Fluid Depth Control/Pump Driver

Mechatronics ENEL417/517 Lab Project

Jonathan Scott & Benson Chang

1 Introduction

This lab involves designing a circuit, constructing the hardware, and programming an embedded microcontroller to control a small DC motor. The motor operates a pump that fills a tank. The project is nominally due at the end of week 7, but can be delayed so as to include the recess. You are *strongly* advised not to leave things to the last day... and the sooner this project is done, the sooner you can have a rest or tackle the next one.

2 The Hardware

The pump is a 12V, submersible type, and resides in a sump. When supplied with sufficient current it lifts water from the sump to the inlet of a glass, cylindrical tank. The tank is also provided with a drain governed by a tap, and an overflow oulet, both of which drain to the sump. Your job is to design, build, and program a small microcontroller circuit (using an IC such as a PIC16F684, '690, or '1825 with a PWM function) to control the depth of the water in the tank. You will also have to arrange a method to sense the depth (conductivity, pressure and sonar have all been used previously with varying degrees of success). The pump typically draws about 1–1.5A, anything in excess of 2.5A represents a fault. You should use a 15V supply.

3 The Specification

Your controller should allow the depth to be set with a potentiometer to about 60% full. The level should be constant as far as possible irrespective of the tap (drain) flow setting. In other words, after adjustment of the tap, the controller should adjust the motor to return to the set level, and not "hunt" up and down around the set point too much. Naturally, the controller should strenuously avoid either allowing the drain flow to fall to zero or the overflow to carry any liquid at all. These would constitute "failure" of your controller.

Students enrolled in ENEL517 should implement full PID control with zero perceptible steady-state depth error.

4 Assessment

As usual, one third of the marks ($\frac{1}{3}$ of 30%) is for self management and business-like conduct. One third is awarded for the demonstration and defence of your design, the finish of the hardware, information provided by indicators, etc. One third will be awarded for technical competence and the report.

4.1 Advice: Written

You should write a very brief report, enough for your colleague at another university to duplicate your work. This should be in PDF format and should contain at least the following—

- Description of how depth is sensed;
- Description of what kind of control has been implemented (Bang-bang, PI position, etc), and how control loop constants were set.
- Description of any key design aspects, such as what events are handled in an interrupt routine.
- Clear circuit diagram. (Appendix?)
- PCB or 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.

Put things like software printouts in appendices. At a minimum, a code listing should be appended.[1]

4.2 Advice: Live Demonstration

You must demonstrate your controller working. Please make an appointment with the course lecturer, in other words, *don't* come racing up an hour before the deadline unannounced.

Your demonstration should highlight the performance of your system. 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.

There is a trade-off between being the first to demonstrate your project, pleasing the boss with your speed, efficiency and punctuality, and getting onto the next one, contrasted against watching others demonstrate theirs so you know what you are in for and can tweak your own system before you commit to demonstration. Once your demonstration is accepted by the boss, you can brainstorm the next project.

[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.