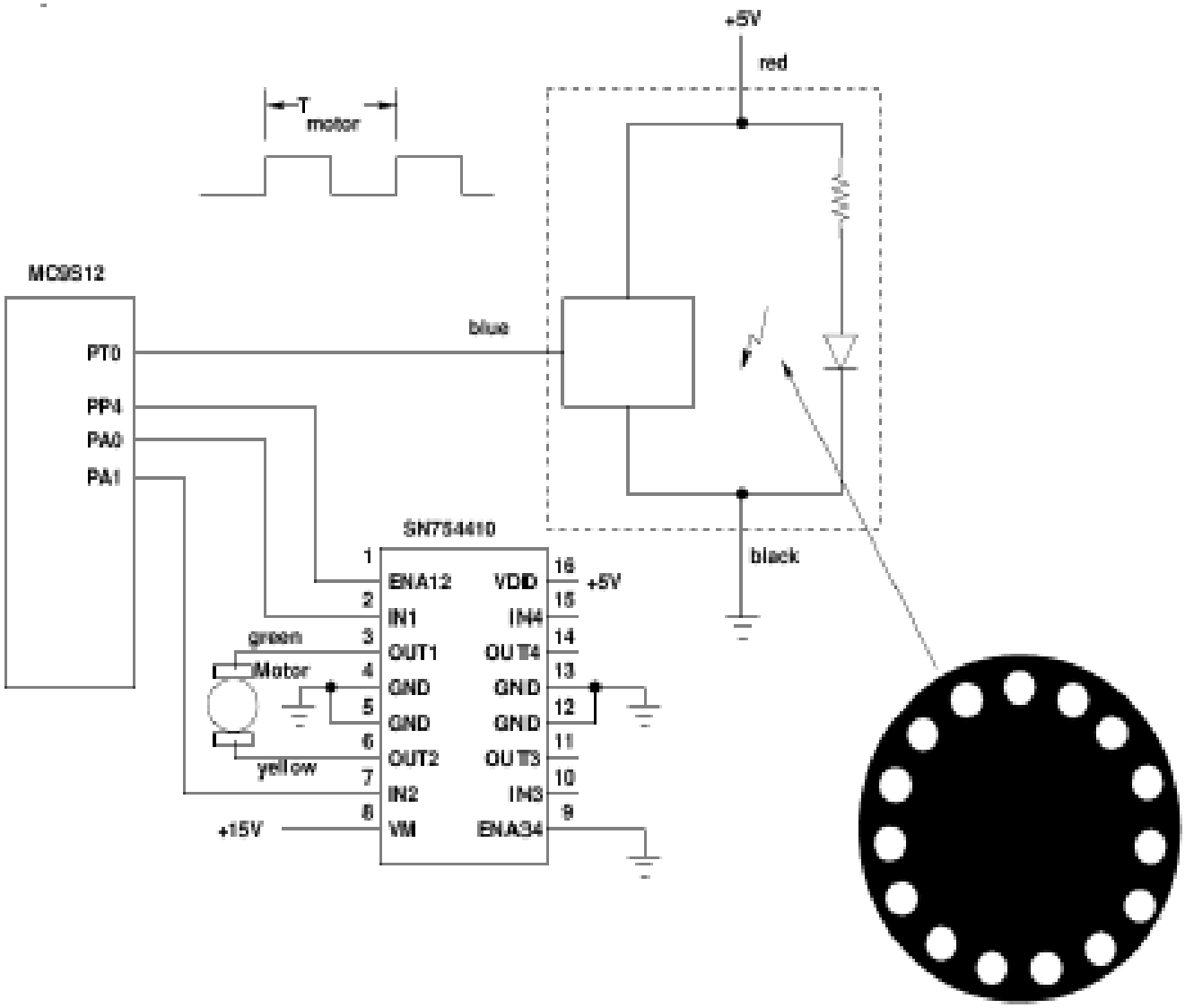

Figure 1: Using a PWM signal to adjust the speed of a motor.

7. Implement closed-loop speed control. The desired speed Sd should be

$$Sd = (0.2 + 0.8 \cdot (AD/ADmax)) \cdot Smax$$

where Smax is the motor speed at 100% duty cycle, AD is the A/D converter reading and ADmax is the maximum A/D converter reading. In this way you will be able to vary the speed between 20% and 100% of Smax.

To set the motor at the desired speed you can use a simple equation (integral control) such as:

$$DCnew = DCold + k \cdot (Sd - Sm)$$

where Sm is the measured speed. Do this calculation inside the real-time routine, and write the new value to PWDTYx. Try different values of k to see how the motor responds. If k is too small, it will take a long time for the motor to get to its steady-state speed. If k is too large, the motor will be jerky as it tries to settle down to its steady-state speed.
It will be much easier to do these calculations using floating point numbers rather than using integers.

8. Set the power voltage to 15V. Measure the motor speed for various values of input voltage to the A/D converter. Take about 10 equally-spaced measurements for input voltage between 0 and 5 V. Use the LCD display to show the raw A/D value and the raw counts between edges on the first line, and show the desired and actual speeds on the second line.

9. With the pot set at about mid-range, vary the voltage of the voltage powering the motor (say between 8V and 14V). With closed-loop control the speed of the motor should stay the same. Verify that this is the case.

10. Using the data from Part 8, plot the speed in RPM vs. the input voltage from the pot ,i.e., convert the speed measurement in time difference between two falling edges to speed in RPM.

11. It is much more effective if you have the data from the previous part recorded automatically, this way you can observe the behavior of the controller and how long it takes to make the motor settle at the right speed. To do that change the BAUD rate to 115,200 then once every 8ms send the input capture difference to the serial port. Set Hyperterm to use 115,200 baud rate, capture the serial data and plot in MATLAB. Set the power voltage back to 15V. Rather than varying the PWM based on the pot, set it manually inside your code for a while and then change it to different value, this will create a step change in the desired set value, and can be used to determine the effectiveness of the controller.

12. Another type of controller that may be used is known as proportional controller. This type of controller, and unlike the integral type control, only uses the current measurements to set the output rather than accumulating any history. This is accomplished by

$$DCnew = k \cdot (Sd - Sm)$$

Similar to the previous step, collect the data due to a step change in the desired speed, and plot in MATLAB. Compare this proportional controller to the integral controller.