EE333 Microcontroller Engineering
Oregon Tech Portland, Fall 2013

Lab Assignment #4 - *Thermostat*
Due November 14

Objectives:

Demonstrate how to configure and use GPIO pins. Demonstrate the use of hysteresis to reduce short cycling. Demonstrate the detection of change.

Equipment Required:

Your starting point is your working lab 3 assignment.

Background Information:

The Dragon12 board has a relay connected to one of the GPIO pins. Investigate the documentation to find which pin is used and how to configure the pin to operate the relay.

We want the relay to control an air conditioner. The relay is to be energized when the temperature goes above the value set by variable *setpoint*. The relay is to be de-energized when the temperature goes below the value set by variable *setpoint*. The variable is initialized (for you) to 80 ° which would be fine for Fahrenheit, but you will want to modify the source code to a value like 27. This means that, if the value isn't changed, we want the relay to be energized when the temperature goes from 80 to 81, and de-energized when it goes from 80 to 79. The range between 79 and 81 is referred to as the "dead-band", and is the range where the relay might or might not be energized. Having a higher value for switching on rising values than for falling values is called hysteresis. We want hysteresis so that when the air conditioner is turned on it will continue to run for a reasonable period of time. Without hysteresis it might turn off quickly, only to turn on again a very short time later, a situation called short cycling. This tends to be inefficient and can cause excessive wear on the equipment.

You can implement the hysteresis by turning on the relay when the temperature is greater than the set-point and turning off the relay when the temperature is less than the set-point. If the temperature is the same as the set-point, do nothing. Nothing happens if you turn on the relay and it is already on or if you turn off the relay and it is already off, so you don't have to worry about this.

In this assignment you will be driving the RGB LED based on if the temperature is rising or falling. Because the temperature can perform slow fluctuations up and down in a "normal" room, we need to measure for the change over a period of time. To do this in my demonstration program, every 2.5 seconds I take the current temperature value (the 16-bit value that is 256 times the actual temperature) and compare it to the value from 2.5 seconds previous. Since the display is being updated every .25 seconds, I use a counter variable to count the number of display

updates and every time the count becomes 10, I reset the value and I do the comparison, light the appropriate LED segment and save the current temperature in the variable that held the old temperature. This way, every 2.5 seconds it does a comparison between the current temperature and the temperature that was the current temperature in the comparison 2.5 seconds in the past.

The documentation for the Dragon12 board explains which port pins are used to drive the LED. Note that the RGB LED must be enabled by forcing Port M pin 2 to 0.

Assignment:

Modify your solution to lab assignment 3 so that the relay is controlled based on the temperature set by variable *setpoint*. There should be a 2 degree hysteresis, as described above. Test your program to be sure it works. There is an LED that indicates if the relay is energized. You don't have to connect the relay to an air conditioner!

Take the temperature measurement every 2.5 seconds to determine rise or fall. Have your program light the red segment of the RGB LED if the temperature is rising to indicate a warning. Have it light the green segment of the RGB LED if the temperature is falling, The RGB LED should remain dark if the temperature is steady.

To turn in:

- Documented program listing.
- Description of how you tested the program.
- Discussion of any problems you had

This should all be placed in a single file (PDF format preferred, Word or Open Office formats also acceptable).