Feedback Control of Brushed DC Motor

Mechatronics Project Lab 2 (Covid extra)

Jonathan Scott, March 2020

1 Introduction

This lab involves designing a circuit, constructing it, and programming an embedded microcontroller to implement a feedback control loop setting some aspect of the system, by driving a small DC motor. This project has been designed so that it can be done individually by students working in isolation using borrowed equipment in response to the virus alert that is in effect.

The project is nominally due at the end of the first half of semester, but can be delayed so you may overflow into the time in the mid-semester recess.

2 The Hardware

Imagine that your company needs to build respirators, fast. Each respirator needs to have an air pump whose pressure is reasonably set by the rotation speed of the motor that drives it. Your part in this exercise is to put together a small system, using what is already available, that will set and regulate the speed of the shaft of a motor, and to present a working circuit and a set of plans that can be handed to a bunch of technicians who will turn out lots of these for assembly into makeshift machines.

The motor is a nominal 12V, brushed, permanent-magnet type. Your job is to design, build, and program a small microcontroller circuit (using an IC such as a PIC16F684 or '690) to control the speed of rotation. The speed of rotation must be measured using the back-EMF of the motor between PWM pulses, as there is no shaft speed sensor.

3 The Specification

Your controller should ideally keep the speed of the shaft constant. The set point must be adjustable by means of a potentiometer. The power supply can vary, as it may have to work using batteries or a power supply built by someone else. Ability to handle at least 8–16 volts is desirable. The controller should not "hunt" up and down around the set point, as this would disturb the patient. You should provide some sort of status indication, so that untrained observers can get "situational awareness", i.e., see if the thing is working or struggling/malfunctioning, etc.

4 Assessment

4.1 Written

You should write a very brief report, enough for someone less experienced than you to duplicate your work, i.e., manufacture more of your gadgets, and know if it is working as desired, i.e., you provide test instructions, and be able to read your firmware, i.e., tackle a bug if one turns up and must be fixed. This should be in PDF format and should contain at least the following—

- Description of what kind of control has been implemented (Bang-bang, proportional, PI, PID, etc).
- Logical layout of your program, key algorithms, etc.
- Description of any key design aspects, such as what events are handled in an interrupt routine, what power the circuit must deliver, etc.
- Clear circuit diagram. (Appendix?)
- Construction layout diagram/photo. (Appendix?)
- Brief description of what the components in the circuit do, a "how it works", indicating how critical component values were chosen, what groups of components achieve, etc.
- What indications your circuit provides that it is operating successfully.
- Description of an electronic test that will demonstrate that the feedback is operating, as a sort of "manufacturing check" that can be applied to a circuit once built.
- A code listing should be appended.[1]

4.2 Live

You must demonstrate your controller working. This should be done over a video call, so as to maintain strict social isolation.

Your demonstration should highlight the performance of your system, allow your company executives to see that you have succeeded, ask questions, and for them to assess how good is your solution. Consider the specifications above, and explain the extent to which you have achieved them. As was implied, you do not need to achieve everything, and an economical, robust, and elegant design is more valuable than one which hits all the ideal performance specs but is flakey or excessively complicated.

[1] Think a little about how your code looks printed out. Lines wrapping can ruin the readability of code. Comments should be informative, not daft replication of the obvious. Don't leave yourself looking as if you work for Microsoft.