Controls & Embedded Systems Lab

In this lab, we will learn about how a microcontroller can take in inputs from the real world, process them, and use them to affect something physical.

1. The Circuit

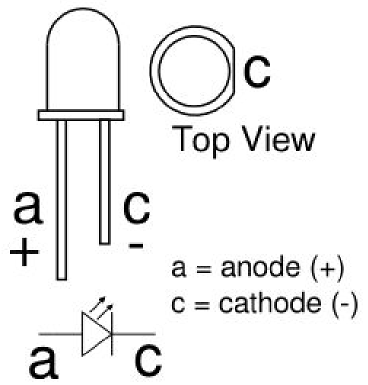
First we will build the circuit. You will build an input circuit and an output circuit. The input circuit takes measurements and sends them to the MSP (the controller). Then the MSP controls the output circuit. Note: While holding the breadboard such that the divider is vertical, all the holes in each horizontal row are connected, meaning in order for the circuit to work, the 2 legs of any one component must be in different rows!

You will need:

1. MSP430 Launchpad
2. 3 LEDs
3. Photoresistor
4. 3 220 Ohm resistors
5. 10k Ohm resistor
6. Breadboard
7. 100 uF capacitor

Schematic:
http://tinyurl.com/y5akjrxw

In case you forgot which way an LED goes:



IMPORTANT: The capacitor is polarized, meaning you **MUST** place the capacitor such that the negative side (the side with the shorter leg, corresponds to a big minus sign on the capacitor) is connected to ground, and the other side/leg is connected to the 10k resistor and wire to the MSP pin.

2. The Code

Before you move on, you must check with one of the instructors to make sure your circuit is correct - it's quite easy to fry the MSPs if you're not careful.

Now we need to implement our controller. Open up Energia and open up the .ino file. Take a look at the code and see if you can get an idea of what it is doing.

a. First try to upload the code to the MSP.
b. Try covering, shining a light on, uncovering, etc, the photoresistor. How does this affect the rate at which the LEDs are turning on and off?
c. If you uncover or cover the photoresistor while the LEDs are turning on, what happens?

3. Delay

a. Play around with the `period` to see how it affects the output. Try values like `period = 100` or `period = 500`. How does the smoothness of the rate of change of the LEDs' brightness change as we decrease the rate at which we take measurements?

4. PID Control Example (Optional)

If you are interested in PID control, here is an online example of how it works: https://sites.google.com/site/fpgaandco/pid

a. You can use the horizontal slider to change the target position of the car.
b. To look at a simple controller (such as the one we implemented) **set all gains but Kp to 0**. Try moving around the setpoint and see the response.
c. How can we reduce oscillation? Try increasing the D term. What happens to the response if the D term is too high?
d. Now try using the vertical slider to create a slope. Notice that the car does not quite reach the desired position (there's steady-state error). Try using the I-term to fix that. What happens if you make the I-term too large?
e. Tuning a PID controller is quite system-specific. If you play around with the car parameters (mass, motor force limit) you might notice different response to the same change in setpoint.
    i. For instance, if you make the mass really low and the motor force limit really high, you will see that high derivative coefficients actually reduce the time it takes to reach the desired setpoint. While this might seem great, in the real world when there is noise, this might end up amplifying the noise (as you jump to desired setpoints near instantaneously, even if those setpoints may not be correct). This is why it may be good to impose artificial limits on the

maximum amount of “force” you can exert on a system. Without limits on the maximum “force” the derivative term can amplify noise (as the instant the setpoint is changed, it provides a large boost).