Lab 1
Creating a Power Plant Monitor using Incremental Building and Modular Coding

January 20, 2022

Learning Objectives

1. Utilize the Microchip Studio IDE and the Arduino to program an AVR microcontroller.
2. Create header files to modularize larger programs into a series of more manageable and coherent source files.
3. Utilize interrupts for time-sensitive functions.
4. Program a microcontroller to interface with analog sensors using the on-board analog-to-digital converter (ADC).
5. Configure timers to generate pulse-width modulated pulse trains for controlling servos.
6. Utilize the incremental build model of software development to incrementally design and test a system or product.

Overview

This lab consists of constructing a power plant monitoring prototype in order to reinforce the above learning objectives. A temperature monitor is an example of an embedded system where the controller is completely encapsulated within the product[1]. In this lab, we will prototype some of the essential components of a digital temperature monitoring system, including its microcontroller (MCU), display, keypad/buttons, and motors/actuators. All of these components will be controlled and integrated using an Atmel AVR MCU.

Lab Requirements

During this lab, you are required to demonstrate a working prototype of your power plant monitor to your instructor. This demonstration will reinforce the incremental build model [2], which is a part of the Agile process that you will follow during your capstone design project. In essence, the incremental build process is about decomposing a system into incremental build-phases that can be tested along the way to the finished product. In general, it is good practice to decompose a

[1] Embedded Computer Systems
[2] Wikipedia

system's requirements into a series of tasks and functional components. Once decomposed, the problem becomes more manageable and incremental steps toward the finished product can be achieved more rapidly and easily.

The programming code for this project must be logically organized within separate C source files. See references for more details on using header files to link together C programming modules. Below are the minimum specifications of your prototype and good incremental stages for testing & demonstrating your design:

- A display for the user. The display should show the current temperature and the desired temperature. It could be a terminal window such as Putty or a LCD display. 10 points.
- A temperature sensor such as a LM35 and using the analog-digital interface. 10 points.
- A PWM-controlled servo to simulate the operation of a ventilation system to perform a cooling operation once the actual temperature is greater than the desired temperature. 10 points.
- Two push buttons to provide a desired temperature (using external interrupts). 10 points.
- A blinking LED that starts once the actual temperature is greater than the desired temperature. 10 points.

Deliverables

- Demonstrate the above experiments to your instructor. The sequence could be: actual and desired temperature is displayed on a terminal window. User presses one of two push buttons connected to external interrupts to increase or decrease the desired temperature. Once the temperature is exceeded, an interrupt service routine causes the LED to blink and the servo to spin continuously (i.e. sets a Boolean variable). Given a simple embedded system, you will develop or make substantial modifications to integrate additional capabilities.
- Submit a detailed laboratory report following the guidelines provided in the lab memo on the course website. As part of your write-up, identify the Cyber concerns associated with such an embedded system. 50 points.
- All code for your project should be submitted as part of your lab report's appendix section.
- Submit an electronically signed lab report with full documentation and sources. Upload to Canvas.
- The lab demonstration will be in groups of two. The lab reports are individual efforts.

Lab Demonstration Grading Rubric

Task	Proficient (10)	Intermediate (8)	Developing (6)	Failure (3)
Temperature Display	Displayed on terminal window and updates periodically (i.e. 1 Hz)			
Push buttons change desired temp	Desired temp increases and decreases			
LED blinks	LED blinks at a rate that is observable by humans			
ADC converts to temperature	ADC captures and converts voltage to temperature			
Servo actuates	The servo moves to a new position to simulate a fan			