Overview

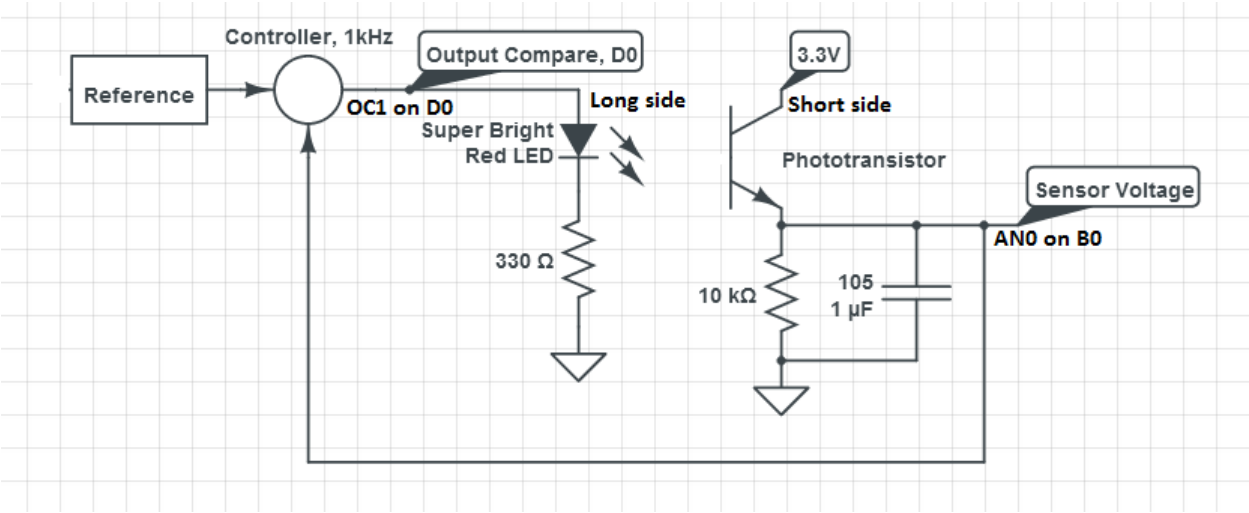
The goal of this project is to control the light emitted by an LED. We provide a signal that specifies the brightness of the LED over time, and program the PIC to make the LED brightness follow the signal. The steps to make this work are outlined here: they are extremely similar to what must be done to control motors.

The project consists of 4 parts.

Parts 1 & 2 are due on Tuesday 2/18 before class. You will demo the response to part 2 question 7 in class. Include answers to all the questions (including nScope snapshots) and the source code for the answer to part 2 question 7.

Parts 3 & 4 will be posted soon and will be due on Tuesday 2/25 before class

Here is a diagram, including the circuits and control loop. The reference signal is stored as an array on the PIC.



The project consists of several components:

1. The LED circuit. We control the brightness of the LED by turning it on and off really fast using a PWM signal. The switching is too fast for the human eye to see, so it appears as different brightness levels, depending on the duty cycle of the signal.
    - Part 1 of the assignment had you control the light level of this LED
    - In Part 2, you will use the Output Compare feature to control the light level, rather than manually changing it in a timer.
2. The reference generator generates the signal (aka the reference) that we want the LED to track. The signal is stored in an array on the PIC, and is played back by a timer interrupt.
    - In Part 2, you will create this reference signal and use a timer to change the PWM duty cycle. This will cause the LED to approximately follow your signal.
3. The sensing circuit. This is so we can measure how much light is emitted, and adjust the brightness of the LED accordingly. The voltage will be measured on the PIC using the analog to digital converter (ADC), which converts voltage into a digital integer value.

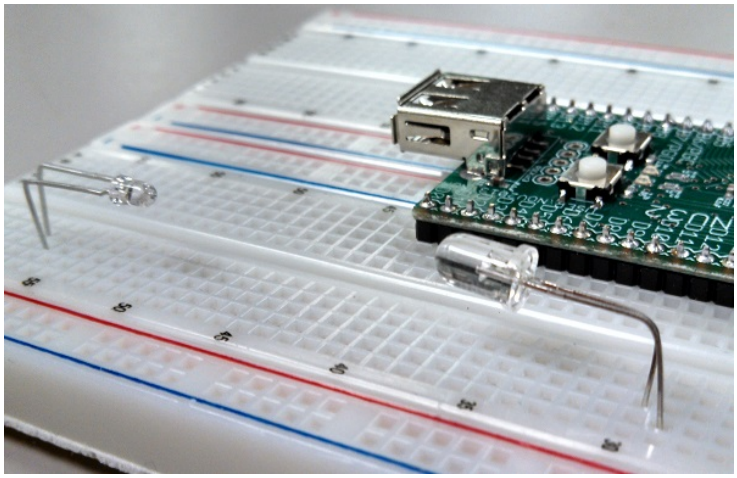
- The sensing circuit includes a low-pass RC filter. In Part 2, you will see how this RC filter converts the on/off signal from the PWM into a voltage that corresponds to the brightness of the led
- In Part 3, you will hook the output of this signal up to an analog input pin on the PIC and read the values of this voltage.

4. The control loop. In the timer where you change the reference signal, you will also read the sensor value. Using the reference and sensed value, you will implement a control law that uses the error between what you want the LED brightness to be and what your sensor reads to change the PWM signal, thus making the LED follow the reference signal.
    - In part 1, the timer you created can be thought of as a primitive control loop. Each time the ISR triggered, you turned the LED either on or off, this controlled the LED
    - In part 2, you will use the control loop to change the PWM signal to the LED, according to a reference signal. This is known as "open loop" control because you are sending a control signal to the LED, but are not using any sensor data.
    - In part 3, you will read sensor data in addition to outputting the reference signal. You will also be able to output the reference and sensed signal on the computer and compare them. You will see that the sensed value does not match the reference signal.
    - In part 4, you will implement the controller. Your data will show that the controller makes the sensor reading match the reference signal. Mission Accomplished!

Step 2 - In class 2/13

1. The sample code we have provided generates a 1kHz PWM signal on OC1 (which is output on pin D0, the pin your LED should be connected to). The OC1RS tick count is entered by the user over serial. View the output of this PWM signal on the nScope (channel A). Play around with different numbers and observe how the wave changes. Approximately how many ticks yield a 25% duty cycle? A 75% duty cycle?

2. Wire up the phototransistor circuit, EXCLUDING THE CAPACITOR. Use a 10kOhm resistor to ground. Place the photo transistor so it is pointed directly at the LED. View the output of the phototransistor on the nScope (Channel B). What do you observe?



Approximate orientation of the LEDs

3. Add a 1uF capacitor in parallel with the 10k resistor. How does the signal change? Try a few different duty cycles.

4. We have provided an array named waveform. We will use this array to set the values of the OC1RS register. This array is designed to be played back at 10 Hz. It is initialized by the makeWaveform function, called from main(). Setup Timer 2 to operate at 10 Hz. In the Timer 2 ISR, cycle through the waveform array and set the OC1RS register according to the value at the current position in the waveform. Observe the output at the LED and at the phototransistor. Are they the same?

Exercises:

4. Generate a 20KHz PWM signal on pin OC1. Use two different combinations of prescaler and PR3 register. Which combination gives you more precise control over the duty cycle?

5. What are the advantages of using the built in PWM function rather than manually performing PWM in a timer, as in Part 1?

6. Setup Timer 2 to run at 1 kHz. Write a function that fills the duty cycle array with the appropriate values to create a 1 Hz square wave, sampled at 1 kHz. Include the ability to specify the minimum and maximum value for the square wave.

7. Generate a square wave with a minimum of roughly 1V and a maximum of roughly 2V. Observe on the nScope. Do the minimum and maximum match with what you specified? What about the frequency?