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. More specifically, you'll learn how to make a basic auto-brightness feature (like the one your phone, or computer uses).

1. The Circuit

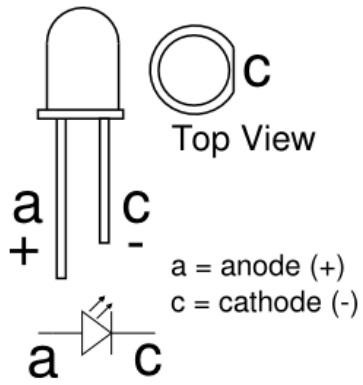
First we will build the circuit. You will build an input circuit and an output circuit. The input circuit takes measurements and sends them to the MSP (the controller). Then the MSP controls the output circuit.

You will need:

1. MSP430 Launchpad
2. LED
3. Photoresistor
4. 220 Ohm resistor
5. Breadboard

Schematic:
http://www.falstad.com/circuit/circuitjs.html?cct=$+1+0.000005+10.200277308269997+50+5+43%0Am+176+112+176+224+0+100+16000+0+1e-8+1e-10+0%0A207+176+160+240+144+0+Photoresistor%0A207+176+112+176+64+0+To+5V+Pin+of+MSP%0Ar+176+224+176+288+0+220%0Ag+176+288+176+336+0%0A207+176+224+112+224+0+To+pin+6.1+of+MSP%0A207+384+112+384+64+0+To+pin+2.5+of+MSP%0A162+384+112+384+192+2+default-led+1+0+0+0.01%0Ar+384+192+384+256+0+220%0Ag+384+256+384+320+0%0A

In case you forgot which way an LED goes:



2. The Code

Now we need to implement our controller. Open up Energia on one of the Lab computers 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. We recommend you do this on a lab computer, otherwise you will need to install Energia and the driver for the MSP (which is on Github).
b. Our first step will be to find the range of our input signal. Try running the code and using the serial monitor to find the highest and lowest signal observations. Find the brightest and darkest environments that you want your circuit to function with and see what values are output as observations. If you are having issues reading the observations (like if the observations are always around 60 then you might want to try using a different pin for the sensor - e.g. `P6_2` instead of `P6_1;` remember to change the code too!`)`. Feel free to ask the lab assistants if you need help debugging your circuit.
c. Now, plug these values into `min_obs` and `max_obs` and re-upload the code. Does your circuit work as expected?

3. Delay

a. Play around with the `period` to see how it affects our filtering. Try values like `period = 100` or `period = 1000`. How does the responsiveness of our controller change as we decrease the rate at which we take measurements?

4. Filtering

a. Set the period back to 1. Play around with the window size to see how it affects our filtering. How does the behavior change when `window = 1`? How about when `window = 20`? Try waving your hand rapidly above the photoresistor (to simulate noise) and to cover / uncover the photoresistor (to simulate an actual change in desired output). Compare the responses for different window sizes. What are the advantages and disadvantages of different window sizes?

5. 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).