CMPE 121L: Microprocessor System Design Lab
Spring 2013
Lab Exercise 4: DC Motor Control

Due: Week of May 20, 2013

In this exercise, you will learn how to control DC motors with the PSoC-5 microcontroller, starting with the open-loop control of a brushed DC motor, ending up with a closed loop PID controller for a 4-wire fan.

Before you start the design, collect the following documents and review them.

1. Documentation for the DC motor in the parts kit: http://www.sparkfun.com/datasheets/Robotics/hobbymotor.JPG
2. H-Bridge reference design: www.robotroom.com/BipolarHBridge.html
3. PWM control of brushed DC motors: http://www.robotroom.com/PWM5.html
4. Components for H-Bridge
    a. NPN transistor, 2N2222: http://www.fairchildsemi.com/ds/PN/PN2222.pdf
    b. PNP transistor, 2N3906: http://www.fairchildsemi.com/ds/2N/2N3906.pdf
    c. Schottky diode for protection: http://www.vishay.com/docs/88525/88525.pdf
5. Documentation for the 4-wire fan in the parts kit: http://media.digikey.com/PDF/Data%20Sheets/NMB-MAT/1611FT.pdf
6. 4-wire fan interface specification: http://www.formfactors.org/developer/specs/4_Wire_PWM_Spec.pdf
7. PID controller tutorial: http://en.wikipedia.org/wiki/PID_controller
8. Rick Bickle's DC motor control tutorial: http://www.dprg.org/tutorials/2003-10a/motorcontrol.pdf
9. PID example code from Arduino blogs: http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

Part 1: Open-loop Control of Brushed DC Motor Using H-Bridge

1. Assemble the H-Bridge circuit on the breadboard using components from the parts kit. No soldering is needed. You can use the same 5V supply of the PSoC-5 to power the H-bridge. Pay attention to the orientation of the circuit components, especially the transistors.
2. Connect the four control points of the H-bridge to PSoC-5 outputs.
3. Using the schematic editor of the PSoC Creator, design a logic circuit to generate the four control outputs for the H-bridge. This circuit should have only two inputs with three control combinations (00 or 11 = stop, 01 = forward, 10 = reverse), and should drive the corresponding valid combinations on the four control points of the H-bridge. The purpose of this circuit is to prevent the four control points of the H-bridge from being set to illegal combinations that could damage the transistors (Hint: you might be able to make use of the LUT block in PSoC Creator for this logic.).
4. Verify the operation of the circuit with a test program. Then connect the H-bridge to the PSoC-5 and verify its operation.

5. **5.** Test the three functions of the controller (running the motor in both directions, and stopping it) using a program.
6. **6.** Add a third *speed control* input to your circuit. This input should accept a PWM control signal and pass it on to the H-bridge based on the setting of the other two inputs. That is, if the other two control inputs are set to 01 (forward), the PWM signal should be passed on to the top right and bottom left transistors in the H-bridge. If the other two control inputs are set to 10 (reverse), the PWM signal should be passed on to the top left and bottom right transistors in the H-bridge
7. **7.** Using the PWM block in PSoC Creator, generate 1 kHz PWM signal and test the operation of the circuit. Use the potentiometer to vary the duty cycle of the PWM block and display the duty cycle on the LCD display. Start the motor at 100% duty cycle, then reduce the duty cycle slowly until it stops. Note the the minimum duty cycle at which the motor continues running, for both the forward and reverse directions.
8. **8.** Try to start the motor from the rest position with less than 100% duty cycle. Determine the minimum duty cycle at which the motor is able to start, for both the forward and reverse directions.
9. **9.** Repeat steps 7 and 8 with a 20 kHz PWM signal.

Part 2: Closed-loop Control of the 4-Wire Fan

1. **1.** Test the operation of the 4-wire fan. Connect the power and ground wires to a 12V power supply. Using PSoC Creator, Generate a 25 kHz PWM signal to drive the PWM control input of the fan. You should be able to change the PWM duty cycle using the potentiometer. Display the duty cycle of the PWM block on the LCD display. Connect the tachometer output of the fan to one of the PSoC-5 inputs. Using the timer and counter blocks in PSoC-5, design a circuit to count the number of pulses on this input over 1-second intervals (note that there will be two pulses during each rotation of the fan). Display the measured speed in RPM (rotations per minute) on the LCD display. The operating range of the fan is approximately from 5,000 RPM to 22,000 RPM. Vary the duty cycle from 20% to 100% and plot a curve showing the speed of the fan with respect to the duty cycle.
2. **2.** Design a controller program that continually measures the speed of the fan and adjusts the duty cycle of the PWM control signal to maintain it at a desired set-point. When this program runs, the display should show two speed values: (i) the set-point, which is the desired speed, and (ii) the measured speed. The desired speed must be selectable over the range of 5,000 to 22,000 RPM using the potentiometer. The controller program should continually sample the speed of the fan, determine the error (difference between the measured speed and the set-point), and use this error to adjust the PWM duty cycle.

   Design a simple PID controller to set the PWM duty cycle based on the error input. The sample code for this is available in the Arduino example referenced above. You can enhance it if you have time, as described in the blog.

Change the three constants in the program (Proportional, Integral, Derivative), and observe how the measured speed changes in response to a change in the potentiometer setting.

3. Set the desired speed to 10,000 RPM and wait for the fan to reach steady speed. Then change the potentiometer setting to 20,000 RPM and observe how long it takes for the fan speed to settle to within 5% of the desired speed. Note this time for the cases of (i) only proportional control, no integral and derivative control, (ii) with all three control components. Tune the constants as needed. You don't have to come up with the "best" possible values for the three constants, the goal is only to study how the dynamics of the system change with the controller parameters.

4. Using the parameters of the controller selected in Step 3, change the speed from 20,000 to 10,000 RPM and observe how long it takes for the fan speed to settle to within 5% of the desired speed. Note this time for the cases of (i) only proportional control, no integral and derivative control, (ii) with all three control components. Are these different from the times measured in step 3?

What to submit?

- Schematics and code for both parts
- Descriptions of your designs.
- Results and graphs as described above.
- You should also demonstrate the working design for both parts to your TA.